EE363 Homework 6 Solutions

Spring 2026

19.2881. Sensor selection and observer design. Consider the system $\dot{x} = Ax$, $y = Cx$, with

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

(This problem concerns observer design so we've simplified things by not even including an input.) We consider observers that (exactly and instantaneously) reconstruct the state from the output and its derivatives. Such observers have the form

$$x(t) = F_0 y(t) + F_1 \frac{dy}{dt}(t) + \cdots + F_k \frac{d^k y}{dt^k}(t),$$

where $F_0, \ldots, F_k$ are matrices that specify the observer. (Of course we require this formula to hold for any trajectory of the system and any t, *i.e.*, the observer has to work!) Consider an observer defined by $F_0, \ldots, F_k$. We say the *degree* of the observer is the largest j such that $F_j \neq 0$. The degree gives the highest derivative of y used to reconstruct the state. If the ith columns of $F_0, \ldots, F_k$ are all zero, then the observer doesn't use the ith sensor signal $y_i(t)$ to reconstruct the state. We say the observer *uses* or *requires* the sensor i if at least one of the ith columns of $F_0, \ldots, F_k$ is nonzero.

a) What is the minimum number of sensors required for such an observer? List all combinations (*i.e.*, sets) of sensors, of this minimum number, for which there is an observer using only these sensors.

b) What is the minimum degree observer? List all combinations of sensors for which an observer of this minimum degree can be found.

Solution.

a) Let c_i denote the ith row of C, so

$$c_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}, \qquad c_2 = \begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}, \qquad c_3 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the system $\dot{x} = Ax$, we have

$$\frac{d^j y}{dt^j}(t) = CA^j x(t), \qquad j = 0, 1, \ldots.$$

Thus an observer of degree at most k exists using a set of sensors S if and only if

$$\operatorname{rank} \begin{bmatrix} C_S \\ C_S A \\ \vdots \\ C_S A^k \end{bmatrix} = 4,$$

where C_S is the matrix formed from the rows of C corresponding to the sensors in S. Indeed, an observer

$$x(t) = F_0 y_S(t) + F_1 \dot{y}_S(t) + \cdots + F_k y_S^{(k)}(t)$$

exists exactly when there is a matrix

$$F = \begin{bmatrix} F_0 & F_1 & \cdots & F_k \end{bmatrix}$$

such that

$$F \begin{bmatrix} C_S \\ C_S A \\ \vdots \\ C_S A^k \end{bmatrix} = I.$$

Equivalently, the stacked observability matrix must have full column rank.

We now check the possible sensor sets by hand. First,

$$c_1 A = \begin{bmatrix} 2 & 1 & 0 & 0 \end{bmatrix}, \qquad c_1 A^2 = \begin{bmatrix} 3 & 1 & 0 & 0 \end{bmatrix}, \qquad c_1 A^3 = \begin{bmatrix} 4 & 1 & 0 & 0 \end{bmatrix}.$$

All rows generated by sensor 1 lie in the span of e_1^T and e_2^T, so sensor 1 alone gives rank only 2.

For sensor 2,

$$c_2 A = \begin{bmatrix} 1 & 2 & 1 & 0 \end{bmatrix}, \qquad c_2 A^2 = \begin{bmatrix} 3 & 3 & 1 & 0 \end{bmatrix}, \qquad c_2 A^3 = \begin{bmatrix} 6 & 4 & 1 & 0 \end{bmatrix}.$$

All rows generated by sensor 2 have fourth entry zero, so sensor 2 alone cannot give rank 4. In fact it gives rank 3.

For sensor 3,

$$c_3 A = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \qquad c_3 A^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \qquad c_3 A^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}.$$

Thus sensor 3 alone gives rank only 2.

So no single sensor is sufficient. We next check pairs of sensors. For sensors 1 and 2, all rows generated by c_1 and c_2 have fourth entry zero, so the rank is at most 3. Thus sensors $\{1, 2\}$ are not sufficient.

For sensors 1 and 3, the rows generated by c_1 lie in $\text{span}\{e_1^T, e_2^T\}$, while the rows generated by c_3 lie in $\text{span}\{e_1^T, e_4^T\}$. Hence all these rows lie in

$$\text{span}\{e_1^T, e_2^T, e_4^T\},$$

so the rank is at most 3. Thus sensors $\{1, 3\}$ are not sufficient.

For sensors 2 and 3, already the outputs and first derivatives give

$$\begin{bmatrix} c_2 \\ c_3 \\ c_2 A \\ c_3 A \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

This matrix has rank 4. Therefore sensors 2 and 3 are sufficient.

(It is acceptable to simply code matrix creation and rank checks instead of doing it by hand.)

It follows that the minimum number of sensors required is 2, and the only pair of sensors of this minimum size that works is $\{2, 3\}$.

b) We now determine the minimum degree. A degree zero observer would use only $y(t)$, so it would require C itself to have rank 4. This is impossible, since C has only three rows. Therefore the degree must be at least 1.

But, as shown above,

$$\begin{bmatrix} c_2 \\ c_3 \\ c_2A \\ c_3A \end{bmatrix}$$

has rank 4, so a degree one observer can be constructed using sensors $\{2, 3\}$. Hence, the minimum degree is 1. Including the first sensor would not reduce the rank so using all three sensors also gives a degree one observer.

For completeness, we can write an explicit degree one observer using only sensors 2 and 3. We have

$$y_2 = x_2 + x_3, \qquad y_3 = x_4.$$

Also,

$$\dot{y}_2 = c_2Ax = x_1 + 2x_2 + x_3, \qquad \dot{y}_3 = c_3Ax = x_1.$$

Therefore

$$x_1 = \dot{y}_3,$$

$$x_4 = y_3,$$

$$x_2 = \dot{y}_2 - \dot{y}_3 - y_2,$$

and

$$x_3 = 2y_2 - \dot{y}_2 + \dot{y}_3.$$

Thus

$$x(t) = \begin{bmatrix} \dot{y}_3(t) \\ \dot{y}_2(t) - \dot{y}_3(t) - y_2(t) \\ 2y_2(t) - \dot{y}_2(t) + \dot{y}_3(t) \\ y_3(t) \end{bmatrix}.$$

Equivalently,

$$x(t) = F_0y(t) + F_1\dot{y}(t),$$

where

$$F_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad F_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

This observer has degree one and uses only sensors 2 and 3.

22.1300. Stabilizing the magnetically-suspended ball. The magnetically suspended ball is modeled by

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = 9.8 - \frac{1}{15}\left(\frac{u}{x_1}\right)^2, \qquad y = x_1,$$

where x_1 and x_2 are the position and velocity of the ball, respectively, and u is the current.

Suppose we want to design the input u to stabilize the ball at $y^{\text{eq}} = 0.1$, *i.e.*, at the equilibrium point

$$x^{\text{eq}} = \begin{bmatrix} 0.1 \\ 0 \end{bmatrix}, \qquad u^{\text{eq}} = 1.212.$$

The linearized model around this equilibrium is

$$\dot{\tilde{x}} = \begin{bmatrix} 0 & 1 \\ 195.9 & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} 0 \\ -16.1 \end{bmatrix} \tilde{u}, \qquad \tilde{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tilde{x},$$

where

$$\tilde{x} = x - x^{\text{eq}} = \begin{bmatrix} x_1 - x_1^{\text{eq}} \\ x_2 - x_2^{\text{eq}} \end{bmatrix}, \qquad \tilde{y} = y - y^{\text{eq}}, \qquad \tilde{u} = u - u^{\text{eq}}.$$

(a) Show that the linearized system is controllable.

(b) Design a state feedback controller $\tilde{u} = K\tilde{x}$ that asymptotically stabilizes the linearized system.

(c) *Optional:* Simulate the nonlinear system model with the controller designed in part (b), applied as $u = u^{\text{eq}} + \tilde{u}$. Select at least three different initial conditions, starting close to the equilibrium point and progressively going farther. Plot the outputs y.

Solution. Let

$$A = \begin{bmatrix} 0 & 1 \\ 195.9 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ -16.1 \end{bmatrix}.$$

(a) The controllability matrix is

$$\mathcal{C} = \begin{bmatrix} B & AB \end{bmatrix} = \begin{bmatrix} 0 & -16.1 \\ -16.1 & 0 \end{bmatrix}.$$

Its determinant is

$$\det \mathcal{C} = -(16.1)^2 \neq 0.$$

Thus $\mathcal{C}$ has rank two, so the linearized system is controllable.

(b) Let

$$K = \begin{bmatrix} k_1 & k_2 \end{bmatrix}.$$

Then

$$A + BK = \begin{bmatrix} 0 & 1 \\ 195.9 - 16.1k_1 & -16.1k_2 \end{bmatrix}.$$

The characteristic polynomial of this closed-loop matrix is

$$\det(sI - A - BK) = s^2 + 16.1k_2 s + (16.1k_1 - 195.9).$$

We can therefore place the two closed-loop poles wherever we like. For example, choose the desired poles to be -2 and -3, so the desired characteristic polynomial is

$$(s+2)(s+3) = s^2 + 5s + 6.$$

Matching coefficients gives

$$16.1k_2 = 5, \qquad 16.1k_1 - 195.9 = 6,$$

and hence

$$K = \begin{bmatrix} \dfrac{201.9}{16.1} & \dfrac{5}{16.1} \end{bmatrix} \simeq \begin{bmatrix} 12.5404 & 0.3106 \end{bmatrix}.$$

With this choice,

$$A + BK = \begin{bmatrix} 0 & 1 \\ -6 & -5 \end{bmatrix},$$

which has eigenvalues -2 and -3. Thus the linearized closed-loop system is asymptotically stable.

(c) With the gain from part (b), the controller applied to the nonlinear model is

$$u = u^{\text{eq}} + \frac{201.9}{16.1}(x_1 - 0.1) + \frac{5}{16.1}x_2.$$

For a numerical simulation, it is best to use the unrounded equilibrium current

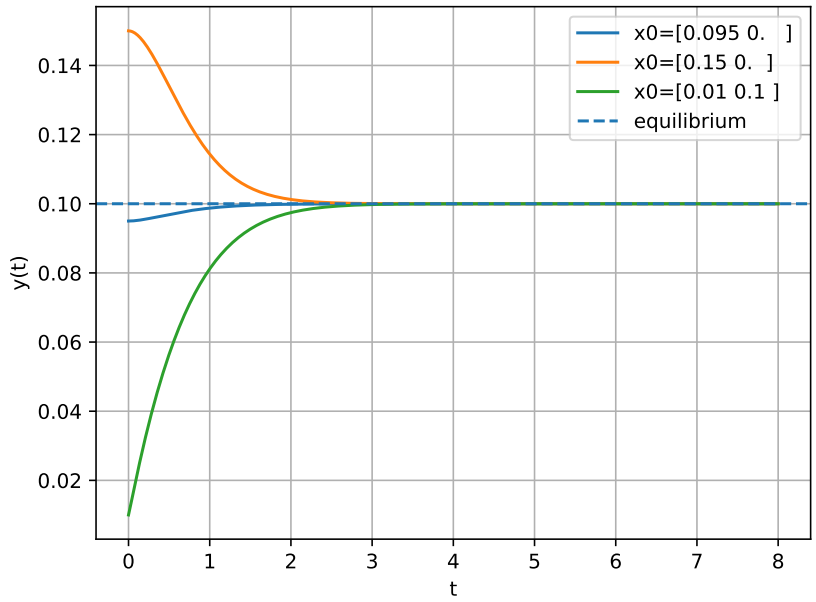
$$u^{\text{eq}} = 0.1\sqrt{147} = 1.2124\ldots,$$

whose rounded value is the 1.212 given in the problem statement. The following Python code simulates the nonlinear closed-loop system and plots $y = x_1$ for three initial conditions.

```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from scipy.integrate import solve_ivp

graphics_dir = Path(__file__).resolve().parent.parent / "graphics"
graphics_dir.mkdir(exist_ok=True)

x_eq = np.array([0.1, 0.0])
u_eq = 0.1*np.sqrt(147.0)
K = np.array([(195.9 + 6.0)/16.1, 5.0/16.1])


def f(t, x):
    u = u_eq + K @ (x - x_eq)
    return np.array([
                    x[1],
```

```
                    9.8 - (1.0/15.0)*(u/x[0])**2
                    ])


initial_conditions = [
    np.array([0.095, 0.0]),
    np.array([0.150, 0.0]),
    np.array([0.01, 0.1]),
]
t_eval = np.linspace(0.0, 8.0, 1000)
fig, ax = plt.subplots()
for x0 in initial_conditions:
    sol = solve_ivp(f, (0.0, 8.0), x0, t_eval=t_eval,
                    rtol=1e-9, atol=1e-11)
    ax.plot(sol.t, sol.y[0], label=f"x0={x0}")

ax.axhline(0.1, linestyle="--", label="equilibrium")
ax.set_xlabel("t")
ax.set_ylabel("y(t)")
ax.legend()
ax.grid(True)
fig.savefig(graphics_dir / "suspended_ball_part_c.pdf", bbox_inches="tight")
plt.show()
```

The generated plot is shown below.



For initial conditions sufficiently close to the equilibrium, the plotted outputs converge to $y^{\text{eq}} = 0.1$.

23.1300. LQR with exponential weighting. A common variation on the LQR problem includes explicit time-varying weighting factors on the state and input costs,

$$J = \sum_{\tau=0}^{N-1} \gamma^\tau \left(x_\tau^T Q x_\tau + u_\tau^T R u_\tau \right) + \gamma^N x_N^T Q_f x_N,$$

where

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

x_0 is given, and, as usual, $Q = Q^T \succeq 0$, $Q_f = Q_f^T \succeq 0$, and $R = R^T \succ 0$ are constant. The parameter γ, called the exponential weighting factor, is positive. For $\gamma = 1$, this reduces to the standard LQR cost function. For $\gamma < 1$, the penalty for future state and input deviations is smaller than in the present; in this case we call γ the discount factor or forgetting factor. When $\gamma > 1$, future costs are accentuated compared to present costs. This gives added incentive for the input to steer the state towards zero quickly.

(a) Note that we can find the input sequence $u_0^\star, \ldots, u_{N-1}^\star$ that minimizes J using standard LQR methods, by considering the state and input costs as time varying, with

$$Q_t = \gamma^t Q, \qquad R_t = \gamma^t R,$$

and final cost matrix $\gamma^N Q_f$. Thus, we know at least one way to solve the exponentially weighted LQR problem. Use this method to find the recursive equations that give $u^\star$.

(b) Exponential weights can also be incorporated directly into a dynamic programming formulation. We define

$$W_t(z) = \min \sum_{\tau=t}^{N-1} \gamma^{\tau-t} \left(x_\tau^T Q x_\tau + u_\tau^T R u_\tau \right) + \gamma^{N-t} x_N^T Q_f x_N,$$

where $x_t = z$, $x_{\tau+1} = Ax_\tau + Bu_\tau$, and the minimum is over $u_t, \ldots, u_{N-1}$. This is the minimum cost-to-go, if we started in state z at time t, with the time weighting also starting at t. Argue that we have

$$W_N(z) = z^T Q_f z,$$

$$W_t(z) = \min_w \left(z^T Q z + w^T R w + \gamma W_{t+1}(Az + Bw) \right),$$

and that the minimizing w is in fact $u_t^\star$. In other words, work out a backwards recursion for W_t, and give an expression for $u_t^\star$ in terms of W_t. Show that this method yields the same $u^\star$ as the first method.

(c) Yet another method can be used to find $u^\star$. Define a new system as

$$y_{t+1} = \gamma^{1/2} A y_t + \gamma^{1/2} B z_t, \qquad y_0 = x_0.$$

Argue that we have

$$y_t = \gamma^{t/2} x_t,$$

provided

$$z_t = \gamma^{t/2} u_t,$$

for $t = 0, \ldots, N-1$. With this choice of z, the exponentially weighted LQR cost J for the original system is given by

$$J = \sum_{\tau=0}^{N-1} \left(y_\tau^T Q y_\tau + z_\tau^T R z_\tau \right) + y_N^T Q_f y_N,$$

i.e., the unweighted LQR cost for the modified system. We can use the standard formulas to obtain the optimal input for the modified system $z^\star$, and from this, we can get $u^\star$. Do this, and verify that once again, you get the same $u^\star$.

Solution.

(a) Using the standard finite-horizon LQR recursion with the time-varying weights

$$Q_t = \gamma^t Q, \qquad R_t = \gamma^t R,$$

and terminal cost matrix $\gamma^N Q_f$, define

$$P_N = \gamma^N Q_f.$$

Then, for $t = N-1, \ldots, 0$, set

$$K_t = -\left(\gamma^t R + B^T P_{t+1} B\right)^{-1} B^T P_{t+1} A$$

and

$$P_t = \gamma^t Q + A^T P_{t+1} A - A^T P_{t+1} B \left(\gamma^t R + B^T P_{t+1} B\right)^{-1} B^T P_{t+1} A.$$

The optimal input is then

$$u_t^\star = K_t x_t, \qquad t = 0, \ldots, N-1.$$

This is just the usual Riccati recursion with the stage costs replaced by Q_t and R_t.

(b) First note that

$$W_N(z) = z^T Q_f z,$$

since no input remains to be chosen at time N. For $t < N$, if we choose $u_t = w$, then the next state is $Az + Bw$, and the remaining cost is γ times the cost-to-go from this next state with the weights restarted. Thus

$$W_t(z) = \min_w \left(z^T Q z + w^T R w + \gamma W_{t+1}(Az + Bw) \right).$$

The true cost-to-go in the original problem, starting from $x_t = z$, is $\gamma^t W_t(z)$. Since $\gamma^t > 0$, multiplying by this constant does not change the minimizing first input. Thus

$$u_t^\star = \operatorname*{argmin}_w \left(x_t^T Q x_t + w^T R w + \gamma W_{t+1}(Ax_t + Bw) \right).$$

Equivalently, the first-order condition before substituting the quadratic form of W_{t+1} is

$$2Ru_t^\star + \gamma B^T \nabla W_{t+1}(Ax_t + Bu_t^\star) = 0.$$

We now work out the recursion explicitly. Write

$$W_t(z) = z^T S_t z.$$

Then $S_N = Q_f$. Assuming $W_{t+1}(z) = z^T S_{t+1} z$, we have

$$W_t(z) = \min_w \left(z^T Q z + w^T R w + \gamma (Az + Bw)^T S_{t+1} (Az + Bw) \right).$$

Since $R \succ 0$, $\gamma > 0$, and $S_{t+1} \succeq 0$, the matrix $R + \gamma B^T S_{t+1} B$ is positive definite. The first-order optimality condition for w is

$$\left(R + \gamma B^T S_{t+1} B\right) w + \gamma B^T S_{t+1} A z = 0,$$

so

$$u_t^\star = K_t x_t,$$

where

$$K_t = -\left(R + \gamma B^T S_{t+1} B\right)^{-1} \gamma B^T S_{t+1} A.$$

Substituting this minimizing value into the cost gives the backward recursion

$$S_t = Q + \gamma A^T S_{t+1} A - \gamma^2 A^T S_{t+1} B \left(R + \gamma B^T S_{t+1} B\right)^{-1} B^T S_{t+1} A,$$

for $t = N-1, \ldots, 0$.

This is the same control law as in part (a). Indeed, if we define

$$P_t = \gamma^t S_t,$$

then $P_N = \gamma^N Q_f$, and the gain from part (a) becomes

$$\begin{aligned} -\left(\gamma^t R + B^T P_{t+1} B\right)^{-1} B^T P_{t+1} A &= -\left(\gamma^t R + \gamma^{t+1} B^T S_{t+1} B\right)^{-1} \gamma^{t+1} B^T S_{t+1} A \\ &= -\left(R + \gamma B^T S_{t+1} B\right)^{-1} \gamma B^T S_{t+1} A, \end{aligned}$$

which is exactly the gain obtained above. Multiplying the recursion for S_t by γ^t also gives the recursion for P_t in part (a).

(c) Let

$$A_\gamma = \sqrt{\gamma} A, \qquad B_\gamma = \sqrt{\gamma} B.$$

The modified system is

$$y_{t+1} = A_\gamma y_t + B_\gamma z_t.$$

We first verify the relation between the original and modified trajectories. Clearly $y_0 = x_0$. If $y_t = \gamma^{t/2} x_t$ and $z_t = \gamma^{t/2} u_t$, then

$$\begin{aligned} y_{t+1} &= \sqrt{\gamma} A y_t + \sqrt{\gamma} B z_t \\ &= \sqrt{\gamma} A \gamma^{t/2} x_t + \sqrt{\gamma} B \gamma^{t/2} u_t \\ &= \gamma^{(t+1)/2} (Ax_t + Bu_t) \\ &= \gamma^{(t+1)/2} x_{t+1}. \end{aligned}$$

Therefore $y_t = \gamma^{t/2} x_t$ for all t. Also,

$$y_t^T Q y_t = \gamma^t x_t^T Q x_t, \qquad z_t^T R z_t = \gamma^t u_t^T R u_t,$$

and

$$y_N^T Q_f y_N = \gamma^N x_N^T Q_f x_N.$$

Thus the original exponentially weighted problem is equivalent, under the one-to-one change of variables $z_t = \gamma^{t/2} u_t$, to the ordinary finite-horizon LQR problem for the modified system.

Applying the standard LQR recursion to the modified system gives

$$S_N = Q_f,$$

and, for $t = N-1, \ldots, 0$,

$$L_t = -\left(R + B_\gamma^T S_{t+1} B_\gamma\right)^{-1} B_\gamma^T S_{t+1} A_\gamma.$$

Substituting $A_\gamma = \sqrt{\gamma} A$ and $B_\gamma = \sqrt{\gamma} B$ yields

$$L_t = -\left(R + \gamma B^T S_{t+1} B\right)^{-1} \gamma B^T S_{t+1} A,$$

and the Riccati recursion is

$$\begin{aligned} S_t &= Q + A_\gamma^T S_{t+1} A_\gamma - A_\gamma^T S_{t+1} B_\gamma \left(R + B_\gamma^T S_{t+1} B_\gamma\right)^{-1} B_\gamma^T S_{t+1} A_\gamma \\ &= Q + \gamma A^T S_{t+1} A - \gamma^2 A^T S_{t+1} B \left(R + \gamma B^T S_{t+1} B\right)^{-1} B^T S_{t+1} A. \end{aligned}$$

The optimal input for the modified system is

$$z_t^\star = L_t y_t.$$

Since $z_t = \gamma^{t/2} u_t$ and $y_t = \gamma^{t/2} x_t$, the corresponding input for the original system is

$$u_t^\star = \gamma^{-t/2} z_t^\star = \gamma^{-t/2} L_t y_t = L_t x_t.$$

This gain L_t is the same gain found in part (b), and hence also the same control law found in part (a).

23.1400. Discrete-time finite-horizon LQR. We consider the scalar discrete-time system

$$x_{t+1} = a x_t + u_t,$$

where x_0 is given. Consider the finite-horizon LQR problem

$$V_0(x_0) = \min_{u_0, u_1} \sum_{t=0}^{1} \left(x_t^2 + r u_t^2\right) + s x_2^2,$$

where $r > 0$ and $s \geq 0$. Compute the optimal control inputs u_0 and u_1 as functions of the given initial state x_0, in terms of a, r, and s.

Solution. We work backwards. At the last stage, with $x_1 = z$, the cost-to-go is

$$V_1(z) = z^2 + \min_w \left(rw^2 + s(az + w)^2\right).$$

The minimizing w satisfies

$$rw + s(az + w) = 0,$$

and hence

$$u_1^\star = -\frac{as}{r+s} x_1.$$

Substituting this into the cost gives

$$V_1(z) = P_1 z^2,$$

where

$$P_1 = 1 + a^2 s - a^2 s^2 (r+s)^{-1} = 1 + \frac{a^2 rs}{r+s}.$$

Now use the one-step cost-to-go from time 0:

$$V_0(x_0) = x_0^2 + \min_{u_0} \left(ru_0^2 + P_1(ax_0 + u_0)^2\right).$$

The first-order optimality condition is

$$ru_0 + P_1(ax_0 + u_0) = 0,$$

so

$$u_0^\star = -\frac{aP_1}{r + P_1} x_0.$$

Using the value of P_1 above, and

$$r + P_1 = \frac{r^2 + rs + r + s + a^2 rs}{r+s},$$

this can be written as

$$u_0^\star = -\frac{a(r + s + a^2 rs)}{r^2 + rs + r + s + a^2 rs} x_0.$$

It remains to express $u_1^\star$ in terms of x_0. The optimal state at time 1 is

$$x_1^\star = ax_0 + u_0^\star = \frac{ar}{r + P_1} x_0.$$

Therefore

$$u_1^\star = -\frac{as}{r+s} x_1^\star = -\frac{a^2 rs}{r^2 + rs + r + s + a^2 rs} x_0.$$

Thus the optimal inputs are

$$u_0^\star = -\frac{a(r + s + a^2 rs)}{r^2 + rs + r + s + a^2 rs} x_0,$$

$$u_1^\star = -\frac{a^2 rs}{r^2 + rs + r + s + a^2 rs} x_0.$$

Since $r > 0$ and $s \geq 0$, the denominator is positive, so these expressions are well-defined.