Final exam solutions

1. *Finding an invariant ellipsoid containing some points but not others.*

   (a) Explain how to find an ellipsoid $\mathcal{E} \subseteq \mathbf{R}^n$, centered at 0, that is invariant for $\dot{x} = Ax$, and satisfies $z^{(i)} \in \mathcal{E}$, $i = 1, \ldots, m$, and $w^{(i)} \notin \mathcal{E}$, $i = 1, \ldots, p$. You are given the data A, $z^{(i)}$, and $w^{(i)}$. Your answer can involve any of the methods from the course, *e.g.*, Lyapunov and Riccati equations, LMIs, etc.

   (b) Carry out the method described in part (a) on the numerical instance with data given in `inv_elp_data.m`. There are many valid numerical solutions, so you must show us the script you used, along with corresponding output that verifies that the ellipsoid you found satisfies all the required conditions. For example, to verify that a matrix `Z` is positive definite, say, you can use `min(eig(Z))`, and we need to see the corresponding output (which should be a positive number).

   Solution. We describe the ellipsoid as $\mathcal{E} = \{z \mid z^T P z \leq 1\}$, where $P > 0$. $\mathcal{E}$ is invariant for $\dot{x} = Ax$ if and only if

   $$A^T P + PA \leq 0.$$

   The constraints $z^{(i)} \in \mathcal{E}$, $i = 1, \ldots, m$, and $w^{(i)} \notin \mathcal{E}$, $i = 1, \ldots, p$ can be written as

   $$z^{(i)T} P z^{(i)} \leq 1, \quad i = 1, \ldots, m, \qquad w^{(i)T} P w^{(i)} > 1, \quad i = 1, \ldots, p.$$

   These are linear inequalities in P.

   Thus, we need to find a P that satisfies

   $$\begin{array}{ll} A^T P + PA \leq 0, & P > 0, \\ z^{(i)T} P z^{(i)} \leq 1, & i = 1, \ldots, m, \\ w^{(i)T} P w^{(i)} > 1, & i = 1, \ldots, p. \end{array}$$

   This is a mixture of strict and nonstrict inequalities.

   We can convert the strict inequalities $P > 0$ and $w^{(i)T} P w^{(i)} > 1$ to the non-strict inequalities $P \geq \epsilon I$ and $w^{(i)T} P w^{(i)} \geq 1 + \epsilon$, where ϵ is chosen to be small. If the inequalities are infeasible for a particular ϵ, we decrease ϵ until we obtain a solution. The following Matlab code solves the problem.

```
inv_elp_data;

eps = 1e-6;
cvx_begin sdp
    variable P(n,n) symmetric
```

```
    A'*P+P*A <= 0;
    diag(Z'*P*Z) <= ones(m,1);
    diag(W'*P*W) >= (1+eps)*ones(p,1);
    P >= eps*eye(n);
cvx_end

% check P satisfies conditions
disp(min(eig(P))); disp(max(eig(A'*P+P*A)));
disp(max(diag(Z'*P*Z)));
disp(min(diag(W'*P*W)));
```

2. *Certifying performance with control-Lyapunov controller.*

   A system has the form

   $$x_{t+1} = A_{i(t)}x_t, \quad t = 1, 2, \ldots,$$

   where $x_t \in \mathbf{R}^n$ is the state, $i(t) \in \{1, \ldots, k\}$ gives the dynamics matrix used in time period t, and $A_1, \ldots, A_k$ are given. In this problem, you can *choose* $i(t)$; you can think of $i(t)$ as the (discrete) input to the system at time period t.

   The following control-Lyapunov scheme is used. We are given a matrix $P = P^T > 0$. At time period t, we choose $i(t)$ to minimize $V(x_{t+1})$, where $V(z) = z^T P z$. (If a tie occurs, you can choose any index which achieves the minimum value.)

   The goal is to certify performance of this control scheme; specifically, to verify that, for any x_t, we have

   $$V(x_{t+1}) \leq \alpha V(x_t),$$

   where α is a given value. (If $\alpha < 1$, this shows the closed-loop system is globally asymptotically stable.) 'Exhaustive simulation' suggests this inequality holds for all $x_t \in \mathbf{R}^n$; your job is to *prove* that it holds.

   (a) Describe a computationally tractable method that can verify the inequality above holds for all $x_t \in \mathbf{R}^n$. Your method does not have to always work; but we will give little or no credit for obvious or silly conditions, such as that $A_i^T P A_i \leq \alpha P$ for some i.

   (b) Carry out your method on the problem instance given in `clf_data.m`. You must submit your Matlab code, along with the corresponding output that verifies your certificate.

   Solution. To show that $V(x_{t+1}) \leq \alpha V(x_t)$ for all x_t, we must show that

   $$\min_{i \in \{1,\ldots,k\}} z^T (A_i^T P A_i - \alpha P) z \leq 0,$$

   for all z. To do this we will use the elementary fact that the minimum of a set of numbers is less than or equal to the weighted average: for any $\theta \in \mathbf{R}^k$ with $\theta \geq 0$ and

$\mathbf{1}^T\theta = 1$, and any $c \in \mathbf{R}^k$, we have

$$\min_{i\in\{1,\ldots,k\}} c_i \leq \sum_{i=1}^k \theta_i c_i.$$

Therefore if we can find $\theta \geq 0$ with $\mathbf{1}^T\theta = 1$, for which

$$\sum_{i=1}^k \theta_i (A_i^T P A_i - \alpha P) \leq 0,$$

then we will shave established the required inequality. Determining such a θ is, evidently, an LMI. If we find such a θ, then for any z,

$$\min_{i\in\{1,\ldots,k\}} z^T (A_i^T P A_i - \alpha P) z \leq \sum_{i=1}^k \theta_i z^T (A_i^T P A_i - \alpha P) z \leq 0,$$

and we have certified the performance.

The same condition can be derived using a lossy S-procedure-like condition.

$$\min_{i\in\{1,\ldots,k\}} z^T (A_i^T P A_i - \alpha P) z \leq 0$$

holds for all z if and only if, for all z,

$$z^T (A_i^T P A_i - \alpha P) z > 0,\ i = 1, \ldots, k-1 \quad \Longrightarrow \quad z^T (A_k^T P A_k - \alpha P) z \leq 0.$$

This is not quite the same as any of the S-procedure conditions we've seen, since it mixes strict and nonstrict inequalities. Ignoring such fine details, we arrive at a condition like this: There exist nonnegative $\tau_1, \ldots, \tau_k$ for which

$$A_k^T P A_k - \alpha P \leq -\sum_{i=1}^{k-1} \tau_i (A_i^T P A_i - \alpha P).$$

We rewrite this as

$$\sum_{i=1}^k \theta_i (A_i^T P A_i - \alpha P) \leq 0,$$

with $\theta_i = \tau_i/(1 + \mathbf{1}^T\tau)$, for $i = 1, \ldots, k-1$, and $\theta_k = 1/(1 + \mathbf{1}^T\tau)$.

Now we turn to the specific instance. We solve the LMI using the following Matlab code, and discover that $\tau = (0.19, 0.35, 0.14, 0.29, 0.03)$ fulfills the requirements.

```
% Load data.
clf_data;

cvx_begin sdp
```

```
    variable theta(K);
    theta >= 0; sum(theta) == 1;
    lhs = 0; for k = 1:K, lhs = lhs + theta(k)*(A{k}'*P*A{k} - alpha*P); end
    lhs <= 0;
cvx_end

% Check resulting combination is positive semi-definite.
disp(min(eig(lhs)));
```

Alternative solution. Here's another valid solution we saw. To show that

$$\min_{i\in\{1,\ldots,k\}} z^T A_i^T P A_i z \leq \alpha z^T P z$$

for all z, we consider k cases, depending on which index gives the minimum. Suppose that index j achieves the minimum, *i.e.*,

$$\min_{i\in\{1,\ldots,k\}} z^T A_i^T P A_i z = z^T A_j^T P A_j z.$$

For such z's we need to have

$$z^T A_j^T P A_j z \leq \alpha z^T P z.$$

We can express this is a way suitable for the S-procedure as: for all z,

$$z^T A_j^T P A_j z \leq z^T A_i^T P A_i z,\ i = 1,\ldots,k,\ i \neq j \implies z^T A_j^T P A_j z \leq \alpha z^T P z.$$

A sufficient condition for this is the existence of nonnegative multipliers τ_i with

$$\alpha P - A_j^T P A_j \geq \sum_{i\neq j} \tau_i (A_i^T P A_i - A_j^T P A_j).$$

A similar condition will have to hold for all values of j, *i.e.*, we need the inequalities to hold no matter which index achieves the minimum. We can, of course, use different sets of multipliers to certify each one. Putting it all together, we need to find nonnegative τ_{ij} for which

$$\alpha P - A_j^T P A_j \geq \sum_{i\neq j} \tau_{ij} (A_i^T P A_i - A_j^T P A_j), \quad j = 1,\ldots,k.$$

We can find these by solving k separate LMIs, or just lump them together into one. And yes, this method does work on our numerical instance.

3. *Optimal time series interpolation.* A scalar time series $y_0, y_1, \ldots$ is the output of a linear Gauss-Markov process

$$x_{t+1} = Ax_t + w_t, \qquad y_t = Cx_t + v_t, \quad t = 0, 1, 2, \ldots,$$

where $y_t \in \mathbf{R}$, w_t are IID $\mathcal{N}(0, W)$, v_t are IID $\mathcal{N}(0, V)$ (and independent of all w_t), and $x_0 = 0$.

You are given the numbers y_{T_0} and y_{T_1}, with $0 \leq T_0 < T_1 - 1$. Your goal is to estimate y_t for $T_0 < t < T_1$, *i.e.*, to (approximately) interpolate the time series for the samples that occur between T_0 and T_1. Let $\hat{y}_t$ denote the minimum mean-square error estimate of y_t, for $T_0 < t < T_1$, given y_{T_0} and y_{T_1}.

(a) Explain how to find $\hat{y}_t$, as well as the RMS estimation error $\sigma_t = (\mathbf{E}(y_t - \hat{y}_t)^2)^{1/2}$.

(b) Plot $\hat{y}_t \pm \sigma_t$ for the specific problem instance with

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0.5 & -0.8 & 1 \end{bmatrix},$$

$W = 0.1I$, $V = 0.1$, $T_0 = 5$, $T_1 = 35$, $y_{T_0} = -4$ and $y_{T_1} = 4$. (To plot a figure with error bars, you can use the Matlab function `errorbar`.)

Solution. We begin by expressing y_t, y_{T_0} and y_{T_1} in terms of the random variables $w_0, \ldots, w_{T_1-1}$, v_t, v_{T_0}, and v_{T_1},

$$\begin{aligned} y_t &= CA^{t-1}w_0 + \cdots + Cw_{t-1} + v_t, \\ y_{T_0} &= CA^{T_0-1}w_0 + \cdots + Cw_{T_0-1} + v_{T_0}, \\ y_{T_1} &= CA^{T_1-1}w_0 + \cdots + Cw_{T_1-1} + v_{T_1}. \end{aligned}$$

Defining $z_t = (y_t, y_{T_0}, y_{T_1})$, we can write this as $z_t = G_t\tilde{w}_t$, where

$$\tilde{w}_t = (w_0, \ldots, w_{T_1-1}, v_t, v_{T_0}, v_{T_1}),$$

and

$$G_t = \begin{bmatrix} CA^{t-1} & \cdots & C & 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \\ CA^{T_0-1} & \cdots & \cdots & C & 0 & \cdots & 0 & 0 & 1 & 0 \\ CA^{T_1-1} & \cdots & \cdots & \cdots & \cdots & \cdots & C & 0 & 0 & 1 \end{bmatrix}.$$

Thus $z_t \sim \mathcal{N}(0, \Sigma_t)$, with $\Sigma_t = G_t\tilde{W}G_t^T$, and

$$\tilde{W} = \begin{bmatrix} W & & & & & & \\ & \ddots & & & & & \\ & & W & & & & \\ & & & V & & & \\ & & & & V & & \\ & & & & & V & \end{bmatrix}.$$

We can write Σ_t as

$$\Sigma_t = \begin{bmatrix} \Sigma_{11}^{(t)} & \Sigma_{12}^{(t)} \\ \Sigma_{12}^{(t)T} & \Sigma_{22}^{(t)} \end{bmatrix}$$

where $\Sigma_{11}^{(t)} \in \mathbf{R}$, $\Sigma_{12}^{(t)} \in \mathbf{R}^{1\times 2}$, $\Sigma_{22}^{(t)} \in \mathbf{R}^{2\times 2}$. Therefore, the minimum mean-square error estimate for y_t, given y_{T_0} and y_{T_1} is

$$\hat{y}_t = \Sigma_{12}^{(t)}\Sigma_{22}^{(t)-1} \begin{bmatrix} y_{T_0} \\ y_{T_1} \end{bmatrix},$$

and the minimum RMS estimation error is

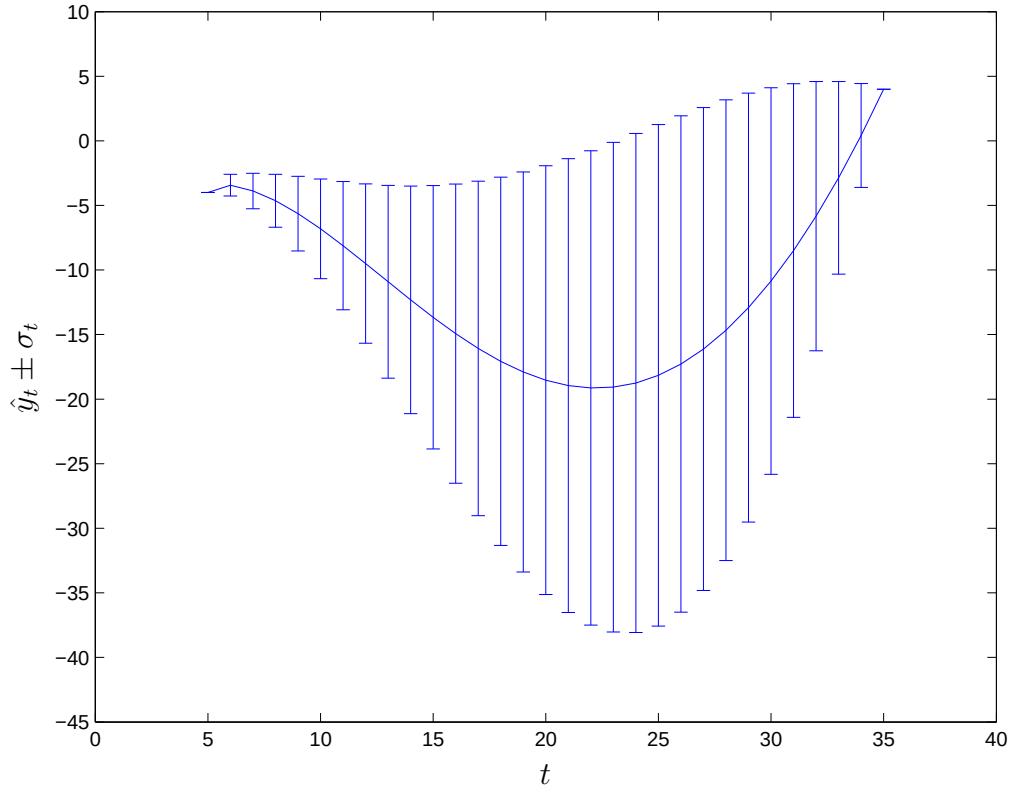
$$\sigma_t = \left(\mathbf{E}(y_t - \hat{y}_t)^2\right)^{1/2} = \left(\Sigma_{11}^{(t)} - \Sigma_{12}^{(t)}\Sigma_{22}^{(t)-1}\Sigma_{12}^{(t)T}\right)^{1/2}.$$

The following Matlab code solves the problem instance in part (b).

```
% problem data
n = 3;
A = [1,1,0;0,1,1;0,0,1];
C = [0.5,-0.8,1];
W = 0.1*eye(n); V = 0.1;
T0 = 5; T1 = 35;
yT0 = -4; yT1 = 4;

G2 = [];
for i = 1:T0 G2 = [G2,C*A^(T0-i)]; end;
for i = 1:T1-T0 G2 = [G2,zeros(1,n)]; end;
G2 = [G2,[0,1,0]];
G3 = [];
for i = 1:T1 G3 = [G3,C*A^(T1-i)]; end;
G3 = [G3,[0,0,1]];

Wtilde = [];
for i = 1:T1 Wtilde = blkdiag(Wtilde,W); end;
Wtilde = blkdiag(Wtilde,V,V,V);

rmse = zeros(T1-T0+1,1);
rmse(1) = 0; rmse(T1-T0+1) = 0;
yhat = zeros(T1-T0+1,1);
yhat(1) = yT0; yhat(T1-T0+1) = yT1;

for t = T0+1:T1-1
    G1 = [];
    for i = 1:t G1 = [G1,C*A^(t-i)]; end;
```

```
    for i = 1:T1-t G1 = [G1,zeros(1,n)]; end;
    G1 = [G1,[1,0,0]];
    G = [G1; G2; G3];
    Sig = G*Wtilde*G';
    rmse(t-T0+1) = ...
        sqrt(Sig(1,1)-Sig(1,2:3)*inv(Sig(2:3,2:3))*Sig(1,2:3)');
    yhat(t-T0+1) = Sig(1,2:3)*inv(Sig(2:3,2:3))*[yT0;yT1];
end

tvec = [T0:1:T1]';
errorbar(tvec,yhat,rmse);
axis([0,40,-45,10]);
ylabel('y'); xlabel('t');
print('-depsc','interp_fig.eps');
```

The following figure shows $\hat{y}_t \pm \sigma_t$.



4. *Optimal control with random dynamics matrix.* Consider a linear discrete-time system

$$x_{t+1} = A_t x_t + B u_t, \quad t = 1, \ldots, N-1,$$

where

$$A_t = \begin{cases} A^{(1)}, & \text{with probability } 1/2 \\ A^{(2)}, & \text{with probability } 1/2. \end{cases}$$

In other words, at each time period t, the dynamics matrix assumes the value $A^{(1)}$ or the value $A^{(2)}$, with equal probability. The cost J, which is to be minimized, is

$$J = \mathbf{E}\left(\sum_{t=1}^{N-1}\left(x_t^T Q x_t + u_t^T R u_t\right) + x_N^T Q x_N\right),$$

where $Q = Q^T \geq 0$ and $R = R^T > 0$ are state and input cost matrices, respectively. The matrices $A^{(1)}$, $A^{(2)}$, B, Q, and R are given.

We will consider two different scenarios.

- At each time period t, you must commit to (*i.e.*, choose) u_t knowing only x_t, *but not* A_t. (In other words: the input is chosen *before* A_t is revealed.)
- At each time period t, you choose u_t knowing x_t *and* A_t. (In other words: the input is chosen *after* A_t is revealed.)

(In both cases, you do not know $A_{t+1}, \ldots, A_{N-1}$.)

Now, the question.

(a) Show that the optimal control in the first scenario has the form $u_t^\star = K_t x_t$, for some matrices $K_1, \ldots, K_{N-1}$. Explain how to find the matrices $K_1, \ldots, K_{N-1}$.

(b) Show that the optimal control in the second scenario has the form

$$u_t^\star = \begin{cases} K_t^{(1)} x_t, & A_t = A^{(1)}, \\ K_t^{(2)} x_t, & A_t = A^{(2)}, \end{cases}$$

for some matrices $K_1^{(1)}, \ldots, K_{N-1}^{(1)}$, $K_1^{(2)}, \ldots, K_{N-1}^{(2)}$. Explain how to find these matrices.

(c) Find the optimal value of J for the two scenarios for the specific problem instance with

$$A^{(1)} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad A^{(2)} = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1.3 \\ 0 & -2 \\ 1 & 0 \end{bmatrix},$$

$Q = I$, $R = I$, $x_1 = (1, 1, 1)$, and $N = 50$.

Solution. We can find the optimal control for the two cases using dynamic programming.

(a) In this scenario we must choose u_t before knowing A_t. Let $V_t(z)$ be the minimum expected cost-to-go starting from state $x_t = z$ at time t; in particular, before knowing the value of A_t. We will show that $V_t(z)$ is quadratic, and can be found from backward recursion. Along the way, we will get the optimal control policy, which is linear in x_t.

We have $V_N(z) = z^T Q z$. Now we look at the inductive step. We assume that $V_{t+1}(z)$ has the form $V_{t+1}(z) = z^T P_{t+1} z$. We'll use this to show that $u_t^\star = K_t x_t$, and that $V_t(z) = z^T P_t z$ for some P_t (which we'll give a formula for).

Now let's imagine that we choose $u_t = v$ as the input at time t. If $A_t = A^{(1)}$, then we have $x_{t+1} = A^{(1)} z + Bv$; if $A_t = A^{(2)}$, we have $x_{t+1} = A^{(2)} z + Bv$. In the first case, the expected optimal cost-to-go from x_{t+1} is

$$(A^{(1)} z + Bv)^T P_{t+1} (A^{(1)} z + Bv),$$

and in the second case, it is

$$(A^{(2)} z + Bv)^T P_{t+1} (A^{(2)} z + Bv).$$

The expected current cost plus cost-to-go, if we choose $u_t = v$, is therefore

$$z^T Q z + v^T R v + (1/2)(A^{(1)} z + Bv)^T P_{t+1} (A^{(1)} z + Bv) + (1/2)(A^{(2)} z + Bv)^T P_{t+1} (A^{(2)} z + Bv).$$

Rewriting as a quadratic form in (v, z), we get

$$\begin{bmatrix} v \\ z \end{bmatrix}^T \begin{bmatrix} R + B^T P_{t+1} B & \frac{1}{2} B^T P_{t+1} (A^{(1)} + A^{(2)}) \\ \frac{1}{2}(A^{(1)} + A^{(2)})^T P_{t+1} B & Q + \frac{1}{2} A^{(1)T} P_{t+1} A^{(1)} + \frac{1}{2} A^{(2)T} P_{t+1} A^{(2)} \end{bmatrix} \begin{bmatrix} v \\ z \end{bmatrix}.$$

Hence the minimum expected cost-to-go, $V_t(z)$, is given by

$$V_t(z) = \min_v \left\{ \begin{bmatrix} v \\ z \end{bmatrix}^T \begin{bmatrix} R + B^T P_{t+1} B & \frac{1}{2} B^T P_{t+1} (A^{(1)} + A^{(2)}) \\ \frac{1}{2}(A^{(1)} + A^{(2)})^T P_{t+1} B & Q + \frac{1}{2} A^{(1)T} P_{t+1} A^{(1)} + \frac{1}{2} A^{(2)T} P_{t+1} A^{(2)} \end{bmatrix} \begin{bmatrix} v \\ z \end{bmatrix} \right\}.$$

Minimizing a quadratic form over some of its variables results in a quadratic form in the remaining variables. Specifically, we get

$$V_t(z) = z^T P_t z,$$

where $P_t \geq 0$, is the Schur complement of $R + B^T P_{t+1} B$ in the matrix above,

$$\begin{aligned} P_t &= Q + (1/2) A^{(1)T} P_{t+1} A^{(1)} + (1/2) A^{(2)T} P_{t+1} A^{(2)} \\ &\quad -(1/4)(A^{(1)} + A^{(2)})^T P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} (A^{(1)} + A^{(2)}). \end{aligned}$$

This gives a backwards recursion for computing P_t, starting from $P_N = Q$.

The minimizer over v is the optimal control. It can be expressed as $u_t^\star = K_t x_t$, where

$$K_t = -(1/2)(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} (A^{(1)} + A^{(2)}).$$

(b) Here A_t is known before choosing u_t. Let $V_t(z)$ be the minimum expected cost-to-go starting from state $x_t = z$ at time t, but *before knowing* $A(t)$. We have $P_N = Q$.

Let's look at the inductive step. We will treat the two cases, $A_t = A^{(1)}$ and $A_t = A^{(2)}$, separately.

If $A_t = A^{(1)}$, we choose $u_t = v^{(1)}$ to minimize the current input cost $v^{(1)T} R v^{(1)}$ plus the optimal cost-to-go from $x_{t+1} = A^{(1)} x_t + B v^{(1)}$, which is

$$(A^{(1)} z + B v^{(1)})^T P_{t+1} (A^{(1)} z + B v^{(1)}).$$

This gives $u_t = K_t^{(1)} x_t$, where

$$K_t^{(1)} = -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A^{(1)}.$$

With this value of u_t, we find that the expected cost-to-go from x_t, conditioned on $A_t = A^{(1)}$, is $x_t^T P^{(1)} x_t$, where

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

(These formulas are the standard ones from LQR.)

Now suppose that (*i.e.*, condition on) $A_t = A^{(2)}$. In the same way we get $u_t = K_t^{(2)} x_t$, where

$$K_t^{(2)} = -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A^{(2)},$$

and the expected cost-to-go from x_t is $x_t^T P_t^{(1)} x_t$, where

$$P_t^{(2)} = Q + A^{(2)T} P_{t+1} A^{(2)} - A^{(2)T} P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A^{(2)}.$$

The optimal expected cost-to-go from x_t is therefore $x_t^T P_t x_t$, where

$$P_t = (1/2)(P_t^{(1)} + P_t^{(2)}).$$

Thus the optimal control policy is

$$\varphi_t^*(x_t) = \begin{cases} K_t^{(1)} x_t, & \text{when } A_t = A^{(1)} \\ K_t^{(2)} x_t, & \text{when } A_t = A^{(2)}, \end{cases}$$

where

$$\begin{aligned} K_t^{(1)} &= -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A^{(1)} \\ K_t^{(2)} &= -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A^{(2)}, \end{aligned}$$

and

$$\begin{aligned} P_t^{(1)} &= Q + A^{(1)T} P_{t+1} A^{(1)} - A^{(1)T} P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A^{(1)} \\ P_t^{(2)} &= Q + A^{(2)T} P_{t+1} A^{(2)} - A^{(2)T} P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A^{(2)}. \end{aligned}$$

(c) The following Matlab code solves the problem for the particular problem instance in part (c)

```
% dimensions and data
n = 3; m = 2; N = 50;
% initial state
x1= ones(n,1);
% system matrices
A1 = [1,1,0;0,1,1;0,0,1];A2 = [2,1,0;0,1,1;0,0,1];B = [0,1.3;0,-2;1,0];
% cost matrices
Q = eye(n);R = eye(m);

disp('Computing optimal cost when At is not known')
% Riccati recursion
P=zeros(n,n,N); P(:,:,N) = Q;
for i = N-1:-1:1
    P(:,:,i)=Q+0.5*A1'*P(:,:,i+1)*A1+0.5*A2'*P(:,:,i+1)*A2-0.25*(A1+A2)'*...
P(:,:,i+1)*B*pinv(R+B'*P(:,:,i+1)*B)*B'*P(:,:,i+1)*(A1+A2);
end
% optimal cost when At is not known
J=x1'*P(:,:,1)*x1;
J

disp('Computing optimal cost assuming At is known')
% Riccati recursion
P=zeros(n,n,N); P(:,:,N) = Q;
P1 = P; P2 = P;
for i = N-1:-1:1
    P1(:,:,i)=Q+A1'*P(:,:,i+1)*A1-A1'*...
P(:,:,i+1)*B*pinv(R+B'*P(:,:,i+1)*B)*B'*P(:,:,i+1)*A1;
    P2(:,:,i)=Q+A2'*P(:,:,i+1)*A2-A2'*...
P(:,:,i+1)*B*pinv(R+B'*P(:,:,i+1)*B)*B'*P(:,:,i+1)*A2;
    P(:,:,i) = 0.5*(P1(:,:,i)+P2(:,:,i));
end
% optimal cost assuming At is known
J=x1'*P(:,:,1)*x1;
J
```

The optimal costs are 355.23 and 191.11, respectively. The difference can be interpreted as the value of knowing A_t before selecting the input u_t.

5. *Control with unreliable actuators.* A control system is modeled as

$$x_{t+1} = Ax_t + B\Delta_t u_t, \qquad u_t = Fx_t,$$

for $t = 0, 1, \ldots$. Here $x_t \in \mathbf{R}^n$ is the plant state, and $u_t \in \mathbf{R}^m$ is the actuator signal. The matrices A, B, and F, and the initial state x_0, are given.

The matrix Δ_t is used to model actuator failures. It is diagonal, and satisfies $(\Delta_t)_{ii} \in [0, 1]$ for all t and all i, but is otherwise unknown. $(\Delta_t)_{ii} = 1$ means the ith actuator is functioning correctly at time period t; $(\Delta_t)_{ii} = 0$ means the ith actuator has failed completely at time period t; $(\Delta_t)_{ii} = 0.8$, for example, means the ith actuator has suffered a 20% reduction in effectiveness at time period t. (This is sometimes called a *soft failure*.)

In this problem, your job is to establish an upper bound on the objective

$$J = \sum_{t=0}^{\infty} x_t^T Q x_t,$$

where $Q = Q^T > 0$ is given. In other words, you must find a number $\tilde{J}$ such that $J \leq \tilde{J}$ for any allowed values of $\Delta_1, \Delta_2, \ldots$.

(a) Describe an effective (*i.e.*, tractable) method for finding an upper bound on J. You method does not have to give the best possible bound, but if there is the possibility of optimizing your bound, please do so. We will give no credit for trivial bounds such as $\tilde{J} = \infty$, or bounds that are not tractable to compute. Your answer can involve any of the methods from the course, *e.g.*, LQR, Lyapunov and Riccati equations, Kalman filter, LMIs, *etc.*.

(b) Carry out your method for the problem instance with data given in the file `unrelact_data.m`. Report the value of your bound for this problem instance. For comparison, give the value of J when there are no actuator failures (*i.e.*, when $\Delta_t = I$ for all t), and when all actuators completely fail all the time (*i.e.*, when $\Delta_t = 0$ for all t).

Solution. We will give two different solutions, both of which we accepted. The two methods differ in the quality of the bound and computational complexity; we'll discuss this after we give both solutions. The first solution is the one we expected, since the second solution requires some knowledge of convex analysis.

An S-procedure solution. We define $v_t = \Delta_t u_t$, so $x_{t+1} = Ax_t + Bv_t$. We can characterize the condition relating v_t and u_t via a set of m quadratic inequalities: for $i = 1, \ldots, m$, we have

$$(v_t)_i = \delta(u_t)_i \text{ for some } \delta \in [0, 1] \iff (v_t)_i^2 - (v_t)_i(u_t)_i \leq 0.$$

To find an upper bound on J, we seek a Lyapunov function $V(z) = z^T P z$, with $P \geq 0$, for which

$$V(Az + Bv) = (Az + Bv)^T P(Az + Bv) \leq V(z) - z^T Q z = z^T (P - Q) z$$

whenever the m quadratic inequalities

$$v_i^2 - v_i e_i^T F z \leq 0, \quad i = 1, \ldots, m,$$

hold. If we can find a $P \geq 0$ that satisfies this condition, then we have $J \leq x_0^T P x_0$, *i.e.*, we have the upper bound $\tilde{J} = x_0^T P x_0$. We write the first inequality as

$$\left[\begin{array}{c} z \\ v \end{array} \right]^T \left[\begin{array}{cc} A^T P A - P + Q & A^T P B \\ B^T P A & B^T P B \end{array} \right] \left[\begin{array}{c} z \\ v \end{array} \right] \leq 0,$$

and the m inequalities as

$$\left[\begin{array}{c} z \\ v \end{array} \right]^T \left[\begin{array}{cc} 0 & -(1/2) F^T e_i e_i^T \\ -(1/2) e_i e_i^T F & e_i e_i^T \end{array} \right] \left[\begin{array}{c} z \\ v \end{array} \right] \leq 0, \quad i = 1, \ldots, m.$$

Now, using the lossy S-procedure, the implication holds when there exist nonnegative $\tau_1, \ldots, \tau_m$ for which

$$\left[\begin{array}{cc} A^T P A - P + Q & A^T P B \\ B^T P A & B^T P B \end{array} \right] + \sum_{i=1}^m \tau_i \left[\begin{array}{cc} 0 & (1/2) F^T e_i e_i^T \\ (1/2) e_i e_i^T F & -e_i e_i^T \end{array} \right] \leq 0.$$

We rewrite this as

$$\left[\begin{array}{cc} A^T P A - P + Q & A^T P B + (1/2) F^T D \\ B^T P A + (1/2) D F & B^T P B - D \end{array} \right] \leq 0,$$

where $D = \mathbf{diag}(\tau_1, \ldots, \tau_m)$. Thus, we can find an upper bound on J by finding $P = P^T \geq 0$, and diagonal $D \geq 0$, with

$$\left[\begin{array}{cc} A^T P A - P + Q & A^T P B + (1/2) F^T D \\ B^T P A + (1/2) D F & B^T P B - D \end{array} \right] \leq 0.$$

This is an LMI problem with variables P and D.

Finally, we can optimize over this family of bounds by solving the SDP

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(P x_0 x_0^T) \\ \text{subject to} & \left[\begin{array}{cc} A^T P A - P + Q & A^T P B + (1/2) F^T D \\ B^T P A + (1/2) D F & B^T P B - D \end{array} \right] \leq 0 \\ & P \geq 0, \quad D \geq 0, \end{array}$$

with symmetric P and diagonal D as variables.

The following Matlab script carries out this method:

```
unrelact_data;

cvx_begin sdp
    variable P(n,n) symmetric
    variable D(m,m) diagonal
    minimize (x0'*P*x0);
      P >= 0;
      D >= 0;
      [A'*P*A-P+Q  A'*P*B+0.5*F'*D; ...
       B'*P*A+0.5*D*F  B'*P*B-D] <= 0;
cvx_end
Jbound = cvx_optval

% find J with no failures
Pnofailures = dlyap((A+B*F)',Q);
Jnofailures = x0'*Pnofailures*x0

% find J with complete failures all the time
Pallfailures = dlyap(A',Q);
Jallfailures = x0'*Pallfailures*x0

% vertex enumeration method (an alternative solution)

cvx_begin sdp
    variable P(n,n) symmetric
    minimize (x0'*P*x0);
      P >= 0;
      (A+B*diag([0 0 0])*F)'*P*(A+B*diag([0 0 0])*F) <= P-Q;
      (A+B*diag([0 0 1])*F)'*P*(A+B*diag([0 0 1])*F) <= P-Q;
      (A+B*diag([0 1 0])*F)'*P*(A+B*diag([0 1 0])*F) <= P-Q;
      (A+B*diag([0 1 1])*F)'*P*(A+B*diag([0 1 1])*F) <= P-Q;
      (A+B*diag([1 0 0])*F)'*P*(A+B*diag([1 0 0])*F) <= P-Q;
      (A+B*diag([1 0 1])*F)'*P*(A+B*diag([1 0 1])*F) <= P-Q;
      (A+B*diag([1 1 0])*F)'*P*(A+B*diag([1 1 0])*F) <= P-Q;
      (A+B*diag([1 1 1])*F)'*P*(A+B*diag([1 1 1])*F) <= P-Q;
cvx_end
Jbound_vertex = cvx_optval
```

For the particular problem instance in `unrelact_data.m`, the best bound obtained by our method is $\tilde{J} = 11.43$. When there are no actuator failures, we have $J = 2.32$. When there all actuators fail all the time, we have $J = 9.88$.

Approximate worst-case simulation. We didn't ask you to do it, but just for fun we worked out a heuristic approximate worst-case simulation method for this problem. We use the Lyapunov function $V(z) = z^T P z$ found to establish the upper bound on J. At each time instance t, we choose Δ_t (within its allowed values) so as to maximize $\dot{V}(x_t)$. Our variables are $(\Delta_t)_{ii}$, each of which must lie between 0 and 1; we have to maximize a convex quadratic function of these variables. It can be shown that the maximum must occur when each $(\Delta_t)_{ii}$ is either 0 or 1. So to maximize $\dot{V}(z_t)$ over the valid choices for Δ_t, we simply check all 2^m cases with $(\Delta_t)_{ii} \in \{0, 1\}$, and take the worst (*i.e.*, largest) one.

To evaluate J with this choice of Δ, we use a simple numerical integration scheme. We choose a small value of h, a time step, and proceed as follows. At time $t = kh$, we choose the value of Δ as described above. Then we use Δ_t to be this value over the interval $[kh, (k+1)h)$. We can update the state using the simple formula $x((k+1)h) \approx (I + h(A + B\Delta F))x(kh)$. We can approximate J as

$$J \approx \sum_{k=0}^{N} hx(kh)^T Q x(kh).$$

(It's also possible to update x exactly, using $x((k+1)h) = \exp h(A + B\Delta F)x(kh)$, and to evaluate the integral over each interval exactly, but for h small, this isn't really needed.)

The code for this is given below.

```
% Approximate worst-case simulation.
% Only works for the case with m = 3.

% Load data and get P from upper bound.
unrelact_sol;

Jawc = 0; x = x0;
while x'*Q*x > 1e-3
    % Add the contribution from this step.
    Jawc = Jawc + x'*Q*x;
    % Find the worst D.
    maxval = -inf;
    for d1 = [0,1]
        for d2 = [0,1]
            for d3 = [0,1]
                D = diag([d1,d2,d3]);
                newval = x'*(F'*D*B'*P*A + A'*P*B*D*F + F'*D*B'*P*B*D*F)*x;
                if newval > maxval
                    maxval = newval;
```

```
                    worstD = D;
                end
            end
        end
    end
    % Propagate with the worst D.
    x = A*x + B*worstD*F*x;
end
Jawc
```

This approximate worst-case simulation gives the value $J = 10.0$. So our upper bound is at most about 14% too high, and perhaps much lower. It also means that usually, but not always, the worst-case scenario is that all actuators suffer total failure.

Vertex enumeration solution. We seek $P \geq 0$ for which

$$z^T(A + B\Delta F)^T P(A + B\Delta F)z \leq z^T(P - Q)z$$

for all z, and all Δ that are diagonal, with $\Delta_{ii} \in [0, 1]$. Let us fix z and work out the maximum possible value of the lefthand side, as Δ ranges over its allowed values.

Consider the lefthand side as a function of $\delta = \mathbf{diag}(\Delta)$. The lefthand side is in fact a quadratic function of δ, and is always nonnegative, since $P \geq 0$. Thus the lefthand side is a convex function of δ. It is a standard fact from convex analysis that the maximum of a convex function over a convex set (in this case a box, described by $0 \leq \delta_i \leq 1$) occurs at one of the vertices (corners) of the set. In other words, the maximum value of the lefthand side is

$$\max_i z^T(A + B\Delta^{(i)}F)^T P(A + B\Delta^{(i)}F)z,$$

for $i = 1, \ldots, 2^m$, where $\Delta^{(i)}$ are the diagonal matrices with $\Delta_{ii} \in \{0, 1\}$. Thus the condition above holds if and only if

$$(A + B\Delta^{(i)}F)^T P(A + B\Delta^{(i)}F) \leq P - Q, \quad i = 1, \ldots, 2^m.$$

This is a set of LMIs. Now we optimize our bound by solving the SDP

$$\begin{array}{ll} \text{minimize} & x_0^T P x_0 \\ \text{subject to} & P \geq 0, \quad D \geq 0, \\ & (A + B\Delta^{(i)}F)^T P(A + B\Delta^{(i)}F) \leq P - Q, \quad i = 1, \ldots, 2^m, \end{array}$$

with symmetric P as variable.

The resulting bound, on the numerical problem instance given, is 11.39, very slightly better than the bound obtained using the S-procedure method.

Comparison of methods. The vertex enumeration method gives the necessary and sufficient condition for the existence of a quadratic Lyapunov function that establishes a given level of bound, so it must give a bound that is better (or the same) as the S-procedure method. On the other hand the size of the LMI grows exponentially in m, the number of actuators. But if m is modest (say, under 15 or so), we can still carry out the required optimization. The numerical instance has only $2^3 = 8$ $\Delta^{(i)}$'s, which is no problem.