MECH 6v29 - Model Predictive Control
Homework 1

Jonas Wagner
jonas.wagner@utdallas.edu

2023, September 15^{th}

Contents

Problem 1

1a)

See attached MATLAB results
The dynamics show a fairly stable underdamped response. The static gain is 3, the overshoot is around 30 %, and the settling time is around 15 s.

1b)

See MATLAB results
The results are not the same, at least for the short prediction horizon

1c)

Se MATLAB Results

1d)

See MATLAB Results

1e)

See MATLAB Results
From the results, it is clear that the closed-loop system results become more stable as the prediction horizon increases and converges upon the LQR-solution. It appears as though the system enters stability around when $N = 6$, although I am uncertain if this is always the case, or just for this state specifically.
This is around half the settling time, so perhaps this would be a good guess.

Table of Contents

MPC HW 1 - Problem 1

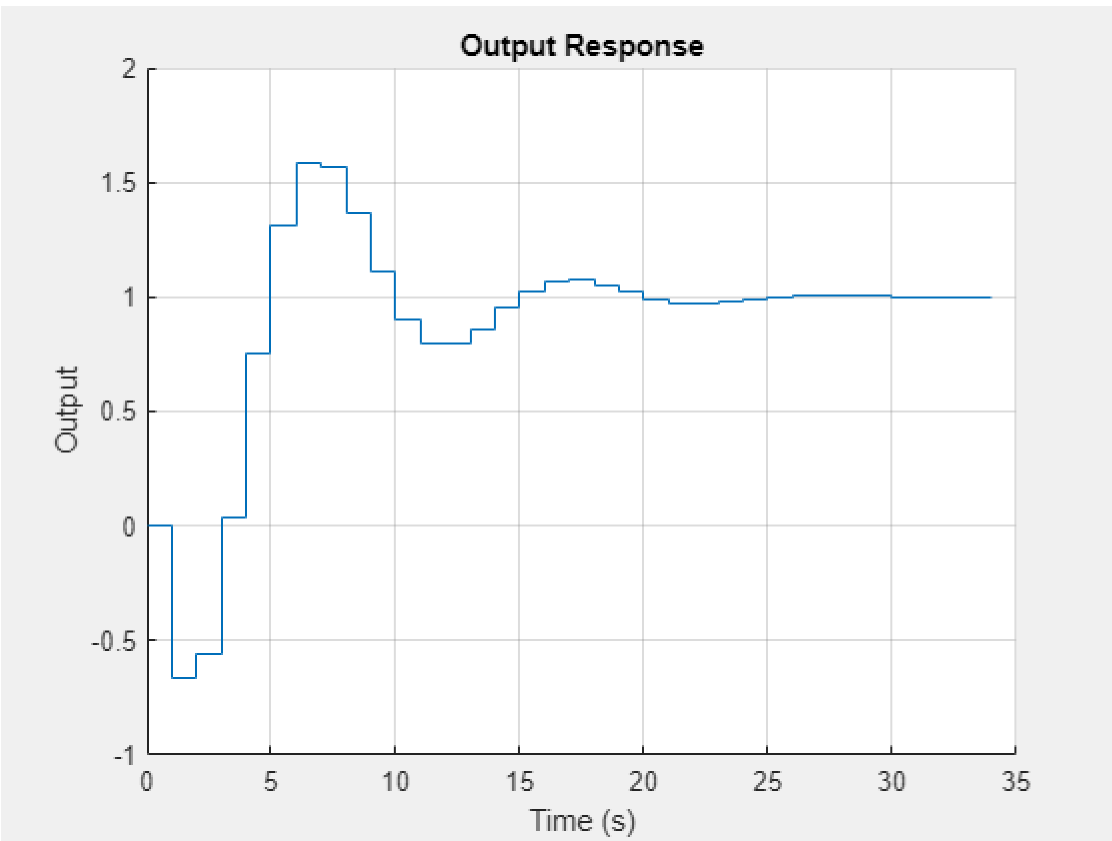
```
clear
close all
subfolder = fileparts(mfilename('fullpath'));
if ~isfolder('figs'); mkdir('figs'); end

% Problem Information
A = [4/3, -2/3; 1, 0];
B = [1; 0];
C = [-2/3, 1];
D = 0;
dt = 1;
sys = ss(A,B,C,D,dt);

% Size parameters
nx = size(A,1);
nu = size(B,2);
```

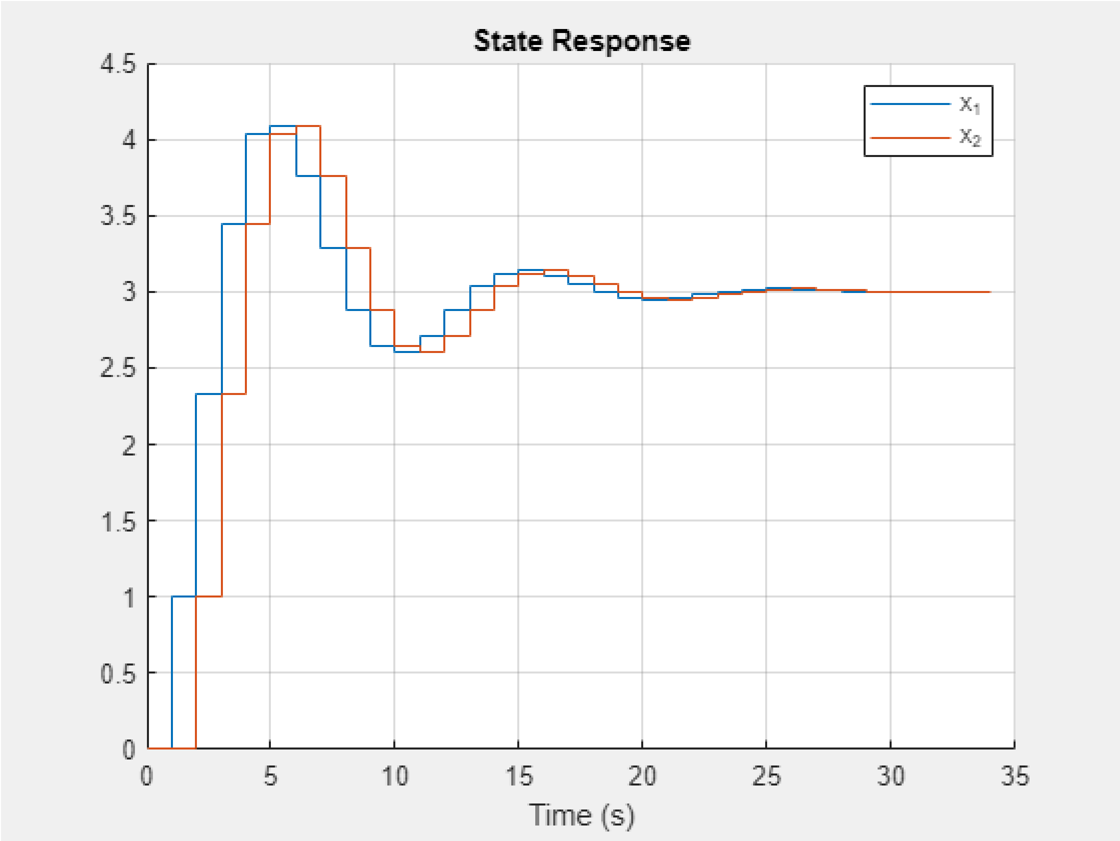
Part 1a

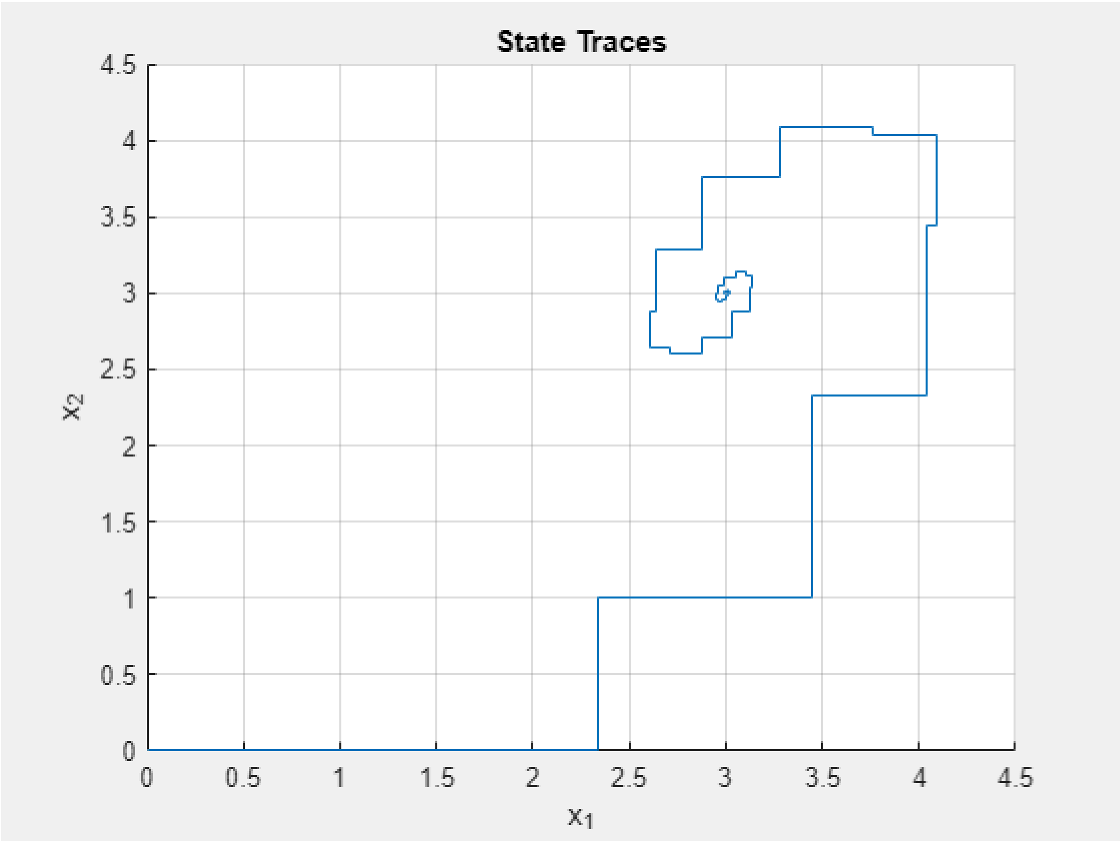
Step Response

```
[y,t,x] = step(sys);

% Output Response
figName = 'pblm1a_fig1';
fig = figure(WindowStyle="normal");
hold on; grid on;
stairs(t,y,"DisplayName",'Output');
title('Output Response')
xlabel('Time (s)')
ylabel('Output')
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')

% State Response
figName = 'pblm1a_fig2';
fig = figure(WindowStyle="normal");
```

```
hold on; grid on;
stairs(t,x(:,1),"DisplayName",'x_1');
stairs(t,x(:,2),"DisplayName",'x_2');
legend
title('State Response')
xlabel('Time (s)')
ylabel('')
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')

% State Response
figName = 'pblm1a_fig3';
fig = figure(WindowStyle="normal");
hold on; grid on;
stairs(x(:,1),x(:,2))
title('State Traces')
xlabel('x_1')
ylabel('x_2')
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
```




State Response
x1
x2
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30
35
Time (s)


State Traces
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
x2
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
x1

Part 1b

MPC Parameters (standard optimization)

```
Q = C'*C + 1e-3*eye(nx);
R = 1e-3;
P = Q;
N = 5; % prediction horizon
```

Batch method

Construct S_x,and S_u

```
for i = 1:N+1
    S_x_{i} = A^(i-1);
    for j = 1:nu
        S_u__{j} = eye(nu)*A^(i-2+j-1)*B;
    end
    S_u_{i} = horzcat(S_u__{:});
end
S_x = vertcat(S_x_{:});
S_u = tril(vertcat(S_u_{:}),-1);

% Construct Qbar and Rbar
Qbar = blkdiag(kron(Q,eye(N)),P);
Rbar = kron(R,eye(N));

% H, F, Y
H = S_u'*Qbar*S_u + R;
F = S_x'*Qbar*S_u;
Y = S_x'*Qbar*S_x;

% Results
Ustar = @(x_0) -H\F'*x_0;
ustar = @(x_0) [eye(nx),zeros(nx,nx*(N-1))]*-H\F'*x_0;
K_0_batch = -H\F';

disp('Batch Method:')
disp('K = '); disp(K_0_batch);
```

```
Batch Method:
K =
   -0.2753    0.6666
```

Dynamic Programing Method

```
F_fun = @(P_k) -(B'*P_k*B+R)\B'*P_k*A;
P_fun = @(P_kp1) A'*P_kp1*A + Q - A'*P_kp1*B*inv(B'*P_kp1*B+R)*B'*P_kp1*A;

P_{N} = Q;

for k = N-1:-1:1
```

```
    P_{k} = P_fun(P_{k+1});
    F_{k} = F_fun(P_{k});
end
P_0 = P_fun(P_{1});
F_0 = F_fun(P_0);

ustar = @(x) F_0*x;
K_0_dynprog = F_0;

disp('Dynamic Programing Method:')
disp('K = '); disp(K_0_dynprog);

% results
disp('They are not the same, or at least not with a time-horrizon on $N=5$')
```

```
Dynamic Programing Method:
K =
   -0.1739    0.6655

They are not the same, or at least not with a time-horrizon on $N=5$
```

Part c

```
disp('Batch version')
A_K_batch = A+B*K_0_batch
eig_batch = eig(A_K_batch)

disp('The system is not closed-loop stable acording to this as there is an
 eigen value >= 1')

disp('Dynamic Programing Method')
A_K_dynprog = A+B*K_0_dynprog
eigh_dynprog = eig(A_K_dynprog)

disp('The system is not closed-loop stable acording to this as there is an
 eigen value >= 1')
```

```
Batch version

A_K_batch =

    1.0580   -0.0001
    1.0000         0


eig_batch =

    1.0579
    0.0001

The system is not closed-loop stable acording to this as there is an eigen
 value >= 1
Dynamic Programing Method
```

```
A_K_dynprog =

    1.1594   -0.0012
    1.0000         0


eigh_dynprog =

    1.1584
    0.0010

The system is not closed-loop stable acording to this as there is an eigen
 value >= 1
```

Part d

```
K_lqr = -dlqr(A,B,Q,R)
A_K_lqr = A+B*K_lqr
eig_lqr = eig(A_K_lqr)

disp('LQR is stable')
```

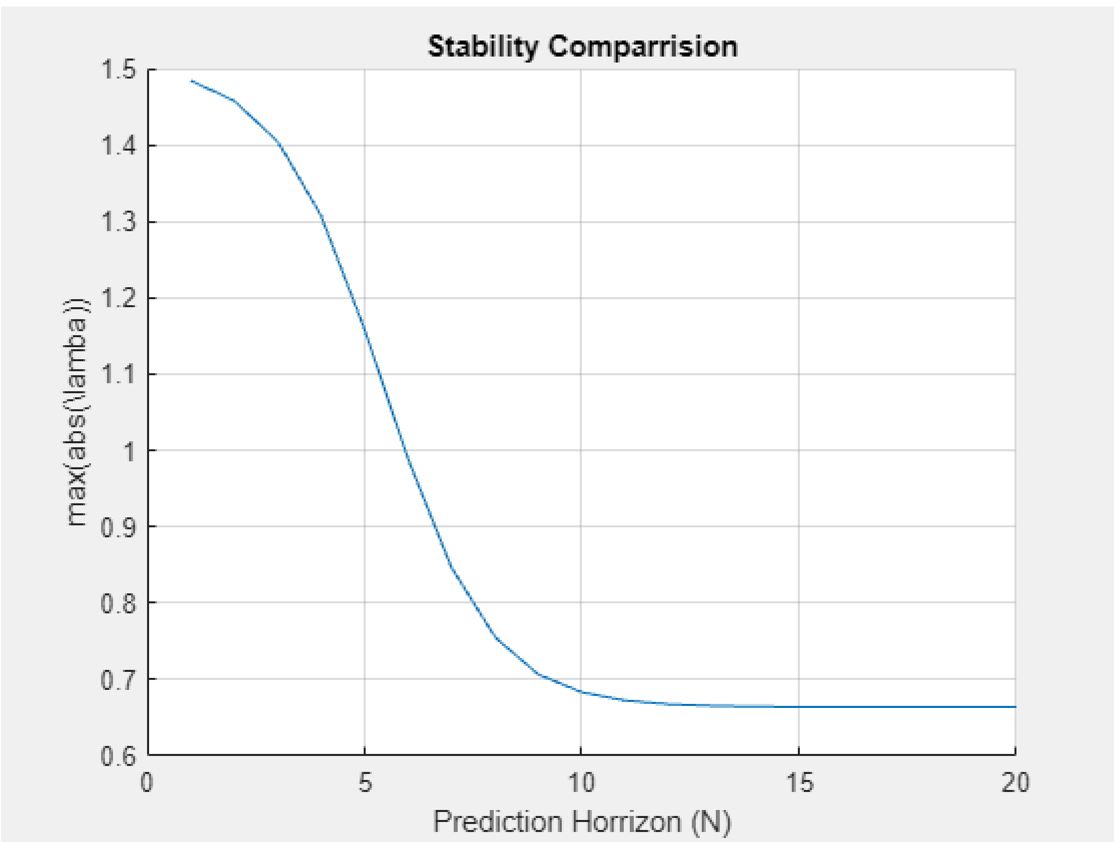
```
K_lqr =

   -0.6683    0.6660


A_K_lqr =

    0.6650   -0.0007
    1.0000         0


eig_lqr =

    0.6640
    0.0010

LQR is stable
```

Part e

```
for N = 1:20
    F_fun = @(P_k) -(B'*P_k*B+R)\B'*P_k*A;
    P_fun = @(P_kp1) A'*P_kp1*A + Q - A'*P_kp1*B*...
        inv(B'*P_kp1*B+R)*B'*P_kp1*A;

    P_{N} = Q;
    for k = N-1:-1:1
        P_{k} = P_fun(P_{k+1});
        F_{k} = F_fun(P_{k});
    end
```

```
    P_0 = P_fun(P_{1});
    F_0 = F_fun(P_0);

    K_0_dynprog_{N} = F_0;
    A_K_dynprog_{N} = A+B*K_0_dynprog_{N};
    eig_dynprog_{N} = eig(A+B*K_0_dynprog_{N});
    max_eig_{N} = max(abs(eig_dynprog_{N}));
end

max_eig = [max_eig_{:}];

figName = 'pblm1e';
fig = figure(WindowStyle="normal");
hold on; grid on;
plot(1:N,max_eig)
title('Stability Comparrision')
xlabel('Prediction Horrizon (N)')
ylabel('max(abs(\lamba))')
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
```

```
Warning: Error updating Text.

 String scalar or character vector must have valid interpreter syntax:
max(abs(\lamba))
```



Published with MATLAB® R2023a

Problem 2

Generally my results to this problem were not as I expected, but I'm pretty sure they are correct given the system listed in the assignment.

2a)

See MATLAB code
The lack of any major transient behavior is weird, but I'm guessing that this is becouse the lack of constraints allow it to reach it's "steady-state" after only one time-step

2b)

See MATLAB code
This is a much much better result (as is often expected with increased time-horizons)

2c)

See MATLAB code
This made it unstable. There is no longer enough input constraints to counteract the main response of the system.

2d)

See MATLAB code
It was unclear whether the input constraints were also to be included, thus both versions were tested.
When only the state is restricted, the response of the system is just slowed and the input decreases as expected until it reaches zero. However, the MPC problem becomes unfeasible when both the input and state constraints are implimented.

Table of Contents

MPC HW 1 - Problem 2

clear

```
close all
subfolder = fileparts(mfilename('fullpath'));
if ~isfolder('figs'); mkdir('figs'); end

yalmip('clear')

% Problem Information
A = [1,1;1,0];
B = [0; 1];
C = [1, 0];
D = 0;
dt = 1;
sys = ss(A,B,C,D,dt);

nx = size(A,1);
nu = size(B,2);
```

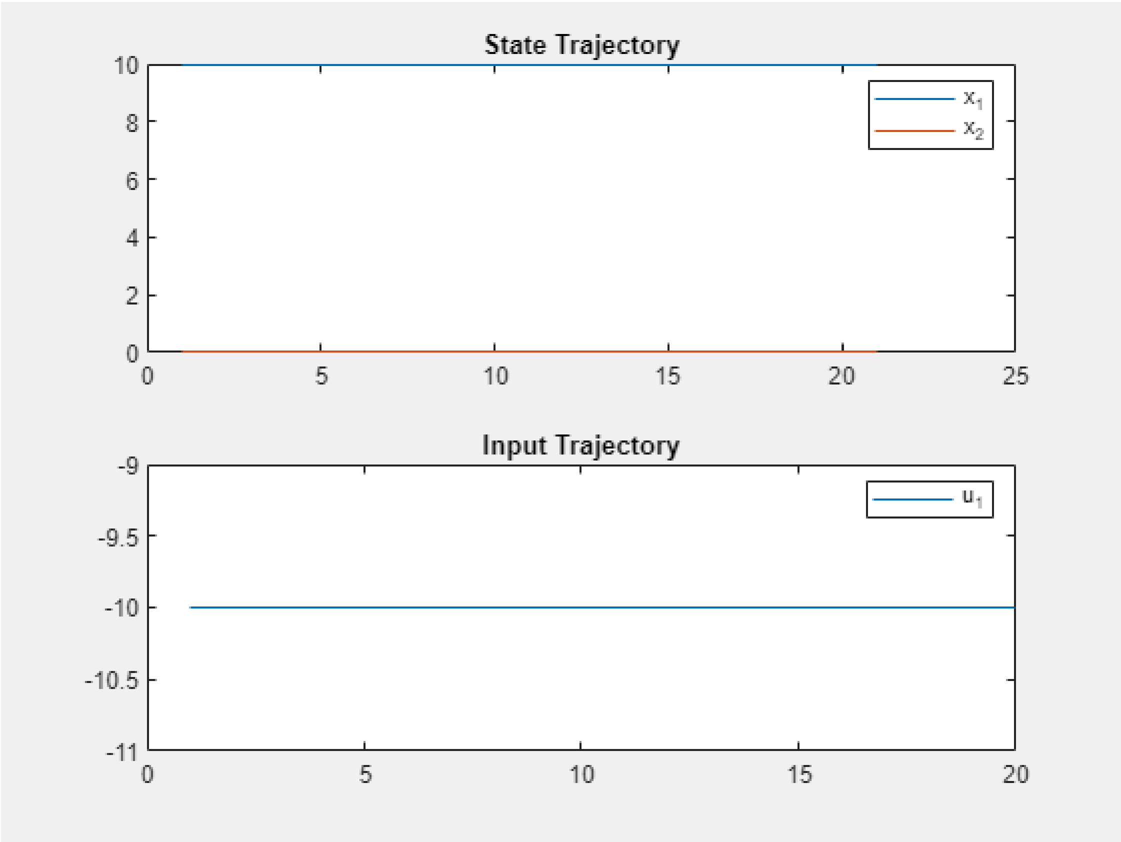
Part a

MPC Parameters

```
Q = eye(size(A,1));
R = 0.1;
N = 3;

u_con = @(u) [];
x_con = @(x) [];

controller = mpc_yalmip_controller(A, B, Q, R, N, u_con, x_con);

% Simulation setting
x0 = [10; 0];
tf = 20;

U_ = cell(tf,1); X_ = cell(tf,1);
X_{1} = x0;
```

```
for k = 1:tf
    U = controller{X_{k}};
    U_{k} = U(1);
    X_{k+1} = A*X_{k} + B*U_{k};
end
```

```
% Results
figName = 'pblm2a';
fig = figure(WindowStyle="normal");
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'})
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
```

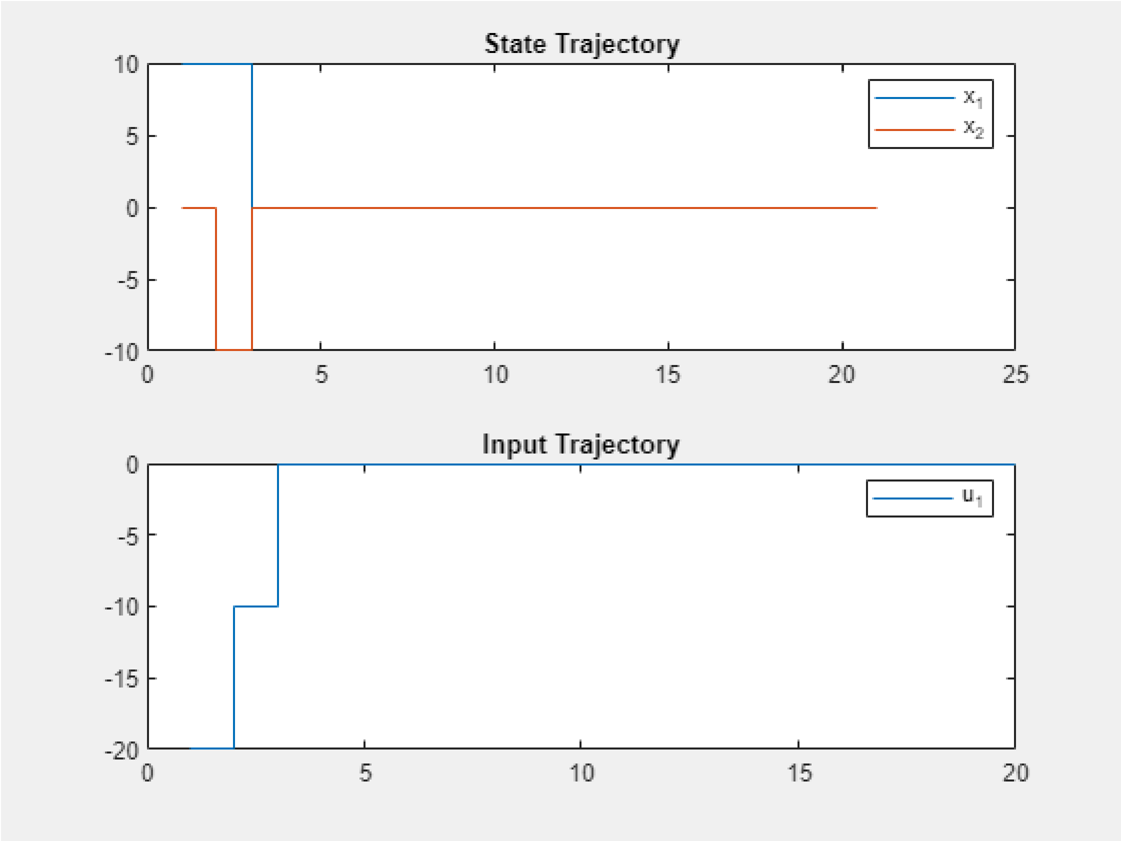


Part b

MPC Parameters

```
Q = eye(size(A,1));
```

```
R = 0.1;
N = 10;

u_con = @(u) [];
x_con = @(x) [];

controller = mpc_yalmip_controller(A, B, Q, R, N, u_con, x_con);

% Simulation setting
x0 = [10; 0];
tf = 20;

U_ = cell(tf,1); X_ = cell(tf,1);
X_{1} = x0;
for k = 1:tf
    U = controller{X_{k}};
    U_{k} = U(1);
    X_{k+1} = A*X_{k} + B*U_{k};
end


% Results
figName = 'pblm2b';
fig = figure(WindowStyle="normal");
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'})
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
```

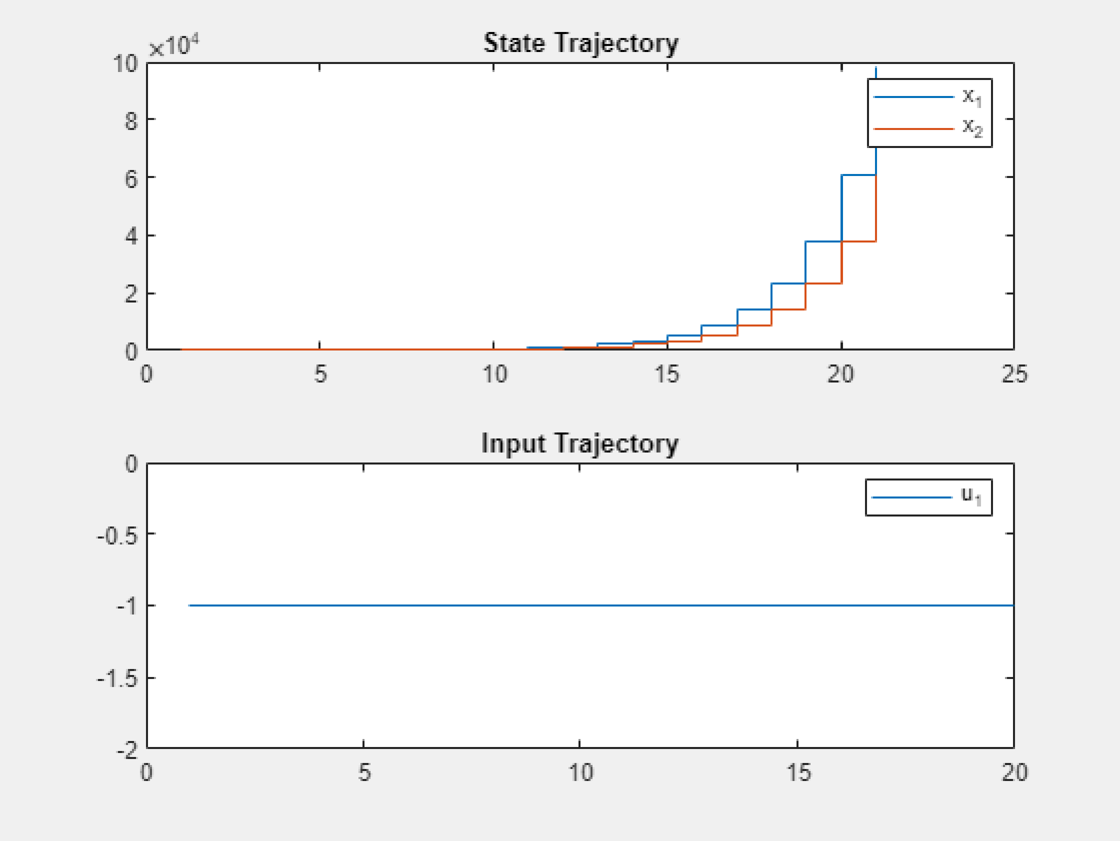


Part c

MPC Parameters

```
Q = eye(size(A,1));
R = 0.1;
N = 10;

u_con = @(u) -1<=u<=1;
x_con = @(x) [];

controller = mpc_yalmip_controller(A, B, Q, R, N, u_con, x_con);

% Simulation setting
x0 = [10; 0];
tf = 20;

U_ = cell(tf,1); X_ = cell(tf,1);
X_{1} = x0;
for k = 1:tf
    U = controller{X_{k}};
    U_{k} = U(1);
    X_{k+1} = A*X_{k} + B*U_{k};
end
```

```
% Results
figName = 'pblm2c';
fig = figure(WindowStyle="normal");
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'})
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
```

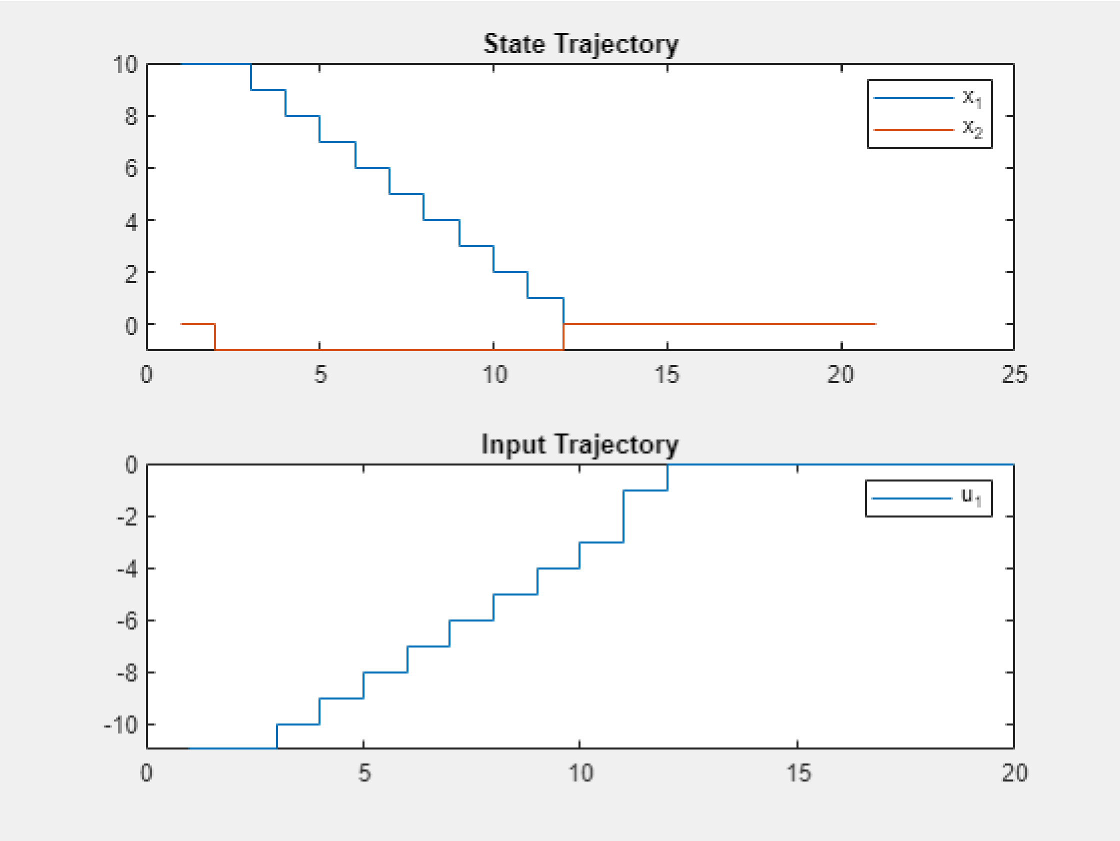


Part d

MPC Parameters

```
Q = eye(size(A,1));
R = 0.1;
N = 10;

u_con = @(u) [];%-1 <= u <=1;
x_con = @(x) -1 <= x(2) <= 1;

controller = mpc_yalmip_controller(A, B, Q, R, N, u_con, x_con);
```

```
% Simulation setting
x0 = [10; 0];
tf = 20;

U_ = cell(tf,1); X_ = cell(tf,1);
X_{1} = x0;
for k = 1:tf
    U = controller{X_{k}};
    U_{k} = U(1);
    X_{k+1} = A*X_{k} + B*U_{k};
end


% Results
figName = 'pblm2d';
fig = figure(WindowStyle="normal");
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'})
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
```

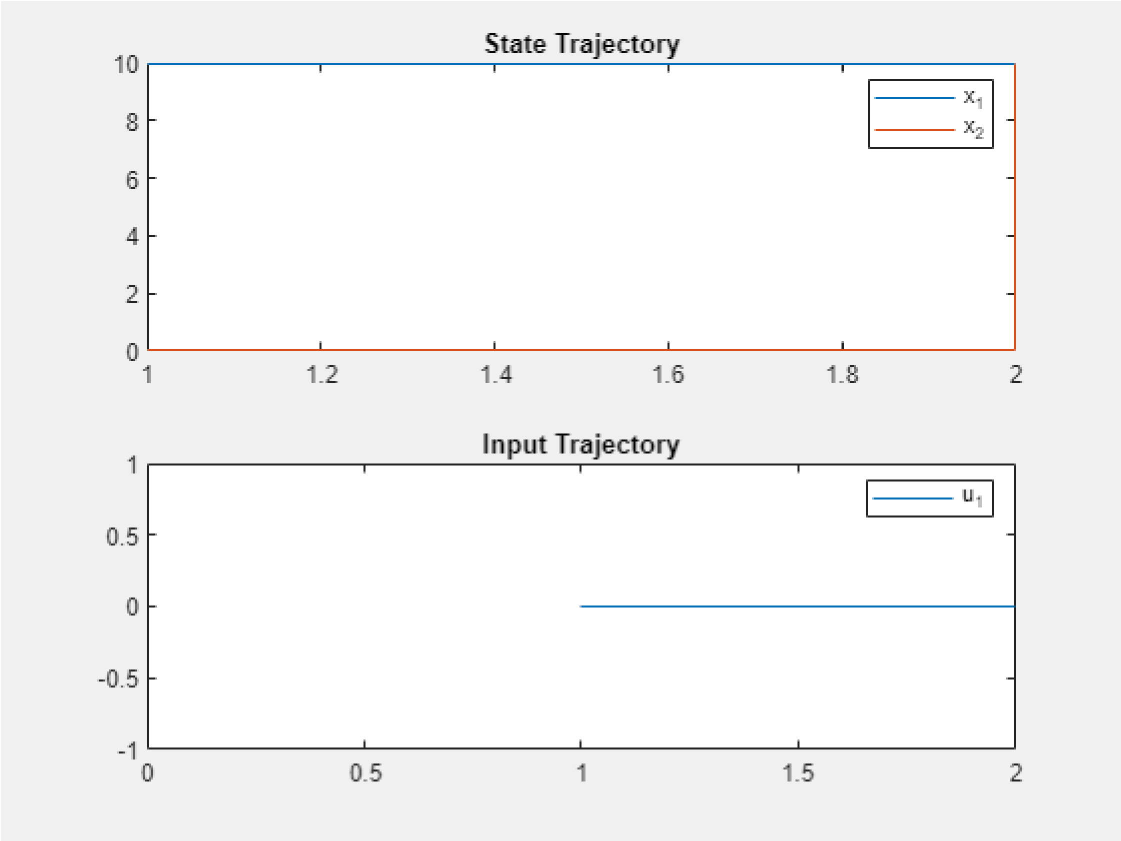


Part d (2nd)

MPC Parameters

```
Q = eye(size(A,1));
R = 0.1;
N = 10;

u_con = @(u) -1 <= u <=1;
x_con = @(x) -1 <= x(2) <= 1;

controller = mpc_yalmip_controller(A, B, Q, R, N, u_con, x_con);

% Simulation setting
x0 = [10; 0];
tf = 20;

U_ = cell(tf,1); X_ = cell(tf,1);
X_{1} = x0;
for k = 1:tf
    U = controller{X_{k}};
    U_{k} = U(1);
    X_{k+1} = A*X_{k} + B*U_{k};
end
```

```
% Results
figName = 'pblm2d_2';
fig = figure(WindowStyle="normal");
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'})
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
```



ending

```
close all
```

Local Functions

```
function controller = mpc_yalmip_controller(A,B, Q,R, N, u_con, x_con)
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));

constraints = [];
objective = 0;
for k = 1:N
    objective = objective + norm(Q*x_{k},1) + norm(R*u_{k},1);
    constraints = [constraints, x_{k+1} == A*x_{k} + B*u_{k}];
    constraints = [constraints, u_con(u_{k})];
    constraints = [constraints, x_con(x_{k})];
end

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

Published with MATLAB® R2023a