Spencer Lyon

Sargent RA: M.O. 3

Due on Friday, August 16, 2013

August 15, 2013

Problem 5.1

Consider the modified version of the optimal linear regulator problem where the objective is to maximize

$$-\sum_{t=0}^{\infty} \beta^t \{x_t' R x_t + u_t' Q u_t + 2 u_t' H x_t\} \tag{1}$$

subject to the law of motion

$$x_{t+1} = A x_t + B u_t \tag{2}$$

Here x_t is an $n \times 1$ state vector, u_t is a $k \times 1$ vector of controls, and x_0 is a given initial condition. The matrices R and Q are positive definite and symmetric. The maximization is with respect to sequences $\{u_r, x_t\}_{t=0}^{\infty}$.

a. Show that the optimal policy has the form

$$u_t = -\left(Q + \beta B' P B\right)^{-1} \left(\beta B' P A + H\right) x_t \tag{3}$$

where P solves the algebraic matrix Riccati equation

$$P = R + \beta A' P A - \left(\beta A' P B + H'\right) \left(Q + \beta B' P B\right)^{-1} \left(\beta B' P A + H\right) \tag{4}$$

b. Write a Matlab (Python) program to solve Equation 4 by iterating on P, starting with P being equal to a matrix of zeros.

a. To begin this part of the problem, I guess that the value function is quadratic in x: $v(x) = -x'Px$. I can then write out the Bellman operator

$$\begin{aligned} -xPx &= \max_u \left\{-x'Rx - u'Qu - 2u'Hx - \beta(Ax+Bu)'P(Ax+Bu)\right\} \\ &= \max_u \left\{-x'Rx - u'Qu - 2u'Hx - \beta x'A'PAx - \beta x'A'PBu - \beta u'B'PAx - \beta u'B'PBu\right\} \end{aligned} \tag{5}$$

The first order conditions of 5 with respect to u can be calculated and simplified as follows (note that P, Q, and R are symmetric):

$$\begin{aligned} 0 &= -(Q+Q')u - 2Hx - \beta B'P'Ax - \beta B'PAx - \beta(B'PB + B'P'B)u \\ (-1)0 &= 2Qu + 2Hx + \beta 2B'PAx + \beta 2B'PBu \\ (2Q + 2\beta B'BP)u &= -2Hx - 2\beta B'PAx \\ (Q + \beta B'BP)u &= -Hx - \beta B'PAx \\ u &= -(Q+\beta B'BP)^{-1}(H + \beta B'PA)x \end{aligned} \tag{6}$$

The expression for u, found in Equation 6 is what I was asked to find. I can now substitute this result back into the Bellman operator in Equation 5 and simplify to get the desired expression for P:

$$\begin{aligned} -xPx &= -x'Rx - u'Qu - 2u'Hx - \beta x'A'PAx - \beta x'A'PBu - \beta u'B'PAx - \beta u'B'PBu \\ &= -x'Rx \\ &\quad - \left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right]' Q \left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right] \\ &\quad - 2\left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right]' Hx \\ &\quad - \beta x'A'PAx \\ &\quad - \beta x'A'PB\left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right] \\ &\quad - \beta\left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right]' B'PAx \\ &\quad - \beta\left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right]' B'PB\left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right] \\ &= x'Rx \\ &\quad + \left[-x'(\beta A'PB + H')(Q+\beta B'PB)^{-1}\right] Q \left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right] \\ &\quad + \beta\left[-x'(\beta A'PB + H')(Q+\beta B'PB)^{-1}\right] B'PB\left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right] \\ &\quad + 2\left[-x'(\beta A'PB + H')(Q+\beta B'PB)^{-1}\right] Hx \\ &\quad + \beta x'A'PAx \\ &\quad + \beta x'A'PB\left[-(Q+\beta B'BP)^{-1}(H+\beta B'PA)x\right] \\ &\quad + \beta\left[-x'(\beta A'PB + H')(Q+\beta B'PB)^{-1}\right] B'PAx \end{aligned}$$

I can now collect on second and third terms (which I define as a variable D) in the previous expression and simplify.

$$
\begin{aligned}
D &= \text{Term}_1 + \text{Term}_3 \\
&\quad \left[-x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}\right] Q \left[-(Q + \beta B'BP)^{-1}(H + \beta B'PA)x\right] \\
&\quad + \beta \left[-x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}\right] B'PB \left[-(Q + \beta B'BP)^{-1}(H + \beta B'PA)x\right] \\
&= \left[-x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}\right] (Q + \beta B'PB) \left[-(Q + \beta B'BP)^{-1}(H + \beta B'PA)x\right] \\
&= x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}(\beta B'PA + H)x
\end{aligned}
$$

I can now substitute D back into the long equation it came from to get the expression below. Note that the simplification I then do comes in three steps: 1) I use symmetric matrices to combine the last two items from the first expression into a single expression, 2) I use the transpose to collect this new last item with the third item to simplify, and 3) I add like terms.

$$
\begin{aligned}
-xPx &= -x'Rx - u'Qu - 2u'Hx - \beta x'A'PAx - \beta x'A'PBu - \beta u'B'PAx - \beta u'B'PBu \\
&= x'Rx \\
&\quad + x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}(\beta B'PA + H)x \\
&\quad + 2\left[-x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}\right] Hx \\
&\quad + \beta x'A'PAx \\
&\quad + \beta x'A'PB \left[-(Q + \beta B'BP)^{-1}(H + \beta B'PA)x\right] \\
&\quad + \beta \left[-x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}\right] B'PAx \\
&= x'Rx \\
&\quad + x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}(\beta B'PA + H)x \\
&\quad + 2\left[-x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}\right] Hx \\
&\quad + \beta x'A'PAx \\
&\quad + 2\beta \left[-x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}\right] B'PAx \\
&= x'Rx \\
&\quad + x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}(\beta B'PA + H)x \\
&\quad + \beta x'A'PAx \\
&\quad - 2x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}(B'PA + H)x \\
&= x'Rx \\
&\quad - x'(\beta A'PB + H')(Q + \beta B'PB)^{-1}(\beta B'PA + H)x \\
&\quad + \beta x'A'PAx
\end{aligned}
$$

I can now collect terms and to pattern matching. I know that the left-hand side is of the form $x'Px$, so I arrange the right-hand side to look the same and simplify to get my final expression for P

$$
\begin{aligned}
x'Px &= x'\left[R + \beta A'PA - (\beta A'PB + H')(Q + \beta B'PB)^{-1}(\beta B'PA + H)\right] x' \\
P &= R + \beta A'PA - (\beta A'PB + H')(Q + \beta B'PB)^{-1}(\beta B'PA + H)
\end{aligned}
$$

□

b. The Python program I wrote for this problem is included in Listing 1.

Listing 1: Python Function `riccati` to solve the algebraic Riccatil equation by iteration

```
from math import sqrt
import numpy as np
from scipy.linalg import eig, solve, norm, inv


def ricatti(beta, A, B, R, Q, H, tol=1e-6, maxiter=1000):
    """
    Calculates F of the feedback law:

    .. math::

         U = -Fx

     that maximizes the function:

     .. math::

        \sum \{beta^t [x'Qx + u'Ru +2x'Wu] \}

     subject to

     .. math::
         x_{t+1} = A x_t + B u_t

    where x is the nx1 vector of states, u is the kx1 vector of controls

    Parameters
    ----------
    beta : float
        The discount factor from above. If there is no discounting, set
        this equal to 1.

    A : array_like, dtype=float, shape=(n, n)
        The matrix A in the law of motion for x

    B : array_like, dtype=float, shape=(n, k)
        The matrix B in the law of motion for x

    R : array_like, dtype=float, shape=(k, k)
        The matrix R from the objective function

    Q : array_like, dtype=float, shape=(n, n)
        The matrix Q from the objective function

    H : array_like, dtype=float, shape=(n, k), optional(default=0)
        The matrix W from the objective function. Represents the cross
        product terms.

    tol : float, optional(default=1e-6)
        Convergence tolerance for case when largest eigenvalue is below
        1e-5 in modulus

    max_iter : int, optional(default=1000)
        The maximum number of iterations the function will allow before
        stopping

    Returns
    -------
    F : array_like, dtype=float
        The feedback law from the equation above.

    P : array_like, dtype=float
        The steady-state solution to the associated discrete matrix
        Riccati equation

    """
    n = A.shape[0]
    k = np.ascontiguousarray(Q).shape[0]

    A, B, R, Q, H = map(np.matrix, [A, B, R, Q, H])

    A = A.reshape(n, n)
    B = B.reshape(n, k)
    Q = Q.reshape(k, k)
    R = R.reshape(n, n)
    H = H.reshape(k, n)
```

```
    # Start with an initial P matrix
    p0 = np.zeros((n, n))
    p1 = np.zeros((n, n))

    # Define some variables necessary to enter while loop
    dist = 10.
    iters = 0

    while dist > tol and iters < maxiter:
        p1 = R + beta*A.T*p0*A - ((beta*A.T*p0*B + H.T) *
                                  inv(Q + beta*B.T*p0*B) *
                                  (beta*B.T*p0*A + H))

        dist = norm(p1 - p0)
        print("Iteration is %i and norm is %.3e" % (iters, dist))
        p0 = p1

    P = p0

    F = inv((Q + beta*B.T.dot(P.dot(B)))).dot(beta*B.T.dot(P.dot(A)) + H)

    return map(np.array, [F, P])
```

□

Problem 5.2

Verify that equations 5.2.10 and 5.2.11 implement the policy improvement algorithm for the discounted linear regulator problem.

Before showing the solution to this problem I need to define a few things:

- The value function is represented as $v(x)_t = -x_t' P_t x_t$.
- The policy is of the form: $u_t = F_t x_t$
- The Bellman is of the following form:

$$\begin{aligned} V(x) &= -x_t' R x_t - u_t' Q u_t - \beta (A x_t + B u_t)' P_t (A x_t + B u_t) \\ &= -x_t' R x_t - x_t' F_t' Q F_t x_t - \beta (A x_t + B F_t x_t)' P_t (A x_t + B F_t x_t) \\ &= -x_t' R x_t - x_t' (F_t' Q F_t) x_t - \beta x_t' \left((A + B F_t)' P_t (A + B F_t) \right) x_t \end{aligned} \tag{7}$$

- The first order conditions for the Bellman are given in equation 5.2.3 (with a β added where needed – see Problem 5.1 for the work), which I repeat here:

$$\left(Q + \beta B' P_t B \right) u_{t+1} = -\beta B' P_t A x_t$$

With those items set up, I am ready to begin. I assume that I have an optimal policy of the form $u_0 = -F_0 x_0$. I know that my goal is to end up at the policy improvement algorithm, so I will assume I will apply that policy forever. In doing this I will need to verify that the eigenvalues of $A - BF_0$ are all less than $\frac{1}{\sqrt{\beta}}$ in modulus; this will ensure that the problem doesn't go off to infinity. The next step is to take the expression for the Bellman in Equation 7, set it equal to the value function from above, equate terms, and get an implicit expression for P_0. I do this below.

$$\begin{aligned} v(x)_0 &= V(x)_0 \\ -x_0' P_0 x_0 &= -x_0' R x_0 - x_0' (F_0' Q F_0) x_0 - \beta x_0' \left((A + BF_0)' P_0 (A + BF_0) \right) x_0 \\ P_0 &= R + F_0' Q F_0 + \beta (A - BF_0)' P_0 (A - BF_0) \end{aligned} \tag{8}$$

I can now use the first order conditions for this equation solve for u_1 like so:

$$\begin{aligned} \left(Q + \beta B' P_t B \right) u_1 &= -\beta B' P_0 A x_0 \\ u_1 &= -\beta \left(Q + \beta B' P_0 B \right)^{-1} B' P_0 A x_0 \\ u_1 &= -F_1 x_t \end{aligned} \tag{9}$$

Where

$$F_1 = \beta \left(Q + \beta B' P_0 B \right)^{-1} B' P_0 \tag{10}$$

Together Equations 8 and 10 define a recursion for P_t and F_J, which I repeat here for completeness:

$$\begin{aligned} P_t &= R + F_t' Q F_t + \beta (A - BF_t)' P_t (A - BF_t) \\ F_{t+1} &= \beta \left(Q + \beta B' P_t B \right)^{-1} B' P_t \end{aligned} \tag{11}$$

These equations are the same as 5.2.10 and 5.2.11 and were derived using the policy improvement algorithm.

□

Problem 5.3

A household chooses $\{c_t, a_{t+1}\}_{t=0}^{\infty}$ to maximize

$$-\sum_{t=0}^{\infty} \beta^t \left\{ (c_t - b)^2 + \gamma i_t^2 \right\} \tag{12}$$

subject to

$$\begin{aligned} c_t + i_t &= r a_t + y_t \\ a_{t+1} &= a_t + i_t \\ y_{t+1} &= \rho_1 y_t + \rho_2 y_{t-1} \end{aligned} \tag{13}$$

Here c_t, i_t, a_t, and y_t are the household's consumption, investment, asset holdings, and exogenous labor income at t; while $b > 0$, $\gamma > 0$, $r > 0$, $\beta \in (0, 1)$, ad ρ_1, ρ_2 are parameters, and a_0, y_0, y_{-1} are initial conditions. Assume that ρ_1, ρ_2 are such that $\left(1 - \rho_1 z - \rho_2 z^2\right) = 0$ implies that $|z| > 1$.

a. Map this problem into an optimal linear regulator problem.

b. For parameter values $\left[\beta, (1 + r), b, \gamma, \rho_1, \rho_2\right] = \left[0.95, 0.95^{-1}, 30, 1, 1.2, -.3\right]$, compute the household's optimal policy function using your Matlab (Python) program from exercise 5.1.

Problem 5.3 continued on next page...

a. To map this into a linear regulator problem we need to come up with matrices A, B, Q, and R that explain the objective function and pin down the law of motion for the state variables. Let the control vector and the state vector be the following:

$$u_t = \begin{pmatrix} i_t \end{pmatrix} = \begin{pmatrix} a_{t+1} - a_t \end{pmatrix}$$
$$x_t = \begin{pmatrix} a_t \\ y_t \\ y_{t-1} \\ 1 \end{pmatrix} \tag{14}$$

Now, I know that I am looking for the matrices A and B that satisfy $x_{t+1} = Ax_t + Bu_t$. Examining the given equations and using the definitions from equation 14, I can pin them down as follows:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \rho_1 & \rho_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{15}$$

In order to determine the values of R, Q, and H I need to substitute out the c_t in the objective function. Doing this yields the following:

$$\begin{aligned} &-\sum_{t=0}^{\infty} \beta^t \left\{(c_t - b)^2 + \gamma i_t^2\right\} \\ &-\sum_{t=0}^{\infty} \beta^t \left\{\left(ra_t - b - i_t + y_t\right)^2 + \gamma i_t^2\right\} \\ &-\sum_{t=0}^{\infty} \beta^t \left\{-2bra_t - 2ra_t i_t + r^2 a_t^2 + 2ra_t y_t + b^2 + 2bi_t - 2by_t + \gamma i_t^2 - 2i_t y_t + i_t^2 + y_t^2\right\} \end{aligned} \tag{16}$$

Using this form I am now ready to match patterns to define the matrices R, Q, and H in the expression $x_t' R x_t + u_t' Q u_t + 2u_t' H x_t$. To show my process, I will do this one item at a time, then show the entire matrices. For the R matrix:

- $R_{1,1}$: This is the coefficient on a_t^2. There is a $r^2 a_t^2$, so this value is r^2.
- $R_{1,2} = R_{2,1}$: This is 1/2 the coefficient on $y_t a_t$. I see that there is a $2r y_t a_t$ so this value is r.
- $R_{1,4} = R_{4,1}$: This is 1/2 the coefficient on a_t by itself. There is a $-2bra_t$, so this value is $-br$.
- $R_{2,2}$: This is the coefficient on y_t^2. I see that this must be 1.
- $R_{2,4} = R_{4,2}$: This is 1/2 the coefficient on y_t by itself. There is a $-2by_t$ so this value is $-b$.
- $R_{:,3} = R_{3,:} = 0$: The second row and column deal with y_{t-1}, which doesn't appear in the equation above.
- $R_{4,4}$: This is the coefficient on 1. There is a b^2, so this value is b^2.

Problem 5.3 continued on next page...

The Q matrix is the scalar coefficient on i_t^2, which is $(1+\gamma)$.

The H matrix is explained below:

- $H_{1,1}$: This is 1/2 the coefficient on $i_t a_t$. There is a $-2r\,i_t a_t$, so this value is $-r$.
- $H_{1,2}$: This is 1/2 the coefficient on $i_t y_t$. There is a $-2i_t y_t$, so this value is -1.
- $H_{1,3}$: This is 1/2 the coefficient on $i_t y_{t-1}$. There are not y_{t-1}, so this value is 0.
- $H_{1,4}$: This is 1/2 the coefficient on i_t by itself. There is a $2bi_t$, so this value is b.

I am now ready to show all the matrices in their complete form.

$$R = \begin{bmatrix} r^2 & r & 0 & -br \\ r & 1 & 0 & -b \\ 0 & 0 & 0 & 0 \\ -br & -b & 0 & b^2 \end{bmatrix} \quad Q = (1+\gamma) \quad H = \begin{bmatrix} -r & -1 & 0 & b \end{bmatrix} \tag{17}$$

These matrices map the problem into an optimal linear regulator problem. Specifically the problem is to maximize with respect to u

$$-\sum_{t=0}^{\infty} \beta^t \{x_t' R x_t + u_t' Q u_t + 2u_t' H x_t\} \tag{18}$$

subject to the law of motion

$$x_{t+1} = Ax_t + Bu_t \tag{19}$$

b. I used the function `riccati` from Problem 5.1 to compute the optimal policy F in $u_{t+1} = -Fx_t$ and the implied value of P. The code to produce the solution is contained in Listing 2. The solution I got was:

$$F = \begin{bmatrix} -0.000 & -0.317 & -0.056 & 0.000 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.055 & 0.403 & -0.115 & -31.579 \\ 0.403 & 4.591 & -1.250 & -229.446 \\ -0.115 & -1.250 & 0.386 & 65.392 \\ -31.579 & -229.446 & 65.392 & 18000.000 \end{bmatrix}$$

Listing 2: Solution to 5.3b

```
import numpy as np
import sys
from rmt_utils import olrp, ricatti
from matrix2latex import matrix2latex as to_tex

bmat_tex = lambda i: to_tex(i, None, 'bmatrix', formatColumn=['%.3f'] * 15)

args = sys.argv
probs = args[1:] if len(args) > 0 else []


def p5_3(beta, r, b, gamma, rho_1, rho_2):
```

```
    """
    Using rmt_utils.olrp to solve the optimal linear regulator problem.
    from RMT4 problem 5.3b
    """
    A = np.array([[1., 0., 0., 0.],
                  [0., rho_1, rho_2, 0.],
                  [0., 1., 0., 0.],
                  [0., 0., 0., 1.]])

    B = np.array([1., 0., 0., 0.]).reshape(4, 1)

    R = np.array([[r**2, r, 0., -b*r],
                  [r, 1., 0, -b],
                  [0., 0., 0., 0.],
                  [-b*r, -b, 0, b**2]])

    Q = 1 + gamma

    H = np.array([[-r, -1., 0., b]]).reshape(1, 4)

    return ricatti(beta, A, B, R, Q, H)

if '53' in probs:
    in_53 = [0.95, 1 / 0.95 - 1., 30, 1., 1.2, -0.3]
    f, p = p5_3(*in_53)
```

□

Problem 5.4

Modify exercise 5.3 by assuming that the household seeks to maximize

$$-\sum_{t=0}^{\infty} \beta^t \left\{(s_t - b)^2 + \gamma i_t^2\right\} \tag{20}$$

Here s_t measure consumption services that are produced by durables or habits according to

$$\begin{aligned} s_t &= \lambda h_t + \pi c_t \\ h_{t+1} &= \delta h_t + \theta c_t \end{aligned} \tag{21}$$

where h_t is the stock of the durable good or habit, $(\lambda, \pi, \delta, \theta)$ are parameters, and h_0 is an initial condition.

a. Map this problem into an optimal linear regulator problem.

b. For the same parameter values as in exercise 5.3 and $(\lambda, \pi, \delta, \theta) = (1, 0.5, 0.95, 1)$ compute the optimal policy for the household.

c. For the same parameter values as in exercise 5.3 and $(\lambda, \pi, \delta, \theta) = (-1, 1, 0.95, 1)$ compute the optimal policy .

d. Interpret the parameter settings in part b as capturing a model of durable consumption goods, and the settings in part c as giving a model of habit persistence.

a. I also begin this problem by defining the state and control vectors. The only change is that we now need

to keep track of h_t as a state variable. This means that u and x are defined as follows:

$$u_t = \begin{pmatrix} i_t \end{pmatrix} \quad x_t = \begin{pmatrix} a_t \\ y_t \\ y_{t-1} \\ h_t \\ 1 \end{pmatrix} \tag{22}$$

I now proceed to define the matrices A, B, R, Q, H in a manner similar to how I did it for the previous problem. A and B are easily apparent and I write them here.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \rho_1 & \rho_2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \theta r & \theta & 0 & \delta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -\theta \\ 0 \end{bmatrix} \tag{23}$$

To identify R, Q, and H I will take a similar approach – I will make substitutions, expand the result, and do pattern matching to determine the matrices. I will not show all the work again on this problem (it is quite tedious and my approach was clear from the last problem), so I will just give the new values for the matrices.

$$R = \begin{bmatrix} \pi^2 r^2 & \pi^2 r & 0 & \pi r \lambda & -b\pi r \\ \pi^2 r & \pi^2 & 0 & \pi\lambda & -b\pi \\ 0 & 0 & 0 & 0 & 0 \\ \pi r \lambda & \pi\lambda & 0 & \lambda^2 & -b\lambda \\ -b\pi r & -b\pi & 0 & -b\lambda & b^2 \end{bmatrix} \quad Q = \gamma + \pi^2 \quad H = \begin{bmatrix} -\pi^2 r & -\pi^2 & 0 & -\pi\lambda & b\pi \end{bmatrix} \tag{24}$$

b. I used my `riccati` equation from above to get the values of F and P that solve this problem. They appear below.

$$F = \begin{bmatrix} 0.330 & 1.735 & -0.663 & -0.314 & 0.000 \end{bmatrix}$$

$$P = \begin{bmatrix} 6.073 & 43.806 & -12.583 & 5.283 & -323.482 \\ 43.806 & 319.130 & -91.547 & 38.687 & -2350.346 \\ -12.583 & -91.547 & 26.318 & -10.932 & 669.849 \\ 5.283 & 38.687 & -10.932 & 5.494 & -307.692 \\ -323.482 & -2350.346 & 669.849 & -307.692 & 18000.000 \end{bmatrix}$$

c. I repeated the process from part b under different parameters and got the following (note the code for this part and part b are in Listing 3):

$$F = \begin{bmatrix} 0.179 & 0.688 & -0.425 & -0.170 & -0.000 \end{bmatrix}$$

Problem 5.4 continued on next page...

$$P = \begin{bmatrix} 5.051 & 36.406 & -10.429 & 3.945 & 276.113 \\ 36.406 & 265.521 & -75.948 & 28.943 & 2006.177 \\ -10.429 & -75.948 & 21.797 & -8.198 & -571.761 \\ 3.945 & 28.943 & -8.198 & 5.996 & 307.692 \\ 276.113 & 2006.177 & -571.761 & 307.692 & 18000.000 \end{bmatrix}$$

d. For parameterizations in parts b and c, it is instructive to write the laws of motion for h_{t+1} and s_t. They appear in the table below.

	Part b	Part c
h_{t+1}	$0.95h_t + c_t$	$0.95h_t + c_t$
s_t	$h_t + 0.05c_t$	$c_t - h_t$

Notice that the law of motion for h_{t+1} is the same in both cases. The main difference is in how s_t changes from one period to the next. In part b, we have that $s_t = h_t + 0.05c_t$. In this case we see that the marginal effect on the objective function of a change in either h_t or c_t is positive. However, we also see that the effect from a unit increase in h_t is much more influential than the effect of a unit increase in c_t. This difference in relative effects results in durable goods not being as meaningful in the period in which they are purchased, thus allowing the parameterization to lead to a model of durable consumption goods.

In part c, h_t represents the value of today's "habit" The objective function is influenced by $s_t = c_t - h_t$, which is clearly interpreted as the value of today's consumption less today's habit. For this reason, an optimizing agent cannot indefinitely increase the value of the habit each period, because that would drive the objective function further and further negative. This effect makes the parameterization capture a model of habit persistence.

Listing 3: Solution to 5.3c

```
def p5_4(beta, r, b, gamma, rho_1, rho_2, lamb, pi, delta, theta):
    """
    Using rmt_utils.olrp to solve the optimal linear regulator problem.
    from RMT4 problem 5.4(b,c)
    """
    A = np.array([[1, 0, 0, 0, 0],
                  [0, rho_1, rho_2, 0, 0],
                  [0,  1,  0, 0,  0],
                  [theta * r, theta, 0, delta, 0],
                  [0,  0,  0,  0,  1]]
                 )

    B = np.array([1, 0, 0, -theta, 0.])

    Q = pi**2 + gamma
    R = np.array([[(pi*r)**2, pi**2*r, 0, lamb*pi*r, -b*pi*r],
                  [pi**2*r, pi**2, 0, lamb*pi, -b*pi],
                  [0, 0, 0, 0, 0],
                  [lamb*pi*r, lamb*pi, 0, lamb**2, -b*lamb],
                  [-b*pi*r, -b*pi, 0, -b*lamb, b**2]])

    H = np.array([-pi**2*r, -pi**2, 0, -pi*lamb, b*pi])

    return ricatti(beta, A, B, R, Q, H)

if '54' in probs:
    in_54b = [0.95, 1 / 0.95 - 1, 30, 1, 1.2, -0.3, 1, 0.5, 0.95, 1.0]
    in_54c = [0.95, 1 / 0.95 - 1, 30, 1, 1.2, -0.3, -1, 1, 0.95, 1.0]
    f4b, p4b = p5_4(*in_54b)
    f4c, p4c = p5_4(*in_54c)
```

□