7.1080. Hovercraft with limited range. We have a hovercraft moving in the plane with two thrusters, each pointing through the center of mass, exerting forces in the x and y directions with 100% efficiency. The hovercraft has mass 1. The discretized equations of motion for the hovercraft are

\[
x(t + 1) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & 0 \\ 0 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}
\]

where \(x_1\) and \(x_2\) are the position and velocity in the x-direction, and \(x_3, x_4\) are the position and velocity in the y-direction. Here

\[
u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}
\]

is the force acting on the hovercraft for time in the interval \([t, t + 1)\). Let the position of the vehicle at time \(t\) be \(q(t) \in \mathbb{R}^2\).

a) The hovercraft starts at the origin. We’d like to apply thrust to make it move through points \(p_1, p_2, p_3\) at times \(t_1, t_2, t_3\), where

\[
p_1 = \begin{bmatrix} 1 \\ \frac{1}{2} \end{bmatrix}, \quad p_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad p_3 = \begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}
\]

\(t_1 = 6, \quad t_2 = 40, \quad t_3 = 50\)

We will run the hovercraft on the time interval \([0, 70]\). We’d like to apply a sequence of inputs \(u(0), u(1), \ldots, u(70)\) to make the hovercraft position pass through the above sequence of points at the specified times.

We would like to find the sequence of inputs that drives the hovercraft through the desired points which has the minimum cost, given by the sum of the squares of the forces:

\[
\sum_{t=0}^{70} \|u(t)\|^2
\]

To do this, pick \(A_{\text{hov}}\) and \(y_{\text{des}}\) to set this problem up as an equivalent minimum-norm problem, where we would like to find the minimum-norm \(u_{\text{seq}}\) which satisfies

\[
A_{\text{hov}}u_{\text{seq}} = y_{\text{des}}
\]

where \(u_{\text{seq}}\) is the sequence of force inputs

\[
u_{\text{seq}} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(70) \end{bmatrix}
\]
Plot the trajectory of the hovercraft using this input, and the way-points $p_1, \ldots, p_3$. Also plot the optimal $u$ against time.

b) Now we would like to compute the trade-off curve between the accuracy with which the mass passes through the waypoints and the norm of the force used. Let our two objective functions be

$$J_1 = \sum_{i=1}^{3} \|q(t_i) - p_i\|^2 = \|A_{\text{hov}} u_{\text{seq}} - y_{\text{des}}\|^2$$

and

$$J_2 = \sum_{t=0}^{70} \|u(t)\|^2$$

By minimizing the weighted sum

$$J_1 + \mu J_2$$

for a range of values of $\mu$, plot the trade-off curve of $J_1$ against $J_2$ showing the achievable performance. This above trade-off curve shows how we can trade-off between how accurately the hovercraft passes through the waypoints and how much input energy is used.

c) For each of the following values of $\mu$

$$\{ 10^p \mid p = -2, 0, 2, \ldots, 10 \}$$

plot the trajectories all on the same plot, together with the waypoints.

d) Now suppose we are controlling the hovercraft by radio control, and the maximum range possible between the transmitter and receiver is 2 (in whatever units we are using for distance.) Notice that, if we use the minimum-norm input then the hovercraft passes out of range, both when making its first turn and on the final stretch (between times 50 and 70).

We’d like to do something about this, but trading off the input norm as above doesn’t do the right thing; if $\mu$ is large then the hovercraft stays within range, but misses the waypoints entirely; if $\mu$ is small then it comes close to the waypoints, but goes out of range. Notice that this is particularly a problem on the final stretch between times 50 and 70; explain why this is.

e) One remedy for this problem is to solve a constrained multiobjective least-squares problem. We would like to impose the constraint that

$$A_{\text{hov}} u_{\text{seq}} = y_{\text{des}}$$

that is, achieve zero waypoint error $J_1 = 0$. We can attempt to keep the hovercraft in range by trading off the sum of the squares of the position

$$J_3 = \sum_{t=0}^{70} \|q(t)\|^2$$
against input cost \( J_2 \) subject to this constraint. To do this, we’ll solve

\[
\begin{align*}
\text{minimize} & \quad J_3 + \gamma J_2 \\
\text{subject to} & \quad A_{hov} u_{seq} = y_{des}
\end{align*}
\]

First, find the matrix \( W \) so that the cost function is given by

\[ J_3 + \gamma J_2 = \| W u_{seq} \|^2 \]

f) Now we have a problem of the form

\[
\begin{align*}
\text{minimize} & \quad \| W u \|^2 \\
\text{subject to} & \quad A u = y_{des}
\end{align*}
\]

This is called a weighted minimum-norm solution; the only difference from the usual minimum-norm solution to \( A u = y_{des} \) is the presence of the matrix \( W \), and when \( W = I \) the optimal \( u \) is just given by \( u_{opt} = A^\dagger y_{des} \). Show that the solution for general \( W \) is

\[
u_{opt} = \Sigma^{-1}A^T(A\Sigma^{-1}A^T)^{-1}y_{des}\]

where \( \Sigma = W^TW \). (One way to do this is using Lagrange multipliers.) Use this to solve the remaining parts of this problem.

g) For each of the following values of \( \gamma \)

\[ \{ 10^p \mid p = 0, 2, 4, \ldots, 20 \} \]

Plot the trajectories all on the same plot, together with the waypoints. Explain what you see.

h) By trying different values of \( \gamma \), you should be able to find a trajectory which just keeps the hovercraft within range. Plot the trajectory of the hovercraft; what is the corresponding value of \( \gamma \)? Is this the smallest-norm input \( u \) that just keeps the hovercraft within range, and drives the hovercraft through the waypoints? Explain why, or why not.

i) For a range of values of \( \gamma \), plot the trade-off curve of \( J_3 \) against \( J_2 \) showing the achievable performance.

**Solution.**

a) Setting

\[
C = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

gives the position of the hovercraft at time \( t \) as

\[
y(t) = \sum_{\tau=0}^{t-1} CA^{t-1-\tau} Bu(\tau)
\]
The parameters for the least-squares problem are therefore

\[
A_{\text{hov}} = \begin{bmatrix} CA^{t_1-1}B & CA^{t_1-1} & \cdots & CB & 0 & 0 & \cdots & 0 \\
CA^{t_2-1}B & CA^{t_2-2}B & \cdots & 0 \\
CA^{t_3-1}B & CA^{t_3-2}B & \cdots & 0 
\end{bmatrix} \quad y_{\text{des}} = \begin{bmatrix} p_1 \\
p_2 \\
p_3 \end{bmatrix}
\]

Solving this least squares problem gives optimal trajectory

![Optimal trajectory graph](image)

The corresponding optimal input sequence is below.

![Input sequence graph](image)

b) The weighted sum objective is

\[
J_1 + \mu J_2 = \left\| \frac{A_{\text{hov}}}{\sqrt{\mu I}} u_{\text{seq}} - \begin{bmatrix} y_{\text{des}} \\
0 \end{bmatrix} \right\|^2
\]
where

$$u_{seq} = \begin{bmatrix} u(0) \\ \vdots \\ u(69) \end{bmatrix}$$

and so the optimal input sequence is given by

$$u_{seq} = [A_{way}]^{\dagger} \frac{\sqrt[n]{I}}{\mu I} \begin{bmatrix} y_{des} \\ 0 \end{bmatrix}$$

Choosing values of $\mu$ between 1 and $10^7$ using $\text{mus=logspace}(0,7,50)$, the trade-off curve is shown below.

c) All of the trajectories together are
We can see clearly that increasing $\mu$ reduces the accuracy with which the trajectory passes through the waypoints.

d) On the final stretch the input is zero, and so is unaffected by increasing $\mu$. We were attempting to use the heuristic 'keeping $u$ small keeps $x$ small' but this fails, because when $u = 0$ the hovercraft just keeps going in a straight line.

e) We would like to minimize $J_3 + \gamma J_2$ subject to the constraints that the hovercraft moves through the waypoints. Denote the sequence of positions of the hovercraft by

$$ y_{seq} = \begin{bmatrix} y(0) \\ \vdots \\ y(T) \end{bmatrix} $$

where $T = 70$. Then we have

$$ y_{seq} = Tu_{seq} $$

where $T$ is the Toeplitz matrix

$$ T = \begin{bmatrix}
0 & 0 & 0 \\
CB & 0 & 0 \\
CAB & CB & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots \\
CA^{T-1}B & CA^{T-2}B & \ldots & CB
\end{bmatrix} $$

Now the cost function is

$$ J_3 + \gamma J_2 = \|Tu_{seq}\|^2 + \gamma\|u_{seq}\|^2 \\
= \|Wu_{seq}\|^2 $$

6
where

\[ W = \begin{bmatrix} T \\ \sqrt{T}I \end{bmatrix} \]

f) We’d like to solve

\[
\begin{align*}
\text{minimize} & \quad \|Wu\|^2 \\
\text{subject to} & \quad Au = y_{\text{des}}
\end{align*}
\]

One way to solve this is using Lagrange multipliers; if we augment the cost function by the Lagrange multipliers multiplied by the constraints, we have

\[ L(u, \lambda) = u^T \Sigma u + \lambda^T (Au - y_{\text{des}}) \]

and the optimality conditions are

\[
\begin{align*}
\frac{\partial L}{\partial u} & = 2u_{\text{opt}}^T \Sigma + \lambda^T A = 0 \\
\frac{\partial L}{\partial \lambda} & = u_{\text{opt}}^T A^T - y_{\text{des}}^T = 0
\end{align*}
\]

The first condition gives

\[ u_{\text{opt}} = -\frac{1}{2} \Sigma^{-1} A^T \lambda \]

and substituting this into the second we have

\[ -\frac{1}{2} A \Sigma^{-1} A^T \lambda = y_{\text{des}} \]

hence

\[ \lambda = -2(A \Sigma^{-1} A^T)^{-1} y_{\text{des}} \]

and

\[ u_{\text{opt}} = \Sigma^{-1} A^T (A \Sigma^{-1} A^T)^{-1} y_{\text{des}} \]

as desired.
g) The trajectory for a range of $\gamma$ values is shown below. (Actually these are clearer on separate plots)

We can see the trade-off clearly; decreasing $\gamma$ causes the hovercraft to try very hard to
stay close to the origin. Also notice the asymmetry caused by the different times at which the hovercraft must be at the waypoints.

h) A good choice of gamma is about $1.7 \times 10^4$. Here the trajectory just remains within range, as shown below.

This is not the smallest-norm $u$ that keeps the hovercraft within range and drives the hovercraft through the waypoints, because we are minimizing the sum of the squares of $\|q(t)\|$, rather than constraining each $\|q(t)\|$ independently. You can see this in the plot, since in the final stretch the hovercraft is expending extra effort to stay well within range, and this excessive input could be reduced.

In fact, one can compute the exact optimal, but this is not required and not covered in this course; (an approximation of) it is below.
i) The trade-off is below.

Notice that the vertical asymptote occurs when $J_2 \approx 0.03$; this is the minimum-norm of $u$ which drives the hovercraft through the desired trajectory, as seen in part (b).

code that solves this problem
helper functions

function y=vec(x)
% VEC produces a vector of length m*n from an m by n matrix
function y=vec(x)
% Given an m by n matrix x, y=vec(x) constructs a vector y
% consisting of the columns of x stacked on top of each other
%
[m,n] = size(x);
y = reshape(x,m*n,1);

function T=sys_toeplitz(A,B,C,D,out_times,in_times);
% SYS_TOEPLITZ computes toeplitz matrices for a discrete-time LDS
% T=sys_toeplitz(A,B,C,D,out_times,in_times);
% A,B,C,D specify a discrete-time state-space realization
% out_times and in_times are row vectors
% for example, if out_times is [1,2,4] and in_times is [0:10]
% then T is the matrix which maps
% [u(0); u(1); ... u(10)] to [y(1); y(2); y(4)]
% Notice that T is Toeplitz if out_times and in_times are
% both of the form a:b

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% setup parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% number of states
n=size(A,1);

% num inputs and outputs
ny=size(C,1);
nu=size(B,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% construct the Toeplitz matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this is neither efficient nor numerically reliable for big matrices
% but is simple and works well for small cases
T=[];
for r=out_times
    \% now create a row for this output time
    T_row=[];

    for s=in_times
        \% three cases; either CA^tB, D, or 0
        if s<r
            \% below the diagonal
            this_block= C*A^(r-s-1)*B;
        elseif s==r
            \% on the diagonal
            this_block=D;
        else
            \% above the diagonal
            this_block=zeros(ny,nu);
        end

        T_row=[T_row, this_block];
    end

    T=[T; T_row];
end

main code

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% compute min-norm input that drives a hovercraft through a
given set of waypoints
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% desired radius
r_max=2;

% desired time steps and positions
way_times=[ 6 40 50 ];

way_points=[ 1, 0, -1.5 ;
             -0.5, 1, 0 ];

% final time step
t_max=70;

% sampling time
h=1;

% discrete-time system
A=[1 h 0 0 ;
   0 1 0 0 ;
   0 0 1 h ;
   0 0 0 1 ];

B=[h^2/2 0;
   h 0;
   0 h^2/2 ;
   0 h ];

C=[ 1 0 0 0 ;
   0 0 1 0 ];

D=[ 0 0 ;
   0 0 ];

% number of states
n=size(A,1);

% num inputs and outputs
ny=size(C,1);
nu=size(B,2);

% real work is here
% toeplitz matrix mapping inputs to position at way_times
A_hov=sys_toeplitz(A,B,C,D, way_times, 0:t_max-1);

% find minimum norm input
u_tmp=pinv(A_hov)*vec(way_points);

% for convenience of simulation, reshape the inputs
% so that u_opt(:,k+1) is the input vector at time k
u_opt=reshape(u_tmp,2,t_max);

% simulate
x=zeros(n,t_max+1);
for k=0:t_max-1
    x(:,k+2)=A*x(:,k+1) + B*u_opt(:,k+1);
end
y=C*x;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% everything from here on is just plotting
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot trajectory in the plane

figure(1);
clf;
hold on;
axis equal;
axis([-2.5,2.5,-2.5,2.5]);
grid;
box on;

% plot radio range
[xs,ys]=ellipse(r_max^2*eye(2),[0;0]);
plot(xs,ys,'r');

% plot way points
for k=1:size(way_times,2)
    plot(way_points(1,k),way_points(2,k),'ko');
end

% plot trajectory
plot(y(1,:),y(2,:),'.-');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot thruster input versus time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2);
clf;
hold on;
grid;

h=plot((0:t_max-1),u_opt(1,1:t_max),'b.-');
h=plot((0:t_max-1),u_opt(2,1:t_max),'r.-');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% for a hovercraft
% compute the trade-off curve of input norm to distance norm
% subject to the constraint that
% the trajectory passes through the waypoints
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% parameters

% desired radius
r_max=2;

% desired time steps and positions
way_times=[ 6 40 50 ];

way_points=[ 1, 0, -1.5 ;
             -0.5, 1, 0 ];

n_way_points=size(way_points,2);

% final time step
t_max=70;

% sampling time
h=1;

% discrete-time system

A=[1 h 0 0 ;
   0 1 0 0 ;
   0 0 1 h ;
   0 0 0 1 ];

B=[h^2/2 0;
   h 0;
   0 h^2/2 ;
   0 h ];
C = [ 1 0 0 0 ;
     0 0 1 0 ];

D = [ 0 0 ;
     0 0 ];

% number of states
n = size(A,1);

% num inputs and outputs
ny = size(C,1);
nu = size(B,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% compute trade-off for weighted min-norm problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% toeplitz matrix mapping inputs to position at way_times
A_hov = sys_toeplitz(A,B,C,D, way_times, 0:t_max-1);

% toeplitz matrix mapping inputs to sequence of positions
A_pos = sys_toeplitz(A,B,C,D, 0:t_max, 0:t_max-1);

% desired gamma values
gammas = logspace(0,5,40);

% space to save costs
J2 = [];
J3 = [];

for i = 1:size(gammas,2)
    this_gamma = gammas(i);

    % weight parameter - sigma inverse
    sig_inv = inv(A_pos'*A_pos + this_gamma*eye(t_max*nu));

    % stack up the way_points
    y_des = reshape(way_points,2*n_way_points,1);

    % compute input that minimizes weighted cost
    lambda = (A_hov*sig_inv*A_hov')\y_des;
    u = sig_inv*A_hov'*lambda;

16
% compute corresponding trajectory in the plane
y=A_pos*u;

% keep track of achieved costs
J3(i) = norm(y)^2;
J2(i) = norm(u)^2;

% store a nicely shaped y for plotting
y_keep{i}=reshape(y,2,t_max+1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% everything from here is just plotting
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% first plot trade off curve

% figure setup
figure(1);
clf;
hold on;
grid;
axis([0,3,0,200]);

% plot
plot(J2,J3,'.-');
xlabel('J_2 position cost');
ylabel('J_3 input cost');
title('trade off curve')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% now plot all the different inputs

% loop over each gamma value
for i=1:size(gammas,2)

    figure(2);
clf;
hold on;
axis equal;
axis([-2.5,2.5,-2.5,2.5]);
grid;

end
box on;
title(sprintf('gamma=%g',gammas(i)));

% plot radio range
[xs,ys]=ellipse(r_max^2*eye(2),[0;0]);
plot(xs,ys,'r');

% plot way points
for k=1:size(way_times,2)
    plot(way_points(1,k),way_points(2,k),'ko');
end

% plot trajectory
y=y_keep{i};
plot(y(1,:),y(2,:),'.-');

% wait for 100 milliseconds between plots
pause(0.1);
end

9.1410. Invariance of the unit square. Consider the linear dynamical system \( \dot{x} = Ax \) with \( A \in \mathbb{R}^{2 \times 2} \). The unit square in \( \mathbb{R}^2 \) is defined by

\[
S = \{ x \mid -1 \leq x_1 \leq 1, \quad -1 \leq x_2 \leq 1 \}.
\]

a) Find the exact conditions on \( A \) for which the unit square \( S \) is invariant under \( \dot{x} = Ax \). Give the conditions as explicitly as possible.

b) Consider the following statement: if the eigenvalues of \( A \) are real and negative, then \( S \) is invariant under \( \dot{x} = Ax \). Either show that this is true, or give an explicit counterexample.

Solution.

a) In order for the unit square to be an invariant set for the system \( \dot{x} = Ax \), all flows should be in going at the boundary of the unit square. In otherwords, we require that for all \( x \) on the boundary, \( \langle \dot{x}, n \rangle < 0 \), where \( n \) is the outward unit normal vector to the boundary at point \( x \). Suppose that

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.
\]

The boundary of the unit square consists of four line segments. The condition \( \langle \dot{x}, n \rangle < 0 \) along each line segment becomes:

\( x_1 = -1, \ -1 \leq x_2 \leq 1 \): in this region we have

\[
\langle \dot{x}, n \rangle = \left( \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, -e_1 \right) = -a_{11}x_1 - a_{12}x_2.
\]
Since $x_1 = -1$ we require that $a_{11} - a_{12}x_2 \leq 0$ for $-1 \leq x_2 \leq 1$. However, when $-1 \leq x_2 \leq 1$, $a_{11} - a_{12}x_2$ is maximum for $x_2 = -\text{sgn}(a_{12})$ and therefore $a_{11} - a_{12}x_2 \leq 0$ if and only if $a_{11} + a_{12}\text{sgn}(a_{12}) \leq 0$, or finally, $a_{11} + |a_{12}| \leq 0$.

$x_1 = 1, -1 \leq x_2 \leq 1$: here $n = e_1$ and using the same reasoning we get the same condition $a_{11} + |a_{12}| \leq 0$.

$x_2 = -1, -1 \leq x_1 \leq 1$: in this case we have $n = -e_2$ and by similar arguments we get $a_{22} + |a_{21}| \leq 0$.

$x_2 = 1, -1 \leq x_2 \leq 1$: here we reach the same condition as in the previous case, i.e., $a_{22} + |a_{21}| \leq 0$.

In summary the conditions for the unit square to invariant are:

$$a_{11} + |a_{12}| \leq 0, \quad a_{22} + |a_{21}| \leq 0.$$

b) It is not true. $A = \begin{bmatrix} -1 & 2 \\ 0 & -1 \end{bmatrix}$ gives a simple counterexample.

11.1820. Optimal control for maximum asymptotic growth. We consider the controllable linear system

$$x(t + 1) = Ax(t) + Bu(t), \quad x(0) = 0,$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$. You can assume that $A$ is diagonalizable, and that it has a single dominant eigenvalue (which here, means that there is one eigenvalue with largest magnitude).
An input $u(0), \ldots, u(T-1)$ is applied over time period $0, 1, \ldots, T-1$; for $t \geq T$, we have $u(t) = 0$. The input is subject to a total energy constraint:

$$\|u(0)\|^2 + \cdots + \|u(T-1)\|^2 \leq 1.$$  

The goal is to choose the inputs $u(0), \ldots, u(T-1)$ that maximize the norm of the state for large $t$. To be more precise, we're searching for $u(0), \ldots, u(T-1)$, that satisfies the total energy constraint, and, for any other input sequence $\tilde{u}(0), \ldots, \tilde{u}(T-1)$ that satisfies the total energy constraint, satisfies $\|x(t)\| \geq \|\tilde{x}(t)\|$ for $t$ large enough. Explain how to do this. You can use any of the ideas from the class, e.g., eigenvector decomposition, SVD, pseudo-inverse, etc. Be sure to summarize your final description of how to solve the problem. Unless you have to, you should not use limits in your solution. For example you cannot explain how to make $\|x(t)\|$ as large as possible (for a specific value of $t$), and then say, “Take the limit as $t \to \infty$” or “Now take $t$ to be really large”.

**Solution.** We have

$$x(T) = \sum_{\tau=0}^{T-1} A^{T-1-\tau} Bu(\tau) = C_T U,$$

where

$$C_T = \begin{bmatrix} B & AB & \cdots & A^{T-1} B \end{bmatrix}, \quad U = \begin{bmatrix} u(T-1) \\ \vdots \\ u(0) \end{bmatrix}.$$

The vector $U$ is our design variable; the energy constraint on the input is just $\|U\| \leq 1$. For $t > T$, there is no input, so we have

$$x(t) = A^{t-T} x(T) = A^{t-T} C_T U.$$

Now, what does the requirement that $x(t)$ be as large as possible for large $t$ mean? Let’s look at what $A^k z$ looks like, for large $k$. Using the hint given, we write the eigenvalue decomposition of $A$ as

$$A = T \Lambda T^{-1},$$

where the columns of $T$ are the eigenvectors of $A$, and $\Lambda$ is a diagonal matrix with entries $\lambda_{ii} = \lambda_i$. Therefore we have

$$A^k = T \Lambda^k T^{-1}.$$  

Let’s say that $\lambda_1$ is the dominant eigenvalue, i.e., $|\lambda_1| > |\lambda_i|$ for $i = 2, \ldots, n$. (This means that $\lambda_1$ is real; if it were complex, then its conjugate would be another eigenvalue with equal magnitude.) Now, for large $k$ we have

$$A^k z = T \Lambda^k T^{-1} z \approx \lambda_1^k v_1^T w_1^T z,$$

where $v_1$ is the first column of $T$ (i.e., the eigenvector associated with $\lambda_1$) and $w_1$ is the first row of $T^{-1}$ (i.e., a left eigenvector associated with $\lambda_1$). Therefore

$$\|A^k z\| \approx |\lambda_1^k| \|v_1\| \|w_1^T z\|,$$
which shows that our goal is to choose $z$ so that $|w_1^T z|$ is as large as possible. Finally we can put it together. Our goal is to choose $\|U\| \leq 1$ that maximizes

$$|w_1^T C_T U| = \left| (C_T w_1)^T U \right|.$$  

This looks like a problem involving the SVD, but it’s not: It’s a problem involving the Cauchy-Schwartz inequality. The solution is

$$U = \frac{1}{\|C_T w_1\|} C_T w_1$$

(or the negative of this vector, which is also a solution). We should mention a common error. Many of you mentioned that $U$ should be chosen such that $z$ lies in the direction of the first eigenvector and then derived the least norm solution

$$U_{ln} = C_T (C_T^T)^{-1} v_1.$$ 

For this to work, $C_T$ needs to be fat and full rank. Some of you mentioned that if $C_T$ were skinny one would use least squares to get “as close as you can”. Although this gives the maximum gain, $C_T$ needs to be full rank.

12.1930. **Properties of trajectories.** For each of the following statements, give the exact (necessary and sufficient) conditions on $A \in \mathbb{R}^{n \times n}$ under which the statement holds.

a) Every trajectory of $\dot{x} = Ax$ converges as $t \to \infty$. This means that, for any $x(0)$, $x(t)$ converges to some value, which need not be zero (and can depend on $x(0)$ and $A$).

b) Every trajectory of $\dot{x} = Ax$ is bounded. This means that, for any $x(0)$, there is an $M$ (that can depend on $x(0)$ and $A$) for which $\|x(t)\| \leq M$ for all $t \geq 0$.

Your answers can refer to any concepts used in the course (eigenvalues, singular values, Jordan form, least-squares, range, nullspace, ...). We will deduct points from answers that are technically correct, but more complicated than they need to be. *You may not make any assumptions about $A$* (e.g., that it is nonsingular, diagonalizable, etc.).

Please give only your final answer; we do not want any justification or discussion. Your answers should have a form similar to “The property in part (a) occurs if and only if all singular values of $A$ are less than one, and $A$ has no real eigenvalues”. (This is not the correct answer; it is only as an example of what your answer should look like.)

**Solution.**

a) *The property in part (a) occurs if and only if all eigenvalues of $A$ either have negative real part or are zero, and each zero eigenvalue is associated with a Jordan block of size $1 \times 1$. The latter condition can be stated several other ways, e.g., the dimension of the nullspace of $A$ is equal to the number of zero eigenvalues of $A.*

Now let’s justify the condition. (We did not ask you to give any justification.) First suppose our condition holds. Then every solution of $\dot{x} = Ax$ is a linear combination of terms with decaying exponentials (possibly oscillatory, and including powers of $t$ as
well), associated with the eigenvalues with negative real part, and terms which are constant, associated with the zero eigenvalues. (Note that we cannot have terms that grow like $t$, since these would be associated with zero eigenvalues with Jordan blocks of size $2 \times 2$ or larger.) Thus, every trajectory converges. In fact, every trajectory converges to an element of the nullspace of $A$. (These are the equilibrium points of $\dot{x} = Ax$.)

Now suppose our condition does not hold. This happens if $A$ has an eigenvalue with nonnegative real part that is not zero, or a zero eigenvalue associated with a Jordan block of size $2 \times 2$ or bigger. Consider the first case. If an eigenvalue $\lambda$ (associated with eigenvector $v$) has a positive real part, then we can find a diverging (and possibly oscillating, if $\lambda$ is complex) trajectory, which therefore doesn’t converge. If the real part of $\lambda$ is zero, but the eigenvalue is different from zero, then it must be pure imaginary, so we can find an oscillating trajectory, which doesn’t converge. Finally, suppose that $A$ has an eigenvalue 0 associated with a Jordan block of size $2 \times 2$ or bigger. In this case we can find a trajectory of the form $x(t) = a + bt$, with $b \neq 0$, which clearly doesn’t converge.

b) The property in part (b) occurs if and only if all eigenvalues of $A$ have real part less than or equal to zero, and those with real part zero (i.e., the zero and pure imaginary ones) are associated with Jordan blocks of size $1 \times 1$. We consider the Jordan decomposition $A = T J T^{-1}$, and the resulting change of coordinates for the dynamical system: $\dot{y} = T^{-1} A T y = J y$. We see that the original system, represented in the new coordinates, is

$$\dot{y} = T^{-1} A T y = J y.$$

Since $T$ is fixed and invertible, every trajectory of $x$ is bounded if and only if every trajectory of $y$ is bounded. We have that

$$y(t) = e^{tJ} y(0),$$

and we see that the trajectory of $y(t)$ is bounded for all $y(0)$ if and only if $\|e^{tJ}\|$ is bounded as $t$ varies.

However, $J$ has block diagonal structure $J = \text{diag}(J_1, \ldots, J_q)$, where $J_i$ is the Jordan block corresponding to eigenvalue $\lambda_i$. Since $J$ is block diagonal, $\|e^{tJ_i}\|$ is bounded if and only if $\|e^{tJ_i}\|$ is bounded for each $i$. Recall that if $J_i$ is a Jordan block of size $k \times k$, then its exponential is given by

$$e^{tJ_i} = e^{t \lambda_i} \begin{bmatrix} 1 & t & \cdots & t^{k-2}/(k-2)! & t^{k-1}/(k-1)! \\ 1 & \cdots & t^{k-2}/(k-2)! \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 & \end{bmatrix}.$$ 

We are now in a position to confirm the condition that was proposed.

If any eigenvalue $\lambda_i$ of $A$ has positive real part, then $e^{t \lambda_i}$ grows without bound, which implies that $\|e^{tJ_i}\|$ is also unbounded, and so there is an unbounded trajectory of $x$.

If an eigenvalue $\lambda_i$ of $A$ has negative real part, then $e^{t \lambda_i} p \to 0$ as $t \to \infty$ for any positive power of $p$, which implies that $\|e^{tJ_i}\| \to 0$, which tells us that $\|e^{tJ_i}\|$ is bounded.
If an eigenvalue $\lambda_i$ has zero real part, then

$$
\| e^{tJ_i} \| = \left\| e^{t\lambda_i} \begin{bmatrix}
1 & t & \cdots & t^{k-1}/(k-1)!
1 & \cdots & t^{k-2}/(k-2)!
\vdots & \ddots & \vdots
1
\end{bmatrix} \right\| = \left\| \begin{bmatrix}
1 & t & \cdots & t^{k-1}/(k-1)!
1 & \cdots & t^{k-2}/(k-2)!
\vdots & \ddots & \vdots
1
\end{bmatrix} \right\},
$$

since $|e^{t\lambda_i}| = 1$. The $t^k$ terms grow without bound, so $\| e^{tJ_i} \|$ is unbounded if the Jordan block has size $k \times k$, for $k > 1$. On the other hand, if $k = 1$, then the matrix is constant, and thus $\| e^{tJ_i} \|$ is bounded.

Thus, we see that if our condition holds, then every trajectory of $x$ is bounded. Conversely, if every trajectory is bounded, then the condition must hold, because if not, we would be able to find a trajectory that is unbounded, which is a contradiction.


$$
x(t+1) = Ax(t) + Bu(t),
$$
$$
y(t) = Cx(t),
$$

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$. In output feedback control we use an input which is a linear function of the output, that is,

$$
u(t) = Ky(t),
$$

where $K \in \mathbb{R}^{m \times p}$ is the feedback gain matrix. The resulting state trajectory is identical to that of an autonomous system,

$$
x(t+1) = \bar{A}x(t).
$$

a) Write $\bar{A}$ explicitly in terms of $A$, $B$, $C$, and $K$.

b) Consider the single-input, single-output system with

$$
A = \begin{bmatrix} 0.5 & 1.0 & 0.1 \\
-0.1 & 0.5 & -0.1 \\
0.2 & 0.0 & 0.9 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\
0 \\
0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}.
$$

In this case, the feedback gain matrix $K$ is a scalar (which we call simply the feedback gain.) The question is: find the feedback gain $K_{\text{opt}}$ such that the feedback system is maximally damped. By maximally damped, we mean that the state goes to zero with the fastest asymptotic decay rate (measured for an initial state $x(0)$ with non-zero coefficient in the slowest mode.) Hint: You are only required to give your answer $K_{\text{opt}}$ up to a precision of $\pm 0.01$, and you can assume that $K_{\text{opt}} \in [-2, 2]$. 

23
Solution. From the equations given in the problem,

\[ x(t + 1) = Ax(t) + Bu(t) = Ax(t) + BKy(t) = Ax(t) + BKCx(t) = (A + BKC)x(t) \]

Therefore, \( \tilde{A} = A + BKC \). Since \( x(t) = (\tilde{A})^t x(0) \), this is a discrete-time autonomous system. Thus, the system is stable if all the eigenvalues of \( \tilde{A} \) are less than 1 in magnitude, that is, \( \left| \lambda (\tilde{A}) \right| < 1 \). Since the largest eigenvalue in magnitude corresponds to the slowest mode, it determines asymptotic decay rate. Therefore, this feedback system is maximally damped when this largest eigenvalue is minimized over \( K \). So

\[ K_{opt} = \arg \min_K \max_i |\lambda_i(\tilde{A})| \]

There is no easy analytic solution for \( K_{opt} \); actually \( \max_i |\lambda_i(\tilde{A})| \) is a non-differentiable function of \( K \) and, in general, can have local minima. So we just compute the \( \max_i |\lambda_i(\tilde{A})| \) as function of \( K \) (at small intervals) and find the minimum point. \( K_{opt} \) is found to be \( 1.93 \pm 0.005 \) and \( \min_K \max_i |\lambda_i(\tilde{A})| \) is \( 0.74895 \). Note: \( \max_i |\lambda_i(\tilde{A})| \) is called the spectral radius of the matrix \( \tilde{A} \) and denoted by \( \rho(\tilde{A}) \). When \( K \) is a matrix instead of a scalar, this simple approach won’t work. (Until recently this was thought to be a very hard problem! You can learn the tools you need to solve this problem very efficiently for any size \( K \) in EE364.) MatLab code for Problem 5

A=[0.5 1.0 0.1; -0.1 0.5 -0.1; 0.2 0.0 0.9]; B=[1;0;0]; C=[0 1 0]; Klist = -2:1e-3:3; spectral_radius = []; for K = Klist
    spectral_radius = [ spectral_radius max(abs(eig(A+K*B*C))) ];
end
[sr_opt,i_opt] = min( spectral_radius ); Kopt = Klist(i_opt);
figure(1); plot( Klist, spectral_radius ); title( ’Spectral Radius’); k_{opt} = ’num2str( Kopt )... ’, \rho_{opt} = ’num2str( sr_opt ) ’); xtick( ’K’); ytick( ’\rho’); grid on
Figure 2: Spectral Radius

Spectral Radius; $K_{\text{opt}} = 1.93$, $\rho_{\text{opt}} = 0.74895$