Table of Contents

MPC HW 1 - Problem 2

clear

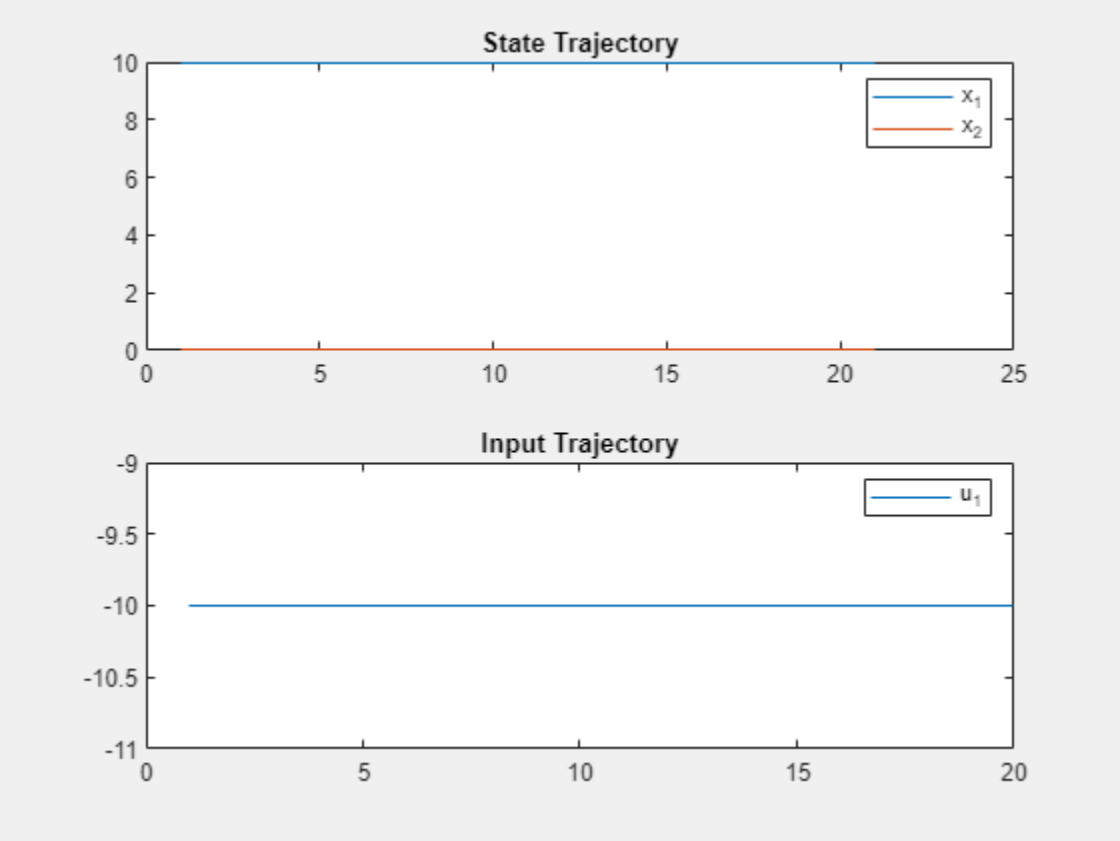
```
close all
subfolder = fileparts(mfilename('fullpath'));
if ~isfolder('figs'); mkdir('figs'); end

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')
```

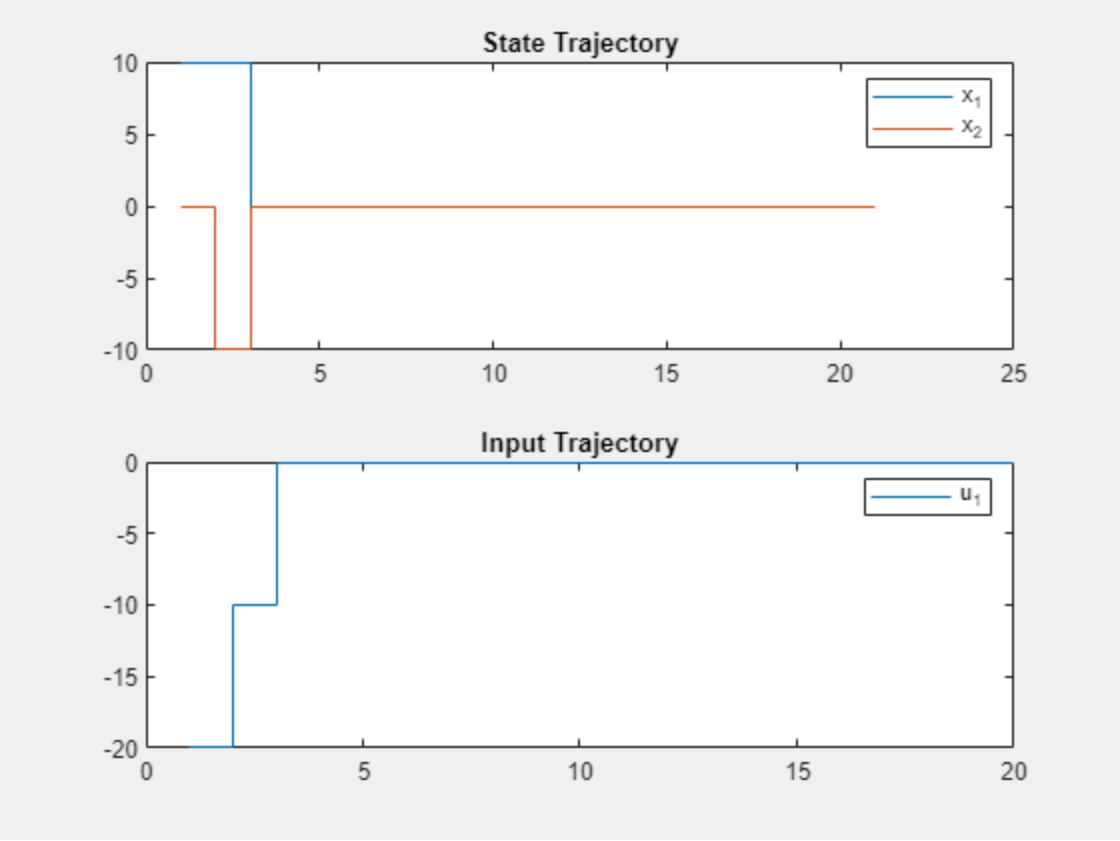


Part b

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')
```

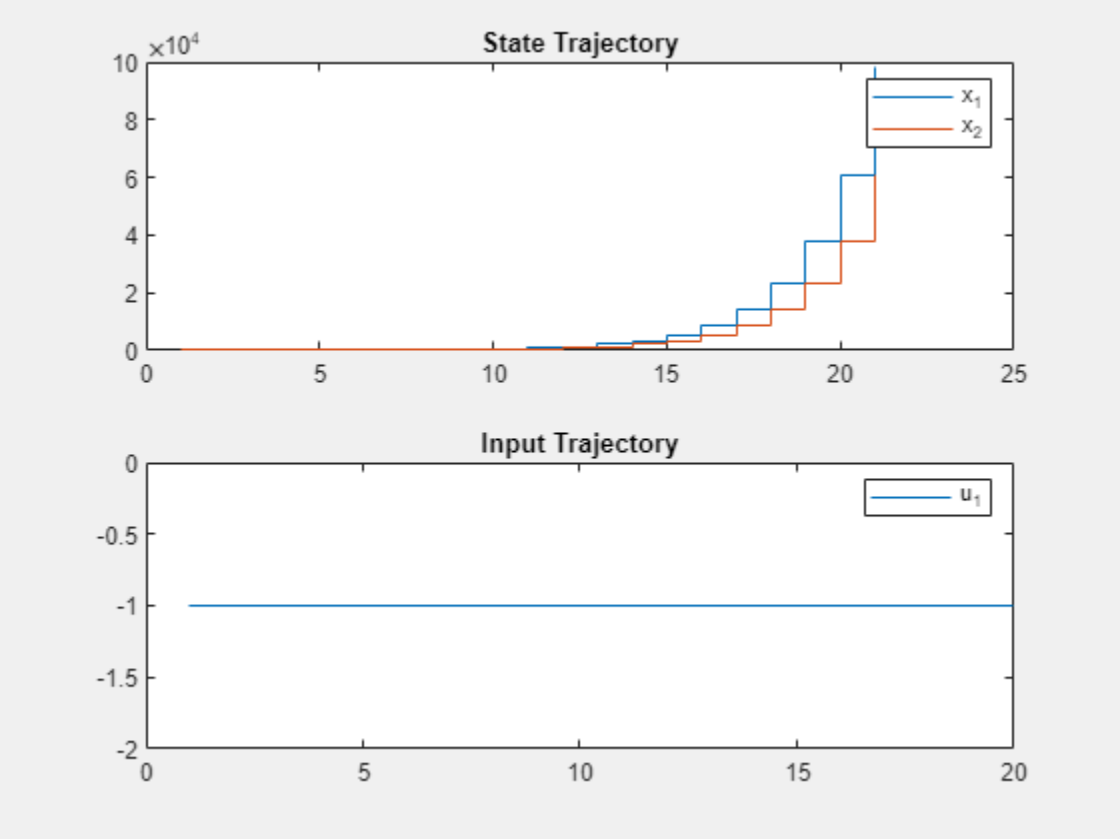


Part c

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')
```

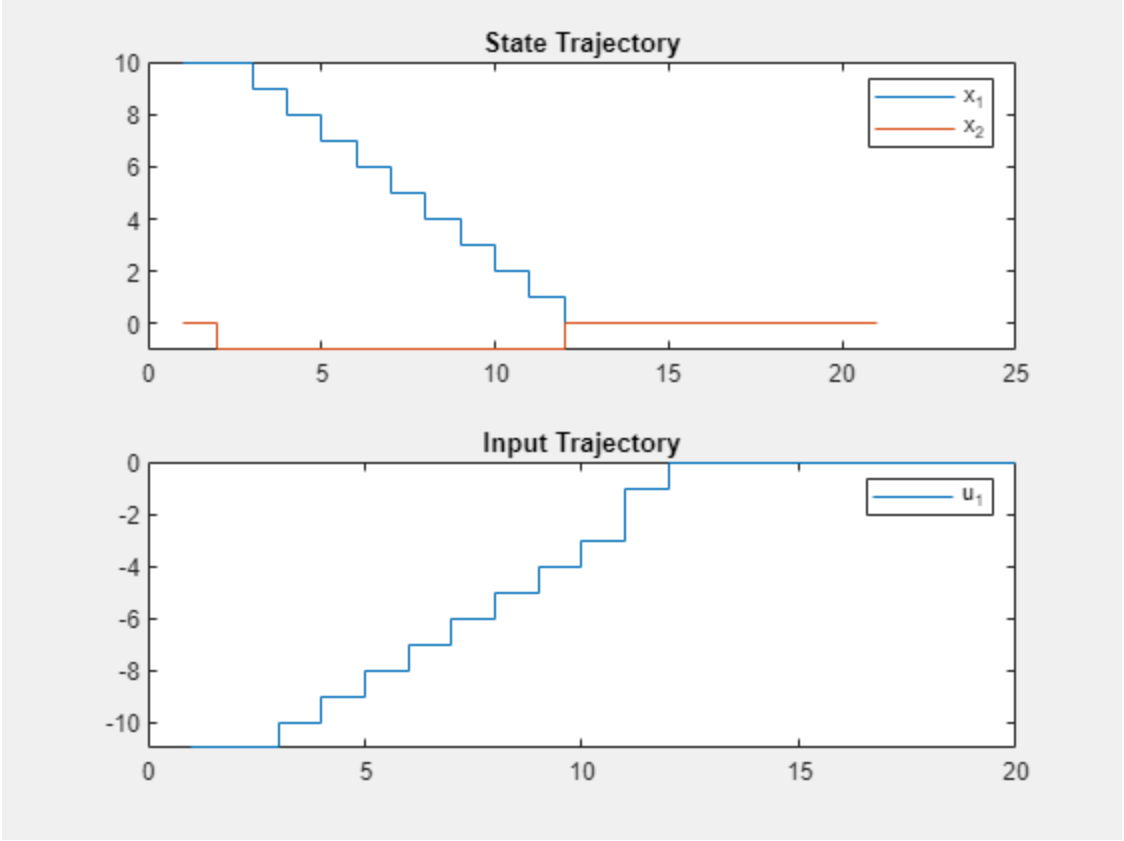


Part d

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')
```

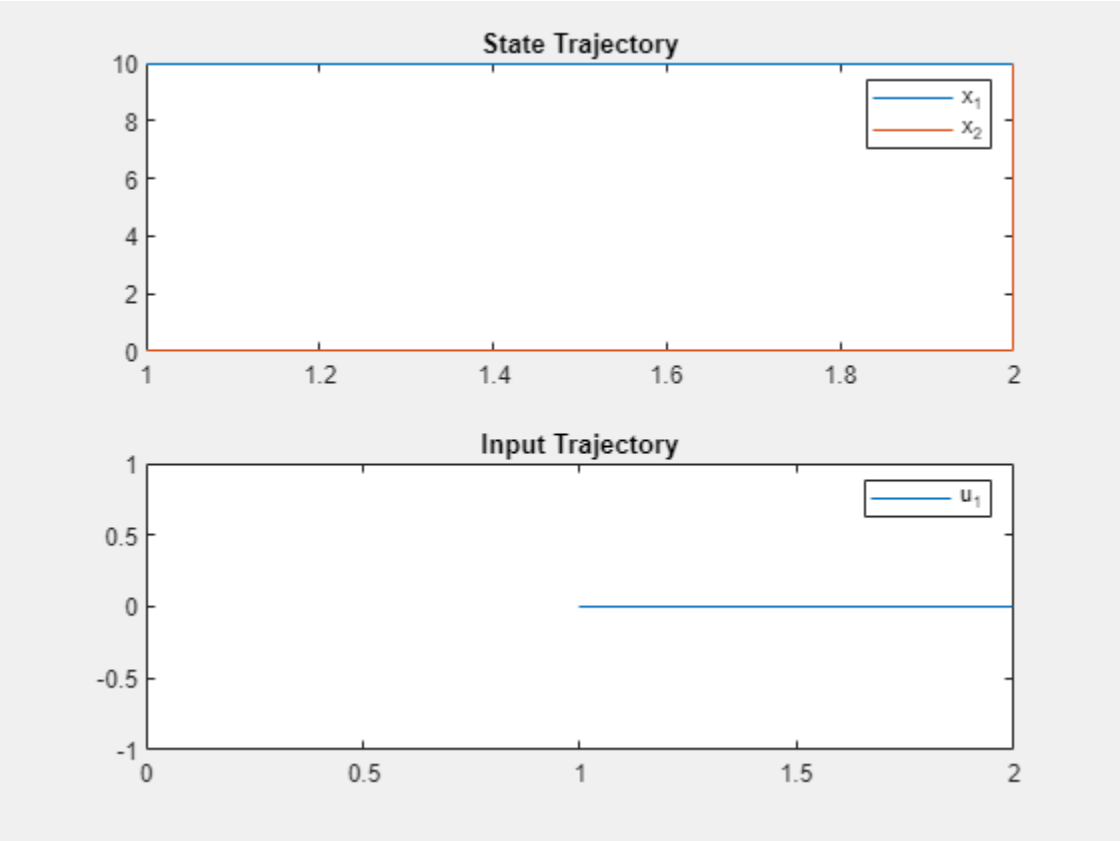


Part d (2nd)

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