Mechanics of Forming and Estimating Dynamic Linear Economies

Evan W. Anderson
University of Chicago

Lars Peter Hansen
University of Chicago

Ellen R. McGrattan
Federal Reserve Bank of Minneapolis

Thomas J. Sargent
University of Chicago
and
Hoover Institution
Stanford University

December 8, 1995

ABSTRACT
This paper catalogues formulas that are useful for estimating dynamic linear economic models. We describe algorithms for computing equilibria of an economic model and for recursively computing a Gaussian likelihood function and its gradient with respect to parameters. We apply these methods to several example economies.

Lars Peter Hansen and Thomas J. Sargent acknowledge financial support from the National Science Foundation, and Evan W. Anderson from a University of Chicago Century graduate fellowship. This report benefited greatly from insightful comments by an anonymous referee. We especially thank Peter Zadrozny for his invaluable comments. Conversations with Sherwin Rosen were very helpful in formulating two of our example economies and in estimating the cattle cycle model. To obtain computer programs that implement the calculations described in the appendices, please send an e-mail message to

erm@ellen.mpls.frb.fed.us. To obtain computer programs that implement the algorithms for solving Riccati and Sylvester equations, please send an e-mail message to ewanders@midway.uchicago.edu. The views expressed in this paper are those of the authors and not necessarily those of the Federal Reserve Bank of Minneapolis or the Federal Reserve System.

Mechanics of Forming and Estimating
Dynamic Linear Economies

Evan W. Anderson, Lars Peter Hansen,
Ellen R. McGrattan and Thomas J. Sargent

1. Introduction

This paper describes recent advances for rapidly and accurately solving matrix Riccati and Sylvester equations and applies them to devise efficient computational methods for solving and estimating dynamic linear economies. The paper surveys the most promising solution methods available and compares their speed and accuracy for some particular economic examples. Except for the simplest dynamic linear models, it is necessary to compute solutions numerically. In estimation contexts, computation speed is important because climbing a likelihood function can require that a model be solved many times. We describe methods that are faster than direct iterations on the Riccati equation and are more reliable than solutions based on eigenvalue-eigenvector decompositions of the state-costate evolution equation. Our survey of these methods draws heavily on Anderson (1978), Gardiner and Laub (1986), Golub, Nash and Van Loan (1979), Laub (1979,1991), and Pappas, Laub and Sandell (1980).

This paper is organized as follows. Section 2 decomposes the optimal linear regulator into sub-problems that are more efficient to solve and describes classes of economic problems that give rise to such problems. Sections 3, 4, 5, and 6 describe recent algorithms for solving these sub-problems. Section 7 extends the range of the basic algorithms to the domain of "distorted economies" whose equilibria do not correspond to solutions of optimum problems. Section 8 describes three particular economic models, one of which is the cattle cycle model of Rosen, Murphy and Scheinkman (1994). Section 9 uses each of these models as contexts for speed and accuracy comparisons of alternative algorithms. Sections 10 and 11 briefly describe innovations representations and recursive computation of Gaussian likelihood functions. Two appendices (A and B) provide formulas for computing derivatives of a Gaussian likelihood with respect to a set of unknown parameters governing the tastes, technology, and information flows of our economic models. These formulas, which build directly from the work of Zadrozny (1988a, 1989), are designed to make numerical search algorithms for maximizing a likelihood function more reliable and to assist in making statistical inferences about the parameters of interest. Section 12 uses these formulas to estimate Rosen, Murphy, and Scheinkman's model.

2. Control Problems

In this section, we pose three optimal control problems. We begin with a problem close to the much studied time-invariant deterministic optimal linear regulator problem. We label this problem the *deterministic regulator problem.* We then consider two progressively more general problems.

The first generalization introduces forcing sequences or "uncontrollable states" into the *deterministic regulator problem.* While this generalization is also a *deterministic regulator problem*, there are computational gains to exploiting the *a priori* knowledge that some components of the state vector are *uncontrollable.* We refer to this generalization as the *augmented regulator problem.* As we will see, a convenient first step for solving an *augmented regulator problem* is to solve a corresponding *deterministic regulator problem* in which the forcing sequence is "zeroed out." In other words, we obtain a piece of the solution to the *augmented regulator problem* by initially solving a problem with a smaller number of state variables.

The second generalization introduces, among other things, discounting and uncertainty into the *augmented regulator problem.* We refer to the resulting problem as the *discounted stochastic regulator problem.* Using well known transformations of the state and control vectors, we show how to convert this problem into a corresponding undiscounted *augmented regulator problem* without uncertainty. Therefore, while our original problem is a *discounted stochastic regulator problem*, we solve it by first solving a *deterministic regulator problem* with a smaller number of state variables, then solving a corresponding *augmented regulator problem*, and finally using this latter solution to construct the solution to the original problem in the manner described below.

2.1. Deterministic Regulator Problem

Choose a control sequence $\{v_t\}$ to maximize

$$-\sum_{t=0}^{\infty}(v_t{}' R v_t + y_t{}' Q_{yy} y_t),$$

subject to

$$y_{t+1} = A_{yy} y_t + B_y v_t$$

$$\sum_{t=0}^{\infty}(|v_t|^2 + |y_t|^2) < \infty. \tag{2.1}$$

This control problem is a standard time-invariant, deterministic optimal linear regulator problem with one modification. We have added a stability condition, (2.1), that is absent in the usual formulation. This stability condition plays a central role in at least one important class of dynamic economic models: permanent income models. More will be said about these models subsequently. In these models, the stability condition can be viewed as an infinite horizon counterpart to a terminal condition on the capital stock.

Following the literature on the time-invariant optimal linear regulator problem, we impose the following:

Definition: The pair (A_{yy}, B_y) is *stabilizable* if $y' B_y = 0$ and $y' A_{yy} = \lambda y'$ for some complex number λ and some complex vector y implies that $|\lambda| < 1$ or $y = 0$.

Assumption 1: (A_{yy}, B_y) is stabilizable.

Stabilizability is equivalent to the existence of a time-invariant control law that stabilizes the state (see Anderson and Moore, 1979, Appendix C). For our applications, it can often be verified by showing that a trivial control law, such as setting investment equal to zero, achieves this stability.

In solving this problem, we are primarily interested in specifications for which all of the state variables are "endogenous," and hence the following stronger restriction is met:

Definition: The pair (A_{yy}, B_y) is *controllable* if $y' B_y = 0$ and $y' A_{yy} = \lambda y'$ for some complex number λ and some complex vector y implies that y is zero.

When (A_{yy}, B_y) is controllable, starting from an initialization of zero, the state vector can attain any arbitrary value in a finite number of time periods by an appropriate setting of the controls (see Anderson and Moore, 1979, Appendix C).[1] For this reason, we can think of a state vector sequence with evolution equation governed by a pair (A_{yy}, B_y) that is controllable as being an *endogenous* state vector sequence.

While Assumption 1 gives us a nonempty constraint set, it is still possible that the *supremum* of the objective is not attained. We assume the following:

Assumption 2: The matrix Q_{yy} is positive semidefinite, and the matrix R is positive definite.

Among other things, this concavity assumption puts an upper bound of zero on the criterion function. Therefore, the *supremum* is finite (and nonpositive). We require that the *supremum* is attained.

Assumption 3: There exists a solution to the *deterministic regulator problem* for each initialization of y_0.

A commonly used sufficient condition in the control theory literature for there to exist a solution is *detectability*. Factor $Q_{yy} = D_y {D_y}'$.

Definition: The pair (A_{yy}, D_y) is *detectable* if $D_y' y = 0$ and $A_{yy} y = \lambda y$ for some complex number λ and some complex vector y implies that $|\lambda| < 1$ or $y = 0$.

When the pair (A_{yy}, D_y) is detectable, it is optimal to choose a control sequence that stabilizes the state vector. In this case, the solution to the control problem is the same with or without the stability constraint (2.1). However, as we mentioned previously, for permanent income models the stability constraint is essential for obtaining an interpretable solution to the problem. For these models, detectability is too strong of a condition to impose. Chan, Goodwin and Sin (1984) give a weaker sufficient condition for there to exist a solution (see (iii) of Theorem 3.10). In the context of a continuous-time formulation, Hansen, Heaton and Sargent (1991) proposed a very similar sufficient condition for stabilizable systems based on a spectral representation of the *deterministic regulator problem.* Unfortunately, these conditions may be tedious to check in practice. Some of the solution algorithms we survey below could in principle be modified to detect a violation of Assumption 3.

A sufficient condition for convergence of one of the solution algorithms that we survey below is that the pair (A_{yy}, D_y) be *observable*:

Definition: The pair (A_{yy}, D_y) is *observable* if ${D_y}' y = 0$ and $A_{yy} y = \lambda y$ for some complex number λ and some complex vector y implies that $y = 0$.

Clearly, observability is stronger than detectability. Moreover, observability is guaranteed when the matrix Q_{yy} is nonsingular. When the pair (A_{yy}, D_y) is observable, the value function associated with the *deterministic regulator problem* is strictly concave in the state vector y (Caines and Mayne 1970, 1971).

[1] This is one of five equivalent characterizations of *reachability* given in Appendix C of Anderson and Moore (1979). However, many other control theorists take one of these characterizations as the definition of *controllability.* For instance, see Kwakernaak and Sivan (1972) and Caines (1988). We choose to follow this latter convention.

The solution to the *deterministic regulator problem* takes the form

$$v_t = -F_y y_t$$

for some feedback matrix F_y. Stability constraint (2.1) guarantees that the eigenvalues of $A_{yy} - B_y F_y$ have absolute values that are strictly less than one because the state evolution equation when the optimal control is imposed is given by

$$y_{t+1} = (A_{yy} - B_y F_y) y_t.$$

2.2. Augmented Regulator Problem

Choose a control sequence $\{v_t\}$ to maximize

$$-\sum_{t=0}^{\infty} (v_t{}' R v_t + y_t{}' Q_{yy} y_t + 2 y_t{}' Q_{yz} z_t),$$

subject to

$$\begin{bmatrix} y_{t+1} \\ z_{t+1} \end{bmatrix} = \begin{bmatrix} A_{yy} & A_{yz} \\ 0 & A_{zz} \end{bmatrix} \begin{bmatrix} y_t \\ z_t \end{bmatrix} + \begin{bmatrix} B_y \\ 0 \end{bmatrix} v_t$$

$$\sum_{t=0}^{\infty} (|v_t|^2 + |y_t|^2) < \infty.$$

We have modified the linear regulator problem by including the *exogenous* forcing sequence $\{z_t\}$. The presumption here is that this partitioning may occur naturally in the specification of the original control problem. Of course, as is well known in the control theory literature, we could always transform an original state vector into controllable and uncontrollable components. Constructing this transformation, however, can be difficult to do in a numerically reliable way. In the next section we will display a class of optimal resource allocation problems associated with dynamic economies for which z_t contains a vector of taste and technology shifters. By assumption, this component of the state vector cannot be influenced by a control vector such as the level of investment.

For the *augmented regulator problem* to be well posed, we require that the forcing sequence be stable:

Assumption 4: The eigenvalues of A_{zz} have absolute values that are strictly less than one.

The solution to the *deterministic regulator problem* gives us a piece of the solution to the *augmented regulator problem*. More precisely, the solution to the augmented problem is

$$v_t = -F_y y_t - F_z z_t,$$

where the matrix F_y is the same as in the solution to the regulator problem for which the forcing sequence $\{z_t\}$ is zeroed out. Consequently, our solution methods entail first computing F_y by solving a deterministic regulator problem of lower dimension and then computing F_z given F_y.

2.3. Discounted Stochastic Regulator Problem

Let $\{\mathcal{F}_t : t = 0, 1, ...\}$ denote an increasing sequence of sigma algebras (information sets) defined on an underlying probability space. We presume the existence of a "building block" process of conditionally homoskedastic martingale differences $\{w_t : t = 1, 2, ...\}$, which obeys

Assumption 5: The process $\{w_t : t = 1, 2, ...\}$ satisfies
(i) $E(w_{t+1}|\mathcal{F}_t) = 0$;
(ii) $E(w_{t+1}{w_{t+1}}'|\mathcal{F}_t) = I$.

The *discounted stochastic regulator problem* is to choose a control process $\{u_t\}$, adapted to $\{\mathcal{F}_t\}$, to maximize

$$-E\left(\sum_{t=0}^{\infty}\beta^t \left[\, {u_t}' \quad {x_t}' \,\right] \begin{bmatrix} R & W' \\ W & Q \end{bmatrix} \begin{bmatrix} u_t \\ x_t \end{bmatrix} \,\middle|\, \mathcal{F}_0 \right),$$

subject to

$$x_{t+1} = Ax_t + Bu_t + Cw_{t+1}$$

$$E\left(\sum_{t=0}^{\infty}\beta^t (|u_t|^2 + |x_t|^2) \,\middle|\, \mathcal{F}_0 \right) < \infty.$$

The state vector x_t is taken to be the composite of the endogenous and exogenous state variables. Let $U_y = [I \ \ 0]$ be a matrix that *selects* the endogenous state vector $U_y x_t$ and $U_z = [0 \ \ I]$ be a matrix that *selects* the exogenous state vector $U_z x_t$ for an optimization problem with discounting. To justify our partitioning, the matrix A is restricted to satisfy $U_z A {U_y}' = 0$, and the matrix B is restricted to satisfy $U_z B = 0$. Notice that in addition to incorporating discounting and uncertainty, the *discounted stochastic regulator* includes cross-product terms between controls and states, which are absent in the *augmented control problem*.

We now apply a standard trick for converting a *discounted stochastic regulator problem* to an *augmented regulator problem*. Using the well known certainty equivalence property of stochastic linear regulator problems, we zero out the uncertainty without altering the optimal control law. That is, we are free to set the matrix C to zero and instead solve the resulting deterministic control problem. We eliminate discounting and cross-product terms between states and controls by using the transformations

$$y_t = \beta^{t/2} U_y x_t, \quad z_t = \beta^{t/2} U_z x_t, \quad v_t = \beta^{t/2}(u_t + R^{-1} W' x_t).$$

As is evident from these formulas, we have absorbed the discounting directly into the construction of the transformed state and control vectors. In addition, the cross-product matrix S is folded into the construction of the transformed control vector. We are left with a version of the *augmented regulator problem* with the following matrices:

$$\begin{bmatrix} A_{yy} & A_{yz} \\ 0 & A_{zz} \end{bmatrix} = \beta^{1/2}(A - BR^{-1}W'), \quad B_y = \beta^{1/2} U_y B,$$

$$\begin{bmatrix} Q_{yy} & Q_{yz} \\ {Q_{yz}}' & Q_{zz} \end{bmatrix} = Q - WR^{-1}W'. \tag{2.2}$$

Assumptions 1 - 4 are imposed on the constructed matrices on the left-hand side of the equal signs in (2.2).

As before, write the solution to the *augmented regulator problem* as

$$v_t = -F_y y_t - F_z z_t.$$

Then the solution to the *discounted stochastic regulator problem* is

$$u_t = -F x_t,$$

where

$$F = \begin{bmatrix} F_y \\ F_z \end{bmatrix} + R^{-1} W'.$$

Also as before, the matrix F_y can be computed by solving the corresponding *deterministic regulator problem* with the forcing sequence "zeroed out." In subsequent sections we will describe methods for computing F_y and F_z.

In macroeconomics, the *discounted stochastic regulator problem* is often obtained in the fashion of Kydland and Prescott (1982), who use it to replace a nonlinear-quadratic problem. Thus consider the nonquadratic optimization problem: choose an adapted (to $\{\mathcal{F}_t\}$) control process $\{u_t\}$ to maximize

$$-E\left(\sum_{t=0}^{\infty} \beta^t r(u_t, x_t) \,\middle|\, \mathcal{F}_0\right), \tag{2.3}$$

subject to

$$x_{t+1} = Ax_t + Bu_t + Cw_{t+1}.$$

Here r is not required to be a quadratic function of u_t and x_t. When the associated constraints are nonlinear, sometimes we can substitute the nonlinear constraints into the criterion function to obtain a problem of the form of (2.3). Kydland and Prescott (1982) simply replace the function r by a quadratic form in $[u_t' \quad x_t']'$ as required for the *discounted stochastic regulator problem*, where the quadratic function is designed to "approximate" r well near a particular value for the state vector.[2] In the next subsection, we describe a different approach where, by design, the initial optimal resource allocation problem can be directly converted into a *discounted stochastic regulator problem*.

2.4. A Class of Linear-Quadratic Economies

We will consider several numerical examples that are members of a class of economies used by Hansen (1987) and Hansen and Sargent (1994). As in the *discounted stochastic regulator problem*, there is an exogenous information vector z_t governed by

$$\hat{z}_{t+1} = \hat{A}_{zz} \hat{z}_t + \hat{C}_z w_{t+1}, \tag{2.4}$$

where $\{w_t\}$ satisfies Assumption 5, and $A_{zz} \equiv \sqrt{\beta} \hat{A}_{zz}$ satisfies Assumption 4. The vector $\hat{z}_t$ determines a time t preference shock b_t and a time t endowment shock d_t via

$$\begin{aligned} d_t &= U_d \ \hat{z}_t \\ b_t &= U_b \ \hat{z}_t. \end{aligned} \tag{2.5}$$

A representative household has preferences ordered by

$$-(1/2)E\left(\sum_{t=0}^{\infty} \beta^t (|s_t - b_t|^2 + |g_t|^2) \,\middle|\, \mathcal{F}_0\right), \tag{2.6}$$

[2] While Kydland and Prescott (1982) apply an *ad hoc* global approximation to r in which the range of approximation is adapted to the amount of underlying uncertainty, many subsequent researchers have instead simply used a local Taylor series approximation around some "nonstochastic" steady state produced by shutting down all randomness in the model. Kydland and Prescott (1982) note that for the range of uncertainty they considered, the two methods gave similar answers.

where g_t is a vector of labor-using intermediate activities (designed to capture generalized adjustment costs), and s_t is a vector of household services produced at time t via the household technology

$$\begin{aligned} s_t &= \Lambda h_{t-1} + \Pi c_t, \\ h_t &= \Delta_h h_{t-1} + \Theta_h c_t. \end{aligned} \tag{2.7}$$

In (2.7), h_t is a vector of stocks of household durable goods at t, c_t is a vector of consumption flows, and Λ, Π, Δ_h, Θ_h are matrices. There is a constant returns to scale production technology

$$\begin{aligned} \Phi_c c_t + \Phi_i i_t + \Phi_g g_t &= \Gamma k_{t-1} + d_t, \\ k_t &= \Delta_k k_{t-1} + \Theta_k i_t, \end{aligned} \tag{2.8}$$

where k_t is a vector of capital goods used in production, i_t is a vector of investment goods, and Δ_k is a matrix.[3] Hansen and Sargent (1994) describe a competitive equilibrium for this economy. Associated with the competitive equilibrium is a social planning problem, namely, to maximize (2.6) over choices of contingency plans for $\{s_t, c_t, i_t, g_t, k_t, h_t\}_{t=0}^{\infty}$ (adapted processes) subject to (2.4), (2.5), (2.7), and (2.8) with given initializations for (z_0, h_{-1}, k_{-1}).

To map this problem into the notation of the previous section, we let

$$x_t \equiv \begin{bmatrix} h_{t-1} \\ k_{t-1} \\ \hat{z}_t \end{bmatrix}.$$

We view the first two components of the state vector to be *endogenous* and the third component to be *exogenous*. The control vector u_t can be chosen to be investment i_t when the matrix $\Phi \equiv [\,\Phi_c \quad \Phi_g\,]$ is nonsingular because in this case[4]

$$\begin{bmatrix} c_t \\ g_t \end{bmatrix} = \Phi^{-1}(\Gamma k_{t-1} + U_d \hat{z}_t - \Phi_i i_t). \tag{2.9}$$

Using this relation, the constraints (2.7) and (2.8) can be rewritten

$$x_{t+1} = Ax_t + Bu_t + Cw_{t+1}$$

for appropriately chosen matrices A, B, C. The matrix A is block triangular and the bottom row block of B is zero as required for the *discounted stochastic regulator problem.* Moreover, using (2.9) and (2.7), the time t terms $|s_t - b_t|^2$ and $|g_t|^2$ in the objective function (2.6) of the social planner both can be expressed as quadratic forms in the control i_t and the augmented state x_t. Therefore, the social planner's problem is a *discounted stochastic regulator problem.*

In permanent income economies, stability of the state vector process is not obtained automatically as an implication of optimality. An example of such an economy is one with a single consumption and capital good and no labor-using intermediate activities. The counterpart to equation (2.9) is

$$c_t = \Gamma k_{t-1} + U_d \hat{z}_t - i_t.$$

We constrain the subjective discount factor to be the reciprical of the physical return to capital: $\beta = \frac{1}{\Gamma + \Delta_k}$. In the absence of a stability constraint, the solution to the resulting control problem does not "stabilize" the capital stock sequence because the sequence of capital stocks often diverges to minus infinity at a rate not even dominated by $\frac{1}{\sqrt{\beta}}$. This

[3] Under the constant returns to scale interpretation, d_t is taken as an additional "input" available in fixed supply.

[4] When Φ is singular, the control vector can be augmented to include some of the components of consumption or the labor-using intermediate activities.

solution to the control problem is not interesting. Therefore, we impose stability as an additional constraint, with the consequence that the solution to the resulting infinite-horizon control problem is equal to the limit of the solutions to a sequence of corresponding finite-horizon problems, each of which has a zero restriction imposed on the terminal capital stock.

3. Solving the Deterministic Linear Regulator Problem

In this section we describe ways to solve for the matrix F_y. Recall that this matrix has a double role. First, it gives the control law for a particular *deterministic regulator problem.* More importantly for us, it also gives a *piece* of the solution to the *discounted stochastic regulator problem.*

In describing methods for computing F_y, it is convenient to work with the state-costate equations associated with the Lagrangian

$$\mathcal{L} = -\sum_{t=0}^{\infty}[y_t' Q_{yy} y_t + v_t' R v_t + 2\mu_{t+1}'(A_{yy} y_t + B_y v_t - y_{t+1})]. \tag{3.1}$$

First-order necessary conditions for the maximization of $\mathcal{L}$ with respect to $\{v_t\}_{t=0}^{\infty}$ and $\{y_t\}_{t=0}^{\infty}$ are

$$v_t : \qquad R v_t + B_y' \mu_{t+1} = 0, \qquad t \geq 0 \tag{3.2}$$

$$y_t : \qquad \mu_t = Q_{yy} y_t + A_{yy}' \mu_{t+1}, \qquad t \geq 0. \tag{3.3}$$

To obtain a composite state-costate evolution equation, solve (3.2) for v_t, substitute the solution into the state evolution equation, and stack the resulting equation and (3.3) and write the state-costate evolution equation as

$$L \begin{bmatrix} y_{t+1} \\ \mu_{t+1} \end{bmatrix} = N \begin{bmatrix} y_t \\ \mu_t \end{bmatrix}, \tag{3.4}$$

where

$$L \equiv \begin{bmatrix} I & B_y R^{-1} B_y' \\ 0 & A_{yy}' \end{bmatrix}, N \equiv \begin{bmatrix} A_{yy} & 0 \\ -Q_{yy} & I \end{bmatrix}.$$

There is also a continuous-time counterpart to this system given by

$$\begin{bmatrix} D y_t \\ D \mu_t \end{bmatrix} = H \begin{bmatrix} y_t \\ \mu_t \end{bmatrix}, \tag{3.5}$$

where

$$H \equiv \begin{bmatrix} A_{yy} & -B_y R^{-1} B_y' \\ -Q_{yy} & -A_{yy}' \end{bmatrix}. \tag{3.6}$$

Equation (3.5) is the state-costate equation corresponding to the continuous-time regulator problem with criterion $-\int_0^{\infty}[y(t)' Q_{yy} y(t) + u(t)' R u(t)]dt$ and law of motion $Dy(t) = A_{yy} y(t) + B_y u(t)$, where D is the time-differentiation operator. We describe several methods for solving equations (3.4) and (3.5). Formally, we will devote most of our attention to the discrete-time system (3.4). As we will see, methods designed for solving the continuous-time system (3.5) can be adapted easily to solve the discrete-time system (3.4), and conversely.

The solution to (3.4) of interest to us is the one that stabilizes the state-costate vector sequence for any initialization y_0. Since we have transformed the state vector to eliminate discounting, we impose stability in the form of square summability:

$$\sum_{t=0}^{\infty} \left| \begin{bmatrix} y_t \\ \mu_t \end{bmatrix} \right|^2 < \infty, \tag{3.7}$$

for the discrete-time system (3.4). (We impose the analogous square integrability restriction on the continuous time system (3.5)).

One way to ascertain the solution to the *deterministic regulator problem* is to find an *initial* costate vector expressed as a function of the initial state vector y_0 that guarantees the stability of system (3.4) or (3.5). The initialization of the costate vector takes the form $\mu_0 = P_y y_0$ and is replicated over time. Substituting $P_y y_t$ for μ_t into (3.4), we find that

$$\begin{aligned} (I + B_y R^{-1} B_y{}' P_y) y_{t+1} &= A_{yy} y_t \\ A_{yy}{}' P_y y_{t+1} &= -Q_{yy} y_t + P_y y_t . \end{aligned} \tag{3.8}$$

It is straightforward to verify that

$$(I + B_y R^{-1} B_y{}' P_y)^{-1} = I - B_y (R + B_y{}' P_y B_y)^{-1} B_y' P_y . \tag{3.9}$$

Solving the first equation in (3.8) for y_{t+1}

$$y_{t+1} = (A_{yy} - B_y F_y) y_t , \tag{3.10}$$

where

$$F_y \equiv (R + B_y{}' P_y B_y)^{-1} B_y' P_y A_{yy} . \tag{3.11}$$

Premultiplying (3.10) by $A_{yy}' P_y$ gives

$$A_{yy}' P_y y_{t+1} = (A_{yy}' P_y A_{yy} - A_{yy}' P_y B_y F_y) y_t . \tag{3.12}$$

For the right-hand side of equation (3.12) to agree with the right-hand side of the second equation of (3.8) for any initialization y_0, it must be that

$$\begin{aligned} P_y &= Q_{yy} + A_{yy}' P_y A_{yy} - A_{yy}' P_y B_y (R + B_y' P_y B_y)^{-1} B_y' P_y A_{yy} \\ &= Q_{yy} + (A_{yy} - B_y F_y)' P_y (A_{yy} - B_y F_y) + F_y{}' R F_y \end{aligned} , \tag{3.13}$$

which is the familiar *Riccati equation*. In other words, the matrix P_y used to set the initial condition on the costate vector is also a solution to the Riccati equation (3.13). With this initialization, the costate relation $\mu_t = P_y y_t$ holds for all $t \geq 0$. Finally, it follows from (3.10) that this state-costate solution is implemented by the control law $v_t = -F_y y_t$.

The remainder of this section is organized as follows. In the first subsection, we initially consider the case in which the matrix A_{yy} is nonsingular. While this case is studied for pedagogical simplicity, it is also of interest in its own right. In the second subsection, we then treat the more general case in which A_{yy} can be singular. As emphasized by Pappas, Laub and Sandell (1980), singularity in A_{yy} occurs naturally in dynamic systems with delays. One of our example economies used in our numerical experiments has a singular matrix A_{yy}. Finally, in the third subsection we study the continuous-time counterpart to the *deterministic regulator problem.* We describe an alternative solution method and show how to convert a discrete-time regulator problem into a continuous-time regulator with the same relation between optimally chosen state and co-state vectors. We defer the discussion of the numerical algorithms used for implementing these methods until the next section.

3.1. Nonsingular A_{yy}

When the matrix A_{yy} is nonsingular, we can solve (3.4) for $\begin{bmatrix} y_{t+1} \\ \mu_{t+1} \end{bmatrix}$:

$$\begin{bmatrix} y_{t+1} \\ \mu_{t+1} \end{bmatrix} = M \begin{bmatrix} y_t \\ \mu_t \end{bmatrix}, \tag{3.14}$$

where

$$M \equiv L^{-1}N = \begin{bmatrix} A_{yy} + B_y R^{-1} B_y' {A_{yy}'}^{-1} Q_{yy} & -B_y R^{-1} B_y' {A_{yy}'}^{-1} \\ -{A_{yy}'}^{-1} Q_{yy} & {A_{yy}'}^{-1} \end{bmatrix}. \tag{3.15}$$

We find the matrix P_y by locating the stable *invariant subspace* of the matrix M.

Definition: An *invariant* subspace of a matrix M is a linear space $\mathcal{C}$ of possibly complex vectors for which $M\mathcal{C} = \mathcal{C}$.

Invariant subspaces are constructed by taking linear combinations of eigenvectors of M. A *stable invariant subspace* is one for which the corresponding eigenvalues have absolute values less than one. To solve the model, we aim to find the matrix P_y such that $\begin{bmatrix} I \\ P_y \end{bmatrix} y$ is in the stable invariant subspace of M for every n dimensional vector y. We now elaborate on how to compute this subspace.

The matrix M has a particular structure that we can exploit in characterizing its eigenvalues. To represent this structure, we introduce a matrix J given by

$$J \equiv \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix}.$$

Notice that $J^{-1} = J' = -J$.

Definition: A matrix M is *symplectic* if $MJM' = J$.

It is straightforward to verify that M given by (3.15) is symplectic. It follows that

$$M' = J^{-1} M^{-1} J. \tag{3.16}$$

Therefore, the transpose of M is *similar* to its inverse. Recall that *similar* matrices define the same linear transformation but with respect to a different coordinate system. Thus M' and M^{-1} share the same eigenvalues. For any matrix M, the eigenvalues of M^{-1} are the reciprocals of the eigenvalues of M, so it follows that the eigenvalues of a real symplectic matrix come in reciprocal pairs, and the number of stable eigenvalues cannot exceed the number of states n. However, merely requiring M to be symplectic permits there to be eigenvalues with absolute values *equal* to one, and so we will need an additional argument to show that there are exactly n stable eigenvalues.

To locate the stable invariant subspace of the symplectic matrix M, we follow Laub (1979) and (block) triangularize M:

$$V^{-1} M V = W$$

$$W = \begin{bmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{bmatrix}, \tag{3.17}$$

where V is a nonsingular matrix. By construction, the matrices M and W are similar. The matrix partitions in (3.17) are built to coincide with the number of stable and unstable eigenvalues. In particular, the absolute values of the eigenvalues of W_{11} are stable.

A special case of this decomposition is an appropriately ordered Jordan decomposition of M as was used by Vaughan (1970) in developing an invariant subspace algorithm for computing P_y. Laub (1991) traces this solution strategy back to the 19th century and credits MacFarlane (1963) and Potter (1966) with introducing it to the control literature. As emphasized by Laub (1991), it is preferable to build algorithms based on other upper triangular decompositions that are more numerically stable. The Jordan decomposition is particularly problematic when the symplectic matrix M has eigenvalues with multiplicities greater than one (see also Golub and Wilkinson 1976). In the next section, we describe alternative Schur decompositions, which are more reliable numerically.

To use this triangularization to calculate P_y, apply V^{-1} to both sides of the state equation (3.14):

$$y^*_{t+1} = W y^*_t,$$

where

$$y^*_t = V^{-1} \begin{bmatrix} y_t \\ \mu_t \end{bmatrix}.$$

This transformation permits us to study asymptotic properties in terms of two smaller uncoupled subsystems. Partition y^*_t into two blocks with dimensions given by the number of stable and unstable eigenvalues:

$$y^*_t \equiv \begin{bmatrix} y^*_{1,t} \\ y^*_{2,t} \end{bmatrix}.$$

Then

$$y^*_{2,t+1} = W_{22} y^*_{2,t},$$

and the solution sequence $\{y^*_{2,t}\}$ fails to converge to zero unless it is initialized at zero. Setting $y^*_{2,0}$ at zero can be accomplished by an appropriate initialization of the costate vector, as we now verify.

Partition the matrices V and V^{-1} as

$$V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix}, V^{-1} = \begin{bmatrix} V^{11} & V^{12} \\ V^{21} & V^{22} \end{bmatrix}.$$

Since V is nonsingular and there exists a (stable) solution to the optimal control problem, we must have

$$V^{21} y_t + V^{22} \mu_t = 0. \tag{3.18}$$

The rank of the matrix $[\,V^{21} \quad V^{22}\,]$ equals the number of unstable eigenvalues of M, and thus the rank of its null space must equal the number of stable eigenvalues. For a solution to exist for every initialization $y_0 = y$, it follows from (3.18) that there must exist a μ such that

$$V^{21} y + V^{22} \mu = 0.$$

Thus the dimensionality of the null space of $[\,V^{21} \quad V^{22}\,]$ must also be at least n. Therefore, M has exactly n stable eigenvalues, and the matrix partition V^{22} is nonsingular. Solving (3.18) for μ_t gives

$$\mu_t = -(V^{22})^{-1} V^{21} y_t.$$

Consequently, the matrix P_y used to initialize the costate vector is given by

$$P_y = -(V^{22})^{-1} V^{21} = V_{21} {V_{11}}^{-1}, \tag{3.19}$$

where the second equality follows from the fact that the rank of $\begin{bmatrix} V_{11} \\ V_{21} \end{bmatrix}$ has rank n, and

$$[\,V^{21} \quad V^{22}\,] \begin{bmatrix} V_{11} \\ V_{21} \end{bmatrix} = 0.$$

3.2. Singular A_{yy}

We now extend the solution method to accommodate singularity in A_{yy}. This method avoids inverting the L matrix in (3.4). Instead of locating the stable invariant subspace of M, a deflating subspace method finds the stable deflating subspace of the pencil $\lambda L - N$.

Definition: A *pencil* $\lambda L - N$ is the family of matrices $\{\lambda L - N\}$ indexed by the complex variable λ.

Definition: A *deflating subspace* of the pencil $\lambda L - N$ is the subspace $\mathcal{C}$ of complex vectors such that the dimension of $\mathcal{C}$ is at least as large as the dimension of the sum of the subspaces $L\mathcal{C}$ and $N\mathcal{C}$.

For the matrices L and N of equation (3.4), it can be verified that the intersection of their null spaces contains only the zero vector.[5] This ensures us that a generalized eigenvalue problem is well posed. When a subspace $\mathcal{C}$ is deflating, there exists a vector x in $\mathcal{C}$ that solves the generalized eigenvalue problem

$$\lambda L y = N y$$

(see Theorem 2.1 in Stewart 1972). Implicitly, we are including the possibility of a solution with $\lambda = \infty$, which occurs when y is in the null space of L but not in the null space of N. As with the previous (invariant subspace) method, the deflating subspace of interest for solving the optimal control problem is the deflating subspace associated with the stable state-costate sequence. The stable deflating subspace is the subspace associated with the stable generalized eigenvectors (the eigenvectors associated with generalized eigenvalues with absolute values strictly less than one.) Hence we solve the model by finding a matrix P_y such that $\begin{bmatrix} I \\ P_y \end{bmatrix} y$ is in the stable deflating subspace of the pencil $\lambda L - N$.

Recall that when A_{yy} is nonsingular, the matrix M is symplectic. More generally, system (3.4) is associated with a symplectic pencil

Definition: A pencil $\lambda L - N$ is *symplectic* if $LJL' = NJN'$.

Pappas, Laub and Sandell (1980, Theorem 4) show that the generalized eigenvalues of the symplectic pencil $(\lambda L - N)$ come in reciprocal pairs, just as the eigenvalues of M do when A_{yy} is nonsingular. Hence we again have that the number of stable generalized eigenvalues is no greater than n. Furthermore, we can imitate our argument in the case in which A_{yy} is nonsingular to show that there are exactly n stable generalized eigenvalues.[6]

We triangularize the state-costate system (3.4) using the solutions to the generalized eigenvalue problem. As in Theorem 2.1 of Stewart (1972), there exists a decomposition of the pencil $\lambda L - N$ such that

$$ULV = T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}, UNV = W = \begin{bmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{bmatrix}, \tag{3.20}$$

[5] See Theorem 3 of Pappas, Laub and Sandell (1980) for the case in which (A_{yy}, D_y) is detectable. As we noted previously, the restriction to a detectable system rules out some interesting economic models. More generally, nonexistence of a common nonzero vector in the null spaces of N and L can be shown by way of contradiction. Suppose there is a common nonzero vector in the null space. Then the matrix $(I + Q_{yy} B_y R^{-1} B_y')$ is singular. However, this singularity contradicts Theorem 1 of Kimura (1988).

[6] Theorems 3 and 4 of Pappas, Laub and Sandell (1980) establish this result when the pair (A_{yy}, D_y) is detectable.

where U and V are unitary matrices and the matrix partitions have the same number, n, of elements as the number of entries in the state vector y_t. Premultiplication of the pencil $\lambda L - N$ by the nonsingular matrix U preserves the solutions to the generalized eigenvalue problem, and postmultiplication by V alters the generalized eigenvectors but not the eigenvalues. A consequence of the triangularization is that the solutions to the generalized eigenvalue problem for the original system are constructed directly from the solutions to the following two smaller problems:

$$\begin{aligned} \lambda T_{11}\tilde{y} &= W_{11}\tilde{y} \\ \lambda T_{22}\tilde{y} &= W_{22}\tilde{y}. \end{aligned} \tag{3.21}$$

As with the invariant subspace method, we build the blocks of the triangularization so that the generalized eigenvalues of the first problem in (3.21) satisfy $|\lambda| < 1$, and for the second problem $|\lambda| > 1$. As a consequence, the span of the first n columns of V gives the vectors of the deflating subspace we seek. The span of the remaining n columns contains the problematic initializations of the state-costate vector for which the implied sequence of state-costate vectors diverges exponentially. In addition, it includes the span of the generalized eigenvectors associated with infinite eigenvalues. Imitating the solution method when A_{yy} is nonsingular, we initialize the costate vector as $\mu_t = P_y y_t$, where the matrix P_y is again given by (3.19).

To understand better the nature of this unstable subspace, recall that an eigenvector associated with an infinite eigenvalue is in the null space of T_{22}. Suppose the triangularization of L and N is built so that we can further partition the matrices:

$$T_{22} = \begin{bmatrix} M_{11} & M_{12} \\ 0 & 0 \end{bmatrix}$$

$$W_{22} = \begin{bmatrix} O_{11} & O_{12} \\ 0 & O_{22} \end{bmatrix},$$

where the matrices M_{11} and O_{22} are nonsingular. Such a triangularization always exists. Consider solving the following equation recursively for a sequence $\{\tilde{y}_{t+1}\}$; for each t solve for $\tilde{y}_{t+1}$ given $\tilde{y}_t$ by using

$$T_{22}\tilde{y}_{t+1} = W_{22}\tilde{y}_t.$$

For this equation to have a solution, the second component of $\tilde{y}_t$ must be zero for all t because

$$O_{22}\tilde{y}_{t,2} = 0, \tag{3.22}$$

and O_{22} is nonsingular. In addition to eliminating the nonexistence problem, imposing this restriction also resolves the multiplicity problem. Note that the multiplicity problem for the triangular system is that for a given t, (3.22) does not restrict $\tilde{x}_{t+1,2}$. However, (3.22) applied to time $t+1$ resolves the problem.

3.3. Continuous-time Systems

To conclude this section, we consider solving continuous-time Hamiltonian systems of the form (3.5). The defining feature of a *Hamiltonian* matrix is:

Definition: A matrix H is *Hamiltonian* if JH is symmetric.

The matrix H in (3.5), (3.6) clearly satisfies this property. It follows that

$$H' = -JHJ^{-1},$$

which in turn implies that the matrix H' is similar to $-H$. Consequently, the eigenvalues of a real Hamiltonian matrix come in pairs that are symmetric about the imaginary axis of the complex plane. The *stable* eigenvalues of a Hamiltonian matrix are those whose real parts are strictly negative. Similar arguments to those given above guarantee that there are exactly n stable eigenvalues of H. Therefore, (3.5) can be solved by using an invariant subspace method and its associated decomposition (3.17), provided that the classification of stable and unstable eigenvalues is modified appropriately.[7]

There is an alternative approach for solving a continuous-time Hamiltonian system. Given a Hamiltonian matrix H, another Hamiltonian matrix G is constructed with the same stable and unstable invariant subspaces. The matrix G is called the "sign" of the matrix H, and is defined as follows. Take the Jordan decomposition of H:

$$H = V \begin{bmatrix} \Lambda_{11} & 0 \\ 0 & \Lambda_{22} \end{bmatrix} V^{-1},$$

where Λ_{11} is an upper triangular matrix with the eigenvalues of H that have strictly negative real parts on the diagonals, and Λ_{22} is an upper triangular matrix with the eigenvalues of H that have strictly positive real parts on the diagonals. Then

$$G = \text{sign}(H) \equiv V \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix} V^{-1}.$$

Thus the sign of a matrix is a new matrix with the same eigenvectors as the original matrix and with eigenvalues replaced by -1 or 1 depending on the signs of the real parts of the original eigenvalues.

The matrix P_y can be inferred directly from G. To see this, we use an insight from Roberts (1980). By construction, all of the stable eigenvalues of G are equal to -1. Consequently, the matrix P_y satisfies the following eigenvalue problem:

$$G \begin{bmatrix} I \\ P_y \end{bmatrix} y = - \begin{bmatrix} I \\ P_y \end{bmatrix} y$$

for any n dimensional vector y, and the matrix P_y solves the affine equation

$$G \begin{bmatrix} I \\ P_y \end{bmatrix} + \begin{bmatrix} I \\ P_y \end{bmatrix} = 0. \tag{3.23}$$

[7] Deflating subspace methods are not needed for solving the class of continuous-time quadratic control problems considered here because we can form directly the Hamiltonian matrix and apply an invariant subspace method. However, as we have formulated it, the continuous-time problem does not permit systems with finite gestation lags in making investment goods productive or systems for which consumption services depend on only a finite interval of past consumptions.

This method is implemented by finding fast ways to compute the "sign" of a matrix.

While the matrix sign method is directly applicable for solving continuous-time Hamiltonian systems, Hitz and Anderson (1972) and Gardiner and Laub (1986) show how to use it to locate deflating subspaces of discrete-time systems. Consider the generalized eigenvalue problem for the symplectic pencil

$$\lambda L y = N.$$

Then

$$(1+\lambda)(L-N)y = (1-\lambda)(L+N)y.$$

Since the only common vector in the null space of L and N is zero, we construct the solution to the eigenvalue problem

$$\delta y = (L-N)^{-1}(L+N),$$

where

$$\delta = \frac{1+\lambda}{1-\lambda}.$$

Consequently, the stability relations (2.1) carry over here as well, and we apply the matrix sign algorithm to $(L-N)^{-1}(L+N)$.

It also turns out that $(L-N)^{-1}(L+N)$ is a Hamiltonian matrix, which we can exploit in computation. To verify the Hamiltonian structure, note that

$$\begin{aligned}
(L-N)J(L'+N') &= LJL' - NJN' - NJL' + LJN' \\
&= -NJL' + LJN' \\
&= NJN' - LJL' - NJL' + LJN'\,{}' \\
&= -(L+N)J(L'-N')
\end{aligned}$$

where we have used the fact that $\lambda L - N$ is a symplectic pencil. Therefore,

$$\begin{aligned}
J(L-N)^{-1}(L+N) &= (L'+N')(L'+N')^{-1}J(L-N)^{-1}(L+N) \\
&= (L'+N')[-(L-N)J(L'+N')]^{-1}(L+N) \\
&= (L'+N')[(L+N)J(L'-N')]^{-1}(L+N) \\
&= (L'+N')(L'-N')^{-1}J',
\end{aligned}$$

which proves that $(L-N)^{-1}(L+N)$ is a Hamiltonian matrix.

In summary, by construction, the stable (unstable) invariant subspace of the Hamiltonian matrix $(L-N)^{-1}(L+N)$ coincides with the stable (unstable) deflating subspace of the symplectic pencil $\lambda L - N$. This coincidence permits us to compute the matrix P_y used for initializing the costate vector for the discrete-time system (3.4) by applying a matrix sign algorithm to $(L-N)^{-1}(L+N)$.

4. Computational Techniques for Solving Riccati Equations

We consider three types of algorithms for computing P_y:

(i) Schur algorithm;
(ii) doubling algorithm;
(iii) matrix sign algorithm.

A Schur algorithm is based on locating a stable subspace using a *Schur decomposition* of the state-costate system. As we noted in the previous section, once a stable subspace is located, the relevant Riccati equation solution P_y is easily computed. There are two versions of a Schur decomposition, depending on whether the matrix A_{yy} is known to be nonsingular or not. A Schur decomposition gives a more reliable way of locating stable spaces than the familiar Jordan decomposition and its generalization for pencils.

A doubling algorithm is an iterative method for speeding up the dynamic programming Riccati equation iteration by *doubling* the number of time periods in each iteration. Recall from our discussion in the previous section that the stable deflating subspace of the pencil $\{\lambda L - N\}$ coincides with the invariant subspace of the sign of the matrix $(L-N)^{-1}(L+N)$ associated with the eigenvalue -1. A matrix sign algorithm is an iterative method for computing the sign of $(L-N)^{-1}(L+N)$ from which we can recover P_y easily.

4.1. Schur Algorithm

Suppose the matrix A_{yy} is nonsingular. As we noted in section 3, the matrix P_y can be found by locating the stable invariant subspace of the matrix M given in (3.15). In some of our numerical calculations, we use what is referred to as a *real* Schur decomposition of M to locate its invariant subspace.

Definition: The *real Schur decomposition* of a real matrix M is an orthogonal matrix $\hat{V}$ and a real upper block triangular matrix $\hat{W}$ such that

$$\hat{V}'M\hat{V} = \hat{W} = \begin{bmatrix} \hat{W}_{11} & \hat{W}_{12} & \dots & \hat{W}_{1m} \\ 0 & \hat{W}_{22} & \dots & \hat{W}_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \hat{W}_{mm} \end{bmatrix}$$

where $\hat{W}_{ii}$ is either a scalar or a 2×2 matrix with complex conjugate eigenvalues.[8]

A real Schur decomposition is a computationally convenient version of the block triangular decomposition (3.17) used to compute P_y when A_{yy} is nonsingular. Golub and Van Loan(1989) describe how to compute the real Schur decomposition (in particular, see sections 7.4 and 7.5). Recall that the block triangular matrix W in (3.17) results from partitioning the eigenvalues into stable and unstable eigenvalues. Algorithms that compute the real Schur decomposition of a matrix typically do not partition the diagonal blocks of $\hat{W}$ according to stability. Instead, given an arbitrary real Schur decomposition $M = \hat{V}\hat{W}\hat{V}'$, one can use the approaches described in either Bai and Demmel (1993) or Stewart (1976) to construct a sequence of orthogonal transformations that reorder the diagonal blocks of $\hat{W}$, while updating $\hat{V}$ so that $M = \hat{V}\hat{W}\hat{V}'$ holds at every step.

In summary, the steps for implementing a Schur algorithm are

[8] There is also a *complex Schur decomposition* of a real or complex matrix in which $\hat{V}$ is a unitary matrix and $\hat{W}$ is upper triangular.

(1) form the matrix M in (3.15);
(2) form a real Schur decomposition of M where the first n columns of $\hat{V}$, written in a partitioned form as $\left[\hat{V_{11}}' \quad \hat{V_{21}}'\right]'$, are a basis for the stable invariant subspace of M;
(3) solve $P_y \hat{V}_{11} = \hat{V}_{21}$ for P_y.

For the numerical computations which follow, we compute the real Schur decomposition of M using the LAPACK[9] function DGEES. For comparisons, we also compute an eigenvector decomposition using the built-in MATLAB function EIG. Our eigenvector routine assumes that the eigenvalues of M are distinct, and we do not attempt to implement an algorithm designed for the more troublesome case in which there are repeated eigenvalues. We compute P_y in step (3) using the built-in MATLAB operator '/', which solves a linear equation using Gaussian elimination with partial pivoting.

A deflating subspace method is required when A_{yy} is singular and likely to be more stable numerically when A_{yy} is nearly singular. To implement this approach in practice, we use an ordered real generalized Schur decomposition to find an appropriate triangularization of the state-costate dynamical system (see Van Dooren, 1982).

Definition: A *generalized real Schur decomposition* of a real matrix pencil $\lambda L - N$ is a pair of orthogonal matrices $\hat{U}$ and $\hat{V}$, a real upper triangular matrix $\hat{T}$, and a real upper block triangular matrix $\hat{W}$, such that

$$\hat{U} L \hat{V} = \hat{T} = \begin{bmatrix} \hat{T}_{11} & \hat{T}_{12} & \dots & \hat{T}_{1m} \\ 0 & \hat{T}_{22} & \dots & \hat{T}_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \hat{T}_{mm} \end{bmatrix}$$

$$\hat{U} N \hat{V} = \hat{W} = \begin{bmatrix} \hat{W}_{11} & \hat{W}_{12} & \dots & \hat{W}_{1m} \\ 0 & \hat{W}_{22} & \dots & \hat{W}_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \hat{W}_{mm} \end{bmatrix},$$

where the pencil $\lambda \hat{T}_{ii} - \hat{W}_{ii}$ is either a 1×1 matrix pencil or a 2×2 matrix pencil with complex conjugate generalized eigenvalues.

As with the real Schur decomposition, we initially compute a generalized real Schur decomposition of $\lambda L - N$ without regard to whether the generalized eigenvalues are stable or not. We then reorder the diagonal blocks of $\hat{T}$ and $\hat{W}$ so that the generalized eigenvalues are partitioned in the manner required by (3.20). This partitioning can be done using the algorithms described in Van Dooren (1981,1982) or in Kågström and Poromaa (1994).

Thus the steps for implementing a generalized Schur algorithm are

(1) form the matrices L and N in (3.4);
(2) form a generalized real Schur decomposition of the pencil $\lambda L - N$ where the first n columns of $\hat{V}$, written in a partitioned form as $\left[\hat{V_{11}}' \quad \hat{V_{21}}'\right]'$, span the deflating subspace of the pencil $\lambda L - N$;
(3) solve $P_y \hat{V}_{11} = \hat{V}_{21}$ for P_y.

[9] The algorithms described in this paper use routines from the FORTRAN packages LAPACK, LINPACK and RICPACK. All of these packages can be obtained by anonymous ftp from netlib.att.com and various mirrors. MATLAB is a commercial matrix algebra package available from The MathWorks, Inc. All of our FORTRAN routines are implemented as MATLAB MEX-files.

For the numerical comparisons which follow, we implement the generalized Schur algorithm by using the routines QZHESW, QZITW, QZIT, and ORDER from RICPACK. We also report results for a method that uses generalized eigenvectors to compute deflating subspaces. This method takes the first n columns of the matrix $\hat{V}$ to be the generalized eigenvectors of $\lambda L - N$ that correspond to stable generalized eigenvalues. We implement this method using the built-in MATLAB function EIG, making no attempt to handle repeated generalized eigenvalues.

4.2. Doubling Algorithm

Dynamic programming solves the infinite horizon problem by backward induction, which leads to iterations on the Riccati equation (3.13). A doubling algorithm can be viewed as a refinement of this approach. It preserves the idea of approximating the solution to the infinite horizon problem by a sequence of finite horizon problems, but instead of increasing the horizon by one time period in each iteration, the number of time periods gets *doubled*.

To see how this approach works, recall that the solution to the finite horizon problem for periods $0 \ldots (\tau - 1)$ can be viewed as a two point boundary value problem where the initial state vector y_0 is set to some arbitrary vector y and the costate vector at the terminal date μ_τ is set to zero. Suppose for simplicity that A_{yy} is nonsingular. By iterating on relation (3.14), we find that

$$\hat{M} \begin{bmatrix} y_\tau \\ 0 \end{bmatrix} = \begin{bmatrix} y_0 \\ \mu_0 \end{bmatrix}, \tag{4.1}$$

where

$$\hat{M} \equiv M^{-\tau}.$$

To approximate the matrix P_y, we solve (4.1) for the initial costate vector μ_0 as a function of y_0. Partitioning $\hat{M}$ conformably to the state-costate partition, we see that

$$\hat{M}_{11} y_\tau = y_0, \qquad \hat{M}_{21} y_\tau = \mu_0.$$

Therefore, the implicit initialization of the costate vector is

$$\mu_0 = \hat{M}_{21} (\hat{M}_{11})^{-1} y_0,$$

and our approximation for the matrix P_y is given by $\hat{M}_{21}(\hat{M}_{11})^{-1}$.

What is needed to implement this approach is a way to compute $\hat{M}$ when the horizon τ is large. Expanding the horizon one period at a time corresponds to multiplying the matrix M^{-1}, τ times in succession. However, when τ is chosen to be a power of two, computations can be sped up by using

$$M^{-2^{k+1}} = (M^{-2^k}) M^{-2^k}. \tag{4.2}$$

As a consequence, when $\tau = 2^j$, the desired matrix can be computed in j iterations instead of 2^j iterations, which explains the name *doubling algorithm*.

Given that the matrix M^{-1} has unstable eigenvalues, direct iterations on (4.2) can be very unreliable. Clearly, the sequence of matrices $\{M^{-2^k}\}$ diverges. One of the features of a doubling algorithm is to transform these computations into matrix iterations that converge. Another feature is that a doubling algorithm exploits the fact that the matrix M is symplectic. Symplectic matrices have several nice properties.[10] We have already seen that their

[10] There is a variation of the Schur algorithm that exploits the symplectic structure of M. See pages 431-434 of Petkov *et al.* (1991) for an overview of this algorithm.

eigenvalues come in reciprocal pairs. In addition, the product of symplectic matrices is symplectic, and the inverse of a symplectic matrix is symplectic. Moreover, for any symplectic matrix S, the matrices $S_{21}(S_{11})^{-1}$ and $(S_{11})^{-1}S_{12}$ are both symmetric and

$$\begin{aligned} S_{22} &= (S'_{11})^{-1} + S_{21}(S_{11})^{-1}S_{12} \\ &= (S'_{11})^{-1} + S_{21}(S_{11})^{-1}S_{11}(S_{11})^{-1}S_{12}. \end{aligned}$$

Therefore, a $(2n \times 2n)$ symplectic matrix can be represented in terms of the three $n \times n$ matrices $\alpha = (S_{11})^{-1}, \beta = (S_{11})^{-1}S_{12}, \gamma = S_{21}(S_{11})^{-1}$, the latter two of which are symmetric.

The doubling algorithm described by Anderson (1978) and Anderson and Moore (1979) exploits such a representation by using the following parameterization of M^{-2^k}:

$$M^{-2^k} = \begin{bmatrix} (\alpha_k)^{-1} & (\alpha_k)^{-1}\beta_k \\ \gamma_k(\alpha_k)^{-1} & \alpha'_k + \gamma_k(\alpha_k)^{-1}\beta_k \end{bmatrix},$$

where the $n \times n$ matrices $\alpha_k, \beta_k, \gamma_k$ are given by the recursions

$$\begin{aligned} \alpha_{k+1} &= \alpha_k(I + \beta_k\gamma_k)^{-1}\alpha_k \\ \beta_{k+1} &= \beta_k + \alpha_k(I + \beta_k\gamma_k)^{-1}\beta_k\alpha'_k \\ \gamma_{k+1} &= \gamma_k + \alpha'_k\gamma_k(I + \beta_k\gamma_k)^{-1}\alpha_k. \end{aligned} \tag{4.3}$$

While this alternative parameterization introduces a matrix inverse into the recursions (4.3) that is absent in (4.2), the matrix $I + \beta_k\gamma_k$ being inverted is only n dimensional. The nonsingularity of this matrix for all k is established in Kimura (1988). To initialize the doubling algorithm, we simply deduce the implicit parameterization of M^{-1} given in partitioned form by

$$M^{-1} = N^{-1}L = \begin{bmatrix} {A_{yy}}^{-1} & {A_{yy}}^{-1}B_yR^{-1}{B_y}' \\ Q_{yy}{A_{yy}}^{-1} & Q_{yy}{A_{yy}}^{-1}B_yR^{-1}{B_y}' + {A_{yy}}' \end{bmatrix}, \tag{4.4}$$

which leads to the initializations

$$\alpha_0 = A_{yy}, \quad \beta_0 = B_yR^{-1}{B_y}', \quad \gamma_0 = Q_{yy}.$$

While our derivation took the matrix A_{yy} to be nonsingular, Anderson (1978) argues that the doubling algorithm is more generally applicable.

A convenient feature of this parameterization is that there are known conditions under which the matrix sequences $\{\alpha_k\}, \{\beta_k\}, \{\gamma_k\}$ converge. When the pair (A_{yy}, D_y) is detectable, then the sequence $\{\gamma_k\}$ is nondecreasing and converges to the matrix P_y. (Here we are adopting the usual partial ordering for positive semidefinite matrices.) As noted by Kimura (1988, Theorem 5), under the same restrictions, the sequence $\{\beta_k\}$ is nondecreasing and converges to a positive semidefinite matrix P_y^* associated with a "dual" to the *deterministic regulator problem.*

The convergence of the $\{\alpha_k\}$ sequence is more problematic. Unfortunately, without simultaneous convergence of $\{\alpha_k\}$, it is not evident that iterations of the form given in (4.3) can be used as the basis of a numerical algorithm. If this latter sequence diverges, small numerical errors may get magnified, causing the resulting algorithm to be poorly behaved. Kimura (1988) provides some sufficient conditions for $\{\alpha_k\}$ to converge to a matrix of zeros. His sufficient conditions are used to guarantee that either P_y or P_y^* is nonsingular.

As we noted previously, a sufficient condition for P_y to be nonsingular is that the pair (A_{yy}, D_y) be observable. Sufficient conditions for the nonsingularity of the matrix P_y^* are that (i) (A_{yy}, B_y) is controllable; and (ii) (A_{yy}, D_y) is detectable (Kimura 1988). Recall that controllability is often achieved by our *a priori* partitioning of the state vector into *endogenous* and *exogenous* components. Thus for our purposes, the restrictions guaranteeing the nonsingularity of P_y^* may be of particular interest. Even so, detectability is too strong for some of our applications.

To apply a doubling algorithm more generally, we sometimes modify the control problem by adding small quadratic penalties to linear combinations of the states and controls. As long as these penalties are sufficient to guarantee that either P_y or P_y^* is nonsingular, we are assured of convergence of all three sequences. Of course, there is a danger that the penalty distorts the solution to the original control problem in a nontrivial way, which must be checked in practice.

4.2.1. Initialization From a Positive Definite Matrix

Instead of adding small quadratic penalties objective function for each calendar date, we could add a terminal penalty to the finite horizon approximation to the control problem. From Chan, Goodwin and Sin (1984), it is known that iterations on the Riccati difference equation converge to the unique stabilizing solution whenever the Riccati equation is initialized at a positive definite matrix.[11] Initializing the Riccati difference equation at a positive definite matrix is equivalent to imposing a terminal penalty that is a negative definite quadratic form in the state vector. We will now show how to initialize the doubling algorithm to impose a terminal penalty. This will permit us to compute P_y via a doubling algorithm for a richer class of control problems.

Consider first a finite time horizon problem with a quadratic penalty on the terminal state. We select this penalty so that the terminal multiplier $\mu_\tau = P_o y_\tau$ for some positive definite matrix P_o. Then equation (4.1) is altered to be

$$\hat{M}\begin{bmatrix} I \\ P_o \end{bmatrix} y_\tau = \begin{bmatrix} y_0 \\ \mu_0 \end{bmatrix}. \tag{4.5}$$

Build a matrix K

$$K \equiv \begin{bmatrix} I & 0 \\ P_o & I \end{bmatrix}.$$

Then equation (4.5) can be rewritten as

$$K^{-1}\hat{M}KK^{-1}\begin{bmatrix} I \\ P_o \end{bmatrix} y_\tau = K^{-1}\begin{bmatrix} y_0 \\ \mu_0 \end{bmatrix}.$$

Equivalently,

$$M^*\begin{bmatrix} y_\tau \\ 0 \end{bmatrix} = \begin{bmatrix} y_0 \\ \mu_0 - P_o y_0 \end{bmatrix},$$

where

$$M^* = K^{-1}\hat{M}K.$$

Partitioning M^* consistently with the state-costate vector, the implicit initialization of the costate vector is now

$$\mu_0 = P_o y_0 + M_{12}^*(M_{11}^*)^{-1} y_0,$$

and our approximation for P_y is given by $M_{12}^*(M_{11}^*)^{-1} + P_o$.

We are now left with computing the matrix M^* when the horizon τ is very large. Notice that

$$M^* = (K^{-1}MK)^{-\tau}.$$

[11] Here we are using the fact that the pair (A_{yy}, B_y) is stabilizable and that there exists a solution to the *deterministic regulator problem* when constraint (2.1) is imposed. The result follows from (i) and (iii) of Theorem 3.1 and Theorem 4.2 of Chan, Goodwin and Sin (1984).

It is straightforward to verify that because M is symplectic, so is $K^{-1}MK$. This means that doubling algorithm (4.3) is applicable for computing $(K^{-1}MK)^{-2^k}$; however, the initializations must be altered. The new initializations can be deduced by looking at the implicit parameterization of the symplectic matrix $K^{-1}M^{-1}K$, and they are given by

$$\begin{aligned}\alpha_0 &= (I + B_y R^{-1} B_y{}' P_o)^{-1} A_{yy} \\ \beta_0 &= (I + B_y R^{-1} B_y{}' P_o)^{-1} B_y R^{-1} B_y{}' \\ \gamma_0 &= Q_{yy} - P_o + A_{yy}{}' P_o (I + B_y R^{-1} B_y{}' P_o)^{-1} A_{yy}.\end{aligned} \tag{4.6}$$

Not suprisingly, the original initializations coincide with setting P_o to zero in (4.6).

There are two related advantages to these initializations over the previous ones. First, the sequence $\{\gamma_j\}$ converges to $P_y - P_o$ whenever P_o is positive definite. This follows from the Riccati difference equation convergence described previously and does not require that (A_{yy}, D_y) be detectable. Second, the sequence $\{\beta_j\}$ converges and satisfies the bounds

$$0 \leq \beta_j \leq (P_o)^{-1}$$

even when (A_{yy}, D_y) is not detectable.[12] Although we do not have a complete characterization of convergence of the resulting algorithm, all three matrix sequences (including $\{\alpha_j\}$) are guaranteed to converge with these alternative initializations if they converge with the original ones.

In summary, the steps for implementing the doubling algorithm are

(1) initialize α_0, β_0, and γ_0 according to (4.6);
(2) iterate in accordance with (4.3);
(3) form P_y as the limit of $\{\gamma_k\} + P_o$.

We implement the doubling algorithm in FORTRAN, exploiting the fact that β_k and γ_k are symmetric matrices for all k.[13] We use two different settings for P_o. To obtain the

[12] The convergence and bound can be established as follows. Let $\{\beta_j^*\}$ denote the sequence starting from the original initialization. Then it is straightforward to show that

$$\beta_j = (I + \beta_j^* P_o)^{-1} \beta_j^*.$$

Exploiting the nonsingularity of P_o, the following equivalent formula can be deduced:

$$\beta_j = (P_o)^{-1} - (P_o + P_o \beta_j^* P_o)^{-1}.$$

The reported bound follows immediately. The sequence $\{\beta_j^*\}$ is monotone increasing because it is a subsequence of Riccati difference equation iterations for a dual problem initialized at zero. Therefore, the sequence $\{\beta_j\}$ is also monotone increasing. Given the upper bound $(P_o)^{-1}$, this latter sequence must converge.

[13] We iterate on (4.3) until

$$\|\gamma_k - \gamma_{k-1}\|_1 \leq \epsilon \, \|\gamma_k\|_1 ,$$

where we set $\epsilon = 1 \times 10^{-15}$ on a computer with a machine precision of $2^{-52} \approx 2.2204 \times 10^{-16}$. Here $\|X\|_1$ denotes the matrix 1-norm of a matrix X:

$$\|X\|_1 = \max_j \sum_i |X_{ij}| .$$

original doubling algorithm, we set P_o to zero; and to investigate the potential advantages of including a terminal penalty, we set P_o to an identity matrix.

4.2.2. Application to Continuous Time

As noted by Anderson (1978) and Kimura (1989), a doubling algorithm for a discrete-time symplectic system can be used to solve a continuous-time Hamiltonian system. Recall that in our discussion of solving control problems via a matrix sign algorithm, we showed how to covert a discrete-time symplectic system into a continuous-time Hamiltonian system. To apply a doubling algorithm, we want to "invert" this mapping, *e.g.,* given a Hamiltonian matrix H, we construct a symplectic pencil with the same stable deflating subspace. The symplectic pencil associated with H is given by $\lambda(I+H)-(I-H)$. By adopting a very similar argument as before, we found it easy to show that the generalized eigenvectors for the constructed pencil coincide with the eigenvectors of the original Hamiltonian matrix H. Moreover, the classification of stable and unstable (generalized) eigenvalues is preserved.

4.3. Matrix Sign Algorithm

In section 3.3 we showed how to compute P_y from the sign of the Hamiltonian matrix for a continuous-time state-costate system. To compute P_y for a symplectic pencil $\lambda L - N$, we first form the Hamiltonian matrix

$$H = (L-N)^{-1}(L+N)$$

and then compute sign(H). For this to be a viable solution method, we must be able to compute sign(H) easily.

There are alternative matrix sign algorithms. An algorithm advocated by Roberts (1980) and Denman and Beavers (1976) is to average a matrix and its inverse:

$$\begin{aligned} G_0 &= H \\ G_{k+1} &= G_k + (1/2)[(G_k)^{-1} - G_k], k = 0, 1, \ldots. \end{aligned} \tag{4.7}$$

To speed up convergence, Gardiner and Laub (1986) suggest using the recursion

$$G_0 = H, \quad G_{k+1} = (1/2\epsilon_k)(G_k + {\epsilon_k}^2 {G_k}^{-1}),$$

where

$$\epsilon_k = |\det G_k|^{1/n}. \tag{4.8}$$

Bierman (1984) and Byers (1987) propose a further refinement, which exploits the fact that the matrix G_k is a Hamiltonian matrix for each k. Recall that if H is a Hamiltonian matrix, then JH is symmetric where

$$J = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix}.$$

Hence

$$JG_{k+1} = \frac{1}{2\epsilon_k}(JG_k + {\epsilon_k}^2 JJ{G_k}^{-1}J), \tag{4.9}$$

where ϵ_k is either set to one as in the original sign algorithm or set via formula (4.8) using JG_k in place of G_k. Consequently, it suffices to compute the sequence of symmetric matrices $\{JG_k\}$ recursively via (4.9) starting from the initialization JH.[14]

[14] Kenney, Laub and Papadopoulos (1993) and Lu and Lin(1993) discuss further improvements to the matrix sign algorithm.

In summary, the steps for implementing a matrix sign algorithm are

(1) form the matrices L and N in (3.4);
(2) compute the sign of $G = (L - N)^{-1}(L + N)$;
(3) compute P_y by solving the over-determined system

$$\begin{bmatrix} G_{12} \\ G_{22} + I \end{bmatrix} P_y = - \begin{bmatrix} G_{11} + I \\ G_{21} \end{bmatrix} \tag{4.10}$$

for P_y.

For our numerical comparisons, we compute the sign of G by iterating on (4.9) until convergence with $\epsilon_k = |\det G_k|^{1/n}$.[15] To compute $(JG_k)^{-1}$ we use the symmetric inversion routines DSIFA and DSIDI from LINPACK. We solve (4.10) for P_y using least squares.

As noted in Anderson (1978), the original sign algorithm (4.7) also can be viewed as a doubling algorithm. Interpreted in this manner, it uses (at least implicitly) an alternative parameterization of the symplectic matrix M^{-1} to that used in doubling algorithm (4.3). Both recursions entail inverting a matrix. While recursion (4.9) requires that a symmetric $(2n \times 2n)$ matrix be inverted in each iteration, the doubling algorithm (4.3) requires that a nonsymmetric $n \times n$ matrix be computed at each iteration.

5. Solving the Augmented Regulator Problem

So far, we have shown how to compute the matrix F_y, which provides us with the optimal control law for the *deterministic regulator problem.* This matrix also gives us a piece of the solution to the *augmented control problem* and, hence, to the problem of interest: the *discounted stochastic regulator problem.* The missing ingredient is the matrix F_z, where the optimal control law for the *augmented regulator problem* is given by $v_t = -F_y y_t - F_z z_t$. In this section, we show that F_z can be calculated by solving a particular Sylvester equation.

We start by forming a Lagrangian modified to incorporate the exogenous state vector sequence $\{z_t\}$:

$$\mathcal{L} = -\sum_{t=0}^{\infty} [y_t' Q_{yy} y_t + 2y_t' Q_{yz} z_t + v_t' R v_t + 2\mu_{t+1}'(A_{yy} y_t + A_{yz} z_t + B_y v_t - y_{t+1})],$$

where the evolution of the forcing sequence is given by

$$z_{t+1} = A_{zz} z_t. \tag{5.1}$$

First-order necessary conditions for the maximization of $\mathcal{L}$ with respect to $\{v_t\}_{t=0}^{\infty}$ and $\{y_t\}_{t=0}^{\infty}$ are

$$v_t : \qquad R v_t + B_y' \mu_{t+1} = 0, \qquad t \geq 0 \tag{5.2}$$

$$y_t : \qquad \mu_t = Q_{yy} y_t + Q_{yz} z_t + A_{yy}' \mu_{t+1}, \qquad t \geq 0. \tag{5.3}$$

[15] More precisely, we iterate on (4.9) until

$$\|JG_k - JG_{k-1}\|_1 \leq \epsilon \|JG_k\|_1 ,$$

where $\epsilon = 1 \times 10^{-15}$.

Solve equation (5.2) for v_t; substitute it into the state equation; and stack the resulting equation along with (5.3) and (5.1) as composite system

$$L^a \begin{bmatrix} y_{t+1} \\ \mu_{t+1} \\ z_{t+1} \end{bmatrix} = N^a \begin{bmatrix} y_t \\ \mu_t \\ z_t \end{bmatrix},$$

where

$$L^a \equiv \begin{bmatrix} I & B_y R^{-1} B_y' & 0 \\ 0 & A_{yy}' & 0 \\ 0 & 0 & I \end{bmatrix}, \qquad N^a \equiv \begin{bmatrix} A_{yy} & 0 & A_{yz} \\ -Q_{yy} & I & -Q_{yz} \\ 0 & 0 & A_{zz} \end{bmatrix}. \tag{5.4}$$

As with the *deterministic regulator problem*, the relevant solution is the one that stabilizes the state-costate vector for any initialization of y_0 and z_0. Hence we seek a characterization of the multiplier μ_t of the form

$$\mu_t = P \begin{bmatrix} y_t \\ z_t \end{bmatrix},$$

such that the resulting composite sequence $[\, y_t' \quad \mu_t' \quad z_t' \,]'$ is in the stable deflating subspace of the augmented pencil $\lambda L^a - N^a$. Assuming for the moment that a solution P exists, it must be the case that $P = [\, P_y \quad P_z \,]$, where P_y is the Riccati equation solution that was characterized in section 3, and P_z is a matrix that has not yet been characterized. To see why this must be the case, note that the solution to the *augmented regulator problem* with $z_0 = 0$ coincides with the solution to the *deterministic regulator problem* . We have previously shown that P_y is a matrix, such that all vectors in the deflating subspace of the pencil $\lambda L - N$ can be represented as $[\, y' \quad y' P_y \,]'$. When the forcing sequence is initialized at zero, so it remains there for all t, it must also be the case that $[\, y' \quad y' P_y \quad 0]'$ is in the stable deflating subspace of the augmented pencil $\lambda L^a - N^a$. This justifies our previous claim that the solution to the *deterministic regulator problem* gives us a piece of the solution to the *augmented regulator problem* .

To deduce the control law associated with the matrix P, we substitute P into (5.4), which yields

$$L^a \begin{bmatrix} y_{t+1} \\ P_y y_{t+1} + P_z z_{t+1} \\ z_{t+1} \end{bmatrix} = N^a \begin{bmatrix} y_t \\ P_y y_t + P_z z_t \\ z_t \end{bmatrix}.$$

If we write the three equations in this composite system separately,

$$\begin{aligned} (I + B_y R^{-1} B_y' P_y) y_{t+1} + B_y R^{-1} B_y' P_z z_{t+1} &= A_{yy} y_t + A_{yz} z_t \\ A_{yy}' P_y y_{t+1} + A_{yy}' P_z z_{t+1} &= (P_y - Q_{yy}) y_t + (P_z - Q_{yz}) z_t \\ z_{t+1} &= A_{zz} z_t. \end{aligned} \tag{5.5}$$

Substitute the last equation into the first and solve for y_{t+1}:

$$y_{t+1} = (I + B_y R^{-1} B_y' P_y)^{-1} [A_{yy} y_t + (A_{yz} - B_y R^{-1} B_y' P_z A_{zz}) z_t].$$

It follows from relation (3.9) that this evolution equation for y_t can be rewritten as

$$y_{t+1} = (A_{yy} - B_y F_y) y_t + (A_{yz} - B_y F_z) z_t, \tag{5.6}$$

where F_y and F_z are given by

$$F_y \equiv (R + B_y' P_y B_y)^{-1} B_y' P_y A_{yy},$$

$$F_z \equiv (R + B_y' P_y B_y)^{-1} B_y' (P_y A_{yz} + P_z A_{zz}). \tag{5.7}$$

For the reasons given previously, our construction of F_y coincides with (3.11) used to represent the optimal control law for the *deterministic regulator problem.* Stability of the state vector sequence $\{y_t\}$ is guaranteed by evolution equation (5.6) because the matrix $A_{yy} - B_y F_y$ is the same matrix that appears in the state evolution equation for the *deterministic regulator problem* under the optimal control law. Since the solution to the *deterministic regulator problem* is stable by design, the eigenvalues of $A_{yy} - B_y F_y$ have absolute values that are strictly less than one. The optimal control law for the *augmented regulator problem* is given by

$$v_t = -F_y y_t - F_z z_t.$$

The matrix F_z can be computed using formula (5.7) once we know P_z. We now show that P_z is the solution to a *Sylvester equation.* Premultiply (5.6) by ${A_{yy}}' P_y$:

$$ {A_{yy}}' P_y y_{t+1} = {A_{yy}}' P_y (A_{yy} - B_y F_y) y_t + {A_{yy}}' P_y (A_{yz} - B_y F_z) z_t. \tag{5.8}$$

Using formula (5.7), we rewrite the coefficient matrix on z_t as

$$ {A_{yy}}' P_y (A_{yz} - F_z) = (A_{yy} - B_y F_y)' (P_y A_{yz} + P_z A_{zz}) - {A_{yy}}' P_z A_{zz}.$$

To obtain an alternative formula for this coefficient, substitute the last equation of (5.5) into the second equation and solve for ${A_{yy}}' P_y y_{t+1}$:

$$ {A_{yy}}' P_y y_{t+1} = (P_z - Q_{yz} - {A_{yy}}' P_z A_{zz}) z_t + (P_y - Q_{yy}) y_t. \tag{5.9}$$

Equating coefficients on z_t in (5.8) and (5.9) results in

$$ (A_{yy} - B_y F_y)' (P_y A_{yz} + P_z A_{zz}) - {A_{yy}}' P_z A_{zz} = P_z - Q_{yz} - {A_{yy}}' P_z A_{zz}.$$

Rewriting this in the form of a *Sylvester equation* (in the unknown matrix P_z), we have that

$$ P_z = Q_{yz} + (A_{yy} - B_y F_y)' P_y A_{yz} + (A_{yy} - B_y F_y)' P_z A_{zz}. \tag{5.10}$$

As we noted previously, the matrix $(A_{yy} - B_y F_y)$ has only stable eigenvalues. Also, we assumed that the matrix A_{zz} has only stable eigenvalues (Assumption 4). These restrictions are sufficient for there to exist a unique solution P_z to (5.10). Up to now, our discussion proceeded under the presumption that there exists a matrix P, such that by setting $\mu_t = P \begin{bmatrix} y_t \\ z_t \end{bmatrix}$, we stabilize the state vector sequence. We can now work backwards using the (unique) solution to the Sylvester equation to show that indeed such a matrix P does exist.

6. Computational Techniques for Solving Sylvester Equations

A Sylvester equation is represented by

$$M = W + SMT, \tag{6.1}$$

where the matrices W, S, and T are specified in advance and M is the matrix to be computed. Consistent with (5.10), the matrices S and T have stable eigenvalues.[16] There is a variety of ways to depict the solution to a Sylvester equation. One is to vectorize (6.1) as

$$[I - T' \otimes S]\text{vec}(M) = \text{vec}(W), \tag{6.2}$$

where vec$(\cdot)$ denotes stacks of the columns of a matrix argument. (To derive (6.2) from (6.1), use the identity vec$(SMT) = [T' \otimes S]$vec(M)). Hence vec(M) is the solution to a linear equation system. Alternatively, M is given by the infinite sum

$$M = \sum_{j=0}^{\infty} S^j W T^j. \tag{6.3}$$

This representation can be deduced by iterating on equation (6.1), starting from any initial matrix with the appropriate dimensions.

We consider two types of algorithms for computing M:

(i) Hessenburg-Schur algorithm;
(ii) doubling algorithm.

The Hessenberg-Schur algorithm uses a Schur decomposition of the matrix T to convert a single Sylvester equation to a collection of much smaller Sylvester equations, each of which can be vectorized as in (6.2). A Hessenberg decomposition of the matrix S is used further to simplify the calculations. The doubling algorithm is an iterative algorithm that approximates the infinite sum on the right-hand side of (6.3) by a finite sum. Similar to the doubling algorithm for solving a Riccati equation, the number of terms included in the finite sum approximation "doubles" at each iteration.

6.1. The Hessenberg-Schur Algorithm

As suggested by Bartels and Stewart (1972), one strategy for solving Sylvester equations entails block triangularizing the matrices T and/or S. We follow Golub, Nash and Van Loan (1979) by forming a Schur decomposition of the matrix T: $V'TV = \hat{T}$, where V is an orthogonal matrix and $\hat{T}$ is upper block triangular with row and column blocks that are either one or two dimensional (see section 4.1 for a formal definition). Postmultiply Sylvester equation (6.1) by V and rewrite the equation as

$$\hat{M} = \hat{W} + \hat{S}\hat{M}\hat{T}, \tag{6.4}$$

where $\hat{M} = MV$, $\hat{W} = WV$, and $\hat{S} = S$. Notice that (6.4) is in the form of a Sylvester equation in the matrix $\hat{M}$.

[16] We offer the following word of caution (or apology) to the reader. We are compelled to recycle some of the notation used in previous sections.

The block triangularity of $\hat{T}$ can now be exploited to reduce (6.4) into m smaller Sylvester equations, where m is the number of row and column blocks of $\hat{T}$. Write the matrix $\hat{T}$ in partitioned form as

$$\hat{T} = \begin{bmatrix} \hat{T}_{11} & \hat{T}_{12} & \dots & \hat{T}_{1m} \\ 0 & \hat{T}_{22} & \dots & \hat{T}_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \hat{T}_{mm} \end{bmatrix}.$$

Use the column partition of W to partition $\hat{M}$ and $\hat{W}$, and let $\hat{M}_j$ and $\hat{W}_j$ denote the corresponding j^{th} partitions. Decompose Sylvester equation (6.4):

$$\hat{M}_1 = \hat{W}_1 + \hat{S}\hat{M}_1\hat{T}_{11} \tag{6.5}$$

$$\hat{M}_j = \hat{W}_j + \hat{S}\sum_{k=1}^{j-1}\hat{M}_k\hat{T}_{kj} + \hat{S}\hat{M}_j\hat{T}_{jj}, \quad j = 2, ..., m. \tag{6.6}$$

Notice that (6.5) is a Sylvester equation in $\hat{M}_1$ and that (6.6) is a Sylvester equation in $\hat{M}_j$ as long as the matrices $\hat{M}_k$ for $k = 1, 2, ..., j-1$ have already been computed. Thus these m Sylvester equations can be solved sequentially as linear equations using vectorization (6.2).

An additional refinement advocated by Golub, Nash and Van Loan (1979) entails taking a Hessenberg decomposition of the matrix S.[17]

Definition: The *Hessenberg decomposition* of the square matrix S is an orthogonal matrix U and a matrix $\hat{S}$ that has all zeros below the first subdiagonal, such that $S = U\hat{S}U'$.

In addition to postmultiplying equation (6.1) by V, we now also premultiply this equation by U'. Equation (6.4) continues to hold with $\hat{M} = U'MV$, $\hat{W} = U'WV$, and $\hat{S} = U'SU$. This Sylvester equation can still be decomposed as in (6.5) and (6.6). With $\hat{S}$ in Hessenberg form, we can solve these latter Sylvester equations more efficiently using an equation solver designed for Hessenberg systems.[18]

In summary, the steps for implementing a Hessenberg-Schur algorithm for computing P_z are

(i) form the matrices $W = Q_{yz} + (A_{yy} - B_yF_y)'P_yA_{yz}$, $S = (A_{yy} - B_yF_y)'$, and $T = A_{zz}$;
(ii) form a Hessenberg decomposition $S = U\hat{S}U'$ and a Schur decomposition $T = V\hat{T}V'$;
(iii) compute the solution $\hat{M}$ to (6.5) and (6.6) and form $P_z = U\hat{M}V'$.

Since the Hessenberg decomposition of a matrix can be computed faster than the real Schur decomposition, one should always arrange the Sylvester equation so that the Hessenberg decomposition is taken of the matrix $(A_{yy} - B_yF_y)'$ or A_{zz}, whichever has more entries. The steps just described should be implemented if there are more elements in the vector y_t than z_t. If z_t has more elements, then the alternative Sylvester equation

$$P_z{}' = Q_{yz}{}' + A_{yz}'P_y(A_{yy} - B_yF_y) + A_{zz}'P_z{}'(A_{yy} - B_yF_y)'$$

should be solved for the matrix $P_z{}'$.

[17] Alternatively, we could take the Schur decomposition of S as proposed by Bartels and Stewart (1972).

[18] Interesting variations on the Hessenberg-Schur algorithm have been proposed by Hammarling(1982) and Gardiner *et al.* (1992).

In the numerical comparisons that follow, we form the Hessenberg decomposition of a matrix using MATLAB subroutine HESS and the Schur decomposition of a matrix with SCHUR. We solve Hessenberg systems using the routines HSFA and HSSL, which are part of the package described in Gardiner *et al.* (1992).[19]

6.2. Doubling Algorithm

The doubling algorithm for Sylvester equations iterates

$$\begin{aligned} \alpha_{k+1} &= \alpha_k \alpha_k \\ \beta_{k+1} &= \beta_k \beta_k \\ \gamma_{k+1} &= \gamma_k + \alpha_k \gamma_k \beta_k \end{aligned} \tag{6.7}$$

to convergence, where $\alpha_0 = S$, $\beta_0 = T$, and $\gamma_0 = W$. By repeated substitution, it can be shown that

$$\gamma_k = \sum_{j=0}^{2^k-1} S^j W T^j .$$

In other words, each iteration doubles the number of terms in the sum.[20]

To use this doubling algorithm to compute P_z

(i) initialize $\alpha_0 = (A_{yy} - B_y F_y)'$, $\beta_0 = A_{zz}$, and $\gamma_0 = Q_{yz} + (A_{yy} - B_y F_y)' P_y A_{yz}$;
(ii) iterate in accordance to (6.7);
(iii) form P_z as the limit of $\{\gamma_k\}$.

We implement the doubling algorithm in FORTRAN.[21]

---

[19] See pages 364-370 of Golub and Van Loan (1989) for a discussion of how to compute the Hessenberg decomposition.

[20] This algorithm is a slight generalization of the doubling algorithm for Lyapunov equations discussed in Anderson and Moore (1979). A Lyapunov equation is a Sylvester equation in which $S = T'$.

[21] We iterate on (6.7) until

$$\|\gamma_k - \gamma_{k-1}\|_1 \leq \epsilon \|\gamma_k\|_1 ,$$

where we set $\epsilon = 1 \times 10^{-15}$.

7. Distorted Economies

Some of the algorithms described previously are directly applicable to solving models whose equilibrium quantity allocations are not the solutions to optimal resource allocation problems. To illustrate this point, we use a simplified version of McGrattan's (1994) model of a *distorted* economy.[22] Consider a setup with a representative agent who chooses a control sequence $\{v_t\}$ to maximize

$$-\sum_{t=0}^{\infty}(v_t{}'Rv_t + y_t{}'Q_{yy}y_t + 2y_t{}'Q_{y\hat{y}}\hat{y}_t),$$

subject to

$$y_{t+1} = A_{yy}y_t + A_{y\hat{y}}\hat{y}_t + B_y v_t \tag{7.1}$$

$$\sum_{t=0}^{\infty}(|v_t|^2 + |y_t|^2) < \infty,$$

where the sequence $\{\hat{y}_t\}$ is viewed by the agent as being beyond his control when making decisions. As an equilibrium condition, $\hat{y}_t$ is an exact function of y_t and v_t:

$$\hat{y}_t = \Omega y_t + \Psi v_t. \tag{7.2}$$

In formulating the decision problem for the representative agent, we have abstracted from uncertainty and used analogous tricks to those described earlier for eliminating discounting and cross products between states and controls. (See McGrattan 1994 and appendix B.3 of this paper for a more complete treatment.) Also, we have zeroed out the forcing sequence $\{z_t\}$, so this setup should be viewed as a distorted equilibrium counterpart to the *deterministic regulator problem*.

To define an equilibrium for this model, we introduce a process $\{y_t^*\}$ that in equilibrium coincides with $\{y_t\}$. This additional process is used to capture the perceived evolution of $\{\hat{y}_t\}$ by economic agents in making their decisions. Formally, the perceived evolution equation is given by

$$\begin{aligned} y_{t+1}^* &= A^* y_t^*, \\ \hat{y}_t &= \Omega^* y_t^* \end{aligned},$$

where the eigenvalues of A^* are assumed to have absolute values that are strictly less than one. Adding this evolution equation to the decision problem of the private agent is sufficient to make his problem a fully specified *deterministic regulator problem*. Write the solution to this decision problem as

$$v_t = -F_y y_t - F_y^* y_t^*. \tag{7.3}$$

Then a rational expectations equilibrium is a specification of $(F_y, F_y^*, A^*, \Omega^*)$ such that

$$\begin{aligned} A^* &= A_{yy} + A_{y\hat{y}}\Omega - (A_{y\hat{y}}\Psi + B_y)(F_y + F_y^*), \\ \Omega^* &= \Omega - \Psi(F_y + F_y^*), \end{aligned}$$

where control law (7.3) solves the decision problem of the private agent.

As an initial step in solving for an equilibrium, we obtain first-order necessary conditions for the private agent's control problem:

$$v_t: \quad Ru_t + B_y{}'\mu_{t+1} = 0, \qquad t \geq 0 \tag{7.4}$$

$$y_t: \quad \mu_t = Q_{yy}y_t + Q_{y\hat{y}}\hat{y}_t + A_{yy}{}'\mu_{t+1}, \qquad t \geq 0 \tag{7.5}$$

[22] In appendix B.3, we take another version of McGrattan's formulation and differentiate the equilibrium law with respect to parameters in the control problem and equilibrium conditions.

where $\{\mu_t\}$ are Lagrange multipliers associated with the constraint Eq. (7.1). At this stage, we are free to substitute for $\hat{y}_t$ from equilibrium condition (7.2). Solving Eq. (7.3) for v_t, substituting it and Eq. (7.2) into Eqs. (7.1) and (7.5), and rearranging gives

$$L \begin{bmatrix} y_{t+1} \\ \mu_{t+1} \end{bmatrix} = N \begin{bmatrix} y_t \\ \mu_t \end{bmatrix}, \tag{7.6}$$

where

$$L = \begin{bmatrix} I & \hat{B}R^{-1}B_y{}' \\ 0 & \tilde{A}' \end{bmatrix}, \; N = \begin{bmatrix} \hat{A} & 0 \\ -\hat{Q} & I \end{bmatrix},$$

and $\hat{A} = A_{yy} + A_{y\hat{y}}\Omega$, $\hat{Q} = Q_{yy} + Q_{y\hat{y}}\Omega$, $\hat{B} = B_y + A_{y\hat{y}}\Psi$, and $\tilde{A} = A_{yy} - B_y R^{-1}\Psi' Q_z{}'$. Note how these equations generalize (3.4) to a distorted equilibrium model. When distortions are active, the pencil $\lambda L - N$ may fail to be symplectic, so the eigenvalues do not necessarily occur in reciprocal pairs. When the eigenvalues can be split with half inside the unit circle and half outside and the analog of V_{11} in (3.19) is nonsingular, then the deflating subspace and matrix sign methods described earlier can be used to compute the unique stable equilibrium.[23] Under the same conditions, if either $\hat{A}$ or $\tilde{A}$ is nonsingular and well conditioned, then invariant subspace methods also can be used. Finally, Anderson (1995) describes a generalization of the doubling algorithm for Riccati equations that can be used to solve distorted equilibria. Since the pencil is not symplectic, this generalized doubling algorithm includes an additional partition.

For economies with a forcing sequence $\{z_t\}$ with first-order dynamics, there is an analogous formulation of a distorted economy equilibrium. As with the *augmented regulator problem*, the equilibrium can be computed in two steps. First, a distorted equilibrium for z_0 set to zero can be computed using one of the methods described above. Then the full equilibrium can be deduced by solving a Sylvester equation analogous to that deduced for the *augmented control problem*. The Hessenburg-Schur algorithm and the doubling algorithm described in section 6 are both applicable in this second step.

8. Example Economies

In preparation for our numerical work, we describe three examples with various features that "stretch" various of our algorithms to the boundaries of their domains of applicability.

[23] When applying matrix sign methods, one should iterate on (4.7) or (4.8) instead of (4.9), since the matrix $J(L-N)^{-1}(L+N)$ is not, in general, symmetric.

8.1. A Model of Permanent Income with Habit Persistence

Our first example is an economy with two interacting unit roots in the endogenous dynamics. As in Hall (1978), Flavin (1981), and Sargent (1987), one unit root comes from the permanent income character of the model. The technology is specified so that the rate of return on capital and the subjective rate of time discount are equated. As in Hansen (1987), Becker and Murphy (1988), and Heaton (1993), we use an extended version of the permanent income model to accommodate preferences that are not time separable. The second unit root occurs because of the special way we model habit persistence.

There is a single consumption good c_t, a single investment good i_t, a single physical capital stock k_t, and a single household capital stock, h_t, in each time period. The household capital stock is constructed to be a geometric average of current and past consumptions:

$$h_t = .9h_{t-1} + .1c_t,$$

where .9 dictates the geometric decay in the average. We capture habit persistence by introducing a service process:

$$s_t = c_t - h_{t-1}.$$

One source for a unit root in the endogenous dynamics is that the magnitude of the time t service is the difference between current and an average of past consumptions.

The production technology is given by the two relations:

$$c_t + i_t = .1k_{t-1} + d_t$$

$$k_t = .95k_{t-1} + i_t.$$

To provide a permanent income character to this model, we set the subjective discount rate $\beta = \frac{1}{1.05}$.

The preference shock process is restricted to be constant over time ($b = 30$), and the technology shock process $\{d_t\}$ is a first-order autoregression with mean 5 and autoregressive coefficient .8. We represent these processes using the setup of section 2.4 by introducing an exogenous state vector $\hat{z}_t$ with two components. Recall that the exogenous state vector process is assumed to have first-order dynamics. The autoregressive matrix for this process is given by

$$\hat{A}_{zz} = \begin{bmatrix} 1 & 0 \\ 0 & .8 \end{bmatrix},$$

where the first component of $\hat{z}_t$ is initialized at one and remains constant over time. While the second component of $\hat{z}_t$ can be subject to shocks in each time period, certainty equivalence makes the magnitude of the uncertainty inconsequential for solving the model. Hence it is unnecessary to specify the matrix $\hat{C}_z$. The selection matrices U_b and U_d are given by $U_b = [30 \quad 0]$ and $U_d = [5 \quad 1]$.[24]

For this particular economy, there are potential problems in applying two of the algorithms we described in sections 3 and 4. Since the economy has repeated unit roots in the endogenous dynamics, an invariant subspace method that uses an eigenvector routine designed for distinct eigenvalues might give a poor approximation to the solution. Also, this is an economy in which the square summability constraint (2.1) is binding. In other words, it is not optimal to stabilize the endogenous state vector process in the absence of such a constraint. As a consequence, Riccati difference equation iterations starting from the zero matrix converge to the wrong solution, as does the corresponding partition of the $P_o = 0$ doubling algorithm.

---

[24] In this economy, there are no intermediate goods g_t. As suggested in section 2.4, we still use i_t as the control vector, and we can clearly solve for c_t as a linear function of the control and state vectors.

As a potential remedy for both of these pitfalls, we "approximate" our economy by one in which there is a very tiny adjustment cost for physical capital. The cost is captured by introducing a single intermediate good g_t, such that

$$\phi i_t - g_t = 0,$$

where we set $\phi = 1 \text{ x } 10^{-7}$. This small adjustment cost is enough to eliminate the repeated unit roots in the endogenous dynamics. Moreover, it makes (A_{yy}, D_y) detectable, so that it is optimal to stabilize the endogenous state vector process. Since the pair (A_{yy}, B_y) is controllable, this small adjustment cost is enough to guarantee convergence of the $P_o = 0$ version of the doubling algorithm. One of the issues considered in our numerical experiments is how well this "fix up" works in practice. Does the introduction of small adjustment costs make either the eigenvector algorithm or the doubling algorithm a viable method for solving the original control problem? We shall also study this economy with the adjustment costs set equal to zero and with the $P_o = I$ version of the doubling algorithm.

8.2. Siow's Time-To-Educate Model

This example is a version of a time-to-build (or time-to-educate) model of wage skill differentials that was formulated by Siow (1984). Siow's model interprets the premium on educated labor as a present-value-equalizing differential required to compensate for the income foregone during training years. To accord with the framework of section 2.4, we reformulate a version of Siow's model as an optimal resource allocation problem.

Suppose there are three skill levels of labor: "low skill," "medium skill," and "high skill." We adopt the notational convention that low skill work is engaged in home production, while the other two skill levels produce market goods. We assume that it takes four periods to train skilled workers and eight periods to train highly skilled workers. Trainees are not permitted to switch training programs. This gives rise to gestation lags in the production technology.

Let $i_{m,t}$ denote the number of workers who choose the medium-skilled training program and $i_{h,t}$ the number who choose the high-skilled training program at time t. Let $k_{m,t}$ and $k_{h,t}$ be the corresponding stocks of workers. Then

$$k_{m,t} = .97 k_{m,t-1} + .97^4 i_{m,t-4}$$

$$k_{h,t} = .97 k_{h,t-1} + .97^8 i_{h,t-8},$$

where $(1 - .97)$ is the exit rate from the labor force. To capture this gestation lag with the first-order specification of section 2.4, we include in k_t the following:

$$k_t = [\, k_{m,t} \quad k_{h,t} \quad .97^3 i_{m,t-3} \quad .97^2 i_{m,t-2} \quad .97 i_{m,t-1} \quad i_{m,t} \quad .97^7 i_{h,t-7} \quad \ldots \quad .97 i_{h,t-1} \quad i_{h,t} \,]' .$$

The first-order evolution equation for $\{k_t\}$ can now be constructed in the obvious way. Hence to capture the delays in the dynamic technology, we are compelled to augment the endogenous state vector. This augmentation is the source of the singularity in the matrix A_{yy}. The control vector is $i_t = [\, i_{m,t} \quad i_{h,t} \,]'$.

The rest of the people engage in home production. Let $d_{1,t}$ denote the time t flow of newborn or raw labor. The difference

$$c_{1,t} = d_{1,t} - i_{m,k} - i_{h,k}$$

is the flow of workers into home production. We include $c_{1,t}$ as a component of the consumption goods vector for notational convenience. In addition to $c_{1,t}$, there are two other components to c_t: goods produced by medium-skilled workers and goods produced by high-skilled workers. These goods are produced according to the (linear) constant returns to scale technology:

$$c_{m,t} = .7k_{m,t-1}$$

$$c_{h,t} = .9k_{h,t-1}.$$

To capture the disutility of working, we introduce two intermediate goods that satisfy

$$g_{m,t} = k_{m,t-1}$$

$$g_{h,t} = k_{h,t-1};$$

and to capture costs associated with matching new entrants with training programs, we introduce two additional intermediate goods that satisfy

$$\hat{g}_{m,t} = .0002i_{m,t}$$

$$\hat{g}_{h,t} = .0003i_{h,t}.$$

When these constraints are combined, the technology for producing intermediate goods and consumption goods is given by

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_{1,t} \\ c_{m,t} \\ c_{h,t} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} g_{mt} \\ g_{ht} \\ \hat{g}_{m,t} \\ \hat{g}_{h,t} \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ .0002 & 0 \\ 0 & .0003 \end{bmatrix} \begin{bmatrix} i_{mt} \\ i_{ht} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 \\ .7 & 0 \\ 0 & .9 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} k_{m,t-1} \\ k_{h,t-1} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} d_{1,t}.$$

Consider next the household technology. Recall that by our notational convention, $c_{1,t}$ denotes the quantity of new entrants into household production. The stock of such workers at time t (after including the new entrants) is denoted h_t. This "household capital stock" evolves according to

$$h_t = .97h_{t-1} + c_{1,t},$$

so the depreciation factor is the same as for the other two types of labor. Consumption services $s_{1,t}$ are produced according to the linear technology

$$s_{1,t} = .5h_{t-1}.$$

To capture the disutility of working in the household, we introduce a second service

$$s_{2,t} = -h_{t-1};$$

and to capture the (utility) costs to matching new entrants to the household technology, we introduce a third consumption service

$$s_{3,t} = -.0001c_{1,t}.$$

All total, there are five components to the consumption service vector s_t because we also include the consumption goods produced by medium-skilled and high-skilled workers:

$$s_t = [\, s_{1,t} \quad s_{2,t} \quad s_{3,t} \quad c_{m,t} \quad c_{h,t} \,]' .$$

The household's subjective rate of time discount is $\beta = \frac{1}{1.05}$. Forcing process $\{d_{1,t}\}$ is given recursively by

$$\begin{aligned} p_t &= \sum_{j=1}^{47} \theta_j p_{t-j} + z_t \\ d_{1,t} &= p_{t-18}, \end{aligned} \tag{8.1}$$

where p_t are the new births at date t and the θ_j's are set to match the birth rates in the United States in 1990 as reported in the *American Almanac: Statistical Abstract of the United States 1993-1994*. We abstract from long term population growth by appropriately scaling the θ_j's to sum to one.[25] The process $\{z_t\}$ has a first-order autoregressive representation with coefficient .9. The variable p_{t-18} occurs with an 18 period lag in the second equation of (8.1) because we assume that it takes 18 periods (years) before a new born person is ready to enter a training program or produce household goods.

The preference shock process has three nondegenerate components:

$$b_t = [\, b_{1,t} \quad 0 \quad 0 \quad b_{m,t} \quad b_{h,t} \,]' .$$

The zeros in the preference shock process b_t are associated with (dis)services to working in the household and to matching labor to household production. The three nondegenerate components are independent first-order autoregressive processes augmented by 300. For each scalar autoregression, the autoregressive coefficient is .9.

8.3. A Model of Cattle Cycles

In this subsection, we present three versions of Rosen, Murphy, and Scheinkman's (1994) model of cattle cycles. The versions differ according to whether the time units are years, quarters, or months. To match the setup of section 2.4, we reformulate Rosen, Murphy, and Scheinkman's market equilibrium model as an optimal resource problem. We initially describe the yearly model. For our numerical speed and accuracy comparisons with the annual version of this model, we estimated some of the parameters using the methods to be described in subsequent sections. The parameters for the versions of the model at the quarterly and monthly timing intervals were deduced in ways described below.

Let $k_{b,t}$ denote the total stock of breeding cows. Each such animal gives birth to η calves, and calves become part of the adult stock after two years. For simplicity, we set the death rate of cattle to zero. Therefore, the law of motion for the breeding stock is given by

$$k_{b,t} = k_{b,t-1} + \eta k_{b,t-3} + i_t, \tag{8.2}$$

[25] Formally, the θ_j's were constructed as follows. We took birthrates for women from Table 93 of the *American Almanac: Statistical Abstract of the United States 1993-1994* in the year 1990 and divided by two. Since birthrates are only recorded for women grouped in five year age brackets, we interpolated linearly from the midpoints of each age bracket. Birthrates for ages 12 and 47 were set to zero when doing this interpolation, and birth rates up to age 12 were set to zero. The resulting birthrates imply an autoregression with an explosive root that induces geometric growth in population. We then scaled the birth rate parameters by the inverse of the growth factor raised to the appropriate powers to eliminate the growth. The resulting autoregressive process has a unit root by construction.

where i_t denotes deletions from the breeding stock due to slaughtering. Stacking the breeding stocks so as to represent this evolution equation as a first-order system, we obtain

$$\begin{bmatrix} k_{b,t} \\ k_{b,t-1} \\ k_{b,t-2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \eta \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} k_{b,t-1} \\ k_{b,t-2} \\ k_{b,t-3} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} i_t.$$

Consumption $c_t = -i_t$. We use one intermediate good to capture slaughtering costs and three additional ones to capture the holding costs. Holding costs differ depending on whether the animal is a calf, a yearling, or an adult. Let

$$\begin{aligned} g_{1,t} &= \epsilon c_t + (1/\epsilon) d_{s,t} \\ g_{2,t} &= \epsilon k_{b,t-1} + (\gamma_1 \eta/\epsilon) d_{h,t} \\ g_{3,t} &= \epsilon k_{b,t-2} + (\gamma_2 \eta/\epsilon) d_{h,t} \\ g_{4,t} &= \epsilon k_{b,t} + (1/\epsilon) d_{h,t}. \end{aligned} \tag{8.3}$$

As specified, the holding and slaughtering costs are quadratic. The parameter ϵ is set to a small positive number to approximate the linear cost structure used by Rosen, Murphy and Scheinkman (1994). The parameters γ_1 and γ_2 dictate the holding costs for calves and yearlings, respectively, relative to those for fully grown animals. For instance, the approximate holding period cost is $d_{h,t}$ for an adult, $\gamma_1 d_{h,t}$ for a calf, and $\gamma_2 d_{h,t}$ for a yearling. In our computational experiments, the parameters γ_1 and γ_2 are set to 1/3 and to 2/3, respectively. Substituting for $k_{b,t}$ in (8.3) using (8.2) and stacking the equations for consumption and intermediate goods into a system, we get

$$\begin{aligned} &\begin{bmatrix} 1 \\ -\epsilon \\ 0 \\ 0 \\ \epsilon \end{bmatrix} c_t + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} i_t + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} g_{1,t} \\ g_{2,t} \\ g_{3,t} \\ g_{4,t} \end{bmatrix} \\ &= \epsilon \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & \eta \end{bmatrix} \begin{bmatrix} k_{b,t-1} \\ k_{b,t-2} \\ k_{b,t-3} \end{bmatrix} + (1/\epsilon) \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & \gamma_1 \eta \\ 0 & \gamma_2 \eta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_{s,t} \\ d_{h,t} \end{bmatrix}. \end{aligned}$$

Consumption goods and services are related trivially by

$$s_t = (1/\alpha_1) c_t,$$

where α_1 is positive. As a consequence, preferences for consumption are time separable, and the slope of the Frisch demand function for beef is $-\alpha_1$.

The exogenous processes are specified as follows. The preference shock process is given by the constant (α_0/α_1). The parameter α_0 is the intercept in the Frisch demand function. The two technology shock processes $\{d_{s,t}\}$ and $\{d_{h,t}\}$ are each scalar first-order autoregressive processes with unconditional means μ_s and μ_h and autoregressive coefficients ρ_s and ρ_h, respectively.

As a device for proliferating endogenous state variables, we construct analogous quarterly and monthly versions of a cattle cycle model. In so doing, we abstract from any (realistic) periodic specification whereby, for example, a certain season of the year is designated as a calving season. Also, we design the higher frequency models to be only roughly compatible with the annual model. The parameter values selected for all three versions are reported in Table 8.1. The higher frequency parameters are obtained from the following algorithms. Let τ denote the number of seasons in a year (either four or twelve). The higher

Table 8.1: Parameter Values for Yearly, Monthly, and Quarterly Formulations of the Cattle Cycles Model

Parameters	Yearly	Quarterly	Monthly
β	0.960	0.990	0.997
α_0	146.0	36.5	12.17
α_1	1.270	0.318	0.106
$1+\eta$	1.938	1.180	1.057
ρ_h	0.888	0.971	0.990
ρ_s	0.699	0.914	0.971
μ_h	37.00	9.250	3.083
μ_s	63.00	63.00	63.00
ϵ	1×10^{-04}	2.5×10^{-05}	8.33×10^{-06}

Table 9.1: Number of State Variables

Economy	Endogenous States	Exogenous States	Norm of P_y	Norm of P_z
Permanent Income	2	2	$2.45 \times 10^{+00}$	$2.08 \times 10^{+02}$
Yearly Cattle Cycles	3	4	$1.37 \times 10^{+00}$	$2.88 \times 10^{+02}$
Quarterly Cattle Cycles	9	4	$3.53 \times 10^{+00}$	$1.26 \times 10^{+03}$
Monthly Cattle Cycles	25	4	$9.67 \times 10^{+00}$	$3.93 \times 10^{+03}$
Education	15	52	$8.76 \times 10^{+01}$	$3.77 \times 10^{+04}$

frequency versions of β, $1+\eta$, ρ_h, and ρ_s are obtained by taking the annual parameters and raising them to the power $1/\tau$. The higher frequency versions of α, ϵ, and μ_h are constructed by dividing the annual parameters by τ. The parameter μ_s is the same for all versions of the model. Finally, as we proliferate time periods, we extend the number of periods it takes for a calf to become a cow. Instead of two periods, it now takes the animal 2τ periods to be an adult. Accordingly, there are 2τ cost parameters $\gamma_j, j = 1 \ldots 2\tau$. As in the annual model, we assumed these parameters increased linearly from zero to one. Hence $\gamma_j = \frac{j}{\tau+1}$.

9. Numerical Comparisons

In this section, we study the performance of algorithms for computing solutions to the optimal resource allocation problems described in section 8. We report results for six different economies: two permanent income/habit persistence economies, three cattle cycle economies, and one time-to-educate model. Recall that the two permanent income economies are very similar except the second one introduces a very small adjustment cost term so that the resulting (A_{yy}, D_y) is detectable. We label these two economies *Permanent Income* and *Permanent Income (with adjustment costs)* in the subsequent tables. The three cattle cycle economies differ with respect to the presumed decision time interval. The three cattle cycle economies are calibrated to be yearly, quarterly, and monthly decision periods and are labeled *Yearly Cattle Cycles*, *Quarterly Cattle Cycles*, and *Monthly Cattle Cycles*, respectively. Finally, the time-to-educate economy is labeled *Education* in our tables.

Table 9.1 gives the number of endogenous and exogenous state variables for each of six optimal resource allocation problems.[26] There are four exogenous state variables for

[26] We also give approximate matrix one-norms for the true solutions. For the *Permanent Income* economy we used the true solutions to calculate the norms. For the other economies

Table 9.2: Performance of Algorithms that Solve Riccati Equations

Economy	Algorithm	CPU time	Residual Norm
Permanent Income	Riccati Iteration	0.0334	2.8×10^{-15}
	Eigenvector	0.0047	1.1×10^{-03}
	Schur	0.0039	4.4×10^{-16}
	Generalized Eigenvector	0.0045	1.5×10^{-04}
	Generalized Schur	0.0037	4.6×10^{-16}
	Doubling($P_0 = I$)	0.0031	6.1×10^{-16}
	Matrix Sign	0.0058	9.7×10^{-16}
Permanent Income (with adjustment costs)	Riccati Iteration	0.0334	1.9×10^{-15}
	Eigenvector	0.0057	2.4×10^{-07}
	Schur	0.0046	1.4×10^{-15}
	Generalized Eigenvector	0.0048	2.9×10^{-04}
	Generalized Schur	0.0037	1.1×10^{-16}
	Doubling($P_0 = 0$)	0.0022	9.2×10^{-16}
	Doubling($P_0 = I$)	0.0030	9.4×10^{-16}
	Matrix Sign	0.0062	3.7×10^{-15}
Yearly Cattle Cycles	Riccati Iteration	0.0056	9.7×10^{-16}
	Eigenvector	0.0076	2.3×10^{-15}
	Schur	0.0079	3.3×10^{-16}
	Generalized Eigenvector	0.0125	1.7×10^{-15}
	Generalized Schur	0.0054	2.1×10^{-15}
	Doubling($P_0 = 0$)	0.0026	5.6×10^{-16}
	Doubling($P_0 = I$)	0.0036	3.9×10^{-16}
	Matrix Sign	0.0089	6.7×10^{-16}
Quarterly Cattle Cycles	Riccati Iteration	0.0520	2.6×10^{-15}
	Eigenvector	0.0400	9.4×10^{-15}
	Schur	0.0373	1.1×10^{-14}
	Generalized Eigenvector	0.1177	6.2×10^{-15}
	Generalized Schur	0.0248	6.9×10^{-15}
	Doubling($P_0 = 0$)	0.0125	6.7×10^{-16}
	Doubling($P_0 = I$)	0.0131	5.6×10^{-16}
	Matrix Sign	0.0314	2.3×10^{-15}
Monthly Cattle Cycles	Riccati Iteration	1.3860	1.0×10^{-14}
	Eigenvector	0.6904	2.9×10^{-14}
	Schur	0.6575	8.2×10^{-14}
	Generalized Eigenvector	1.3100	5.9×10^{-14}
	Generalized Schur	0.3370	6.1×10^{-14}
	Doubling($P_0 = 0$)	0.1435	3.7×10^{-15}
	Doubling($P_0 = I$)	0.1437	1.4×10^{-15}
	Matrix Sign	0.2569	2.2×10^{-14}
Education	Riccati Iteration	0.2554	8.2×10^{-14}
	Generalized Eigenvector	0.2437	$2.2 \times 10^{+04}$
	Generalized Schur	0.0394	2.2×10^{-06}
	Doubling($P_0 = 0$)	0.0371	3.1×10^{-07}
	Doubling($P_0 = I$)	0.0447	2.7×10^{-07}
	Matrix Sign	0.0841	1.9×10^{-07}

we used the solutions computed by the Riccati Iteration algorithm and the doubling algorithm for Sylvester equations. Given the tables that follow, these norms allow a reader to construct a relative measure of accuracy for the candidate solutions.

Table 9.3: Accuracy of Solutions to the Permanent Income Model

Economy	Algorithm	Absolute error of P_y^c	Absolute error of F_y^c
Permanent Income	Riccati Iteration	6.6×10^{-14}	8.8×10^{-15}
	Eigenvector	2.4×10^{-02}	3.0×10^{-03}
	Schur	8.8×10^{-15}	1.1×10^{-15}
	Generalized Eigenvector	3.1×10^{-03}	4.0×10^{-04}
	Generalized Schur	1.9×10^{-14}	2.6×10^{-15}
	Doubling($P_0 = I$)	8.2×10^{-13}	1.3×10^{-13}
	Matrix Sign	2.8×10^{-14}	3.7×10^{-15}
Permanent Income (with adjustment costs)	Riccati Iteration	5.7×10^{-13}	8.2×10^{-14}
	Eigenvector	4.9×10^{-06}	6.4×10^{-07}
	Schur	5.0×10^{-14}	1.1×10^{-15}
	Generalized Eigenvector	6.0×10^{-03}	7.8×10^{-04}
	Generalized Schur	1.4×10^{-13}	1.5×10^{-14}
	Doubling($P_0 = 0$)	5.0×10^{-13}	7.3×10^{-14}
	Doubling($P_0 = I$)	1.7×10^{-12}	2.8×10^{-13}
	Matrix Sign	5.7×10^{-13}	8.3×10^{-14}

Table 9.4: Performance of Algorithms that Solve Sylvester Equations

Economy	Algorithm	CPU time	Residual Norm
Permanent Income	Hessenberg-Schur	0.0017	3.6×10^{-15}
	Doubling	0.0010	3.6×10^{-15}
Yearly Cattle Cycles	Hessenberg-Schur	0.0027	3.3×10^{-13}
	Doubling	0.0014	2.8×10^{-14}
Quarterly Cattle Cycles	Hessenberg-Schur	0.0041	7.8×10^{-13}
	Doubling	0.0028	2.6×10^{-13}
Monthly Cattle Cycles	Hessenberg-Schur	0.0154	2.6×10^{-12}
	Doubling	0.0186	6.5×10^{-13}
Education	Hessenberg-Schur	0.2601	4.3×10^{-11}
	Doubling	0.1233	5.2×10^{-12}

the cattle cylce economy because we included a state that could be used to represent a preference shock. The autogressive parameter for this state was set to zero. Since the gestation time period for a newborn calf to become a cow is held fixed across the three cattle cycle economies, the number of endogenous state variables is larger for *Monthly Cattle Cycles* than for the other two cattle cycle economies. Recall that the number of exogenous state variables and endogenous state variables is large for the *Education* economy because of the presumed population dynamics and the number of time periods it takes to get *highly skilled*.

Associated with each of the six optimal resource allocation problems is a Riccati equation and a Sylvester equation that are solved in finding the optimal decision rule. We report the Riccati equation comparisons in the first subsection and the Sylvester equation comparisons in the second subsection. Recall that Sylvester equations take as one of their inputs a matrix constructed from the solution to the corresponding Riccati equation. To simplify comparisons, we use the same input matrix for each of the two Sylvester equation algorithms.

9.1. Solutions to Riccati Equations

We compare the performance of seven of the Riccati equation solving algorithms described in section 4. We consider two invariant subspace algorithms: one is based on an eigenvector decomposition labeled *Eigenvector* and the other on the Schur decomposition labeled *Schur* in the tables described below. We study two deflating subspace algorithms that are generalizations of the two invariant subspace algorithms designed to permit the state evolution matrix (A_{yy}) to be singular. (In fact, this matrix is singular for the *Education* resource allocation problem.) We label these deflating subspace algorithms *Generalized Eigenvector* and *Generalized Schur*. We investigate two doubling algorithms that differ with respect to how they are initialized. The first doubling algorithm uses the standard initialization ($P_o = 0$), and the second one initializes the doubling algorithm so that the terminal state and costate vectors coincide ($P_o = I$). Since the ($P_o = 0$) doubling algorithm gives the wrong solution to the *Permanent Income* resource allocation problem, it is not included for that contol problem. Both of these algorithms are labeled *Doubling* with the specification of P_o given in parentheses. Our seventh algorithm is the matrix sign algorithm and is labeled accordingly. As a benchmark, one of the algorithms iterates on the Riccati difference equation from dynamic programming.[27] This algorithm is labeled *Riccati Iteration* in the tables.

Table 9.2 reports comparisons of the performance of the eight algorithms used to compute compute candidate solutions $(P_y^c, F(P_y^c))$ to the associated *deterministic regulator problem*'s given the inputs (A_{yy}, B_y, Q_{yy}, R). Here $F(P) = (R + B_y{}'PB_y)^{-1}B_y{}'PA_{yy}$. [28] To measure the accuracy of the computed solutions, we use the matrix one-norm of the Riccati equation residual $P_y^c - T(P_y^c)$ where

$$T(P) = Q_{yy} + A_{yy}'PA_{yy} - A_{yy}'PB_y(R + B_y{}'PB_y)^{-1}B_y{}'PA_{yy}.$$

Gudmundsson, Kenney and Laub(1992) show that P_y^c is an accurate solution of the Riccati equation (3.13) if it has a small residual and the Riccati equation is "well-conditioned."

For the *Permanent Income* resource allocation problems, Table 9.3 reports the absolute errors

$$\left\|P_y - P_y^c\right\|_1, \quad \left\|F_y - F(P_y^c)\right\|_1.$$

These errors were computed under the presumption that the first problem (without adjustment costs) is the problem of interest. That is, we compare the true solutions to the *Permanent Income Economy* to the computed solutions to the *Permanent Income Economy* and the *Permanent Income Economy(with adjustment costs)*. Recall that the primary reason we introduced the adjustment costs is to make the doubling ($P_y = 0$) algorithm applicable. For the *Permanent Income* economy, we calculated the true solutions for F_y and P_y by hand:

$$P_y = \begin{bmatrix} 7/3 & -7/60 \\ -7/60 & 7/1200 \end{bmatrix}, \quad F_y = [\,-1/3 \quad 1/60\,].$$

[27] The Riccati iteration algorithm iterates on

$$P_{j+1} = Q_{yy} + (A_{yy} - B_yF_j)'P_j(A_{yy} - B_yF_y) + F_j{}'RF_j,$$

where

$$F_j = (R + B_y{}'P_jB_y)^{-1}B_y{}'P_jA_{yy}$$

until $\|P_{j+1} - P_j\|_1 \le \epsilon\|P_j\|_1$, where we set $\epsilon = 1 \text{ x } 10^{-15}$. We initialize this algorithm at $P_o = I$.

[28] All comparisons reported in the section were performed on an HP-9000/730 computer with 64MB of memory using version 4.2a of MATLAB and HP's FORTRAN compiler. We base our CPU times on 1100 replications.

The results verify that (for the *Permanent Income* economy) the residual errors reported in Table 9.2 are close proxies for the absolute errors reported in Table 9.3. Since solutions to *Permanent Income (with adjustment costs)* approximate closely the solutions to *Permanent Income*, applying the doubling algorithm to the adjustment cost version gives a reliable solution to the resource allocation problem without adjustment costs.

Returning now to the result in Table 9.2, the following comparisons are noteworthy.

(1) The eigenvector and generalized eigenvector algorithms are unreliable for three of our six economies. Not suprisingly, the presence of repeated roots in the solution to the *Permanent Income* control problem caused the eigenvector algorithm to give unreliable solutions. Shifting to the generalized eigenvector algorithm resulted only in marginal improvements in accuracy. While introducing tiny adjustment costs to the *Permanent Income* control problem improved the accuracy of the eigenvector method, it failed to make the eigenvector method as accurate as the other methods. The generalized eigenvector method performed poorly for both this control problem and the *Education* problem.

(2) The Riccati iteration algorithm computed accurate solutions for all of the control problems and, in particular, computed the most accurate solution for the *Education* problem. Hence if accuracy is the primary concern, rather than speed, this algorithm is a reasonable choice. However, in situations in which repeated solutions are required, other algorithms can save the researcher a significant amount of time.[29] Speed gains are likely to be important in econometric estimation and in determining the sensitivity of solutions to changes in parameter settings.

(3) Algorithms that allow A_{yy} to be singular do not suffer any "penalties" in speed or in accuracy. Hence for our discrete-time control problems, there does not seem to be a good reason to use the invariant subspace algorithms.

(4) Both doubling algorithms performed relatively well across the six economies. The $P_o = 0$ algorithm is a little faster than the $P_o = I$ algorithm for the *Permanent Income (with adjustment costs)* and for the *Yearly Cattle Cycles* control problems with comparable accuracy. The $P_o = I$ algorithm is the quickest of the seven applicable algorithms in solving the original *Permanent Income* control problem. The $P_o = 0$ doubling algorithm outperforms the generalized Schur and matrix sign algorithms. A possible reason it is faster than the generalized Schur algorithm is that the generalized Schur algorithm does not exploit the symplectic structure of the control problem.

[29] The speed of the Riccati iteration algorithm can be increased by lowering the tolerance ϵ. For instance, if ϵ is changed to 1×10^{-07}, for the *Permanent Income Economy* the CPU is reduced to 0.0163 with an absolute error of 5.3×10^{-06} for P_y. Comparable changes in tolerance settings for the other iterative algorithms had very minor changes in speed and accuracy for the *Permanent Income Economy.* Our experience with the matrix sign algorithm applied to other economies is that significantly lowering the tolerance can have disastrous consequences for accuracy.

9.2. Solutions to Sylvester Equations

Table 9.4 compares the performance of the Sylvester equation algorithms discussed in section 6 applied to the five control problems. The algorithms take as inputs the matrices (S, T, W). To assess the accuracy of the solutions, we use the matrix one-norm of the Sylvester equation residual $W + SM^cT - M^c$, where M^c is a candidate solution. For the *Permanent Income* control problem, the absolute error, $\|M - M^c\|_1$, of the Hessenberg-Schur solution is 9.1×10^{-13} and the absolute error of the doubling algorithm's solution is 1.0×10^{-12}.

The accuracy of the doubling and Hessenberg-Schur algorithms are comparable. While the doubling algorithm is faster in solving four of the five Sylvester equations, the Hessenberg-Schur algorithm is faster in solving the Sylvester equation for the *Monthly Cattle Cycles* control problem. Recall that this problem has 25 endogenous states but only four exogenous states. The Hessenberg-Schur algorithm is apparently better at exploiting this asymmetry.

10. Innovations Representations

Constructing an *innovations representation* is a key step in deducing the implications of a model for vector autoregressions and for evaluating a Gaussian likelihood function.[30] An innovations representation is a state-space representation in which the vector white noise driving the system is of the correct dimension (equal to that of the vector of observables) and lives in the proper space (the space spanned by current and lagged values of the observables).

Suppose that our theorizing and data collection lead us to a system of the form[31]

$$\begin{aligned} x_{t+1} &= A_o x_t + C w_{t+1} \\ z_t &= G x_t + v_t \\ v_{t+1} &= D v_t + H w_{t+1}, \end{aligned} \tag{10.1}$$

where D is a matrix whose eigenvalues are bounded in modulus by unity, and $\{w_t\}$ is a martingale difference sequence with $E(w_{t+1}w_{t+1}{}'|\mathcal{F}_t) = I$, where $\mathcal{F}_t$ is the sigma field generated by the history of w_s up to t. We take z_t to be the time t vector of variables on which an econometrician has observations, and we interpret v_t as a serially correlated measurement error vector. We let $R = HH'$, which is the covariance matrix of Hw_{t+1}. We impose $CH' = 0$, by way of assuming that the "state" and "measurement" errors are uncorrelated.

We define the following quasi-differenced process

$$\bar{z}_t \equiv z_{t+1} - Dz_t. \tag{10.2}$$

From Eq. (10.1) and the definition (10.2), it follows that

$$\bar{z}_t = (GA_o - DG)x_t + (GC + H)w_{t+1}.$$

Then $(x_t, \bar{z}_t)$ is governed by the state space system

$$\begin{aligned} x_{t+1} &= A_o x_t + C w_{t+1} \\ \bar{z}_t &= \bar{G} x_t + (GC + H) w_{t+1}, \end{aligned} \tag{10.3}$$

[30] The calculations in this section are versions of ones described by Anderson and Moore (1979). We alert the reader that we are "recycling" or "reinitializing" some of notation used in earlier sections, such as z_t, v_t, u_t, D, R.

[31] In particular, the solution to the *discounted stochastic regulator problem* can be expressed as $x_{t+1} = A_o x_t + C w_{t+1}$ where $A_o = A - BF$.

where $\bar{G} = GA_o - DG$. This system has nonzero covariance between the state noise Cw_{t+1} and the "measurement noise" $(GC + H)w_{t+1}$. Let $[K_t, \Sigma_t]$ be the Kalman gain and state covariance matrix associated with the Kalman filter, namely,

$$\begin{aligned} K_t &= (CC'G' + A_o\Sigma_t\bar{G}')\Omega_t^{-1} && (10.4)\\ \Omega_t &= \bar{G}\Sigma_t\bar{G}' + R + GCC'G' && (10.5)\\ \Sigma_{t+1} &= A_o\Sigma_t A_o{}' + CC' - (CC'G' + A_o\Sigma_t\bar{G}')\Omega_t^{-1}(\bar{G}\Sigma_t A_o{}' + GCC'). && (10.6)\end{aligned}$$

Then an innovations representation for system (10.3) is

$$\begin{aligned} \hat{x}_{t+1} &= A_o\hat{x}_t + K_t u_t \\ \bar{z}_t &= \bar{G}\hat{x}_t + u_t, \end{aligned} \tag{10.7}$$

where

$$\begin{aligned} \hat{x}_t &= \hat{E}[x_t \mid \bar{z}_{t-1}, \bar{z}_{t-2}, \ldots, \bar{z}_0, \hat{x}_0] \\ u_t &= \bar{z}_t - \hat{E}[\bar{z}_t \mid \bar{z}_{t-1}, \ldots, \bar{z}_0, \hat{x}_0] \\ \Omega_t &\equiv E u_t u_t' = \bar{G}\Sigma_t\bar{G}' + R + GCC'G'. \end{aligned} \tag{10.8}$$

Initial conditions for the system are $\hat{x}_0$ and Σ_0. From definition (10.2), it follows that $[z_{t+1}, z_t, \ldots, z_0, \hat{x}_0]$ and $[\bar{z}_t, \bar{z}_{t-1}, \ldots, \bar{z}_0, \hat{x}_0]$ span the same space, so that

$$\begin{aligned} \hat{x}_t &= \hat{E}[x_t \mid z_t, z_{t-1}, \ldots, z_0, \hat{x}_0] \\ u_t &= z_{t+1} - \hat{E}[z_{t+1} \mid z_t, \ldots, z_0, \hat{x}_0]. \end{aligned}$$

The process u_t is said to be an *innovation process* in z_{t+1}.

Equation (10.6) is a matrix Riccati difference equation. The Kalman filter has a steady-state solution if there exists a time-invariant positive semi-definite matrix Σ which satisfies Eq. (10.6) with $\Sigma_{t+1} = \Sigma_t$, i.e., one that satisfies the algebraic matrix Riccati equation. In this case, the same computational procedures used for the optimal linear regulator problem apply: a benefit of the duality of filtering and control. The steady-state Kalman gain K is given by Eq. (10.4) with $\Sigma_t = \Sigma$ and $\Omega_t = \bar{G}\Sigma\bar{G}' + R + GCC'G'$.

10.1. Wold and Autoregressive Representations

The innovations representation is associated with a *Wold representation* or *vector autoregression*. Estimates of these representations are recovered in empirical work using the vector autoregressive techniques promoted by Sims (1980) and Doan, Litterman, and Sims (1984). Wold and vector autoregressive representations are easy to obtain when $A - K\bar{G}$ is a stable matrix. To get a Wold representation for z_t, substitute Eq. (10.2) into Eq. (10.7) to obtain

$$\begin{aligned} \hat{x}_{t+1} &= A_o\hat{x}_t + Ku_t \\ z_{t+1} - Dz_t &= \bar{G}\hat{x}_t + u_t. \end{aligned} \tag{10.9}$$

A Wold representation for z_t is

$$z_{t+1} = [I - DL]^{-1}[I + \bar{G}(I - A_oL)^{-1}KL]u_t, \tag{10.10}$$

where, again, L is the lag operator. From Eq. (10.9) a recursive whitening filter for obtaining $\{u_t\}$ from $\{z_t\}$ is given by

$$\begin{aligned} u_t &= z_{t+1} - Dz_t - \bar{G}\hat{x}_t \\ \hat{x}_{t+1} &= A_o\hat{x}_t + Ku_t \end{aligned}. \tag{10.11}$$

Hansen and Sargent (1994) show that an autoregressive representation for z_t is

$$z_{t+1} = \{D + (I - DL)\,\bar{G}[I - (A_o - K\bar{G})L]^{-1}KL\}\,z_t + u_t \tag{10.12}$$

or

$$\begin{aligned} z_{t+1} =& [D + \bar{G}K]z_t + \sum_{j=1}^{\infty}[\bar{G}(A_o - K\bar{G})^j K \\ & - D\bar{G}(A_o - K\bar{G})^{j-1}K]z_{t-j} + u_t. \end{aligned} \tag{10.13}$$

This equation expresses z_{t+1} as the sum of the one-step-ahead linear least squares forecast and the one-step prediction error.

11. The Likelihood Function

Obtaining the Kalman gain sequence $\{K_t\}$ of the previous section is a key step in constructing and manipulating a recursive representation of a Gaussian quasi-likelihood function. It is often necessary to transform the observations into a form matching the linear state-space form. Thus, we start with a "raw" time series $\{y_t\}$ that determines an adjusted series z_t according to

$$z_t = f(y_t, \Theta),$$

where Θ is the vector containing the free parameters of the model, including parameters determining particular detrending procedures. For example, if our raw series has a geometric growth trend equal to μ^t which is to be removed before estimation, then the adjusted series is $z_t = y_t/\mu^t$. We assume that the state space model of the form (10.3) and the associated innovations representation (10.7) pertain to the adjusted data $\{z_t\}$. We can use the innovations representation (10.7) recursively to compute the innovation series, then calculate the conditional Gaussian log-likelihood function

$$L(\Theta) = \sum_{t=0}^{T-1}\{\log|\Omega_t| + \mathrm{trace}(\Omega_t^{-1}u_t u_t') - 2\log|\frac{\partial f(y_t,\Theta)}{\partial y_t}|\} \tag{11.1}$$

and find estimates, $\hat{\Theta} = \mathrm{argmin}_{\Theta} L(\Theta)$, where $\Omega_t = Eu_t u_t'$ is the covariance matrix of the innovations computed from (10.8).[32] To find the minimizer $\hat{\Theta}$, we can use a standard optimization program. In practice, it is best if we can calculate both the log-likelihood function and its derivatives analytically. First, the computational burden is much lower with analytical derivatives. Consider, for example, the model of McGrattan, Rogerson, and Wright (1995), which has 64 elements in Θ. For each step of a quasi-Newton optimization routine, L and $\frac{\partial L}{\partial \theta}$ are computed. To obtain $\frac{\partial L}{\partial \theta}$ numerically for the McGrattan, Rogerson, Wright (1995) example, the log-likelihood function must be evaluated 128 times if central differences are used in computing an approximation for $\frac{\partial L}{\partial \theta}$, e.g.,

$$\frac{\partial L}{\partial \theta} \approx \frac{L(\Theta + \epsilon e) - L(\Theta - \epsilon e)}{2\epsilon}, \tag{11.2}$$

[32] This is the log likelihood function conditional on an initial estimate $\hat{x}_0$ of the state vector. The conditional log likelihood is conveniently factored as

$$\log \Pr(z_t, z_{t-1}, \ldots, z_0|\hat{x}_0) = \log \Pr(z_t|z_{t-1}, \ldots, z_0, \hat{x}_0)\cdots\log\Pr(z_1|z_0,\hat{x}_0)\log\Pr(z_0|\hat{x}_0).$$

For alternative ways of modelling $\hat{x}_0$, see Ansley and Kohn (1985), Hamilton (1994), and Hansen and Sargent (1994).

where e is a vector of zeros except for a 1 in the element corresponding to θ and ϵ is some positive number. Usually, the costs of computing L a large number of times far outweigh the costs of computing $\frac{\partial L}{\partial \theta}$ once. If L and $\frac{\partial L}{\partial \theta}$ are to be computed many times, which is typically the case, then the costs of computing numerical derivatives can be quite large. A second advantage to analytical derivatives is numerical accuracy. If the log-likelihood function is not very smooth for the entire parameter space, there may be problems with the accuracy of approximations such as Eq. (11.2). With inaccurate derivatives, it is difficult to determine the curvature of the function and, hence, to find a minimum.

For $L(\Theta)$ in Eq. (11.1), the derivatives $\partial L(\Theta)/\partial\theta$ can be derived by following procedures of Kashyap (1970), Wilson and Kumar (1982), and Zadrozny (1988a, 1989, 1992). We display these derivatives in Appendix B and distinguish formulas that are steps in the derivation from those that would be put into a computer code. Note that although the final expression for $\frac{\partial L}{\partial \theta}$ derived in Appendix B is complicated, we can use numerical approximations such as Eq. (11.2) to uncover coding errors.

Once we have the log-likelihood function and its derivatives, we can apply standard optimization methods to the problem of finding the maximum likelihood estimates. In practice, we will have a constrained optimization problem since the equilibrium is not typically computable for all possible parameterizations. For example, we may have simple constraints such as $\ell < \Theta < u$, where ℓ and u are the lower and upper bounds for the parameter vector. In this case, we use either a constrained optimization package or penalty functions (see Fletcher 1987).

After computing the maximum likelihood estimates, we need to compute their standard errors,

$$S_e(\Theta) = \mathrm{diag}\left(\sqrt{(\sum_t \frac{\partial L_t}{\partial\Theta}\frac{\partial L_t}{\partial\Theta}')^{-1}}\right), \tag{11.3}$$

where $L_t(\Theta)$ is the logarithm of the density function of the date t innovation, i.e.,

$$L_t(\Theta) = \log|\Omega_t| + u_t'\Omega_t^{-1}u_t - 2\log|\frac{\partial f(y_t,\Theta)}{\partial y_t}|. \tag{11.4}$$

The formula for $\frac{\partial L_t}{\partial \theta}$ is also given in Appendix B.

12. Estimating the Cattle Cycles Model

In this section, we present estimates of some of the parameters of Rosen, Murphy, and Scheinkman's (1994) model. [33] We let p_t be the price of freshly slaughtered beef, $d_{s,t}$ the feeding cost of preparing an animal for slaughter, $d_{h,t}$ the one-period holding cost for a mature animal, $\gamma_1 d_{h,t}$ the one-period holding cost for a yearling, and $\gamma_2 d_{h,t}$ the one-period holding cost for a calf. The costs $\{d_{h,t}, d_{s,t}\}_{t=0}^{\infty}$ are exogenous stochastic processes, while the stochastic process $\{p_t\}_{t=0}^{\infty}$ is determined by an equilibrium. Let $k_{b,t}$ be the breeding stock and y_t be the total stock of animals. Each animal that is reserved for breeding gives birth to η calves. Calves that survive become part of the adult stock after 2 years. Letting t index years, the law of motion for stocks is[34]

$$k_{b,t} = k_{b,t-1} + \eta k_{b,t-3} - c_t, \tag{12.1}$$

[33] We have used estimates of key parameters from this section in the numerical experiments for the annual model.

[34] We have set the death probability in Rosen, Murphy and Scheinkman's(1994) model to zero.

Table 12.1: Parameter estimates for "Cattle Cycle" example.

Parameters	Estimates	Standard Errors
α_0	146	33.4
α_1	1.27	0.323
γ_1	0.647	11.5
γ_2	1.77	12.0
η	0.938	0.0222
ρ_h	0.888	0.115
ρ_s	0.699	0.0417
σ_h	6.82	10.6
σ_s	4.04	1.05
σ_y	0.273	0.0383
σ_c	4.82	0.531

where c_t is a rate of slaughtering. The total head count of cattle is

$$y_t = k_{b,t} + \eta k_{b,t-1} + \eta k_{b,t-2}, \tag{12.2}$$

which is the sum of adults, yearlings, and calves, respectively.

A representative farmer maximizes

$$E_0 \sum_{t=0}^{\infty} \beta^t \left\{ p_t c_t - d_{h,t} k_{b,t} - (\gamma_1 d_{h,t})(\eta k_{b,t-1}) - (\gamma_2 d_{h,t})(\eta k_{b,t-2}) - d_{s,t} c_t - \frac{\epsilon}{2} \Psi_t \right\} \tag{12.3}$$

where

$$\Psi_t = \left(k_{b,t}^2 + k_{b,t-1}^2 + k_{b,t-2}^2 + c_t^2 \right).$$

Here ϵ is a small positive parameter which measures the quadratic costs of carrying stocks and slaughtering.

Demand is governed by

$$c_t = \alpha_0 - \alpha_1 p_t \tag{12.4}$$

where $\alpha_0 > 0$ and $\alpha_1 > 0$. The stochastic processes $\{d_{h,t}, d_{s,t}\}$ are univariate autoregressions with orthogonal innovations

$$\begin{aligned} d_{h,t+1} &= (1-\rho_h)\mu_h + \rho_h d_{h,t} + \epsilon_{h,t}, \\ d_{s,t+1} &= (1-\rho_s)\mu_s + \rho_s d_{m,t} + \epsilon_{s,t}, \end{aligned}$$

where $E\epsilon_{h,t}^2 = \sigma_h^2$ and $E\epsilon_{s,t}^2 = \sigma_s^2$. The disturbance processes $\{\epsilon_{h,t}\}$ and $\{\epsilon_{s,t}\}$ are white noises that are uncorrelated at all lags.

To compute parameter estimates, we use the data of Rosen, Murphy, and Scheinkman (1994), which include annual observations for y_t, c_t, and p_t for the United States during the period 1900-1990.[35] We assume that there is error in measuring the total stock of cattle y_t and the slaughter rate c_t. In particular, we assume that the (1,1) element of R, the variance-covariance matrix of the measurement errors, is equal to σ_y^2, and we assume that the (2,2) element of R is equal to σ_c^2. All other elements of R are set equal to zero.

We are now equipped to estimate the parameters of this model by applying the formulas of the previous sections. We start with some a priori restrictions. Assume that $\beta = 0.96$,

[35] The sources of these data are the *Historical Statistics of the United States, Colonial Times to 1970* and *Agricultural Statistics*. In the data, y is the total stock of cattle excluding milk cows, c is the cattle slaughtered, and p is price of slaughtered cattle.

$\epsilon = 1 \times 10^{-4}$, $\mu_h = 37$, and $\mu_s = 63$. The remaining parameters are elements of Θ, i.e., $\Theta = [\alpha_0, \alpha_1, \gamma_1, \gamma_2, \eta, \rho_h, \rho_s, \sigma_h, \sigma_s, \sigma_y, \sigma_c]$. In Table 12.1, we report estimates of these parameters and standard errors for the estimates. Note that from the values for α_0 and α_1 we can get an estimate of the demand elasticity. For this model, the elasticity is given by -0.61.[36] The values of γ_1 and γ_2 give us information about the holding costs. The estimates indicate that the costs are higher for calves than for yearlings. However, the standard errors on γ_1 and γ_2 indicate that these parameters are not precisely estimated. The value of η implies that $0.94k_{b,t-1}$ calves are born at date t, where $k_{b,t-1}$ is the breeding stock at $t-1$. This estimate is higher than Rosen, Murphy, and Scheinkman's (1994) estimate of 0.85. The estimates of ρ_h and ρ_s imply that there is persistence in the processes for holding and feeding costs. Finally, the estimates of σ_y and σ_c indicate that the measurement error is higher for the slaughter rate than for the total stock.

In Figures 1 through 3, we plot the predicted and actual time series for the stock of cattle, the slaughter rate, and the price. The predicted series are the one-step-ahead forecasts. Using the notation of section 10 these are given by the vector $\bar{G}\hat{x}_t$.

[36] This estimate is $\alpha_1 \times p_0/c_0$ (-1.27$\times$0.48).

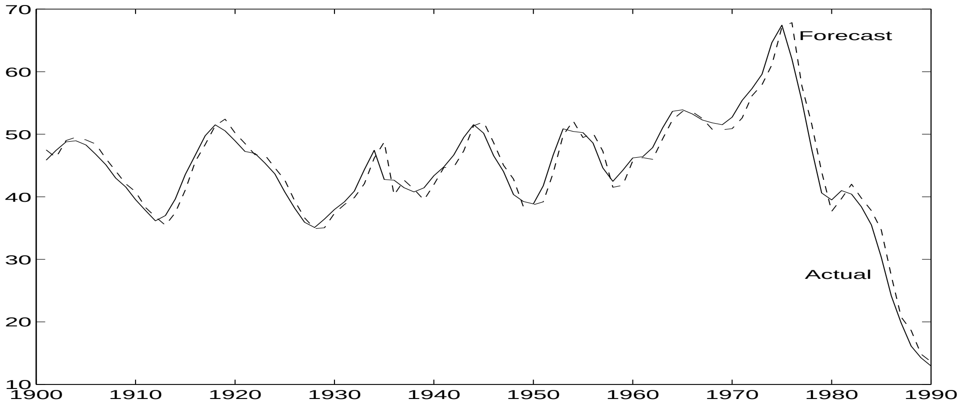

Figure 1. One-step-ahead forecast and actual total stock.

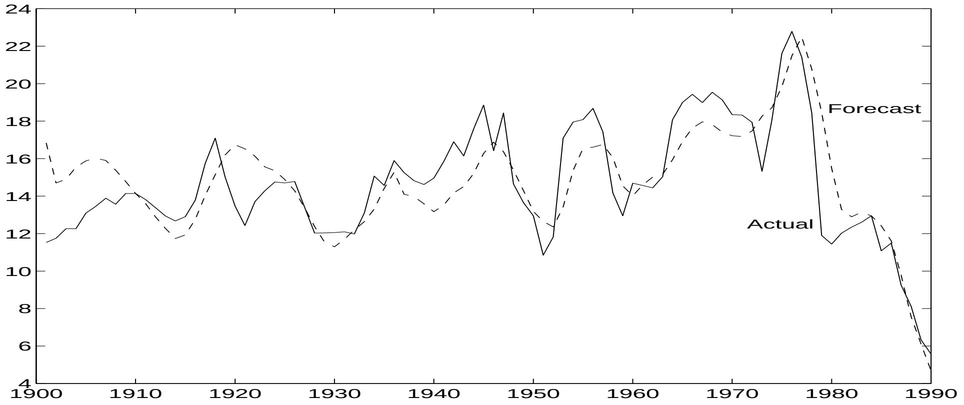

Figure 2. One-step-ahead forecast and actual slaughter rate.

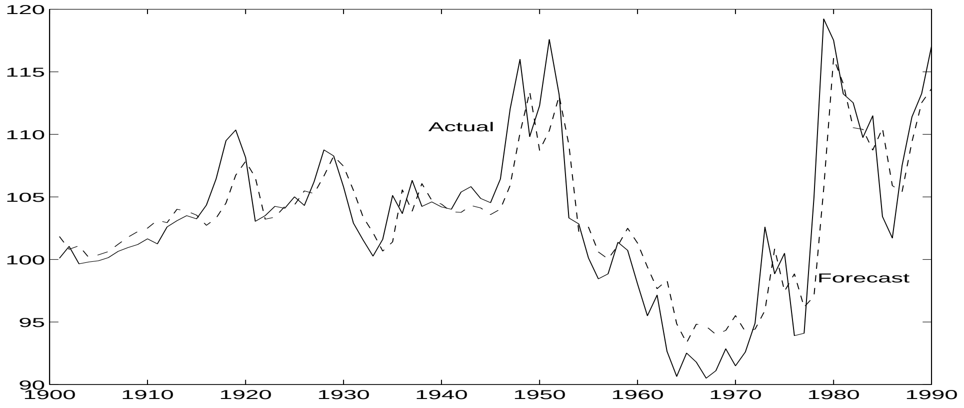

Figure 3. One-step-ahead forecast and actual price of slaughtered beef.

A. Computing $\frac{\partial L}{\partial \theta}$ and $\frac{\partial L_t}{\partial \theta}$ for a State Space Model

Differentiating the log-likelihood function with respect to the free parameters of the economic model can be broken into two steps: first, differentiating the log-likelihood function with respect to matrices appearing in the state space model (10.7); and second, differentiating the parameters of the state space model (10.3) with respect to the free parameters of the underlying economic model. In this appendix, we derive $\frac{\partial L}{\partial \theta}$ in terms of the derivatives of A_o, C, G, D, R, $\hat{x}_0$, Σ_0, and $\{z_t, t=0,\ldots T\}$. We ignore the Jacobian in Eq. (11.1) since it differs for each problem. In Appendix B, we show how to compute derivatives of A_o for the linear-quadratic and nonlinear economies with and without distortions.

A.1. The Formula for $\frac{\partial L}{\partial \theta}$

For the first step, we take as given A_o, C, G, D, R, $\hat{x}_0$, Σ_0, and $\{z_t, t=0,\ldots T\}$ and their derivatives with respect to the deeper economic parameters. We shall show that the derivative of the log-likelihood function is

$$
\begin{aligned}
\frac{\partial L}{\partial \theta} &= \sum_{t=0}^{T-1} \big[2 \text{ trace}\{\frac{\partial A_o}{\partial \theta}\Sigma_t \bar{G}' M_t G - \hat{x}_t u_t' \Omega_t^{-1} G\} + 2 \text{ trace}\{\frac{\partial C}{\partial \theta} C' G' M_t G\} \\
&+ 2 \text{ trace}\{\frac{\partial G}{\partial \theta}(Ao\Sigma_t \bar{G}' M_t - \Sigma_t \bar{G}' M_t D + CC'G'M_t - A_o \hat{x}_t u_t' \Omega_t^{-1} \\
&\qquad + \hat{x}_t u_t' \Omega_t^{-1} D)\} \\
&- 2 \text{ trace}\{\frac{\partial D}{\partial \theta} G\Sigma_t \bar{G}' M_t - z_t u_t' \Omega_t^{-1} + G\hat{x}_t u_t' \Omega_t^{-1})\} \\
&+ \text{ trace}\{\frac{\partial R}{\partial \theta} M_t\} + \text{ trace}\{\frac{\partial \Sigma_t}{\partial \theta}\bar{G}' M_t \bar{G}\} - 2 \text{ trace}\{\frac{\partial \hat{x}_t}{\partial \theta} u_t' \Omega_t^{-1} \bar{G}\} \\
&+ 2 \text{ trace}\{\frac{\partial z_{t+1}}{\partial \theta} u_t' \Omega_t^{-1}\} - 2 \text{ trace}\{\frac{\partial z_t}{\partial \theta} u_t' \Omega_t^{-1} D\}\big], \qquad (A.1)
\end{aligned}
$$

where

$$
\begin{aligned}
\frac{\partial \Sigma_{t+1}}{\partial \theta} &= \frac{\partial A_o}{\partial \theta}\Sigma_t A_o' + A_o \frac{\partial \Sigma_t}{\partial \theta} A_o' + A_o \Sigma_t \frac{\partial A_o}{\partial \theta}' + \frac{\partial C}{\partial \theta} C' + C \frac{\partial C'}{\partial \theta} \\
&\quad - \Big(\frac{\partial C}{\partial \theta} C' G' + C \frac{\partial C'}{\partial \theta} G' + CC' \frac{\partial G'}{\partial \theta} + \frac{\partial A_o}{\partial \theta} \Sigma_t \bar{G}' \\
&\quad + A_o \frac{\partial \Sigma_t}{\partial \theta} \bar{G}' + A_o \Sigma_t \frac{\partial \bar{G}}{\partial \theta}' \Big) K_t' + K_t \frac{\partial \Omega_t}{\partial \theta} K_t' \\
&\quad - K\Big(\frac{\partial \bar{G}}{\partial \theta}\Sigma_t A_o' + \bar{G}\frac{\partial \Sigma_t}{\partial \theta} A_o' + \bar{G}\Sigma_t \frac{\partial A_o}{\partial \theta}' \\
&\quad + \frac{\partial G}{\partial \theta} CC' + G \frac{\partial C}{\partial \theta} C' + GC \frac{\partial C'}{\partial \theta}\Big) \qquad (A.2) \\
\frac{\partial \hat{x}_{t+1}}{\partial \theta} &= \bar{A}_o \frac{\partial \hat{x}_t}{\partial \theta} + (\frac{\partial A_o}{\partial \theta} - \frac{\partial K_t}{\partial \theta}\bar{G} - K_t \frac{\partial \bar{G}}{\partial \theta})\hat{x}_t + \frac{\partial K_t}{\partial \theta}\bar{z}_t \\
&\quad + K_t(\frac{\partial z_{t+1}}{\partial \theta} - D\frac{\partial z_t}{\partial \theta}). \qquad (A.3)
\end{aligned}
$$

The expressions in $(A.2)$ and $(A.3)$ follow from the definitions of Σ_t in Eq. (10.6) and $\hat{x}_t$ in Eq. (10.7). The initial conditions $\hat{x}_0$ and Σ_0 and their derivatives are assumed to be given.

If Σ_0 is given by the steady state solution of the Riccati equation, then the computation can be simplified. The formula for the derivative of the log-likelihood function is given by

$$
\begin{aligned}
\frac{\partial L}{\partial \theta} &= 2T \text{ trace}\{\frac{\partial A_o}{\partial \theta}(\Sigma\bar{G}'MG - \Gamma_{\hat{x}u}\Omega^{-1}G - \Gamma_{\hat{x}\lambda}(I-KG) \\
&\qquad - \Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}(I-KG) - \Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1}G \\
&\qquad + \Sigma\bar{A}_o'\Pi(I-KG))\} \\
&+ 2T \text{ trace}\{\frac{\partial C}{\partial \theta}C'(G'MG - G'\Omega^{-1}\Gamma_{u\lambda}(I-KG) \\
&\qquad - (I-G'K')\Gamma_{u\lambda}'\Omega^{-1}G + (I-G'K')\Pi(I-KG))\} \\
&+ 2T \text{ trace}\{\frac{\partial G}{\partial \theta}(A_o\Sigma\bar{G}'M - \Sigma\bar{G}'MD + CC'G'M \\
&\qquad - A_o\Gamma_{\hat{x}u}\Omega^{-1} + \Gamma_{\hat{x}u}\Omega^{-1}D + A_o\Gamma_{\hat{x}\lambda}K \\
&\qquad - \Gamma_{\hat{x}\lambda}KD - CC'(I-G'K')\Gamma_{u\lambda}'\Omega^{-1} \\
&\qquad + CC'G'\Omega^{-1}\Gamma_{u\lambda}K - A_o\Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1} \\
&\qquad + \Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1}D + A_o\Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}K \\
&\qquad - \Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}KD - A_o\Sigma\bar{A}_o'\Pi K + \Sigma\bar{A}_o'\Pi KD \\
&\qquad - CC'\Pi K + CC'G'K'\Pi K)\} \\
&- 2T \text{ trace}\{\frac{\partial D}{\partial \theta}(G\Sigma\bar{G}'M + (\Gamma_{zu} - G\Gamma_{\hat{x}u})\Omega^{-1} \\
&\qquad + G\Gamma_{\hat{x}\lambda}K - \Gamma_{z\lambda}K - G\Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1} \\
&\qquad + G\Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}K - G\Sigma\bar{A}_o'\Pi K)\} \\
&+ 2T \text{ trace}\{\frac{\partial R}{\partial \theta}(\frac{1}{2}M + \Omega^{-1}\Gamma_{u\lambda}K + \frac{1}{2}K'\Pi K)\} \\
&+ 2 \text{ trace}\{\sum_{t=0}^{T-1}(\frac{\partial z_{t+1}}{\partial \theta} - D\frac{\partial z_t}{\partial \theta})u_t'\Omega^{-1}\} \\
&- 2 \text{ trace}\{\sum_{t=1}^{T-1}(\frac{\partial z_t}{\partial \theta} - D\frac{\partial z_{t-1}}{\partial \theta})\lambda_t'K\} - 2 \text{ trace}\{\frac{\partial \hat{x}_0}{\partial \theta}\lambda_0'\}, \qquad (A.4)
\end{aligned}
$$

where Σ is the asymptotic state covariance matrix found by iterating on Eq. (10.6) and $\bar{G}$, K, Ω, u_t and $\hat{x}_t$ are defined in Eqs. (10.3), (10.4), (10.5), and (10.7), and

$$
\begin{aligned}
\lambda_t &= (A_o - K\bar{G})'\lambda_{t+1} + \bar{G}'\Omega^{-1}u_t, \qquad t = 0,\ldots,T-2 \\
\lambda_{T-1} &= \bar{G}'\Omega^{-1}u_{T-1} \\
\Gamma_{uu} &= \frac{1}{T}\sum_{t=0}^{T-1} u_t u_t' \\
\Gamma_{\hat{x}u} &= \frac{1}{T}\sum_{t=0}^{T-1} \hat{x}_t u_t' \\
\Gamma_{zu} &= \frac{1}{T}\sum_{t=0}^{T-1} z_t u_t' \\
\Gamma_{u\lambda} &= \frac{1}{T}\sum_{t=1}^{T-1} u_{t-1}\lambda_t' \qquad (A.5)
\end{aligned}
$$

$$
\begin{aligned}
\Gamma_{\hat{x}\lambda} &= \frac{1}{T}\sum_{t=1}^{T-1} \hat{x}_{t-1}\lambda_t' && (A.6)\\
\Gamma_{z\lambda} &= \frac{1}{T}\sum_{t=1}^{T-1} z_{t-1}\lambda_t' && (A.7)\\
M &= \Omega^{-1} - \Omega^{-1}\Gamma_{uu}\Omega^{-1}\\
\bar{A}_o &= A_o - K\bar{G}\\
\Pi &= \bar{A}_o'\Pi\bar{A}_o + \bar{G}'M\bar{G} - \bar{G}'\Omega^{-1}\Gamma_{u\lambda}\bar{A}_o - \bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1}\bar{G}.
\end{aligned}
$$

In the remainder of this appendix, we derive the formulas in Eq. $(A.1)$ and Eq. $(A.4)$. Readers who are not interested in this derivation can skip the rest of this appendix.

A.2. Derivation of the Formula

The derivative of the log-likelihood function with respect to any element θ of the parameter vector is given by

$$
\begin{aligned}
\frac{\partial L}{\partial \theta} &= \sum_{t=0}^{T-1} \text{trace}\{\frac{\partial \Omega_t}{\partial \theta} M_t\} + \sum_{t=0}^{T-1} \text{trace}\{(\frac{\partial u_t}{\partial \theta} u_t' + u_t \frac{\partial u_t'}{\partial \theta})\Omega_t^{-1}\}, \\
&= S_1 \qquad\qquad\qquad\qquad + S_2
\end{aligned}
\qquad (A.8)
$$

where $M_t = \Omega_t^{-1} - \Omega_t^{-1}u_t u_t'\Omega_t^{-1}$ and $\Omega_t = Eu_t u_t'$. We start with the first term in the expression for the derivative of the log-likelihood function S_1. For this, we need the derivative of the covariance matrix Ω_t which satisfies

$$
\begin{aligned}
\frac{\partial \Omega_t}{\partial \theta} &= \frac{\partial \bar{G}}{\partial \theta}\Sigma_t \bar{G}' + \bar{G}\frac{\partial \Sigma_t}{\partial \theta}\bar{G}' + \bar{G}\Sigma_t \frac{\partial \bar{G}'}{\partial \theta} + \frac{\partial R}{\partial \theta} + \frac{\partial G}{\partial \theta}CC'G'\\
&\quad + G\frac{\partial C}{\partial \theta}C'G' + GC\frac{\partial C'}{\partial \theta}G' + GCC'\frac{\partial G'}{\partial \theta}\\
&= (\frac{\partial G}{\partial \theta}A_o + G\frac{\partial A_o}{\partial \theta} - \frac{\partial D}{\partial \theta}G - D\frac{\partial G}{\partial \theta})\Sigma_t \bar{G}' + \bar{G}\frac{\partial \Sigma_t}{\partial \theta}\bar{G}'\\
&\quad + \bar{G}\Sigma_t(A_o'\frac{\partial G'}{\partial \theta} + \frac{\partial A_o{}'}{\partial \theta}G' - G'\frac{\partial D'}{\partial \theta} - \frac{\partial G'}{\partial \theta}D')\\
&\quad + \frac{\partial R}{\partial \theta} + \frac{\partial G}{\partial \theta}CC'G' + G\frac{\partial C}{\partial \theta}C'G' + GC\frac{\partial C'}{\partial \theta}G' + GCC'\frac{\partial G'}{\partial \theta}. \qquad (A.9)
\end{aligned}
$$

The second equality follows from the definition of $\bar{G}$. If we post-multiply the derivative of Ω_t by M_t and take the trace of the result, we have the first term of the derivative of the log-likelihood function in Eq. $(A.8)$:

$$
\begin{aligned}
S_1 &= \sum_{t=0}^{T-1}\big[2\ \text{trace}(\frac{\partial A_o}{\partial \theta}\Sigma_t \bar{G}'M_t G) + 2\ \text{trace}(\frac{\partial C}{\partial \theta}C'G'M_t G)\\
&\quad + 2\ \text{trace}(\frac{\partial G}{\partial \theta}\{A_o\Sigma_t\bar{G}'M_t - \Sigma_t\bar{G}'M_t D + CC'G'M_t\})\\
&\quad - 2\ \text{trace}(\frac{\partial D}{\partial \theta}G\Sigma_t\bar{G}'M_t) + \text{trace}(\frac{\partial R}{\partial \theta}M_t)\\
&\quad + \text{trace}(\frac{\partial \Sigma_t}{\partial \theta}\bar{G}'M_t\bar{G})\big]. \qquad (A.10)
\end{aligned}
$$

Note that the formula for S_1 depends on derivatives $\frac{\partial A_o}{\partial \theta}$, $\frac{\partial C}{\partial \theta}$, $\frac{\partial G}{\partial \theta}$, $\frac{\partial D}{\partial \theta}$, and $\frac{\partial R}{\partial \theta}$, which are known, and $\frac{\partial \Sigma_t}{\partial \theta}$, which is yet to be derived.

We now turn to the second term of the log-likelihood function derivative, $S_2 = \text{trace}(\partial u_t u_t'/\partial\theta \Omega_t^{-1})$. Let $\Gamma_{uu}(t) = u_t u_t'$. By definition, $\Gamma_{uu}(t) = (\bar{z}_t - \bar{G}\hat{x}_t)(\bar{z}_t - \bar{G}\hat{x}_t)'$ and, therefore, its derivative is given by

$$\begin{aligned}
\frac{\partial \Gamma_{uu}(t)}{\partial\theta} &= (\frac{\partial \bar{z}_t}{\partial\theta} - \frac{\partial \bar{G}}{\partial\theta}\hat{x}_t - \bar{G}\frac{\partial \hat{x}_t}{\partial\theta})u_t' + u_t(\frac{\partial \bar{z}_t}{\partial\theta} - \frac{\partial \bar{G}}{\partial\theta}\hat{x}_t - \bar{G}\frac{\partial \hat{x}_t}{\partial\theta})' \\
&= (\frac{\partial z_{t+1}}{\partial\theta} - \frac{\partial D}{\partial\theta} z_t - D\frac{\partial z_t}{\partial\theta} - \frac{\partial G}{\partial\theta}A_o\hat{x}_t - G\frac{\partial A_o}{\partial\theta}\hat{x}_t \\
&+ \frac{\partial D}{\partial\theta}G\hat{x}_t + D\frac{\partial G}{\partial\theta}\hat{x}_t - \bar{G}\frac{\partial \hat{x}_t}{\partial\theta})u_t' \\
&+ u_t(\frac{\partial z_{t+1}}{\partial\theta}' - z_t'\frac{\partial D'}{\partial\theta} - \frac{\partial z_t}{\partial\theta}'D' - \hat{x}_t'A_o'\frac{\partial G'}{\partial\theta} - \hat{x}_t'\frac{\partial A_o}{\partial\theta}'G' \\
&+ \hat{x}_t'G'\frac{\partial D'}{\partial\theta} + \hat{x}_t'\frac{\partial G'}{\partial\theta}D' - \frac{\partial \hat{x}_t}{\partial\theta}'\bar{G}'). \qquad (A.11)
\end{aligned}$$

If we post-multiply this derivative by Ω_t^{-1}, take the trace of the resulting matrix, and sum over t, then we have the second term of the derivative of the log-likelihood function, i.e.,

$$\begin{aligned}
S_2 = &-\sum_{t=0}^{T-1}\big[2\ \text{trace}\{\frac{\partial A_o}{\partial\theta}\hat{x}_t u_t'\Omega_t^{-1}G\} + 2\ \text{trace}\{\frac{\partial G}{\partial\theta}(A_o\hat{x}_t u_t'\Omega_t^{-1} - \hat{x}_t u_t'\Omega_t^{-1}D)\} \\
&+ 2\ \text{trace}\{\frac{\partial D}{\partial\theta}(z_t u_t' - G\hat{x}_t u_t')\Omega_t^{-1}\} - 2\ \text{trace}\{\sum_{t=0}^{T-1}\frac{\partial z_{t+1}}{\partial\theta}u_t'\Omega_t^{-1}\} \\
&+ 2\ \text{trace}\{\sum_{t=0}^{T-1}\frac{\partial z_t}{\partial\theta}u_t'\Omega_t^{-1}D\} + 2\ \text{trace}\{\sum_{t=0}^{T-1}\frac{\partial \hat{x}_t}{\partial\theta}u_t'\Omega_t^{-1}\bar{G}\}\big]. \qquad (A.12)
\end{aligned}$$

Sum the expressions in Eqs. $(A.10)$ and $(A.12)$ to get the expression for the derivative of the log-likelihood function in $(A.1)$.

For the time-invariant case, several more steps are needed. First, we derive the last term in Eq. $(A.12)$ in terms of the derivatives that are taken as inputs. Following Kashyap (1970), Wilson and Kumar (1982), and Zadrozny (1988a), we can simplify the computations by working with sequences $\{d_t\}$ and $\{\lambda_t\}$ defined as follows

$$\begin{aligned}
d_t &= (\frac{\partial A_o}{\partial\theta} - \frac{\partial K}{\partial\theta}\bar{G} - K\frac{\partial \bar{G}}{\partial\theta})\hat{x}_t + \frac{\partial K}{\partial\theta}\bar{z}_t + K\frac{\partial \bar{z}_t}{\partial\theta}, \quad t = 0,\ldots,T-1 \\
\lambda_t &= (A_o - K\bar{G})'\lambda_{t+1} + \bar{G}'\Omega^{-1}u_t, \qquad t = 0,\ldots,T-2 \\
\lambda_{T-1} &= \bar{G}'\Omega^{-1}u_{T-1}. \qquad (A.13)
\end{aligned}$$

Notice that the time subscripts have been dropped from K and Ω since the time-invariant case assumes that $\Sigma_t = \Sigma$ for all t. Let $\bar{A}_o = A_o - K\bar{G}$. Notice that since $\hat{x}_{t+1} = \bar{A}_o\hat{x}_t + K\bar{z}_t$, its derivative is given by

$$\frac{\partial \hat{x}_{t+1}}{\partial\theta} = \bar{A}_o\frac{\partial \hat{x}_t}{\partial\theta} + d_t. \qquad (A.14)$$

Write out the last term in Eq. $(A.12)$ and substitute in $\hat{x}_t = \bar{A}_o^t + \sum_{s=0}^{t-1}\bar{A}_o^{s-1}d_{t-s}$. Then group terms involving $\hat{x}_0$ and d_t, $t = 0,\ldots T-2$. These steps lead to

$$\begin{aligned}
-\frac{2}{T}\text{trace}\ (\sum_{t=0}^{T-1}\frac{\partial \hat{x}_t}{\partial\theta}u_t'\Omega^{-1}\bar{G}) &= -\frac{2}{T}\text{trace}(\frac{\partial \hat{x}_0}{\partial\theta}\lambda_0' + \sum_{t=1}^{T-1}d_{t-1}\lambda_t') \\
&= -\frac{2}{T}\ \text{trace}(\frac{\partial \hat{x}_0}{\partial\theta}\lambda_0') - 2\ \text{trace}\{(\frac{\partial A_o}{\partial\theta} - \frac{\partial K}{\partial\theta}\bar{G} - K\frac{\partial G}{\partial\theta}A_o
\end{aligned}$$

$$
\begin{aligned}
&- KG\frac{\partial A_o}{\partial \theta} + K\frac{\partial D}{\partial \theta}G + KD\frac{\partial G}{\partial \theta})\Gamma_{\hat{x}\lambda} + \frac{\partial K}{\partial \theta}\Gamma_{\bar{z}\lambda} \\
&+ \frac{1}{T}K\sum_{t=1}^{T-1}\frac{\partial z_t}{\partial \theta}\lambda_t' - K\frac{\partial D}{\partial \theta}\Gamma_{z\lambda} - \frac{1}{T}KD\sum_{t=1}^{T-1}\frac{\partial z_{t-1}}{\partial \theta}\lambda_t'\} \\
= &- 2\ \mathrm{trace}\{\frac{\partial A_o}{\partial \theta}\Gamma_{\hat{x}\lambda}(I-KG)\} \\
&+ 2\ \mathrm{trace}\{\frac{\partial G}{\partial \theta}(A_o\Gamma_{\hat{x}\lambda}K - \Gamma_{\hat{x}\lambda}KD\} \\
&- 2\ \mathrm{trace}\{\frac{\partial D}{\partial \theta}(G\Gamma_{\hat{x}\lambda}K - \Gamma_{z\lambda}K)\} \\
&- \frac{2}{T}\ \mathrm{trace}\{K\sum_{t=1}^{T-1}\frac{\partial z_t}{\partial \theta}\lambda_t' - KD\sum_{t=1}^{T-1}\frac{\partial z_{t-1}}{\partial \theta}\lambda_t'\} \\
&- \frac{2}{T}\ \mathrm{trace}(\frac{\partial \hat{x}_0}{\partial \theta}\lambda_0') - 2\ \mathrm{trace}\{\frac{\partial K}{\partial \theta}\Gamma_{u\lambda}\}, \qquad (A.15)
\end{aligned}
$$

where $\Gamma_{u\lambda}$, $\Gamma_{\hat{x}\lambda}$, and $\Gamma_{z\lambda}$ are the sums defined in Eqs. $(A.5)$ through Eq. $(A.7)$ and $\Gamma_{\bar{z}\lambda} = \sum_{t=1}^{T-1}\bar{z}_{t-1}\lambda_t'/T$. The second equality follows from the definitions of d_{t-1} and $\bar{G}$ and some algebraic manipulation. The last term in Eq. $(A.15)$ uses the fact that $u_t = \bar{z}_t - \bar{G}\hat{x}_t$. With the exception of $\frac{\partial K}{\partial \theta}$, the expression in Eq. $(A.15)$ is a function of known derivatives. The expression for $\frac{\partial K}{\partial \theta}$ follows from the definition in Eq. (10.4) and is given by

$$
\begin{aligned}
\frac{\partial K}{\partial \theta} &= \Big[\frac{\partial C}{\partial \theta}C'G' + C\frac{\partial C'}{\partial \theta}G' + CC'\frac{\partial G'}{\partial \theta} + \frac{\partial A_o}{\partial \theta}\Sigma\bar{G}' + A_o\frac{\partial \Sigma}{\partial \theta}\bar{G}' \\
&+ A_o\Sigma A_o'\frac{\partial G'}{\partial \theta} + A_o\Sigma\frac{\partial A_o{}'}{\partial \theta}G' - A_o\Sigma G'\frac{\partial D'}{\partial \theta} - A_o\Sigma\frac{\partial G'}{\partial \theta}D'\Big]\Omega^{-1} \\
&- (CC'G' + A_o\Sigma\bar{G}')\Omega^{-1}\Big[\frac{\partial G}{\partial \theta}A_o\Sigma\bar{G}' + G\frac{\partial A_o}{\partial \theta}\Sigma\bar{G}' - \frac{\partial D}{\partial \theta}G\Sigma\bar{G}' \\
&- D\frac{\partial G}{\partial \theta}\Sigma\bar{G}' + \bar{G}\frac{\partial \Sigma}{\partial \theta}\bar{G}' + \bar{G}\Sigma A_o'\frac{\partial G'}{\partial \theta} + \bar{G}\Sigma\frac{\partial A_o{}'}{\partial \theta}G' \\
&- \bar{G}\Sigma G'\frac{\partial D'}{\partial \theta} - \bar{G}\Sigma\frac{\partial G'}{\partial \theta}D' + \frac{\partial R}{\partial \theta} + \frac{\partial G}{\partial \theta}CC'G' \\
&+ G\frac{\partial C}{\partial \theta}C'G' + GC\frac{\partial C'}{\partial \theta}G' + GCC'\frac{\partial G'}{\partial \theta}\Big]\Omega^{-1}. \qquad (A.16)
\end{aligned}
$$

Note that we have written $\frac{\partial \bar{G}}{\partial \theta}$ in terms of $\frac{\partial G}{\partial \theta}$, $\frac{\partial A_o}{\partial \theta}$, and $\frac{\partial D}{\partial \theta}$. Substituting $\frac{\partial K}{\partial \theta}$ into the expression in Eq. $(A.15)$ and rearranging terms, we have

$$
\begin{aligned}
&-\frac{2}{T}\ \mathrm{trace}(\sum_{t=0}^{T-1}\frac{\partial \hat{x}_t}{\partial \theta}u_t'\Omega^{-1}\bar{G}) = \\
&\quad - 2\ \mathrm{trace}\{\frac{\partial A_o}{\partial \theta}(\Gamma_{\hat{x}\lambda}(I-KG) + \Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}(I-KG) \\
&\qquad + \Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1}G\} \\
&\quad - 2\ \mathrm{trace}\{\frac{\partial C}{\partial \theta}C'(G'\Omega^{-1}\Gamma_{u\lambda}(I-KG) + (I-G'K')\Gamma_{u\lambda}'\Omega^{-1}G)\} \\
&\quad + 2\ \mathrm{trace}\{\frac{\partial G}{\partial \theta}(A_o\Gamma_{\hat{x}\lambda}K - \Gamma_{\hat{x}\lambda}KD - CC'(I-G'K')\Gamma_{u\lambda}'\Omega^{-1}
\end{aligned}
$$

$$
\begin{aligned}
& + CC'G'\Omega^{-1}\Gamma_{u\lambda}K - A_o\Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1} + \Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1}D \\
& + A_o\Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}K - \Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}KD)\} \\
- & 2 \text{ trace}\{\frac{\partial D}{\partial\theta}(G\Gamma_{\hat{x}\lambda}K - \Gamma_{z\lambda}K - G\Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1} + G\Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}K)\} \\
+ & 2 \text{ trace}\{\frac{\partial R}{\partial\theta}\Omega^{-1}\Gamma_{u\lambda}K\} \\
- & \frac{2}{T} \text{ trace}\{K\sum_{t=1}^{T-1}\frac{\partial z_t}{\partial\theta}\lambda_t' - KD\sum_{t=1}^{T-1}\frac{\partial z_{t-1}}{\partial\theta}\lambda_t'\} \\
- & \frac{2}{T} \text{ trace}\{\frac{\partial \hat{x}_0}{\partial\theta}\lambda_0'\} - 2 \text{ trace}\{\frac{\partial\Sigma}{\partial\theta}(\bar{G}'\Omega^{-1}\Gamma_{u\lambda}\bar{A}_o)\}.
\end{aligned} \qquad (A.17)
$$

Therefore, the expression for the second term of the log-likelihood function derivative S_2 is given by

$$
\begin{aligned}
S_2 = & -2 \text{ trace}\{\frac{\partial A_o}{\partial\theta}(\Gamma_{\hat{x}u}\Omega^{-1}G + \Gamma_{\hat{x}\lambda}(I-KG) + \Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}(I-KG) \\
& + \Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1}G\} \\
& - 2 \text{ trace}\{\frac{\partial C}{\partial\theta}C'(G'\Omega^{-1}\Gamma_{u\lambda}(I-KG) + (I-G'K')\Gamma_{u\lambda}'\Omega^{-1}G)\} \\
& - 2 \text{ trace}\{\frac{\partial G}{\partial\theta}(A_o\Gamma_{\hat{x}u}\Omega^{-1} - \Gamma_{\hat{x}u}\Omega^{-1}D - A_o\Gamma_{\hat{x}\lambda}K + \Gamma_{\hat{x}\lambda}KD \\
& + CC'(I-G'K')\Gamma_{u\lambda}'\Omega^{-1} - CC'G'\Omega^{-1}\Gamma_{u\lambda}K \\
& + A_o\Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1} - \Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1}D \\
& - A_o\Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}K + \Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}KD)\} \\
& - 2 \text{ trace}\{\frac{\partial D}{\partial\theta}((\Gamma_{zu} - G\Gamma_{\hat{x}u})\Omega^{-1} + G\Gamma_{\hat{x}\lambda}K - \Gamma_{z\lambda}K - G\Sigma\bar{A}_o'\Gamma_{u\lambda}'\Omega^{-1} \\
& + G\Sigma\bar{G}'\Omega^{-1}\Gamma_{u\lambda}K)\} \\
& + 2 \text{ trace}\{\frac{\partial R}{\partial\theta}\Omega^{-1}\Gamma_{u\lambda}K\} \\
& + \frac{2}{T} \text{ trace}\{\sum_{t=0}^{T-1}\frac{\partial z_{t+1}}{\partial\theta}u_t'\Omega^{-1}\} - \frac{2}{T} \text{ trace}\{\sum_{t=0}^{T-1}\frac{\partial z_t}{\partial\theta}u_t'\Omega^{-1}D\} \\
& - \frac{2}{T} \text{ trace}\{K\sum_{t=1}^{T-1}\frac{\partial z_t}{\partial\theta}\lambda_t' - KD\sum_{t=1}^{T-1}\frac{\partial z_{t-1}}{\partial\theta}\lambda_t'\} \\
& - \frac{2}{T} \text{ trace}\{\frac{\partial \hat{x}_0}{\partial\theta}\lambda_0'\} \\
& - 2 \text{ trace}\{\frac{\partial\Sigma}{\partial\theta}\bar{G}'\Omega^{-1}\Gamma_{u\lambda}\bar{A}_o\}.
\end{aligned} \qquad (A.18)
$$

Our expressions for S_1 in Eq. $(A.10)$ and S_2 in Eq. $(A.18)$ depend on $\frac{\partial A_o}{\partial\theta}$, $\frac{\partial C}{\partial\theta}$, $\frac{\partial G}{\partial\theta}$, $\frac{\partial D}{\partial\theta}$, $\frac{\partial R}{\partial\theta}$, which are known, and $\frac{\partial\Sigma}{\partial\theta}$, which we will now derive. Using the expression in Eq. $(A.2)$ with $\Sigma_{t+1} = \Sigma_t = \Sigma$, we get

$$
\frac{\partial\Sigma}{\partial\theta} = \bar{A}_o\frac{\partial\Sigma}{\partial\theta}\bar{A}_o' + W + W', \qquad (A.19)
$$

where

$$
\begin{aligned}
W \ = \ & \frac{\partial A_o}{\partial \theta}\Sigma A_o' + \frac{\partial C}{\partial \theta}C' - \frac{\partial C}{\partial \theta}C'G'K' - C\frac{\partial C'}{\partial \theta}G'K' - CC'\frac{\partial G'}{\partial \theta}K' \\
& - \frac{\partial A_o}{\partial \theta}\Sigma \bar{G}'K' - A_o\Sigma A_o'\frac{\partial G'}{\partial \theta}K' - A_o\Sigma\frac{\partial A_o{}'}{\partial \theta}G'K' \\
& + A_o\Sigma G'\frac{\partial D'}{\partial \theta}K' + A_o\Sigma\frac{\partial G'}{\partial \theta}D'K' + \frac{1}{2}K\frac{\partial R}{\partial \theta}K' \\
& + K\frac{\partial G}{\partial \theta}A_o\Sigma\bar{G}'K' + KG\frac{\partial A_o}{\partial \theta}\Sigma\bar{G}'K' - K\frac{\partial D}{\partial \theta}G\Sigma\bar{G}'K' \\
& - KD\frac{\partial G}{\partial \theta}\Sigma\bar{G}'K' + K\frac{\partial G}{\partial \theta}CC'G'K' + KG\frac{\partial C}{\partial \theta}C'G'K'. \qquad (A.20)
\end{aligned}
$$

The terms W and W' in Eq. $(A.19)$ include all derivatives but $\frac{\partial \Sigma}{\partial \theta}$. To get the expression in Eq. $(A.20)$, we substituted the expressions for $\frac{\partial \Omega}{\partial \theta}$ and $\frac{\partial G}{\partial \theta}$ into Eq. $(A.19)$. Let Π be a symmetric matrix that satisfies

$$
\Pi = \bar{A}_o'\Pi\bar{A}_o + \frac{1}{2}(H + H'), \qquad (A.21)
$$

where

$$
H = \bar{G}'M\bar{G} - 2\bar{G}'\Omega^{-1}\Gamma_{u\lambda}\bar{A}_o. \qquad (A.22)
$$

Then

$$
\begin{aligned}
\text{trace}(\frac{\partial \Sigma}{\partial \theta}H) \ & = \text{trace}\{\frac{\partial \Sigma}{\partial \theta}\frac{1}{2}(H + H')\} \\
& = \text{trace}\{\frac{\partial \Sigma}{\partial \theta}(\Pi - \bar{A}_o'\Pi\bar{A}_o)\} \\
& = \text{trace}\{\frac{\partial \Sigma}{\partial \theta}\Pi\} - \text{trace}\{\bar{A}_o\frac{\partial \Sigma}{\partial \theta}\bar{A}_o'\Pi\} \\
& = \text{trace}\{(\frac{\partial \Sigma}{\partial \theta} - \bar{A}_o\frac{\partial \Sigma}{\partial \theta}\bar{A}_o')\Pi\} \\
& = \text{trace}\{(W + W')\Pi\} \\
& = 2\ \text{trace}\{W\Pi\}. \qquad (A.23)
\end{aligned}
$$

If we post-multiply W by Π and take 2 times the trace, then we have an expression for trace$(\frac{\partial \Sigma}{\partial \theta}H)$ in terms of known derivatives, i.e.,

$$
\begin{aligned}
\text{trace}(\frac{\partial \Sigma}{\partial \theta}H) \ & = 2\ \text{trace}\{\frac{\partial A_o}{\partial \theta}\Sigma\bar{A}_o'\Pi(I - KG)\} \\
& + 2\ \text{trace}\{\frac{\partial C}{\partial \theta}C'(I - G'K')\Pi(I - KG)\} \\
& - 2\ \text{trace}\{\frac{\partial G}{\partial \theta}(A_o\Sigma\bar{A}_o'\Pi K - \Sigma\bar{A}_o'\Pi KD + CC'(I - G'K')\Pi K\} \\
& + 2\ \text{trace}\{\frac{\partial D}{\partial \theta}G\Sigma\bar{A}_o'\Pi K\} + \text{trace}\{\frac{\partial R}{\partial \theta}K'\Pi K\}. \qquad (A.24)
\end{aligned}
$$

Sum S_1, which appears in Eq. $(A.10)$ with $\Sigma_t = \Sigma$ and $\Omega_t = \Omega$, and S_2 in $(A.18)$. Substitute in the expression for trace$(\frac{\partial \Sigma}{\partial \theta}H)$ from Eq. $(A.24)$. The result is the derivative of the log-likelihood function which is given in Eq. $(A.4)$.

A.3. Standard Errors

After we have computed parameter estimates, we want to compute their standard errors as given in Eq. (11.3). For this, we need to compute the derivative of

$$L_t(\Theta) = \log|\Omega_t| + u_t'\Omega_t^{-1}u_t$$

with respect to any element θ of the parameter vector.[37] This derivative is given by

$$\begin{aligned}
\frac{\partial L_t}{\partial \theta} &= \mathrm{trace}(\Omega_t^{-1}\frac{\partial \Omega_t}{\partial \theta}) + \frac{\partial u_t}{\partial \theta}' \Omega_t^{-1}u_t + u_t'\Omega_t^{-1}\frac{\partial u_t}{\partial \theta} - u_t'\Omega_t^{-1}\frac{\partial \Omega_t}{\partial \theta}\Omega_t^{-1}u_t \\
&= \mathrm{trace}\{(\Omega_t^{-1} - \Omega_t^{-1}u_tu_t'\Omega_t^{-1})\frac{\partial \Omega_t}{\partial \theta}\} + \mathrm{trace}\{\frac{\partial u_t}{\partial \theta}' \Omega_t^{-1}u_t + u_t'\Omega_t^{-1}\frac{\partial u_t}{\partial \theta}\} \\
&= \mathrm{trace}\{\frac{\partial \Omega_t}{\partial \theta}M_t\} + \mathrm{trace}\{\Omega_t^{-1}\frac{\partial (u_tu_t')}{\partial \theta}\}, \qquad (A.25)
\end{aligned}$$

where $M_t = \Omega_t^{-1} - \Omega_t^{-1}u_tu_t'\Omega_t^{-1}$. Above, we calculated $\frac{\partial \Omega_t}{\partial \theta}$ and $\frac{\partial (u_tu_t')}{\partial \theta}$. These expressions are given in Eq. $(A.9)$ and Eq. $(A.11)$.

[37] Note that we are again ignoring the Jacobian since the relationship between z and y differs for each problem.

B. Differentiating the State-Space Model with Respect to Economic Parameters

In this appendix, we describe how to compute derivatives of A_o with respect to the free parameters of an economic model. We do this for four economies: a linear-quadratic economy without distortions; a nonlinear economy without distortions; a linear-quadratic economy with distortions; and a nonlinear economy with distortions. Because we use linear approximations for the nonlinear economies, most of the work is in deriving the formulas for the linear-quadratic economies.

B.1. A Linear-Quadratic Economy without Distortions

We consider a *discounted stochastic regulator problem*. The optimization problem is

$$\max_{\{u_t\}} E_0 \sum_{t=0}^{\infty} \beta^t (x_t' Q x_t + u_t' R u_t + 2x_t' W u_t), \tag{B.1}$$
$$\text{subject to } x_{t+1} = Ax_t + Bu_t + Cw_{t+1}.$$

We assume that the matrices Q, R, W, A, and B depend on a vector of parameters Θ. For the remainder of this section we assume that $C = 0$. Typically, the number of elements in Θ is small relative to the combined number of elements in these matrices. We also assume that the derivatives of the matrices in Eq. $(B.1)$ with respect to the elements of Θ are known.

The optimal decision function is given by $u_t = -Fx_t$, where

$$F = (R + \beta B'PB)^{-1}(\beta B'PA + W') \tag{B.2}$$

for P satisfying

$$P = Q + \beta A'PA - (W + \beta A'PB)(R + \beta B'PB)^{-1}(\beta B'PA + W'). \tag{B.3}$$

The law of motion for x in equilibrium is

$$x_{t+1} = A_o x_t, \quad A_o = A - BF. \tag{B.4}$$

Therefore, the derivative of A_o with respect to an element of Θ is

$$\frac{\partial A_o}{\partial \theta} = \frac{\partial A}{\partial \theta} - \frac{\partial B}{\partial \theta}F - B\frac{\partial F}{\partial \theta}. \tag{B.5}$$

The derivatives $\frac{\partial A}{\partial \theta}$ and $\frac{\partial B}{\partial \theta}$ depend on the specification of the problem in Eq. $(B.1)$ and are assumed to be known. The derivative of F is

$$\begin{aligned}\frac{\partial F}{\partial \theta} &= -(R + \beta B'PB)^{-1}\Big(\frac{\partial R}{\partial \theta} + \beta\frac{\partial B'}{\partial \theta}PB + \beta B'\frac{\partial P}{\partial \theta}B + \beta B'P\frac{\partial B}{\partial \theta}\Big)F \\ &\quad + (R + \beta B'PB)^{-1}\Big(\beta\frac{\partial B'}{\partial \theta}PA + \beta B'\frac{\partial P}{\partial \theta}A + \beta B'P\frac{\partial A}{\partial \theta} + \frac{\partial W'}{\partial \theta}\Big).\end{aligned} \tag{B.6}$$

Notice that this formula depends on the derivative of P, with the remaining derivatives provided by the modeler. The derivative $\frac{\partial P}{\partial \theta}$ satisfies the following equation:

$$\begin{aligned}\frac{\partial P}{\partial \theta} &= \frac{\partial Q}{\partial \theta} + \beta\frac{\partial A'}{\partial \theta}PA + \beta A'\frac{\partial P}{\partial \theta}A + \beta A'P\frac{\partial A}{\partial \theta} - \Big(\frac{\partial W}{\partial \theta} + \beta\frac{\partial A'}{\partial \theta}PB \\ &\quad + \beta A'\frac{\partial P}{\partial \theta}B + \beta A'P\frac{\partial B}{\partial \theta}\Big)F + F'\Big(\frac{\partial R}{\partial \theta} + \beta\frac{\partial B'}{\partial \theta}PB + \beta B'\frac{\partial P}{\partial \theta}B \\ &\quad + \beta B'P\frac{\partial B}{\partial \theta}\Big)F - F'\Big(\beta\frac{\partial B'}{\partial \theta}PA + \beta B'\frac{\partial P}{\partial \theta}A + \beta B'P\frac{\partial A}{\partial \theta} + \frac{\partial W'}{\partial \theta}\Big) \\ &= \beta A_o'\frac{\partial P}{\partial \theta}A_o + \frac{\partial Q}{\partial \theta} + \beta\Big[\frac{\partial A'}{\partial \theta} - F'\frac{\partial B'}{\partial \theta}\Big]PA_o + \beta A_o'P\Big[\frac{\partial A}{\partial \theta} - \frac{\partial B}{\partial \theta}F\Big] \\ &\quad - \frac{\partial W}{\partial \theta}F - F'\frac{\partial W'}{\partial \theta} + F'\frac{\partial R}{\partial \theta}F.\end{aligned} \tag{B.7}$$

Although this formula determines only an implicit function for $\frac{\partial P}{\partial \theta}$, the gradient of P can be represented explicitly in terms of things we know. Define the gradient operator as follows: for any matrix A that depends on the parameter θ, $\nabla_\theta A = vec(\frac{\partial A}{\partial \theta})$. Then,

$$\begin{aligned}\nabla_\theta P &= (I - \beta A_o' \otimes A_o')^{-1}\{\nabla_\theta Q + \beta(A_o' P \otimes I)\nabla_\theta A' + \beta(I \otimes A_o' P)\nabla_\theta A \\ &\quad - \beta(A_o' P \otimes F')\nabla_\theta B' - \beta(F' \otimes A_o' P)\nabla_\theta B - (F' \otimes I)\nabla_\theta W \\ &\quad - (I \otimes F')\nabla_\theta W' + (F' \otimes F')\nabla_\theta R\}, \qquad (B.8)\end{aligned}$$

which is a function of the gradients of A, B, Q, R, and W. The gradient of P can then be substituted into the following formula for $\nabla_\theta F$:

$$\begin{aligned}\nabla_\theta F &= \beta(I \otimes \mathcal{R}B'P)\nabla_\theta A - \beta(F' \otimes \mathcal{R}B'P)\nabla_\theta B + \beta(A_o' P \otimes \mathcal{R})\nabla_\theta B' \\ &\quad - (F' \otimes \mathcal{R})\nabla_\theta R + (I \otimes \mathcal{R})\nabla_\theta W' + \beta(A_o' \otimes \mathcal{R}B')\nabla_\theta P, \qquad (B.9)\end{aligned}$$

where $\mathcal{R} = (R + \beta B'PB)^{-1}$. Finally, we substitute this expression for $\nabla_\theta F$ into

$$\nabla_\theta A_o = \nabla_\theta A - (F' \otimes I)\nabla_\theta B - (I \otimes B)\nabla_\theta F. \qquad (B.10)$$

B.2. A Nonlinear Economy without Distortions

The optimization problem that we start with is

$$\begin{aligned}&\max_{\{u_t\}} E_0 \sum_{t=0}^{\infty} \beta^t \; r(z_t, \theta), \qquad (B.11)\\ &\text{subject to } \; x_{t+1} = Ax_t + Bu_t + Cw_{t+1} \\ &\qquad\qquad\quad z_t = [x_t', u_t']',\end{aligned}$$

where $\{w_{t+1}\}$ is a martingale difference sequence and E_0 is the mathematical expectation conditioned on time 0 information. We solve a related problem, namely:

$$\begin{aligned}&\max_{\{u_t\}} E_0 \sum_{t=0}^{\infty} \beta^t z_t' M z_t \qquad (B.12)\\ &x_{t+1} = Ax_t + Bu_t,\end{aligned}$$

where

$$\begin{aligned}M &= e(r(\bar z, \theta) - \frac{\partial r(\bar z, \theta)'}{\partial \bar z}\bar z + \frac{1}{2}\bar z' \frac{\partial^2 r(\bar z, \theta)}{\partial \bar z^2}\bar z)e' + \frac{1}{2}(e\frac{\partial r(\bar z, \theta)'}{\partial \bar z} \\ &\quad + \frac{\partial r(\bar z, \theta)}{\partial \bar z}e' - e\bar z' \frac{\partial^2 r(\bar z, \theta)}{\partial \bar z^2} - \frac{\partial^2 r(\bar z, \theta)}{\partial \bar z^2}\bar z e' + \frac{\partial^2 r(\bar z, \theta)}{\partial \bar z^2}), \qquad (B.13)\end{aligned}$$

and where e is a vector of zeros except for a 1 in the element corresponding to the constant term in x_t, $\bar z$ and $\bar w$ are the steady state values of z_t and w_t, and $S_x = [I_n; 0_{k,n}]$ and $S_u = [0_{n,k}; I_k]$ (where the ";" denotes stacking) are selector matrices and imply $z_t = S_x x_t + S_u u_t$, where n is the dimension of x_t and k is the dimension of u_t. The latter problem yields the same decision function as that of Eq. $(B.1)$ (where $Q = S_x' M S_x$, $R = S_u' M S_u$, and $W = S_x' M S_u$).

In the nonlinear case, however, the derivatives are slightly more complicated. To derive $\frac{\partial A_o}{\partial \theta}$, we need to calculate derivatives of the coefficient matrices of the objective function. For this, we need the derivative of M with respect to θ:

$$\begin{aligned}\frac{\partial M}{\partial \theta} &= e\Big(\frac{\partial r(\bar z,\theta)}{\partial \theta} - \frac{\partial^2 r(\bar z,\theta)'}{\partial \bar z \partial \theta}\,\bar z + \frac{1}{2}\bar z'\Big(\nabla_{\bar z}\frac{\partial^2 r(\bar z,\theta)}{\partial \bar z^2}\frac{\partial \bar z}{\partial \theta}\Big)(:)\,\bar z \\ &+ \frac{1}{2}\bar z'\frac{\partial^3 r(\bar z,\theta)}{\partial \bar z^2 \partial \theta}\bar z\Big)e' + \frac{1}{2}\Big(e\frac{\partial^2 r(\bar z,\theta)'}{\partial \bar z \partial \theta} + \frac{\partial^2 r(\bar z,\theta)}{\partial \bar z \partial \theta}e' \\ &- e\bar z'\Big(\nabla_{\bar z}\frac{\partial^2 r(\bar z,\theta)}{\partial \bar z^2}\frac{\partial \bar z}{\partial \theta}\Big)(:) - \Big(\nabla_{\bar z}\frac{\partial^2 r(\bar z,\theta)}{\partial \bar z^2}\frac{\partial \bar z}{\partial \theta}\Big)(:)\,\bar z e' \\ &- e\bar z'\frac{\partial^3 r(\bar z,\theta)}{\partial \bar z^2 \partial \theta} - \frac{\partial^3 r(\bar z,\theta)}{\partial \bar z^2 \partial \theta}\bar z e' + \frac{\partial^3 r(\bar z,\theta)}{\partial \bar z^2 \partial \theta} \\ &+ \Big(\nabla_{\bar z}\frac{\partial^2 r(\bar z,\theta)}{\partial \bar z^2}\frac{\partial \bar z}{\partial \theta}\Big)(:)\Big),\end{aligned} \qquad (B.14)$$

where $\nabla_z A(z) = [\frac{\partial}{\partial z_1}A(z), \ldots, \frac{\partial}{\partial z_n}A(z)]$ for $A(z)$ which is $n \times n$ and $b(:)$ is an $n \times n$ matrix created from a vector of length n^2 by stacking the first n elements of b into column 1, the next n elements of b into column 2, etc. As this formula indicates, the modeler must provide first, second, and third-order derivatives of the return function. The derivatives of Q, R, and W follow immediately from $\frac{\partial M}{\partial \theta}$, e.g., $\frac{\partial Q}{\partial \theta} = S_x' \frac{\partial M}{\partial \theta} S_x$. The remaining derivations are the same as in the linear-quadratic case.

B.3. A Linear-Quadratic Economy with Distortions

The optimization problem that we start with is given by

$$\max_{\{\bar u_t\}} E_0 \sum_{t=0}^{\infty} \beta^t \Big\{ \begin{bmatrix} \bar y_t \\ \bar z_t \end{bmatrix}' \begin{bmatrix} \bar Q_y & \bar Q_z \\ \bar Q_z' & \bar Q_{22} \end{bmatrix} \begin{bmatrix} \bar y_t \\ \bar z_t \end{bmatrix} + \bar u_t' \bar R \bar u_t + 2 \begin{bmatrix} \bar y_t \\ \bar z_t \end{bmatrix}' \begin{bmatrix} \bar W_y \\ \bar W_z \end{bmatrix} \bar u_t \Big\}, \qquad (B.15)$$

subject to

$$\bar y_{t+1} = \bar A_y \bar y_t + \bar A_z \bar z_t + \bar B_y \bar u_t + C \bar w_{t+1}.$$

Equilibrium conditions are imposed in the form of a set of linear equations

$$\bar z_t = \bar\Theta \bar y_t + \bar\Psi \bar u_t.$$

In the notation of this subsection (which differs from that used in section 7 in the text), $\bar y_t$ denotes the endogenous state variables affected by the representative agent, and $\bar z_t$ denotes variables that the agent takes as beyond its control. To ease notation, we convert the problem to one without cross-products or discounting. Let

$$\begin{aligned} y_t &= \beta^{t/2}\bar y_t \\ z_t &= \beta^{t/2}\bar z_t \\ u_t &= \beta^{t/2}\bar u_t \\ w_t &= \beta^{t/2}\bar w_t \\ R &= \bar R \\ Q_y &= \bar Q_y - \bar W_y \bar R^{-1} \bar W_y' \\ Q_z &= \bar Q_z - \bar W_y \bar R^{-1} \bar W_z' \\ Q_{22} &= \bar Q_{22} - \bar W_z \bar R^{-1} \bar W_z' \end{aligned}$$

$$\begin{aligned}
A_y &= \sqrt{\beta}(\bar{A}_y - \bar{B}_y \bar{R}^{-1} \bar{W}_y')\\
A_z &= \sqrt{\beta}(\bar{A}_z - \bar{B}_y \bar{R}^{-1} \bar{W}_z')\\
B_y &= \sqrt{\beta} \bar{B}_y\\
\Theta &= (I + \bar{\Psi} \bar{R}^{-1} \bar{W}_z')^{-1} (\bar{\Theta} - \bar{\Psi} \bar{R}^{-1} \bar{W}_y')\\
\Psi &= (I + \bar{\Psi} \bar{R}^{-1} \bar{W}_z')^{-1} \bar{\Psi}.
\end{aligned} \tag{B.16}$$

With these definitions, we can restate the optimization problem as follows

$$\max_{\{u_t\}} \sum_{t=0}^{\infty} \{ \begin{bmatrix} y_t \\ z_t \end{bmatrix}' \begin{bmatrix} Q_y & Q_z \\ Q_z' & Q_{22} \end{bmatrix} \begin{bmatrix} y_t \\ z_t \end{bmatrix}' + u_t' R u_t \}, \tag{B.17}$$

subject to

$$y_{t+1} = A_y y_t + A_z z_t + B_y u_t.$$

Let $\hat{A} = A_y + A_z \Theta$, $\hat{Q} = Q_y + Q_z \Theta$, $\hat{B} = B_y + A_z \Psi$, and $\tilde{A} = A_y - B_y R^{-1} \Psi' Q_z'$. The decision function in this case is given by

$$F = \left(R + B_y' P \hat{B}\right)^{-1} B_y' P \hat{A}, \tag{B.18}$$

where P satisfies

$$P = \hat{Q} + \tilde{A}' P \hat{A} - \tilde{A}' P \hat{B} (R + B_y' P \hat{B})^{-1} B_y' P \hat{A}. \tag{B.19}$$

The decision function for the original problem is given by

$$\bar{F} = (\bar{R} + \bar{W}_z' \bar{\Psi})^{-1} (\bar{R} F + \bar{W}_y' + \bar{W}_z' \bar{\Theta}), \tag{B.20}$$

and the equilibrium law of motion for $\bar{y}_t$ is

$$\bar{y}_{t+1} = A_o \bar{y}_t, \quad A_o = \bar{A}_y + \bar{A}_z \bar{\Theta} - \bar{A}_z \bar{\Psi} \bar{F} - \bar{B}_y \bar{F} = \beta^{-\frac{1}{2}} (\hat{A} - \hat{B} F). \tag{B.21}$$

Therefore, the derivative of A_o with respect to a parameter θ is given by

$$\frac{\partial A_o}{\partial \theta} = \beta^{-\frac{1}{2}} (\frac{\partial \hat{A}}{\partial \theta} - \frac{\partial \hat{B}}{\partial \theta} F - \hat{B} \frac{\partial F}{\partial \theta}). \tag{B.22}$$

To calculate $\frac{\partial A_o}{\partial \theta}$ requires several steps. First, we need the derivatives of $\hat{A}$, $\hat{B}$, and F with respect to θ:

$$\frac{\partial \hat{A}}{\partial \theta} = \frac{\partial A_y}{\partial \theta} + \frac{\partial A_z}{\partial \theta} \Theta + A_z \frac{\partial \Theta}{\partial \theta} \tag{B.23}$$

$$\frac{\partial \hat{B}}{\partial \theta} = \frac{\partial B_y}{\partial \theta} + \frac{\partial A_z}{\partial \theta} \Psi + A_z \frac{\partial \Psi}{\partial \theta} \tag{B.24}$$

$$\begin{aligned}
\frac{\partial F}{\partial \theta} &= -\left(R + B_y' P B_y + B_y' P A_z \Psi\right)^{-1} (\frac{\partial R}{\partial \theta} + \frac{\partial B_y}{\partial \theta}' P B_y\\
&\quad + B_y' \frac{\partial P}{\partial \theta} B_y + B_y' P \frac{\partial B_y}{\partial \theta} + \frac{\partial B_y}{\partial \theta}' P A_z \Psi\\
&\quad + B_y' \frac{\partial P}{\partial \theta} A_z \Psi + B_y' P \frac{\partial A_z}{\partial \theta} \Psi + B_y' P A_z \frac{\partial \Psi}{\partial \theta}) F\\
&\quad + \left(R + B_y' P B_y + B_y' P A_z \Psi\right)^{-1} (\frac{\partial B_y}{\partial \theta}' P A_y + B_y' \frac{\partial P}{\partial \theta} A_y + B_y' P \frac{\partial A_y}{\partial \theta}\\
&\quad + \frac{\partial B_y}{\partial \theta}' P A_z \Theta + B_y' \frac{\partial P}{\partial \theta} A_z \Theta + B_y' P \frac{\partial A_z}{\partial \theta} \Theta + B_y' P A_z \frac{\partial \Theta}{\partial \theta})\\
&= \left(R + B_y' P \hat{B}\right)^{-1} (-\frac{\partial R}{\partial \theta} F + \frac{\partial B_y}{\partial \theta}' P (\hat{A} - \hat{B} F)\\
&\quad + B_y' \frac{\partial P}{\partial \theta} (\hat{A} - \hat{B} F) + B_y' P (\frac{\partial A_y}{\partial \theta} - \frac{\partial B_y}{\partial \theta} F)\\
&\quad + B_y' P \frac{\partial A_z}{\partial \theta} (\Theta - \Psi F) + B_y' P A_z (\frac{\partial \Theta}{\partial \theta} - \frac{\partial \Psi}{\partial \theta} F)).
\end{aligned} \tag{B.25}$$

Note that these derivatives are functions of of $\frac{\partial R}{\partial\theta}$, $\frac{\partial B_y}{\partial\theta}$, $\frac{\partial A_y}{\partial\theta}$, $\frac{\partial A_z}{\partial\theta}$, $\frac{\partial\Theta}{\partial\theta}$, $\frac{\partial\Psi}{\partial\theta}$, and $\frac{\partial P}{\partial\theta}$. The derivative of R is given since $R = \bar{R}$. The derivatives for B_y, A_y, A_z, Θ, and Ψ follow from their definitions above, e.g.,

$$\begin{aligned}
\frac{\partial B_y}{\partial\theta} &= \sqrt{\beta}\frac{\partial\bar{B}_y}{\partial\theta}, && (B.26)\\
\frac{\partial A_y}{\partial\theta} &= \sqrt{\beta}(\frac{\partial\bar{A}_y}{\partial\theta} - \frac{\partial\bar{B}_y}{\partial\theta}\bar{R}^{-1}\bar{W}_y' + \bar{B}_y\bar{R}^{-1}\frac{\partial\bar{R}}{\partial\theta}\bar{R}^{-1}\bar{W}_y' - \bar{B}_y\bar{R}^{-1}\frac{\partial\bar{W}_y}{\partial\theta}'), && (B.27)\\
\frac{\partial A_z}{\partial\theta} &= \sqrt{\beta}(\frac{\partial\bar{A}_z}{\partial\theta} - \frac{\partial\bar{B}_y}{\partial\theta}\bar{R}^{-1}\bar{W}_z' + \bar{B}_y\bar{R}^{-1}\frac{\partial\bar{R}}{\partial\theta}\bar{R}^{-1}\bar{W}_z' - \bar{B}_y\bar{R}^{-1}\frac{\partial\bar{W}_z}{\partial\theta}'), && (B.28)\\
\frac{\partial\Theta}{\partial\theta} &= -(I + \bar{\Psi}\bar{R}^{-1}\bar{W}_z')^{-1}(\frac{\partial\bar{\Psi}}{\partial\theta}\bar{R}^{-1}\bar{W}_z'\Theta - \bar{\Psi}\bar{R}^{-1}\frac{\partial\bar{R}}{\partial\theta}\bar{R}^{-1}\bar{W}_z'\Theta\\
&\quad + \bar{\Psi}\bar{R}^{-1}\frac{\partial\bar{W}_z}{\partial\theta}'\Theta - \frac{\partial\bar{\Theta}}{\partial\theta} + \frac{\partial\bar{\Psi}}{\partial\theta}\bar{R}^{-1}\bar{W}_z' - \bar{\Psi}\bar{R}^{-1}\frac{\partial\bar{R}}{\partial\theta}\bar{R}^{-1}\bar{W}_z'\\
&\quad + \bar{\Psi}\bar{R}^{-1}\frac{\partial\bar{W}_z}{\partial\theta}'), && (B.29)\\
\frac{\partial\Psi}{\partial\theta} &= -(I + \bar{\Psi}\bar{R}^{-1}\bar{W}_z')^{-1}(\frac{\partial\bar{\Psi}}{\partial\theta}\bar{R}^{-1}\bar{W}_z'\Psi - \bar{\Psi}\bar{R}^{-1}\frac{\partial\bar{R}}{\partial\theta}\bar{R}^{-1}\bar{W}_z'\Psi\\
&\quad + \bar{\Psi}\bar{R}^{-1}\frac{\partial\bar{W}_z}{\partial\theta}'\Psi - \frac{\partial\bar{\Psi}}{\partial\theta}). && (B.30)
\end{aligned}$$

The derivative for P is given by

$$\begin{aligned}
\frac{\partial P}{\partial\theta} &= \sqrt{\beta}\tilde{A}_o'\frac{\partial P}{\partial\theta}A_o + \frac{\partial\hat{Q}}{\partial\theta} + \sqrt{\beta}[\frac{\partial\tilde{A}}{\partial\theta}' - \tilde{F}'\frac{\partial B_y}{\partial\theta}]PA_o\\
&\quad + \tilde{A}_o'P[\frac{\partial\hat{A}}{\partial\theta} - \frac{\partial\hat{B}}{\partial\theta}F] + \tilde{F}'\frac{\partial R}{\partial\theta}F, && (B.31)
\end{aligned}$$

where $\tilde{F} = (R + B_y'P\hat{B})^{-1}\hat{B}P'\tilde{A}$, $\tilde{A}_o = \tilde{A} - B_y\tilde{F}$, and

$$\begin{aligned}
\frac{\partial\hat{Q}}{\partial\theta} &= \frac{\partial Q_y}{\partial\theta} + \frac{\partial Q_z}{\partial\theta}\Theta + Q_z\frac{\partial\Theta}{\partial\theta} && (B.32)\\
\frac{\partial\tilde{A}}{\partial\theta} &= \frac{\partial A_y}{\partial\theta} - \frac{\partial B_y}{\partial\theta}R^{-1}\Psi'Q_z' + B_yR^{-1}\frac{\partial R}{\partial\theta}R^{-1}\Psi'Q_z'\\
&\quad - B_yR^{-1}\frac{\partial\Psi}{\partial\theta}'Q_z' - B_yR^{-1}\Psi'\frac{\partial Q_z}{\partial\theta}'. && (B.33)
\end{aligned}$$

The last two derivatives needed are $\frac{\partial Q_y}{\partial\theta}$ and $\frac{\partial Q_z}{\partial\theta}$:

$$\begin{aligned}
\frac{\partial Q_y}{\partial\theta} &= \frac{\partial\bar{Q}_y}{\partial\theta} - \frac{\partial\bar{W}_y}{\partial\theta}\bar{R}^{-1}\bar{W}_y' + \bar{W}_y\bar{R}^{-1}\frac{\partial\bar{R}}{\partial\theta}\bar{R}^{-1}\bar{W}_y' - \bar{W}_y\bar{R}^{-1}\frac{\partial\bar{W}_y}{\partial\theta}', && (B.34)\\
\frac{\partial Q_z}{\partial\theta} &= \frac{\partial\bar{Q}_z}{\partial\theta} - \frac{\partial\bar{W}_y}{\partial\theta}\bar{R}^{-1}\bar{W}_z' + \bar{W}_y\bar{R}^{-1}\frac{\partial\bar{R}}{\partial\theta}\bar{R}^{-1}\bar{W}_z' - \bar{W}_y\bar{R}^{-1}\frac{\partial\bar{W}_z}{\partial\theta}'. && (B.35)
\end{aligned}$$

We now have everything that we need to compute the derivatives of the matrices in the decision rule and the law of motion for the state vector. To avoid iterating on Eq. $(B.31)$ for $\frac{\partial P}{\partial\theta}$, we instead take the gradient, e.g.,

$$\begin{aligned}
\nabla_\theta P &= (I - \sqrt{\beta}A_o' \otimes \tilde{A}_o')^{-1}\{\nabla_\theta\hat{Q} + (I \otimes \tilde{A}_o'P')\nabla_\theta\hat{A}\\
&\quad + \sqrt{\beta}(A_o'P' \otimes I)\nabla_\theta\tilde{A}' - (F' \otimes \tilde{A}_o'P')\nabla_\theta\hat{B}\\
&\quad - \sqrt{\beta}(A_o'P' \otimes \tilde{F}')\nabla_\theta B_y' + (F' \otimes \tilde{F}')\nabla_\theta R. && (B.36)
\end{aligned}$$

Thus the gradient of F is given by

$$\begin{aligned}
\nabla_\theta F \ &= (I \otimes \mathcal{R} B_y' P)\, \nabla_\theta A_y + \big((\Theta - \Psi F') \otimes \mathcal{R} B_y' P\big)\, \nabla_\theta A_z \\
&\quad - (F' \otimes \mathcal{R} B_y' P)\, \nabla_\theta B_y + \sqrt{\beta}(A_o' P' \otimes \mathcal{R})\, \nabla_\theta B_y' \\
&\quad + \sqrt{\beta}(A_o' \otimes \mathcal{R} B_y')\, \nabla_\theta P - (F' \otimes \mathcal{R}')\, \nabla_\theta R \\
&\quad + (I \otimes \mathcal{R} B_y' P A_z)\, \nabla_\theta \Theta - (F' \otimes \mathcal{R} B_y' P A_z)\, \nabla_\theta \Psi, \qquad (B.37)
\end{aligned}$$

where $\mathcal{R} = (R + B_y' P \hat{B})^{-1}$. In terms of the computer code, we start with Eqs. $(B.26)$-$(B.30)$ and Eqs. $(B.34)$-$(B.35)$, which relate the derivatives of the original problem to those of the problem without discounting or cross-product terms. To compute the gradients of these objects in terms of our inputs, we use the fact that $vec(ABC) = (C' \otimes A)vec(B)$ for any matrices A, B, and C with the appropriate dimensions such that ABC exists. We next compute the derivatives for $\hat{A}$, $\hat{B}$, $\hat{Q}$, and $\tilde{A}$ which appear in Eqs. $(B.23)$, $(B.24)$, $(B.32)$, and $(B.33)$. Finally, we compute $\nabla_\theta P$ in Eq. $(B.36)$, $\nabla_\theta F$ in Eq. $(B.37)$, and

$$\nabla_\theta A_o = \beta^{-\frac{1}{2}}(\nabla_\theta \hat{A} - (F' \otimes I)\, \nabla_\theta \hat{B} - (I \otimes \hat{B})\, \nabla_\theta F).$$

15. References

Anderson, B.D.O (1978). 'Second-order Convergent Algorithms for the Steady-state Riccati Equation'. *International Journal of Control*, Vol. 28, pp. 295–306.

Anderson, B.D.O. and J.B. Moore (1979). *Optimal Filtering*. Englewood Cliffs: Prentice Hall.

Anderson, E.W. (1995). 'Computing Equilibria in Linear-Quadratic Dynamic Games and Models with Distortions'. Mimeo. University of Chicago.

Ansley, C.F. and R. Kohn (1985). 'Estimation, Filtering, and Smoothing in State-Space Models with Incompletely Specified Initial Conditions'. *Annals of Statistics*, Vol. 13, pp. 1286–1316.

Bai, Z. and J.W. Demmel (1993). 'On Swapping Diagonal Blocks in Real Schur Form'. *Linear Algebra and its Applications*, Vol. 186, pp. 73–95.

Bartels, R.H. and G.W. Stewart (1972). 'Algorithm 432 Solution of the Matrix Equation $AX + XB = C$'. *Communications of the ACM*, Vol. 15, pp. 820–826.

Becker, G.S. and K.M. Murphy (1988). 'A Theory of Rational Addiction'. *Journal of Political Economy*, Vol. 96, pp. 675–700.

Bierman, G.J. (1984). *Computational Aspects of the Matrix Sign Function Solution to the ARE*. Proceedings 23rd IEEE Conference on Decision Control,. pp. 514–519.

Byers, R. (1987). 'Solving the Algebraic Riccati Equation with the Matrix Sign Function'. *Linear Algebra and its Applications*, Vol. 85, pp. 267–279.

Caines, P.E. (1988). *Linear Stochastic Systems*. New York: John Wiley & Sons, Inc.

Caines, P.E. and D.Q. Mayne (1970). 'On the Discrete Time Matrix Riccati Equation of Optimal Control'. *International Journal of Control*, Vol. 12:5, pp. 785–794.

Caines, P.E. and D.Q. Mayne (1971). 'Correspondence: "On the Discrete Time Matrix Riccati Equation of Optimal Control-a Correction"'. *International Journal of Control*, Vol. 14:1, pp. 205–207.

Chan, S.W., G.C. Goodwin, and K.S. Sin (1984). 'Convergence Properties of the Riccati Difference Equation in Optimal Filtering of Nonstabilizable Systems'. *IEEE Transactions on Automatic Control*, Vol. AC-29:2, pp. 110-118.

Denman, E.D. and A.N. Beavers (1976). 'The Matrix Sign Function and Computations in Systems'. *Applications of Mathematical Computations*, Vol. 2, pp. 63–94.

Doan, T., R. Litterman, and C. Sims (1984). 'Forecasting and Conditional Projection Using Realistic Prior Distributions'. *Econometric Reviews*, Vol. 3, No. 1, pp. 1–100.

Economic and Statistics Administration, and Bureau of the Census (1993). *The American Almanac 1993-1994* . Austin: The Reference Press.

Flavin, M.A. (1981). 'The Adjustment of Consumption to Changing Expectations About Future Income'. *Journal of Political Economy*, Vol. 89, 975–1009.

Fletcher, R. (1987). *Practical Methods of Optimization*. New York: John Wiley & Sons.

Gardiner, J.D. and A.J. Laub (1986). 'A Generalization of the Matrix-Sign-Function Solution for Algebraic Riccati Equations'. *International Journal of Control*, Vol. 44, pp. 823–832.

Gardiner, J.D., M.R. Wette, A.J. Laub, J.J Amato, and C.B. Moler (1992). 'A FORTRAN-77 Software Package for Solving the Sylvester Matrix Equation $AXB^T + CXD^T = E$'. *ACM Transactions on Mathematical Software*, Vol. 18, pp. 232–238.

Golub, G.H., S. Nash, and C. Van Loan (1979). 'A Hessenberg-Schur Method for the Matrix Problem $AX + XB = C$'. *IEEE Transactions on Automatic Control*, AC-24, pp. 909–913.

Golub, G.H and C. Van Loan (1989). *Matrix Computations*. Baltimore: Johns Hopkins University Press.

Golub, G.H. and J.H. Wilkinson (1976). 'Ill-conditioned Eigensystems and the Computation of the Jordan Canonical Form'. *SIAM Review*, Vol. 18, pp. 578–619.

Gudmundsson, T., C. Kenney, and A.J. Laub (1992). 'Scaling of the Discrete-Time Algebraic Riccati Equation to Enhance Stability of the Schur Method'. *IEEE Transactions on Automatic Control*, Vol. 37, pp. 513–518.

Hall, R.E. (1978). 'Stochastic Implications of the Life Cycle-Permanent Income Hypothesis: Theory and Evidence'. *Journal of Political Economy*, Vol. 86, 971–987.

Hamilton, James D. (1994). *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Hammarling, S.J. (1982). 'Numerical Solution of the Stable Nonnegative Lyapunov Equation'. *IMA Journal of Numerical Analysis*, Vol. 2, pp. 303–323.

Hansen, L.P. (1987). 'Calculating Asset Prices in Three Exchange Economies'. *Advances in Econometrics, Fifth World Congress*, Cambridge: Cambridge University Press.

Hansen, L.P., J. Heaton, and T.J. Sargent (1991). 'Faster Methods for Solving Continuous Time Recursive Linear Models of Dynamic Economies'. *Rational Expectations Econometrics*, pp. 177–208.

Hansen, L.P. and T.J. Sargent (1994). 'Recursive Linear Models of Dynamic Economies'. Mimeo. University of Chicago.

Heaton, J. (1993). 'The Interaction Between Time-Nonseparable Preferences and Time Aggregation'. *Econometrica*, Vol. 61:2, pp. 353–385.

Hitz, K.L. and B.D.O. Anderson (1972). 'Iterative Method of Computing the Limiting Solution of the Matrix Riccati Differential Equation'. *Proc. IEEE*, Vol. 60, pp. 1402–1406.

Kågström, B. and P. Poromaa (1994). 'Computing Eigenspaces with Specified Eigenvalues of a Regular Matrix Pair (A, B) and Condition Estimation: Theory, Algorithms and Software'. Mimeo. LAPACK Working Note 87.

Kashyap, R.L. (1970). 'Maximum Likelihood Identification of Stochastic Linear Systems'. *IEEE Transactions on Automatic Control*, Vol. AC–15: pp. 25–34.

Kenney, C.S., A.J. Laub, and P.M. Papadopoulos (1993). 'A Newton-Squaring Algorithm for Computing the Negative Invariant subspace of a Matrix'. *IEEE Transactions on Automatic Control*, Vol. 38, pp. 1284–1289.

Kimura, M. (1988). 'Convergence of the Doubling Algorithm for the Discrete-Time Algebraic Riccati Equation'. *Int. J. Systems Sci.*, Vol. 19, No. 5, pp. 701–711.

Kimura, M. (1989). 'Doubling Algorithm for Continuous-Time Algebraic Riccati Equation'. *Int. J. Systems Sci.*, Vol. 20, No. 2, pp. 191–202.

King, R.G., C.I. Plosser, and S.T. Rebelo (1988a). 'Production, Growth, and Business Cycles: I. The Basic Neoclassical Model'. *Journal of Money, Credit, and Banking*, Vol. 21, No. 2/3, pp. 195–232.

King, R.G., C.I. Plosser, and S.T. Rebelo (1988b). 'Production, Growth, and Business Cycles: II. New Directions'. *Journal of Money, Credit, and Banking*, Vol. 21, No. 2/3, March, May 1988, 309–342.

Kwakernaak, H. and R. Sivan (1972). *Linear Optimal Control Systems*. New York: Wiley Interscience.

Kydland, F. and E.C. Prescott (1982). 'Time to Build and Aggregate Fluctuations'. *Econometrica*, Vol. 50, pp. 1345–1370.

Laub, A.J. (1979). 'A Schur Method for Solving Algebraic Riccati Equations'. *IEEE Trans. Auto. Control*, Vol. AC-24, pp. 913–921.

Laub, A.J. (1991). 'Invariant Subspace Methods for the Numerical Solution of Riccati Equations'. In S. Bittanti, A.J. Laub and J.C. Willems (eds.), *The Riccati Equation*. Springer Verlag.pp. 163–196.

Lu, L. and W. Lin (1993). 'An Iterative Algorithm for Solution of the Discrete-Time Algebraic Riccati Equation'. *Linear Algebra and Its Applications*, Vol 188,189, pp. 465-488.

MacFarlane, A.G.J. (1963). 'An Eigenvector Solution of the Optimal Linear Regulator Problem'. *J. Electron. Contr.*, Vol. 14: pp. 643–654.

McGrattan, E. (1994). 'A Note on Computing Competitive Equilibria in Linear Models'. *Journal of Economic Dynamics and Control*, Vol. 18, pp. 149–160.

McGrattan, E., R. Rogerson, and R. Wright (1995). 'An Equilibrium Model of the Business Cycle with Household Production and Fiscal Policy'. Mimeo. Staff Report 166, Federal Reserve Bank of Minneapolis.

Pappas, T., A.J. Laub, and N.R. Sandell, Jr. (1980). 'On the Numerical Solution of the Discrete-Time Algebraic Riccati Equation'. *IEEE Transactions on Automatic Control*, Vol. AC-25, No. 4, pp. 631–641.

Petkov, P. Jr., N.D. Christov, and M.M. Konstantinov (1991). *Computational Methods for Linear Control Systems*. Englewood Cliffs: Prentice Hall.

Potter, J.E. (1966). 'Matrix Quadratic Solutions'. *SIAM J. Appl. Math.*, Vol. 14, pp. 496–501.

Roberts, J.D. (1980). 'Linear Model Reduction and Solution of the Algebraic Equation by Use of the Sign Function'. *International Journal of Control*, Vol. 32, pp. 677-687. (reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971)

Rosen, S., K.M. Murphy, and J.A. Scheinkman (1994). 'Cattle Cycles'. *Journal of Political Economy*, Vol. 102, No. 3, pp. 468–492.

Sargent, T.J. (1987). *Macroeconomic Theory*. 2nd ed. New York: Academic Press.

Sims, C.A. (1980). 'Macroeconomics and Reality'. *Econometrica*, Vol. 48, No. 1, pp. 1–48.

Siow, A. (1984). 'Occupational Choice Under Uncertainty'. *Econometrica*, Vol. 52, No. 3, pp. 631–645.

Stewart, G.W. (1972). 'On the Sensitivity of the Eigenvalue Problem $Ax = \lambda Bx$'. *SIAM Journal Numer Anal*, Vol. 9, pp. 669–668.

Stewart, G.W. (1976). 'Algorithm 506 - HQR3 and EXCHNG: FORTRAN Subroutines for Calculating and Ordering the Eigenvalues of a Real Upper Hessenberg Matrix'. *ACM Trans. Math Software*, Vol. 2, pp. 275–280.

United States, Bureau of the Census (1975). *Historical Statistics of the United States, Colonial Times to 1970*. Washington, D.C.: U.S. Department of Commerce.

United States, Bureau of the Census (1989). *Agricultural Statistics*. Washington, D.C.: U.S. Department of Commerce.

Van Dooren, P. (1981). 'A Generalized Eigenvalue Approach for Solving Riccati Equations'. *SIAM J. Sci. Stat. Comput.*, 2:121–135.

Van Dooren, P. (1982). 'Algorithm 590-DSUBSP and EXCHQZ: Fortran Subroutines for Computing Deflating Subspaces with Specified Spectrum'. *ACM Trans. Math. Software*, Vol. 8, pp. 376–382.

Vaughan, D.R. (1970). 'A Nonrecursive Algebraic Solution for the Discrete Riccati Equation'. *IEEE Transactions on Automatic Control*, Vol. AC-15, No. 5, pp. 597–599.

Ward, R.C. (1981). 'Balancing the Generalized Eigenvalue Problem'. *SIAM J. Sci. Stat. Comput.*, 2: pp. 141-152.

Whiteman, C. (1983). *Linear Rational Expectations Models: A User's Guide*. Minneapolis: University of Minnesota Press.

Wilson, D.A. and A. Kumar (1982). 'Derivative Computations for the Log Likelihood Function'. *IEEE Transactions on Automatic Control*, AC-27: pp. 230–232.

Zadrozny, Peter (1988a). 'Analytic Derivatives for Estimation of Discrete-Time, Linear Quadratic, Dynamic, Optimization Models'. *Econometrica*, 56: pp. 467–472.

Zadrozny, P.A. (1988b). 'Gaussian Likelihood of Continuous-Time ARMAX Models when Data are Stocks and Flows at Different Frequencies'. *Econometric Theory*, Vol. 4: pp. 108–124.

Zadrozny, P.A. (1989). 'Analytic Derivatives for Estimation of Linear Dynamic Models'. *Computers and Mathematics with Applications*, Vol. 18: pp. 539–553.

Zadrozny, P.A. (1992). 'Errata to Analytic Derivatives for Estimation of Linear Dynamic Models'. *Computers and Mathematics with Applications*, Vol. 24: pp. 289–290.