MECH 6v29 - Model Predictive Control
Homework 3

Jonas Wagner
jonas.wagner@utdallas.edu

2023, October 20th

Contents

Problem 1

Problem Data:

1a)

Problem Data:

$$\begin{aligned} \mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{B} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \quad \mathcal{W} = \{\mathbf{w} = \mathbf{B}z \ : \ |z| \leq 0.3\} \\ \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathcal{Y} = \{\mathbf{y} \in \mathbb{R}^3 \ : \ \|\mathbf{y}\|_\infty \leq 1\} \end{aligned} \tag{1}$$

Prediction Horizon: $N = 10$
Initial Condition: $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

1b) Nilpotent candidate controller

$\Lambda(\mathbf{A} + \mathbf{BK}) = 0$
Result w/ Acker: $\mathbf{K} = \begin{bmatrix} -1 & -1.5 \end{bmatrix}$
Same as in [**?**].

1c) Output constraint tightening

From reference (using different notation):

$$\begin{aligned} \mathcal{Y}_0 &= \mathcal{Y} \\ \mathcal{Y}_{j+1} &= \mathcal{Y}_j \ominus (\mathbf{C} + \mathbf{DK})\mathbf{L}_j\mathcal{W}, \quad \forall_{j \in \{0,\dots,N-1\}} \end{aligned} \tag{2}$$

where $\mathbf{L}_j = (\mathbf{A} + \mathbf{BK})^j$.[1]
Or equivalently, using the time-invarience of K and some version of the Cayley-Hamilton theorem,

$$\mathcal{Y}_j = \mathcal{Y} \ominus \bigoplus_{i=1,\dots,n} (\mathbf{C} + \mathbf{DK})(\mathbf{A} + \mathbf{BK})^{i-1} \tag{3}$$

(eliminating if the power is negative...)
TODO: double check this... (pretty sure this falls under some distributed property...)
For this system, $(\mathbf{C} + \mathbf{DK}) = \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{K} \end{bmatrix}$

$$\begin{aligned} \mathcal{Y}_0 &= \{\mathbf{y} \in \mathbb{R}^3 \ : \ \|\mathbf{y}\|_\infty \leq 1\} \\ &= \{\mathbf{y} \in \mathbb{R}^3 \ : \ |y_1| \leq 1, |y_2| \leq 1, |y_3| \leq 1\} \\ \mathcal{Y}_1 &= \mathcal{Y}_0 \ominus (\mathbf{C} + \mathbf{DK})\mathcal{W} \\ &= \mathcal{Y}_0 \ominus \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{K} \end{bmatrix} \{\mathbf{B}w \in \mathbb{R} \ : \ |w| \leq 0.3\} \\ &= \{\mathbf{y} \in \mathbb{R}^3 \ : \ |y_1| \leq 0.85, |y_2| \leq 0.7, |y_3| \leq 0.4\} \\ \mathcal{Y}_2 &= \mathcal{Y}_1 \ominus (\mathbf{C} + \mathbf{DK})(\mathbf{A} + \mathbf{BK})\mathcal{W} \\ &= \mathcal{Y}_1 \ominus \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{K} \end{bmatrix} \{\mathbf{B}w \in \mathbb{R} \ : \ |w| \leq 0.3\} \\ &= \{\mathbf{y} \in \mathbb{R}^3 \ : \ |y_1| \leq 0.7, |y_2| \leq 0.4, |y_3| \leq 0.1\} \end{aligned} \tag{4}$$

which is the same for the remaining since $(\mathbf{A} + \mathbf{BK})^2 = \mathbf{0}$.

[1] not explicitly, but eliminating time-variance that's what it is...

1d) Controller Implementation

Using everything provided, the controller was implimented in YALMIP as described.

1e) Closed-loop Simulation

The system was simulated in MATLAB and the figures requested were recreated:

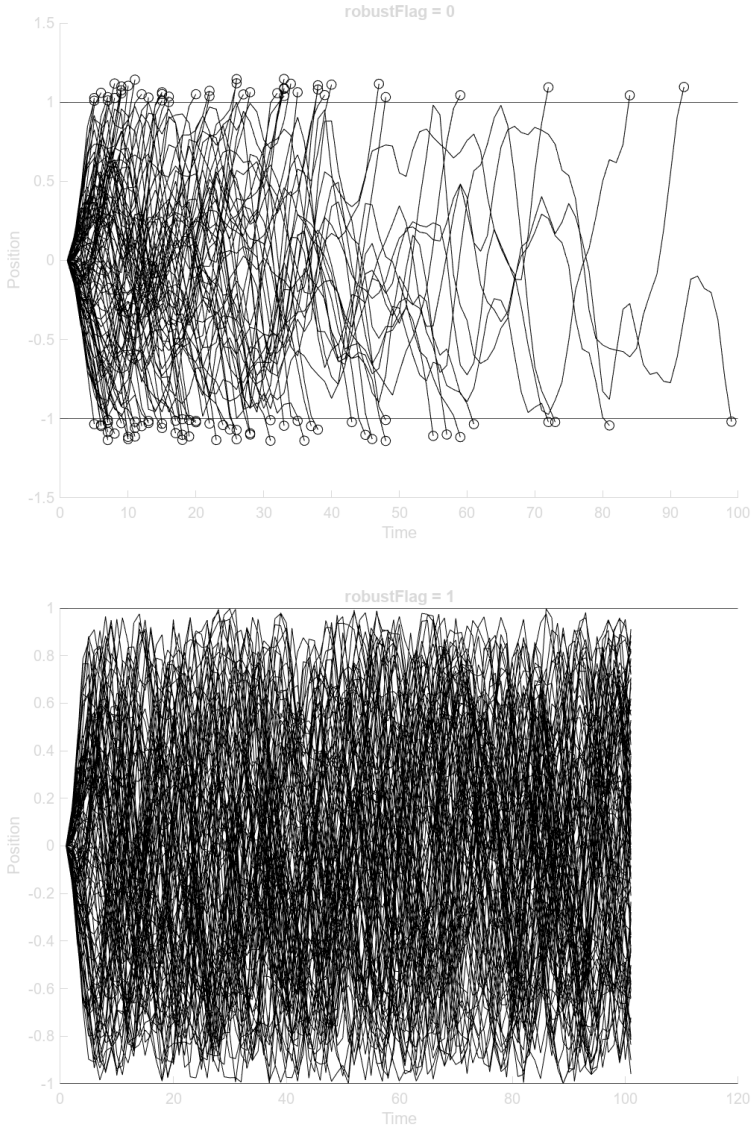


Figure 1: Recreated results from output constraint tightening.

1f) Simulation Result Analysis

The results for the Tube-MPC approach was a mean of 239 and max of 357.

Problem 2

Problem Data:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{B} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \quad \mathcal{W} = \{\mathbf{w} = \mathbf{B}z \ : \ |z| \leq 0.3\} \\ \mathbf{C} &= \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathcal{Y} = \{\mathbf{y} \in \mathbb{R}^3 \ : \ \|\mathbf{y}\|_\infty \leq 1\} \end{aligned} \tag{5}$$

Prediction Horizon: $N = 10$

Initial Condition: $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Candidate Feedback Controller: $K = \begin{bmatrix} -1 & -1.5 \end{bmatrix}$

2a) Set Definitions

State/Input Constraints: From the output constraints the individual state/input constraints can be found by satisfying $C\mathcal{X} = \mathcal{Y} \ominus D\mathcal{U}$ and $D\mathcal{U} = \mathcal{Y} \ominus C\mathcal{X}$; but C and D would have to be invertible for a direct solution, so I'm not sure if it's good.
TODO: double check this...
Alternatively, we can look at the individual dimensions/values and do it simply by observation as we can decompose the dimensions:

$$\begin{aligned} \mathcal{X} &= \{x \in \mathbb{R}^n \ : \ \|Cx\|_\infty \leq 1\} = \{x \in \mathbb{R}^n \ : \ |x_1| \leq 1, \ |x_2| \leq 1\} \\ \mathcal{U} &= \{u \in \mathbb{R}^m \ : \ \|Du\|_\infty \leq 1\} = \{u \in \mathbb{R}^m \ : \ |u_1| \leq 1\} \end{aligned} \tag{6}$$

In H-rep this becomes:

$$\begin{aligned} \mathcal{X} &= \left\{ A = \begin{bmatrix} \mathbf{I}_2 \\ -\mathbf{I}_2 \end{bmatrix}, \ b = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \\ \mathcal{U} &= \left\{ A = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \ b = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} \end{aligned} \tag{7}$$

Disturbance Set: $\mathcal{W}$ is defined by

$$\mathcal{W} = \{\mathbf{w} = \mathbf{B}z \ : \ |z| \leq 0.3\} \tag{8}$$

and in H-rep it becomes

$$\mathcal{W} = \left\{ A = \begin{bmatrix} B \\ -B \end{bmatrix}, \ b = \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix} \right\} \tag{9}$$

2b) RPI Sets

Constructing the RPI set was done in MATLAB following process in provided code (as instructed):

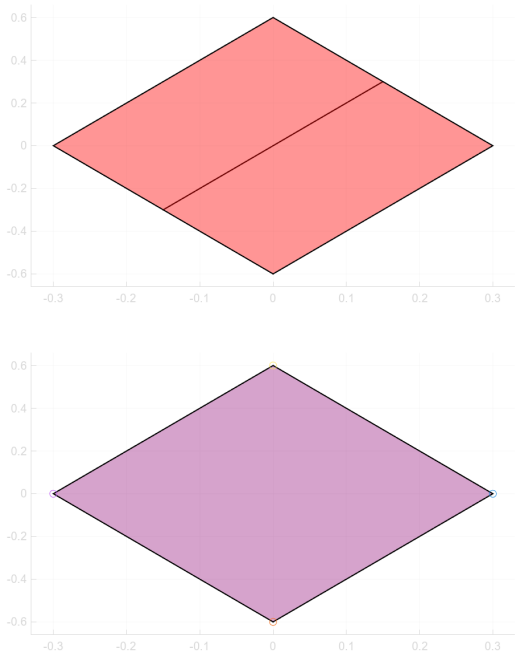


Figure 2: First one is the iterative portion and the second one is the one using the provided Approx_RPI() function

Testing different epsilon values resulted in an RPI set that didn't look all that different and the vertices all appeared to line up. (Tested many epsilon values all less than 1)
The result of Z is calculated in MATLAB as:

$$Z = \begin{bmatrix} 0.8639 & -0.4319 & 0.2592 \\ 0.8639 & 0.4319 & 0.2592 \\ -0.8639 & 0.4319 & 0.2592 \\ -0.8639 & -0.4319 & 0.2592 \end{bmatrix} \tag{10}$$

2c) Tightened Input/Output Constraint Sets

From [**?**], the disturbance invarient set for $x_{k+1} = A_K x_k + w$ is defined by $A_K Z \oplus W \subseteq Z$, where $A_K = A + B * K$ is stable.
This Z is the RPI set for $x_{k+1} = A_K x_k + w$. The proposition from the reference says that if $x_k \in \bar{x}_k$ and $u_k \in \bar{u}_k + K(x - \bar{x})$, then $x_{k+1} \in \bar{x}_{k+1} \oplus Z \; \forall_{w_k \in W}$. Where $x_{k+1} = Ax_k + Bu_k + w_k$. [2]
Result makes it so that $u_k = \bar{u}_k + K(x_k = \bar{x}_k)$ keeps the state close to the nominal one. [3]
The new state and input constraints are based on Z_k for each time-step.
To ensure that $x_k^* = x_k \oplus Z$ satisfies the original constraint set, the new constraints of

$$\bar{\mathcal{X}} = \mathcal{X} \ominus Z = \left\{ x \in \mathbb{R}^2 \; : \; |x_1| \leq 0.7, |x_2| \leq 0.4 \right\} \tag{11}$$

For the input, the state constraints are satisfied as

$$\bar{\mathcal{U}} = \mathcal{U} \ominus KZ = \{ u \in \mathbb{R} \; : \; |u| \leq 0.1 \} \tag{12}$$

As expected, both constraint sets are smaller than the original sets.

2d) Controller Formulation

The controller is now described differently then before as follows: Using the RPI set and newly derived tightened state/input constraints, the nominal trajectory values are derived in the same way as before. Then the difference from nominal has the feedback gain to correct it. (Summarized in previous subsections)

[2] some of the time-invarience isn't included
[3] this also means it can ensure that it satisfies state/input constraints as well

2e) Simulation

The system was simulated in MATLAB. Results requested:

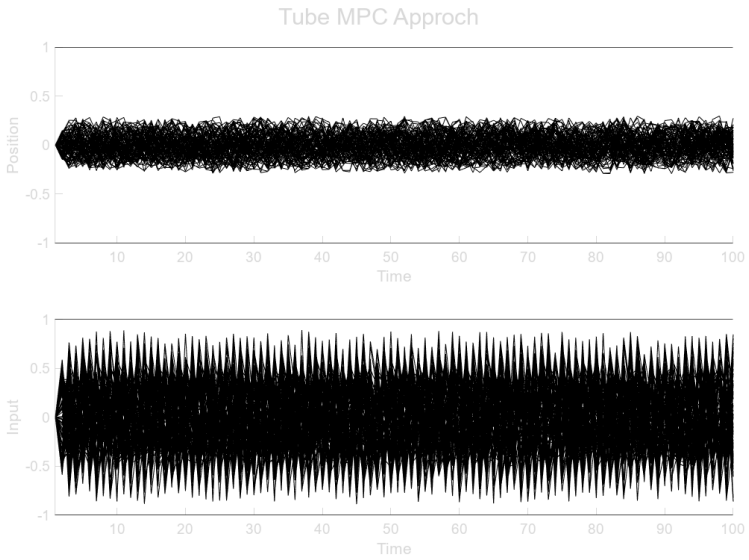


Figure 3: Simulation results of Tube-MPC method

2f) Result Analysis

The results for the Tube-MPC approach was a mean of 1465 and max of 1973.
Compared to the constraint tightened approach this was much higher in terms of cost. I'm not certain why this is, but my guess is because the input is not directly tied to the cost-function in the 2nd approach. It is also possible that I did the 2nd-approach incorrectly and I was supposed to account for the actual input signal; although I'm not sure.

A Code

1a) Github

See my github repo for all my course related materials: `https://github.com/jonaswagner2826/MECH6v29_MPC`

1b) Matlab results

MATLAB code and results are attached.

MECH 6V29 - MPC - Homework 3

Table of Contents

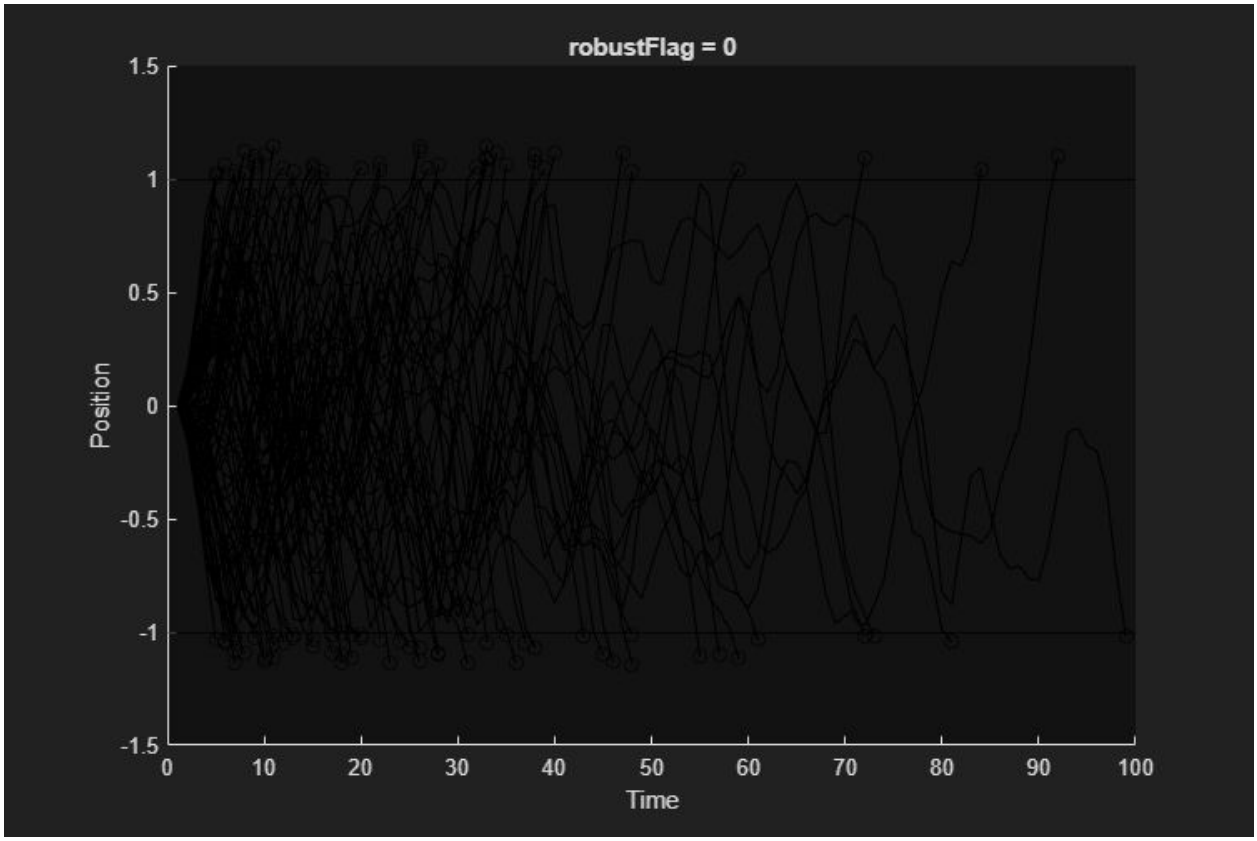
Problem 1

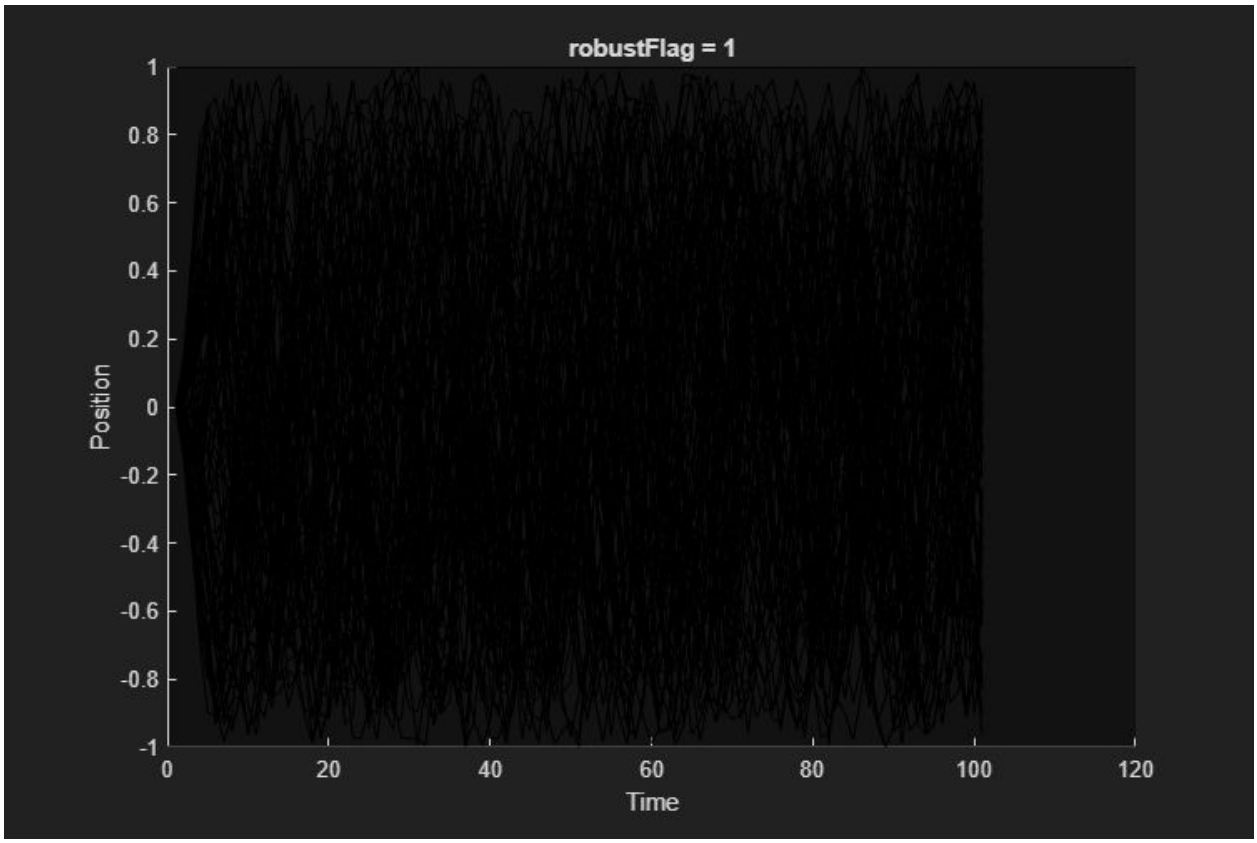
```
% 1a
A = [1, 1;
     0, 1];
B = [0.5;
     1];
C = eye(3,2);%<--- [1,0;0,1;0,0]
D(3,1) = 1; %<--- [0;0;1]
sys = ss(A,B,C,D,1)

N = 10;

% sizes
nx = size(A,1);
nu = size(B,2);
ny = size(C,1);


% 1b
K = -acker(A,B,zeros(nx,1))

% 1c
Y = Polyhedron('A', [eye(ny);-eye(ny)], 'b', ones(2*ny,1));
W = B*Polyhedron('A',[1;-1],'b',[0.3;0.3]);

Y_{1} = Y;% - (C+D*K)*W;
for j = 1:N
    Y_{j+1} = Y_{j} - (C+D*K)*(A+B*K)^(j-1)*W;
end

for robustFlag = [false, true]
```

1d ----- Setup Controller

```
P=0;
Q = 1e-3*eye(nx);
R = 100;

yalmip('clear'); clear('controller');
```

```
u_ = sdpvar(repmat(nu,1,N),ones(1,N));
x_ = sdpvar(repmat(nx,1,N),ones(1,N));

constraints = []; objective = 0;
for k = 1:N-1
    objective = objective + x_{k}'*Q*x_{k} + u_{k}'*R*u_{k};
    constraints = [constraints, x_{k+1} == A*x_{k} + B*u_{k}];
    if robustFlag
        constraints = [constraints, Y_{k}.A*(C*x_{k}+D*u_{k})<= Y_{k}.b];
    else
        constraints = [constraints, Y.A*(C*x_{k}+D*u_{k})<= Y.b];
    end
end
k = k + 1;
constraints = [constraints, x_{k} == 0];
if robustFlag; constraints = [constraints, Y_{k}.A*(C*x_{k}+D*u_{k})<=
Y_{k}.b]; end
objective = objective + x_{k}'*P*x_{k};

opts = sdpsettings;
controller = optimizer(constraints,objective,opts,x_{1},u_{1});


% simulate and plot
fig = figure(...
        WindowStyle="normal",...
        Position=[0 0 750 500]);
hold on
for i = 1:100
    rng(i);
    x0 = zeros(nx,1); tf = 100;
    V = num2cell(0.6*rand(nx,tf)-0.3);
    [X{i},U{i},~] = run_sim(A,B,V,controller, x0, tf);
    k_fail = find(~isfinite(U{i}),1,"first");
    plot(X{i}(1,:),'k')
    plot(k_fail,X{i}(1,k_fail),'ko')
end
yline(1,'k'); yline(-1,'k');
ylabel('Position');
xlabel('Time');
title(sprintf('robustFlag = %d',robustFlag))
saveas(fig,strcat('figs',filesep,sprintf('pblm1_robust=%d',robustFlag),'.png'
));
```


robustFlag = 0
Position
Time
1.5
1
0.5
0
-0.5
-1
-1.5
0
10
20
30
40
50
60
70
80
90
100


robustFlag = 1
Position
Time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120

Result Analysis

Cost

```
J_{100} = [];

for i = 1:100
    J_{i} = 0;
    for k = 1:tf-1
        J_{i} = J_{i} + X{i}(:,k)'*Q*X{i}(:,k) + U{i}(:,k)'*R*U{i}(:,k);
    end
    J_{i} = J_{i} + X{i}(:,k+1)'*P*X{i}(:,k+1);
end
J = [J_{:}];

J_mean = mean(J)
J_max = max(J)
```

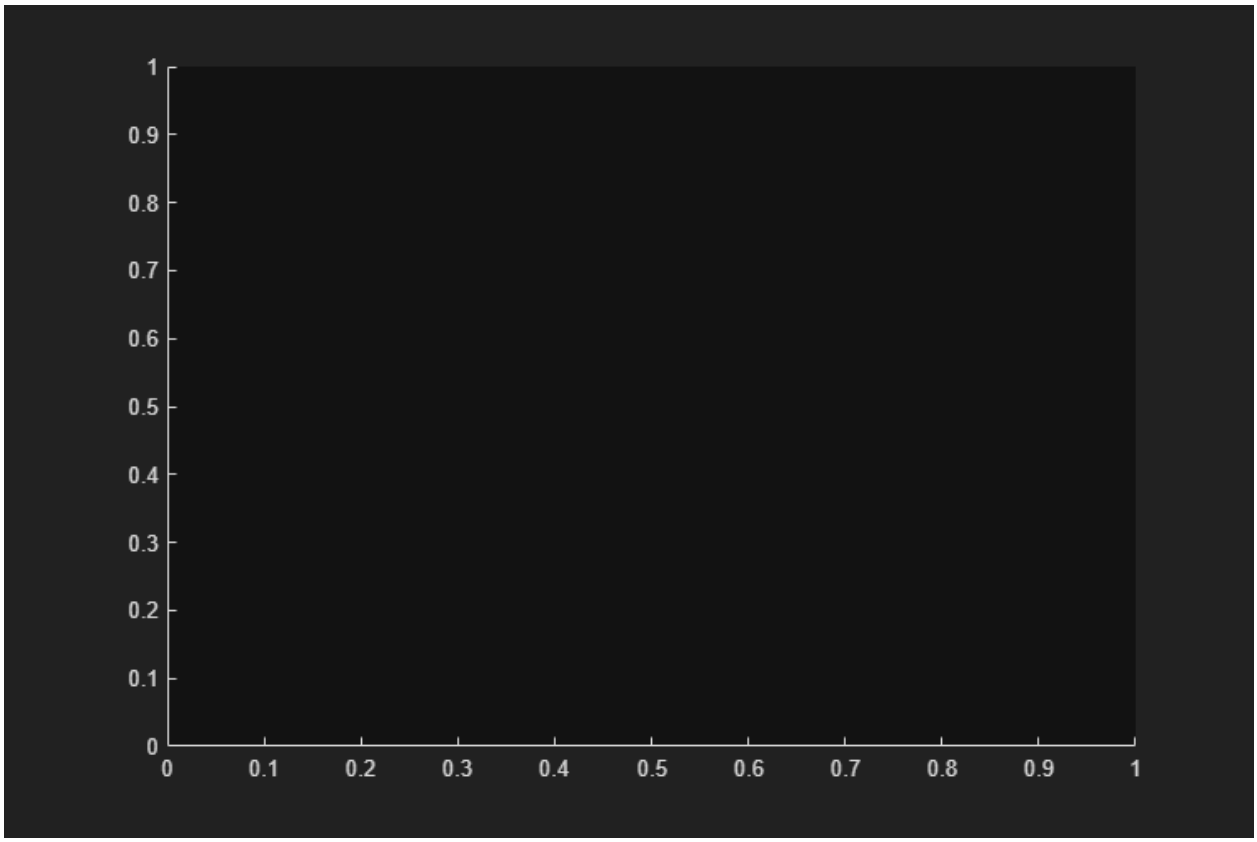
```
J_mean =

   NaN


J_max =

   NaN
```

```
J_mean =

  238.7968


J_max =

  357.6807
```

```
end
```

```
sys =

  A =
       x1  x2
   x1   1   1
   x2   0   1

  B =
        u1
   x1  0.5
   x2    1
```

```
  C =
       x1  x2
   y1   1   0
   y2   0   1
   y3   0   0

  D =
       u1
   y1   0
   y2   0
   y3   1

Sample time: 1 seconds
Discrete-time state-space model.

K =

   -1.0000   -1.5000
```



Local functions

```
function [X,U,diagnostics_] = run_sim(A,B,V,controller,x0, tf)

    X_{tf+1} = []; U_{tf} = []; diagnostics_{tf} = [];
```

```
    X_{1} = x0;
    for k = 1:tf
        [U_{k},diagnostics_{k}] = controller{X_{k}};
        X_{k+1} = A*X_{k} + B*U_{k} + B*V{k};
    end
    X = [X_{:}]; U = [U_{:}];
end
```

Published with MATLAB® R2023b

MECH 6V29 - MPC - Homework 3

Table of Contents

Problem 2

2a

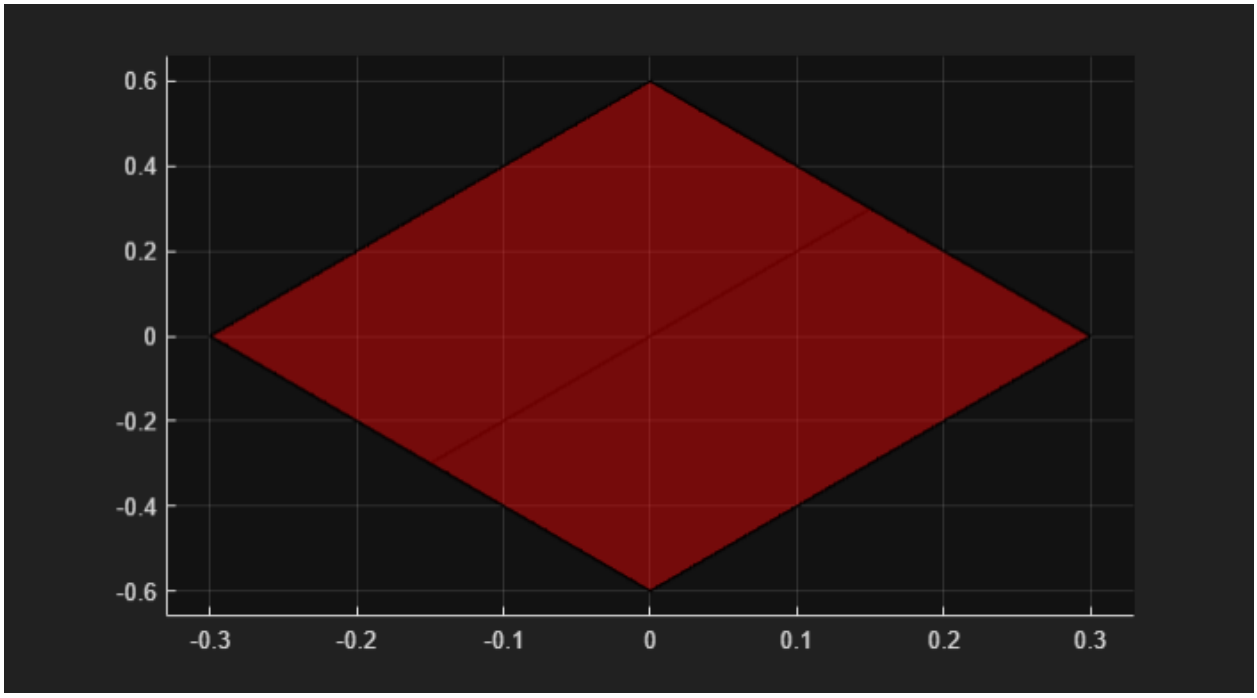
```
A = [1, 1;
    0, 1];
B = [0.5;
    1];
C = eye(3,2);%<--- [1,0;0,1;0,0]
D(3,1) = 1; %<--- [0;0;1]
sys = ss(A,B,C,D,1)

N = 10;

% sizes
nx = size(A,1);
nu = size(B,2);
ny = size(C,1);

% controller
K = -acker(A,B,zeros(nx,1));

% sets
X = Polyhedron('A',[eye(nx);-eye(nx)], 'b', ones(2*nx,1));
U = Polyhedron('A', [1;-1], 'b', ones(2*nu,1));
W = B*Polyhedron('A',[1;-1],'b',[0.3;0.3]);


sys =

  A =
       x1  x2
   x1   1   1
   x2   0   1
```

```
  B =
        u1
   x1  0.5
   x2    1

  C =
       x1  x2
   y1   1   0
   y2   0   1
   y3   0   0

  D =
       u1
   y1   0
   y2   0
   y3   1

Sample time: 1 seconds
Discrete-time state-space model.
```

2b - RPI Set

```
A_K = A+B*K;
F = W;
F.minHRep;
fig = figure; hold on;
F.plot
% drawnow
for i = 1:5
    F = F + (A_K)^(i)*W;
    F.minHRep;
    F.plot; alpha(0.1);
    % drawnow
end
hold off
saveas(fig,strcat('figs',filesep,'pblm2b_1','.png'));

% Approx code
epsilon = 1; %<== all epsilon smaller then 1 appear to make it good
F_approx = Approx_RPI(A_K,W,epsilon);

% Plot
fig = figure; hold on
F_approx.plot('color', 'blue'); alpha(1)

F.plot; alpha(0.2);
for i = 1:size(F.V,1)
    plot(F.V(i,1),F.V(i,2),'o');
end
saveas(fig,strcat('figs',filesep,'pblm2b_2','.png'))


s =
```

```
     1


s =

     2
```



2c - tightened state/input sets

```
Z = F_approx;

X_bar = X - Z; X_bar.minHRep;
U_bar = U - K*Z; U_bar.minHRep;
```

2d ----- Setup Controller

```
P=0;
Q = 1e-3*eye(nx);
R = 100;

yalmip('clear'); clear('controller');
u_bar_ = sdpvar(repmat(nu,1,N),ones(1,N));
x_bar_ = sdpvar(repmat(nx,1,N+1),ones(1,N+1));
x_1 = sdpvar(nx,1);
u_1 = sdpvar(nu,1);

constraints = []; objective = 0;
constraints = [constraints,Z.A*(x_bar_{1}-x_1) <= Z.b];
```

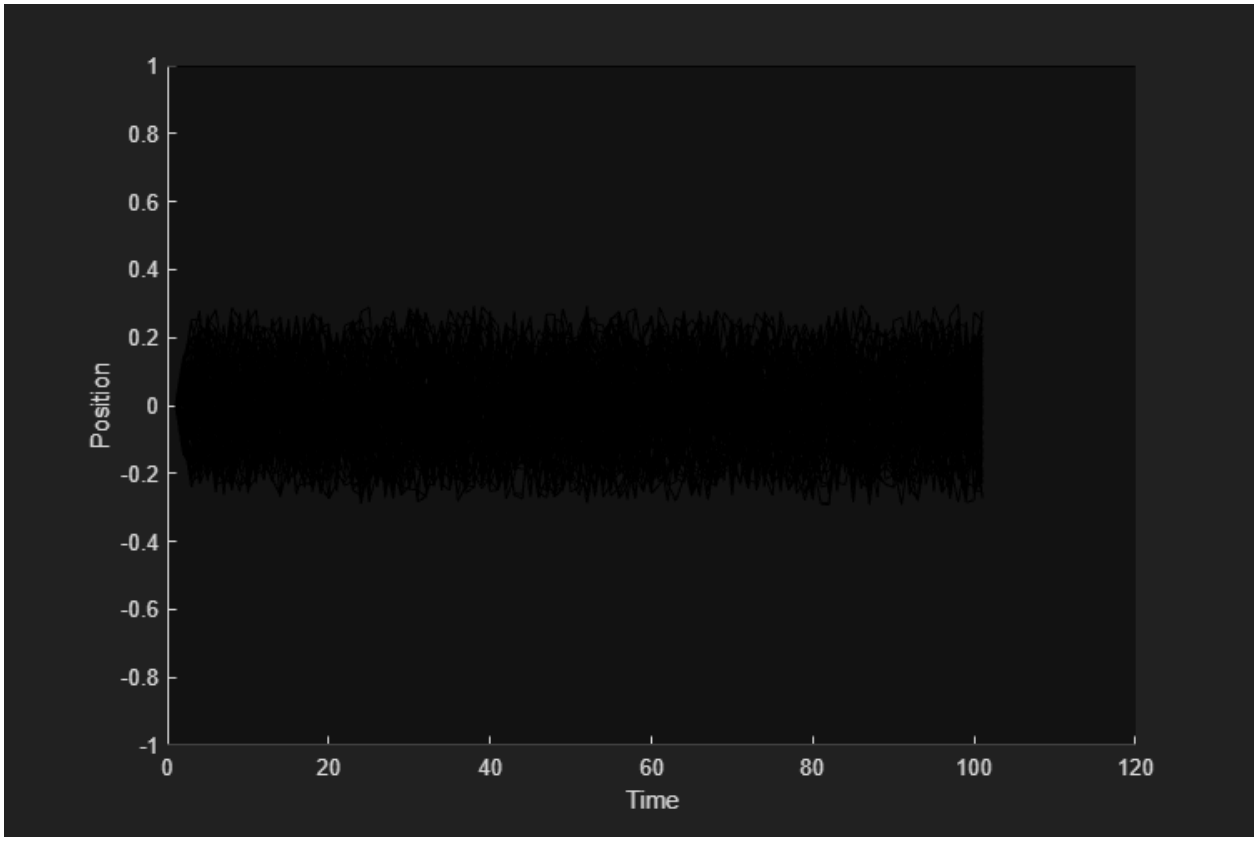
```
% constraints = [constraints, Z.A*x_bar_{1} <= Z.b]; %<-- initial condition
constraint
for k = 1:N
    objective = objective + x_bar_{k}'*Q*x_bar_{k} + u_bar_{k}'*R*u_bar_{k};
    constraints = [constraints, x_bar_{k+1} == A*x_bar_{k} + B*u_bar_{k}];
    constraints = [constraints, X_bar.A*x_bar_{k} <= X_bar.b];
    constraints = [constraints, U_bar.A*u_bar_{k} <= U_bar.b];
end
constraints = [constraints, Z.A*(x_bar_{k+1}+0)<= Z.b];
objective = objective + x_bar_{k+1}'*P*x_bar_{k+1};

constraints = [constraints, u_1 == u_bar_{1} + K*(x_1 - x_bar_{1})];

opts = sdpsettings;
controller = optimizer(constraints,objective,opts,x_1,u_1);
```

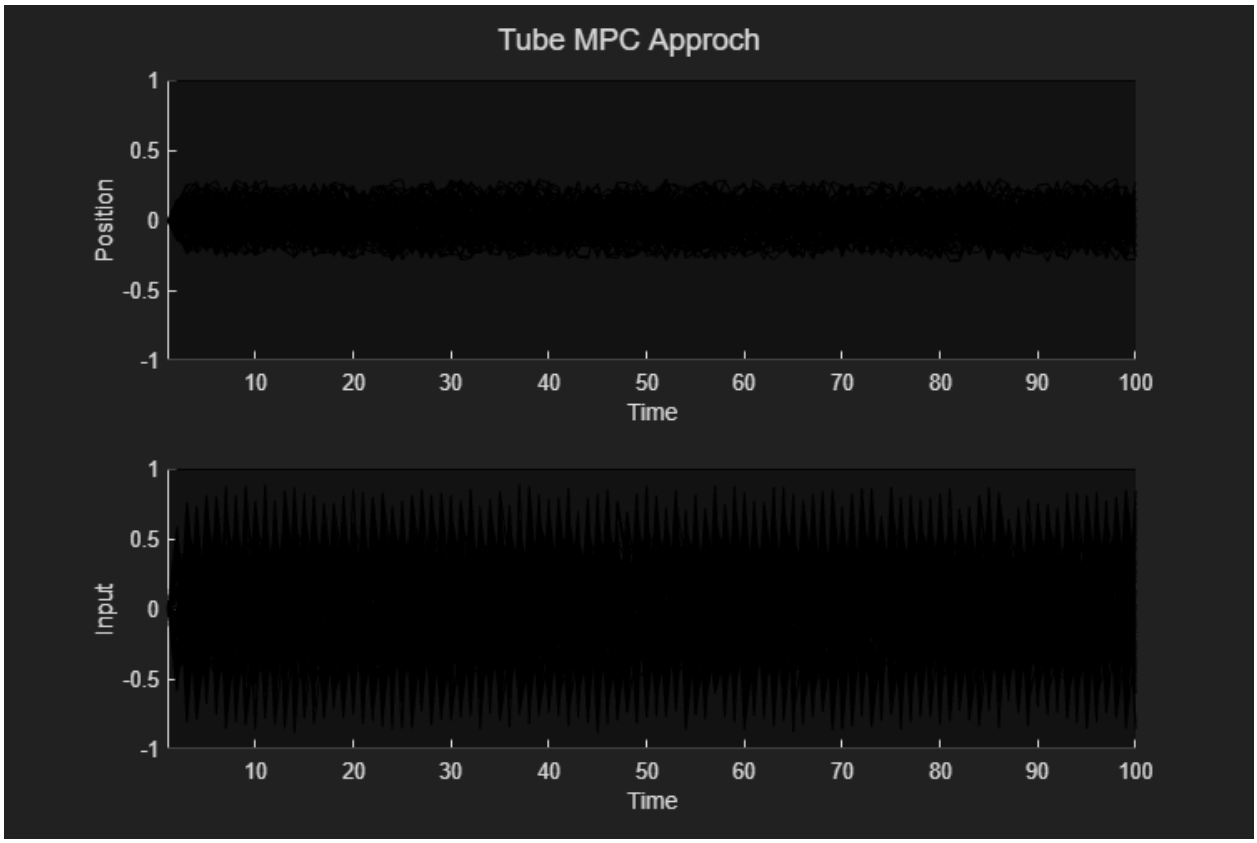
2e ----- Simulation

```
clear X U
for i = 1:100
    rng(i);
    x0 = zeros(nx,1); tf = 100;
    V = num2cell(0.6*rand(nx,tf)-0.3);
    [X{i},U{i},~] = run_sim(A,B,V,controller, x0, tf);
end
```

Ploting

```
fig = figure(...
        WindowStyle="normal",...
        Position=[0 0 750 500]);
% States
subplot(2,1,1); hold on;
yline(1,'k'); yline(-1,'k');
ylabel('Position');
xlabel('Time');
xlim([1,100]);
for i = 1:length(X); plot(X{i}(1,:),'k'); end
% Input
subplot(2,1,2); hold on;
yline(1,'k'); yline(-1,'k');
ylabel('Input');
xlabel('Time');
xlim([1,100]);
for i = 1:length(U); plot(U{i}(1,:),'k'); end

% save fig
sgtitle('Tube MPC Approch')
saveas(fig,strcat('figs',filesep,'pblm2e_results','.png'));
```


Position
Time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120


Tube MPC Approch
Position
Time
Input
Time
1
0.5
0
-0.5
-1
10
20
30
40
50
60
70
80
90
100

2f ------- Result Analysis

Cost

```
J_{100} = [];

for i = 1:100
    J_{i} = 0;
    for k = 1:tf-1
        J_{i} = J_{i} + X{i}(:,k)'*Q*X{i}(:,k) + U{i}(:,k)'*R*U{i}(:,k);
    end
    J_{i} = J_{i} + X{i}(:,k+1)'*P*X{i}(:,k+1);
end
J = [J_{:}];

J_mean = mean(J)
J_max = max(J)
```

```
J_mean =

   1.4650e+03


J_max =

   1.9730e+03
```

Local functions

function controller = mpc_yalmip_controller(A,B,P,Q,R,N,cons,cons_f) yalmip('clear') nx = size(A,1); nu = size(B,2);

```
   u_ = sdpvar(repmat(nu,1,N),ones(1,N));
   x_ = sdpvar(repmat(nx,1,N+1),ones(1,N+1));
   s_ = sdpvar(ones(1,N+1),ones(1,N+1));

   constraints = [];
   objective = 0;
   for k = 1:N
       objective = objective + x_{k}'*Q*x_{k} + u_{k}'*R*u_{k} + s_{k};
       constraints = [constraints, s_{k} >= 0];
       constraints = [constraints, x_{k+1} == A*x_{k} + B*u_{k}];
       constraints = [constraints, cons(x_{k+1},u_{k},s_{k})];
   end
   constraints = [constraints,cons_f(x_{k+1},u_{k},s_{k})];
   objective = objective + x_{k+1}'*P*x_{k+1};

   opts = sdpsettings;
   controller = optimizer(constraints,objective,opts,x_{1},u_{1});
end
```

```
function [X,U,diagnostics_] = run_sim(A,B,V,controller,x0, tf)

    X_{tf+1} = []; U_{tf} = []; diagnostics_{tf} = [];
    X_{1} = x0;
    for k = 1:tf
        [U_{k},diagnostics_{k}] = controller{X_{k}};
        X_{k+1} = A*X_{k} + B*U_{k} + B*V{k};
    end
    X = [X_{:}]; U = [U_{:}];
end

% function fig = plot_trajectory(X, U)
%     fig = figure(...
%         WindowStyle="normal",...
%         Position=[0 0 750 500]);
%     hold on; grid on;
%     subplot(2,1,1);
%     stairs(X')
%     title('State Trajectory')
%     legend({'x_1','x_2'})
%     subplot(2,1,2);
%     stairs(U');
%     title('Input Trajectory')
%     legend({'u_1'})
% end
```

Published with MATLAB® R2023b