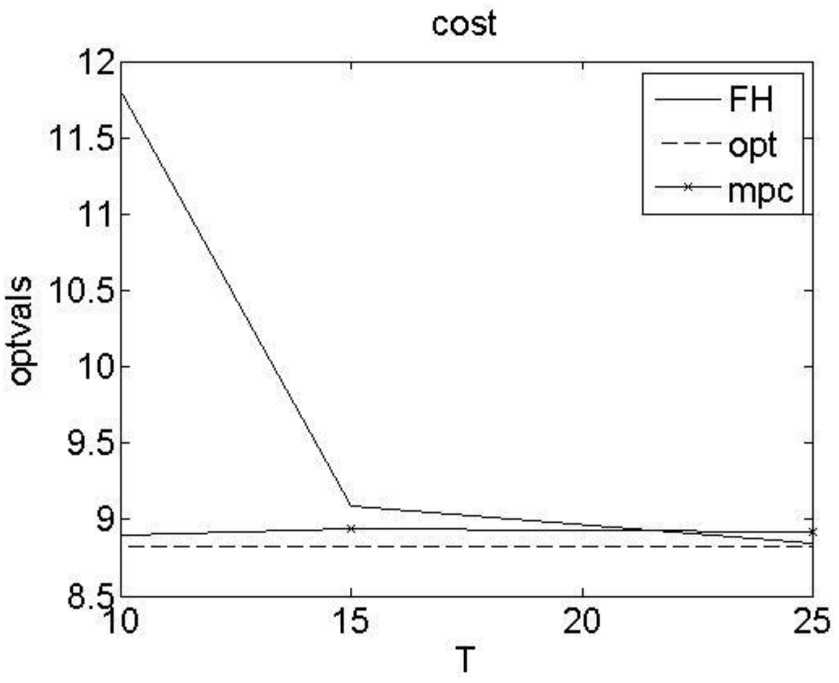
AA 203 Problem Set 7

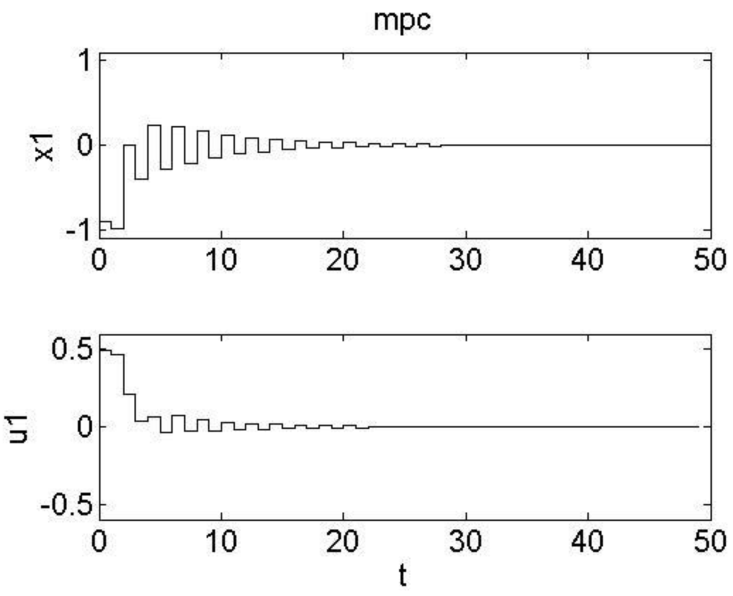
Somrita Banerjee

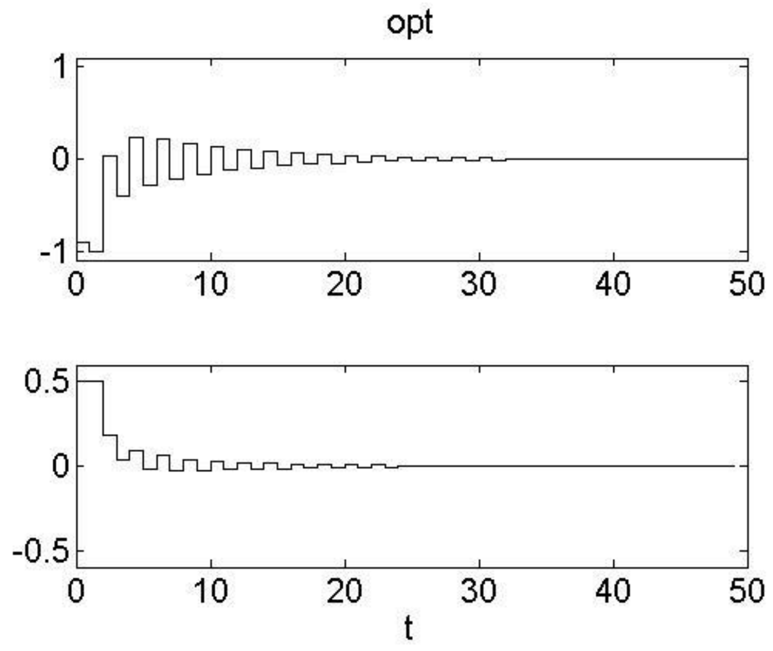
Problem 1

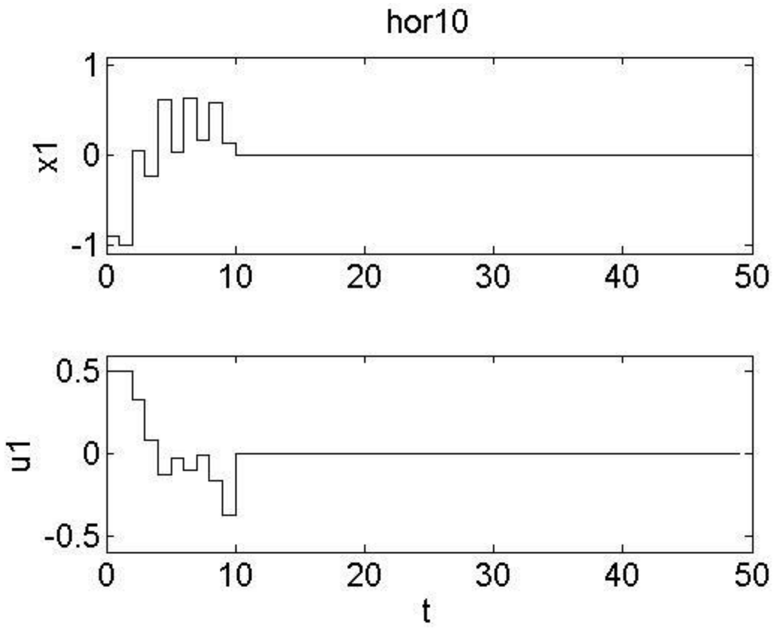
We see that the MPC controller solution cost comes close to the infinite horizon solution cost. On the other hand, the finite horizon cost can be significantly higher, especially when we only predict out to shorter horizons.



In addition, we can see that the evolution of state and control in the MPC case is much closer to the optimal (infinite horizon) case as compared to the finite horizon case.




opt
1
0
-1
0
10
20
30
40
50
0.5
0
-0.5
0
10
20
30
40
50
t


hor10
x1
1
0
-1
0
10
20
30
40
50
u1
0.5
0
-0.5
0
10
20
30
40
50
t

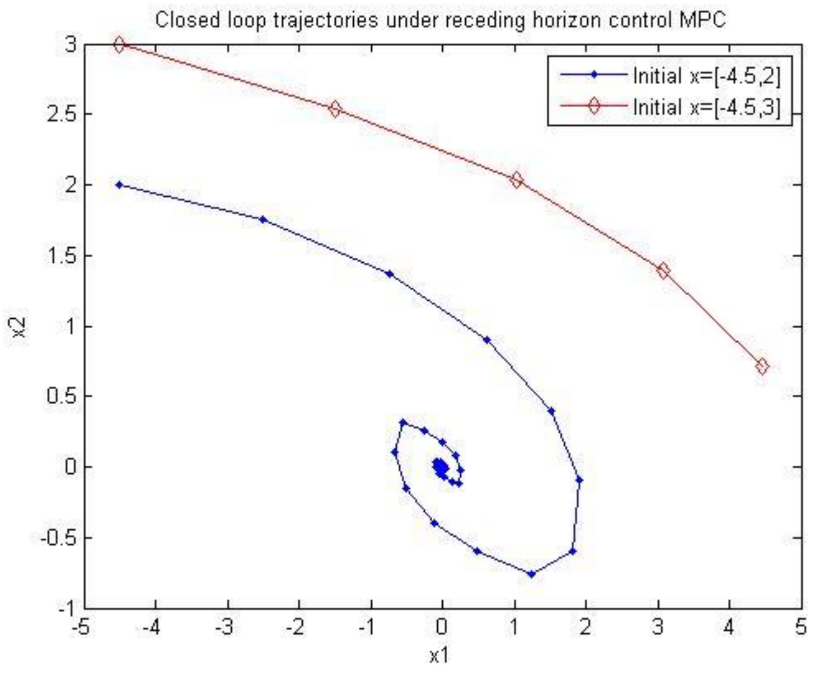
Problem 2

Part a

See code for implementation of the receding horizon controller with a terminal cost and a calcul ation horizon of 50.

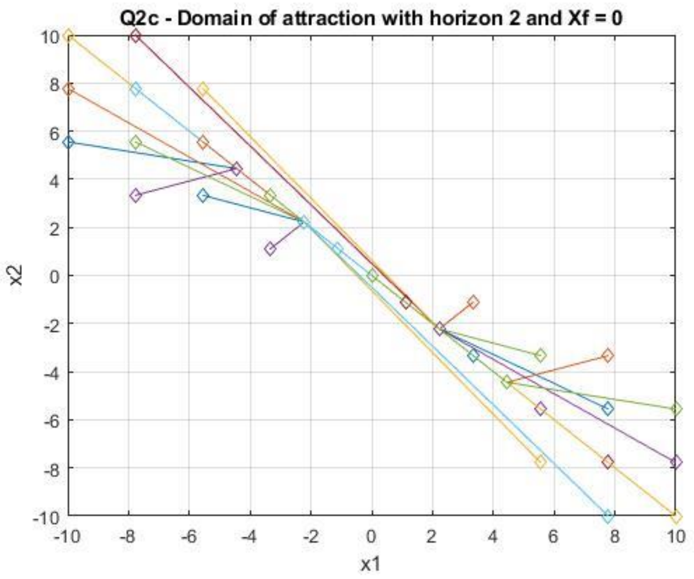
Part b

These trajectories match the ones in the book. The trajectory starting from [-4.5,2] converges to the origin and the trajectory starting from [-4.5,3] is infeasible after 5 steps.



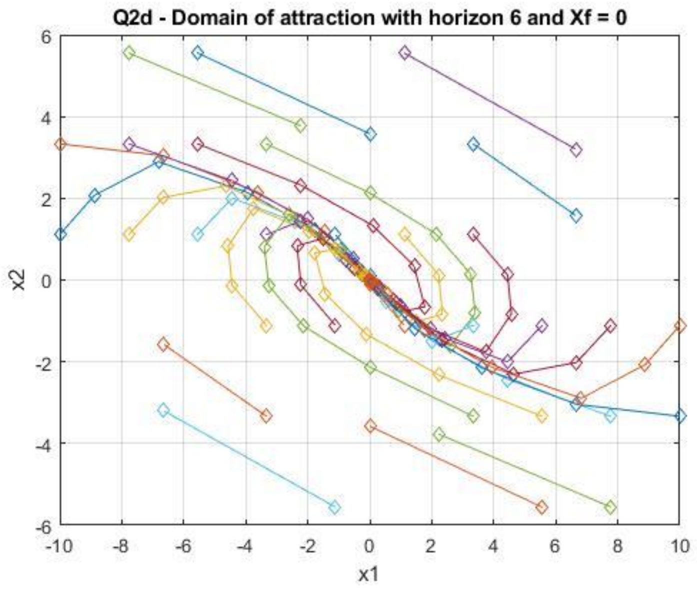
Part c

I discretized the state space with 100 points, 10 along each dimension x1 and x2.



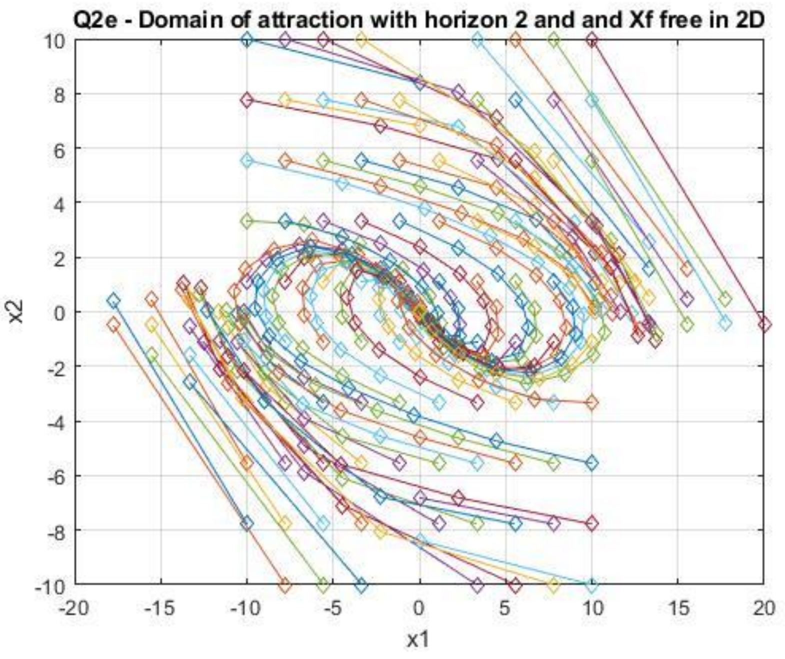
Domain of attraction: Points close to the line joining (-10,10) to (10,-10) that can **reach the origin in exactly 2 time steps**. In particular, points in the regions $\{x_1 < 0,\ x_2 > -x_1\}$ and $\{x_1 > 0,\ x_2 < -x_1\}$ are roughly in the domain of attraction.

Part d



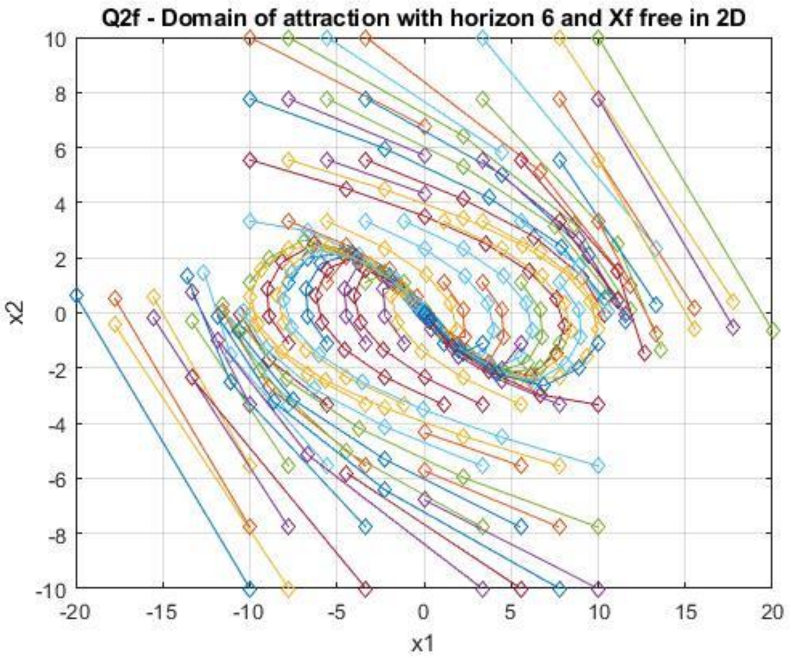
Domain of attraction: Slightly wider range of points close to origin that **can reach the origin in exactly 6 time steps**. The domain of attraction is this spiral region or polygon roughly bounded by $x_2 = 4$, $x_2 = -4$, $x_2 = -x_1 - 2$, and $x_2 = -x_1 + 2$.

Part e



Domain of attraction: This is a much larger range of points close to origin in a spiral of radius approximately 5.

Part f



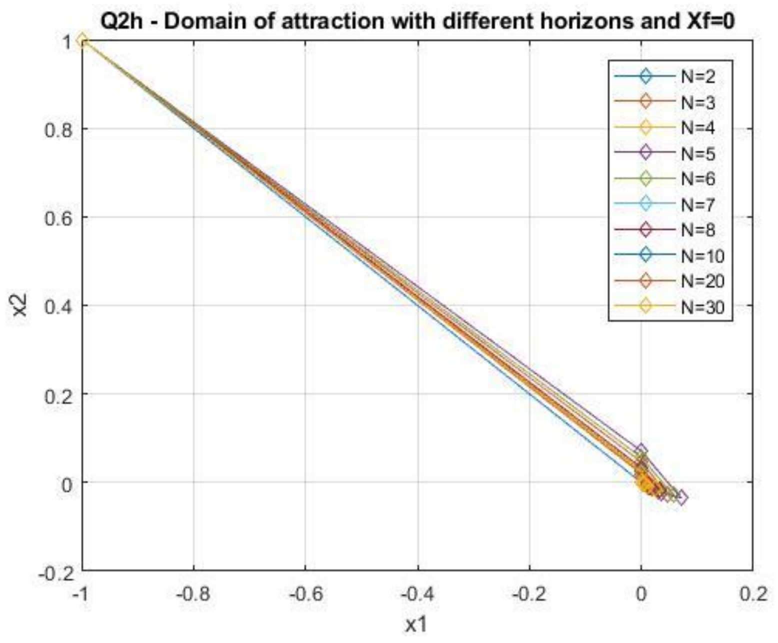
Domain of attraction: This is approximately the same range of points in a spiral of radius approximately 5.

Part g

Forcing the terminal state to be exactly the origin imposes a much stricter constraint on the MPC problem and results in fewer starting points that will converge to a solution. If we relax the terminal state to be all of the $\mathbb{R}^2$ space, the domain of attraction gets bigger because the points no longer have to be exactly at the origin in only 2 or 6 timesteps. If the terminal state has to be exactly 0, changing the MPC horizon from 2 to 6 also increases the domain of attraction.

Part h

I used the initial point [-1,1] because I knew that it could easily converge to the origin (which is the terminal set) in as few as 2 time steps.



```
 N       Cost 
___    ______

  2      2.01
  3    2.0148
  4    2.0115
  5     2.022
  6    2.0175
  7    2.0123
  8    2.0123
 10    2.0111
 20    2.0107
 30     2.011
```

The choice of N keeps both the trajectory and the costs approximately the same.

Contents

```
% HW 7 Question 2
clc
% clear all
close all
dbstop if error
% x(t+1) = Ax + Bu
A = [1 1;
    0 1];
B = [0;1];
Q = [1 0;
    0 1];
R = 0.01;
P = [1 0;
    0 1];
% cost = xN'PxN + x'Qx + u'Ru
ubar = 1;
xbar = 1;
umin = -ubar;
umax = ubar;
xmin = [-xbar; -xbar];
xmax = [xbar; xbar];
% Pinf = solution to Ricatti
Pinf = solveDiscreteRiccati(A, B, Q, R);
```

Part b

xbar = 5; ubar = 0.5; umin = -ubar; umax = ubar; xmin = [-xbar; -xbar]; xmax = [xbar; xbar]; N = 3; P = [1 0; 0 1]; R = 10; x0 = [-4.5; 2]; Xf = []; % Represents no constraint on Xf [Xallmpc, Uallmpc, horvalmpc, ~] = solveMPC(N, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf); tvec = 0:50; figure plot(Xallmpc(1,:),Xallmpc(2,:),'.-b') hold on x0 = [-4.5; 3]; [Xallmpc, Uallmpc, horvalmpc, lastIter] = solveMPC(N, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf); figure(1) plot(Xallmpc(1,1:lastIter),Xallmpc(2,1:lastIter),'-dr') title('Closed loop trajectories under receding horizon control MPC') legend('Initial x=[-4.5,2]','Initial x=[-4.5,3]') xlabel('x1') ylabel('x2') grid on

Part c

xbar = 10; ubar = 1; umin = -ubar; umax = ubar; xmin = [-xbar; -xbar]; xmax = [xbar; xbar]; N = 2; R = 0.01; P = Pinf; Xf = 0; numPoints = 10; % TODO change to 10 (or more) figure for x1=linspace(-xbar,xbar,numPoints) for x2=linspace(-xbar,xbar,numPoints) x0 = [x1;x2]; fprintf('Initial point (%.2f, %.2f)\n',x1,x2); solvehorizon=50; % TODO change to 50 [Xallmpc, Uallmpc, horvalmpc, lastIter] = solveMPC(N, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf, solvehorizon); if lastIter>1 plot(Xallmpc(1,1:lastIter),Xallmpc(2,1:lastIter),'-d')% hold on end end end title('Q2c - Domain of attraction with horizon 2 and Xf = 0') xlabel('x1') ylabel('x2') grid on

Part d

xbar = 10; ubar = 1; umin = -ubar; umax = ubar; xmin = [-xbar; -xbar]; xmax = [xbar; xbar]; N = 6; R = 0.01; P = Pinf; Xf = 0; numPoints = 10; % TODO change to 10 (or more) figure for x1=linspace(-xbar,xbar,numPoints) for x2=linspace(-xbar,xbar,numPoints) x0 = [x1;x2]; fprintf('Initial point (%.2f, %.2f)\n',x1,x2); solvehorizon=50; % TODO change to 50 [Xallmpc, Uallmpc, horvalmpc, lastIter] = solveMPC(N, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf, solvehorizon); if lastIter>1 plot(Xallmpc(1,1:lastIter),Xallmpc(2,1:lastIter),'-d') hold on end end end title('Q2d - Domain of attraction with horizon 6 and Xf = 0') xlabel('x1') ylabel('x2') grid on

Part e

```
xbar = 10;
ubar = 1;
umin = -ubar;
umax = ubar;
xmin = [-xbar; -xbar];
xmax = [xbar; xbar];
N = 2;
```

```
R = 0.01;
P = Pinf;
Xf = [];
numPoints = 10; % TODO change to 10 (or more)
figure
for x1=linspace(-xbar,xbar,numPoints)
    for x2=linspace(-xbar,xbar,numPoints)
        x0 = [x1;x2];
        fprintf('Initial point (%.2f, %.2f)\n',x1,x2);
        solvehorizon=50; % TODO change to 50
        [Xallmpc, Uallmpc, horvalmpc, lastIter] = solveMPC(N, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf, solvehorizon);
        if lastIter>1
            plot(Xallmpc(1,1:lastIter),Xallmpc(2,1:lastIter),'-d')
            hold on
        end
    end
end
title('Q2e - Domain of attraction with horizon 2 and and Xf free in 2D')
xlabel('x1')
ylabel('x2')
grid on
```

Part f

```
xbar = 10;
ubar = 1;
umin = -ubar;
umax = ubar;
xmin = [-xbar; -xbar];
xmax = [xbar; xbar];
N = 6;
R = 0.01;
P = Pinf;
Xf = [];
numPoints = 10; % TODO change to 10 (or more)
figure
for x1=linspace(-xbar,xbar,numPoints)
    for x2=linspace(-xbar,xbar,numPoints)
        x0 = [x1;x2];
        fprintf('Initial point (%.2f, %.2f)\n',x1,x2);
        solvehorizon=50; % TODO change to 50
        [Xallmpc, Uallmpc, horvalmpc, lastIter] = solveMPC(N, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf, solvehorizon);
        if lastIter>1
            plot(Xallmpc(1,1:lastIter),Xallmpc(2,1:lastIter),'-d')
            hold on
        end
    end
end
title('Q2f - Domain of attraction with horizon 6 and Xf free in 2D')
xlabel('x1')
ylabel('x2')
grid on
```

Part h

```
xbar = 10;
ubar = 1;
umin = -ubar;
umax = ubar;
xmin = [-xbar; -xbar];
xmax = [xbar; xbar];
R = 0.01;
P = Pinf;
Xf = 0;
numPoints = 5; % TODO change to 10 (or more)
x0 = [-1;1]; % TODO change required?
% N_vals = [2;4;8];
N_vals = [2; 3; 4; 5; 6; 7; 8; 10; 20; 30];
mpccosts = zeros(length(N_vals),1);
```

```
figure
for i = 1: length(N_vals)
    N = N_vals(i);
    fprintf('N=%d\n',N);
    solvehorizon=50; % TODO change to 50
    [Xallmpc, Uallmpc, horvalmpc, lastIter] = solveMPC(N, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf, solvehorizon);
    mpccosts(i) = horvalmpc;
    lblstr = sprintf('N=%d',N);
    if lastIter>1
        plot(Xallmpc(1,1:lastIter),Xallmpc(2,1:lastIter),'-d','DisplayName',lblstr)% TODO this produces different colors right?
%             plot(Xallmpc(1,1:lastIter),Xallmpc(2,1:lastIter),'-d','color',rand(1,3),'markersize',10,'DisplayName',lblstr)
        hold on
    end
end
title('Q2h - Domain of attraction with different horizons and Xf=0')
xlabel('x1')
ylabel('x2')
legend show
grid on
T = array2table([N_vals, mpccosts],...
    'VariableNames',{'N','Cost'})
```

Published with MATLAB® R2017b

```
% Check Riccati solver
function Pinf = solveDiscreteRiccati(A, B, Q, R)
if nargin == 0
    A = [1 1;
    0 1];
    B = [0;1];
    Q = [1 0;
        0 1];
    R = 0.01;
end
% Own function
Pinf = [1 1; 1 1];
for i = 1:10
    Pinfnew = Q + A'*Pinf*A - A'*Pinf*B*inv(B'*Pinf*B + R)*B'*Pinf*A;
    Pinf = Pinfnew;
end

% MATLABfunctions
% [X,K,L] = idare(A,B,Q,R,S,E)
[X,L,G] = dare(A,B,Q,R,[],[]);
[K,S,E] = dlqr(A, B, Q, R);

if ~(norm(X-S)<1e-3 && norm(X-Pinf)<1e-3)
    fprintf('Not close to equal.\n')
end
end
```

Published with MATLAB® R2017b

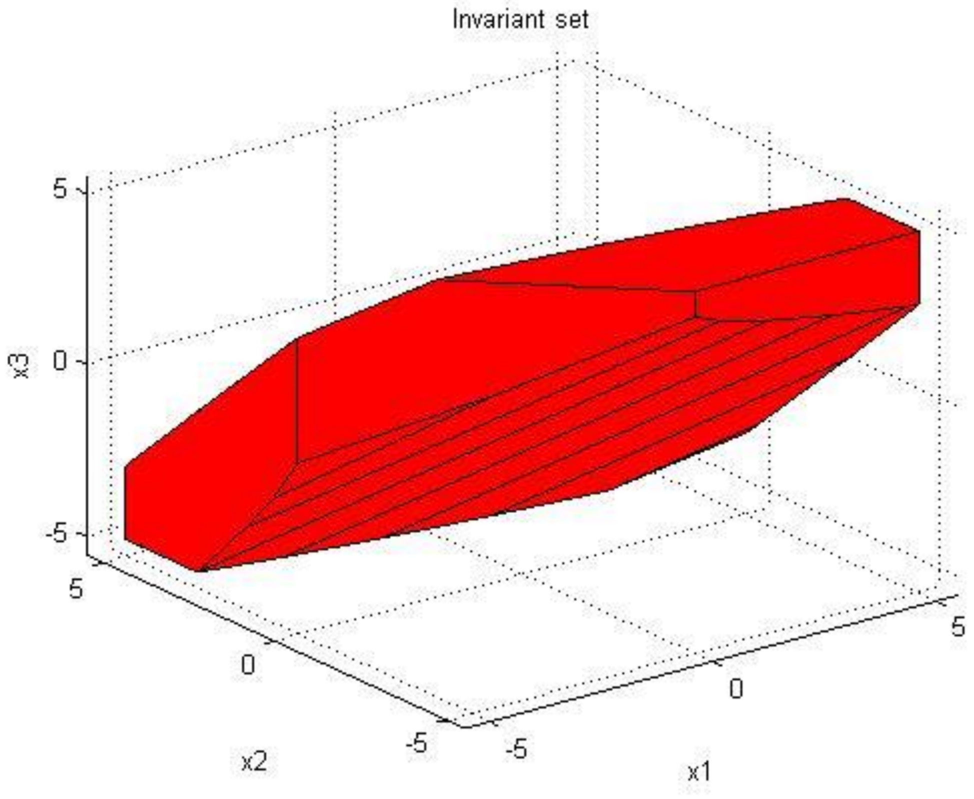
```
function [Xallmpc, Uallmpc, horvalmpc, i] = solveMPC(horizon, x0, A, B, P, Q, R, xmax, xmin, umax, umin, Xf, solvehorizon)
optvalmpc = 0;
if nargin < 13
    N = 50; % solving horizon for solution
else
    N = solvehorizon;
end
%store solutions
n = size(B,1);
m = size(B,2);
Xallmpc = zeros(n,N+1);
Uallmpc = zeros(m,N);
T = horizon; % mpc horizon passed in
x = x0; %reset initial state
Xallmpc(:,1) = x;

Qhalf = sqrtm(Q); Rhalf = sqrtm(R);

fprintf('MPC horizon=%d ; timestep = ',T);

%step through time
for i = 1:N
    fprintf('%d, ',i-1);

    %cvx precision
    cvx_precision(min(max(min(abs(x))/10,1e-6),0.99999))
    if Xf == 0
        cvx_begin quiet
            variables X(n,T+1) U(m,T)
            max(X') <= xmax'; max(U') <= umax';
            min(X') >= xmin'; min(U') >= umin';
            X(:,2:T+1) == A*X(:,1:T)+B*U;
            X(:,1) == x; %initial state constraint
            X(:,T+1) == 0; %terminal state constraint
            minimize (X(:,T+1)'*P*X(:,T+1)+pow_pos(norm([Qhalf*X(:,1:T);Rhalf*U],2),2))
        cvx_end
    else
        cvx_begin quiet
            variables X(n,T+1) U(m,T)
            max(X') <= xmax'; max(U') <= umax';
            min(X') >= xmin'; min(U') >= umin';
            X(:,2:T+1) == A*X(:,1:T)+B*U;
            X(:,1) == x; %initial state constraint
            minimize (X(:,T+1)'*P*X(:,T+1)+pow_pos(norm([Qhalf*X(:,1:T);Rhalf*U],2),2))
        cvx_end
    end

    %check feasibility
    if strcmp(cvx_status,'Solved')

        %store control
        u= U(:,1);
        Uallmpc(:,i) = u;

        %accumulate cost
        optvalmpc = optvalmpc + x'*Q*x + u'*R*u;

        %forward propagate state
        x = A*x+B*u;

        %record state
        Xallmpc(:,i+1) = x;

    else
       fprintf('ERROR')
       % break from loop
       optvalmpc = Inf;
```

```
        break;
    end
end
fprintf('\n');


horvalmpc = 0;
for k = 1:(horizon-1)
    xk = Xallmpc(:,k);
    uk = Uallmpc(:,k);
    horvalmpc = horvalmpc + xk'*Q*xk + uk'*R*uk;
end
x_last = Xallmpc(:,horizon);
horvalmpc = horvalmpc + x_last'*P*x_last;

end
```

Published with MATLAB® R2017b

Problem 3

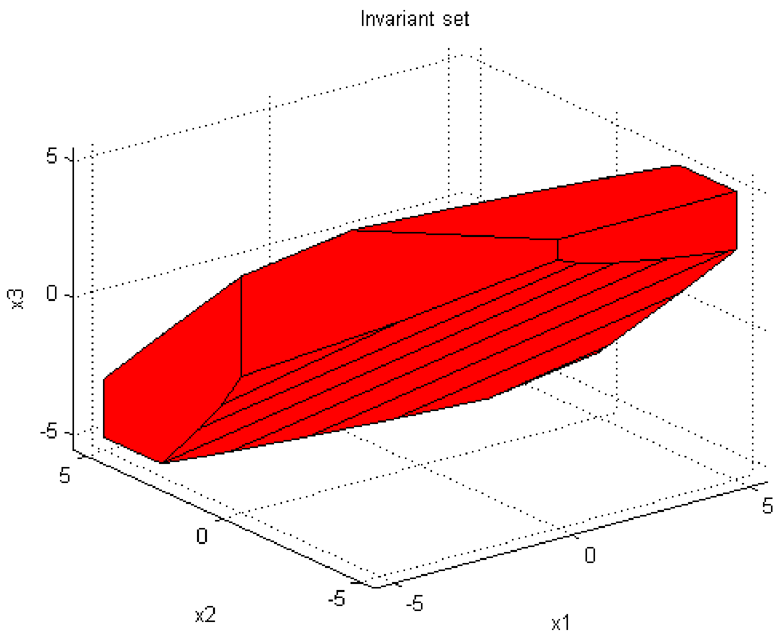
This is the invariant set for the problem. See code attached.



HW 7 Question 3

```
clear all
close all
clc
A = [1 1 0;
    0 0.9 1;
    0 0.2 0];
B = [0; 1; 0];
% computes a control invariant set for LTI system x^+ = A*x+B*u
system = LTISystem('A', A, 'B', B);
system.x.min = [-5; -5; -5];
system.x.max = [5; 5; 5];
system.u.min = -0.5;
system.u.max = 0.5;
InvSet = system.invariantSet()
InvSet.plot()
xlabel('x1')
ylabel('x2')
zlabel('x3')
title('Invariant set')
```

```
Iteration 1...
Iteration 2...
Iteration 3...
Iteration 4...
Iteration 5...
Iteration 6...
Iteration 7...
Iteration 8...
Iteration 9...
Iteration 10...
Polyhedron in R^3 with representations:
    H-rep (irredundant) : Inequalities  24 | Equalities   0
    V-rep               : Unknown (call computeVRep() to compute)
Functions : none
```



Published with MATLAB® R2014a

Problem 4

Question 1

Here the eigenvalues of A are {0.99, 0.99} so A is asymptotically stable.

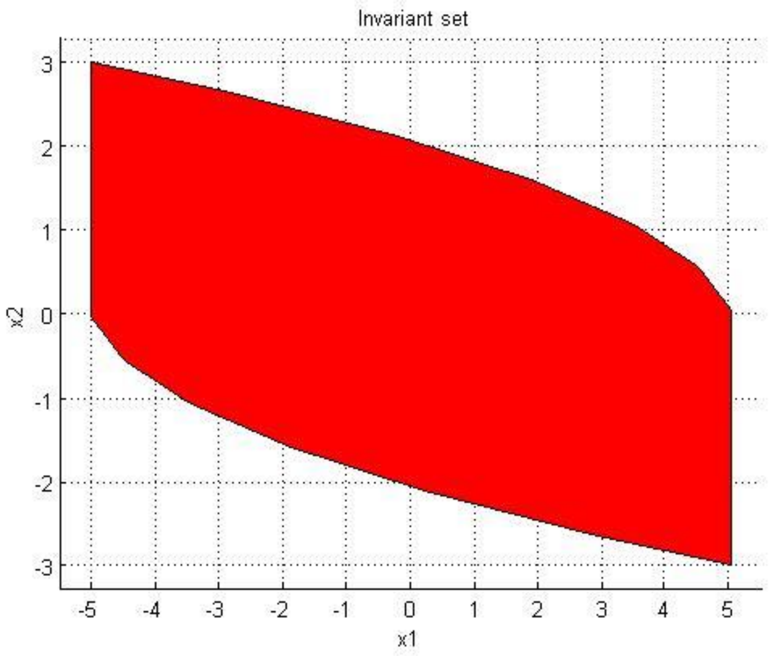
We can choose X_f to be the maximally positive invariant set O_∞ for the system $x(t+1) = Ax(t)$ which is also a control invariant set for $x(t+1) = Ax(t) + Bu(t)$ because $u(t)$ can be 0. Using MPT, we can simply calculate the maximal control invariant set for $x(t+1) = Ax(t) + Bu(t)$ directly.

We choose P such that $-P + Q + A'PA = 0$, i.e. solution to the Lyapunov equation. This is solved in MATLAB using P = `dlyap(A',Q)`.

We get that

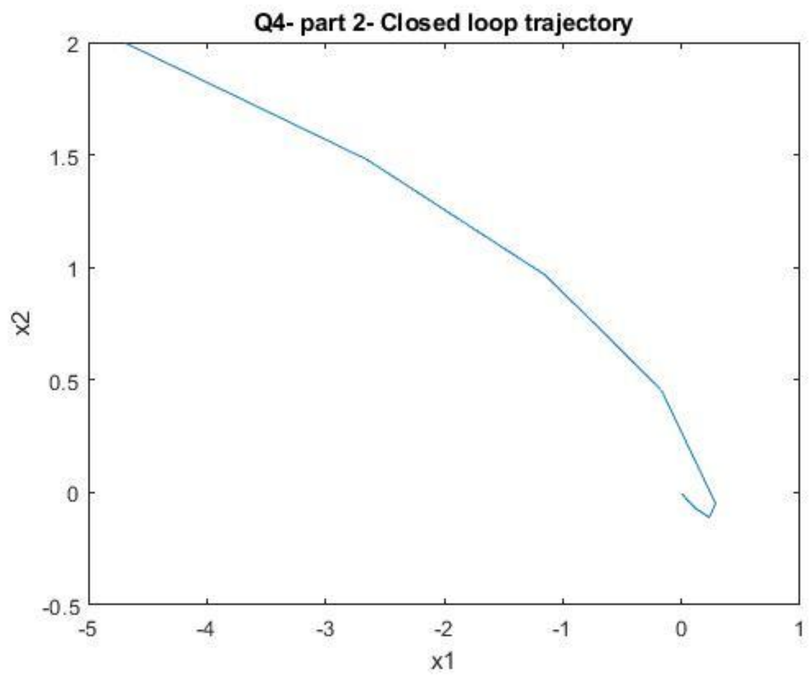
```
P =

1.0e+05 *

   0.0005    0.0250

   0.0250    2.5131
```

The invariant set X_f is plotted below.



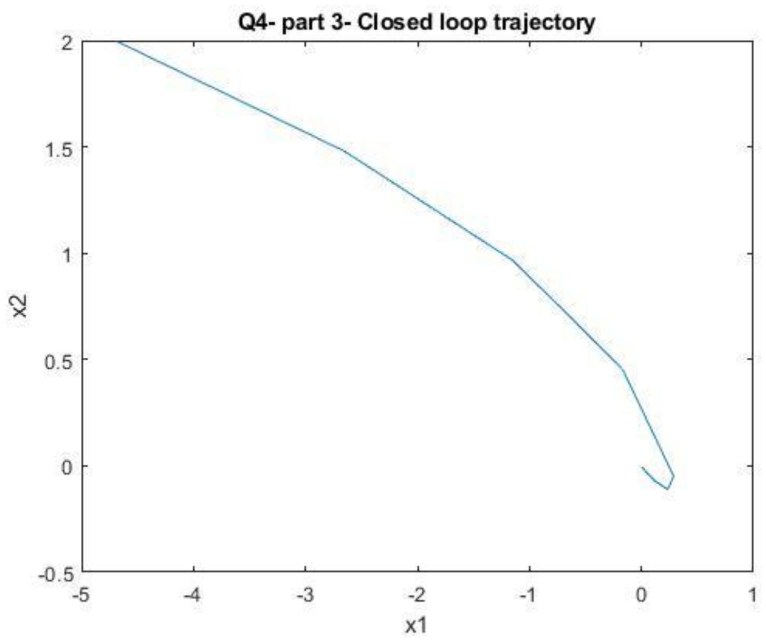
Question 2

This is for the online controller.



Question 3

This is for the explicit controller.



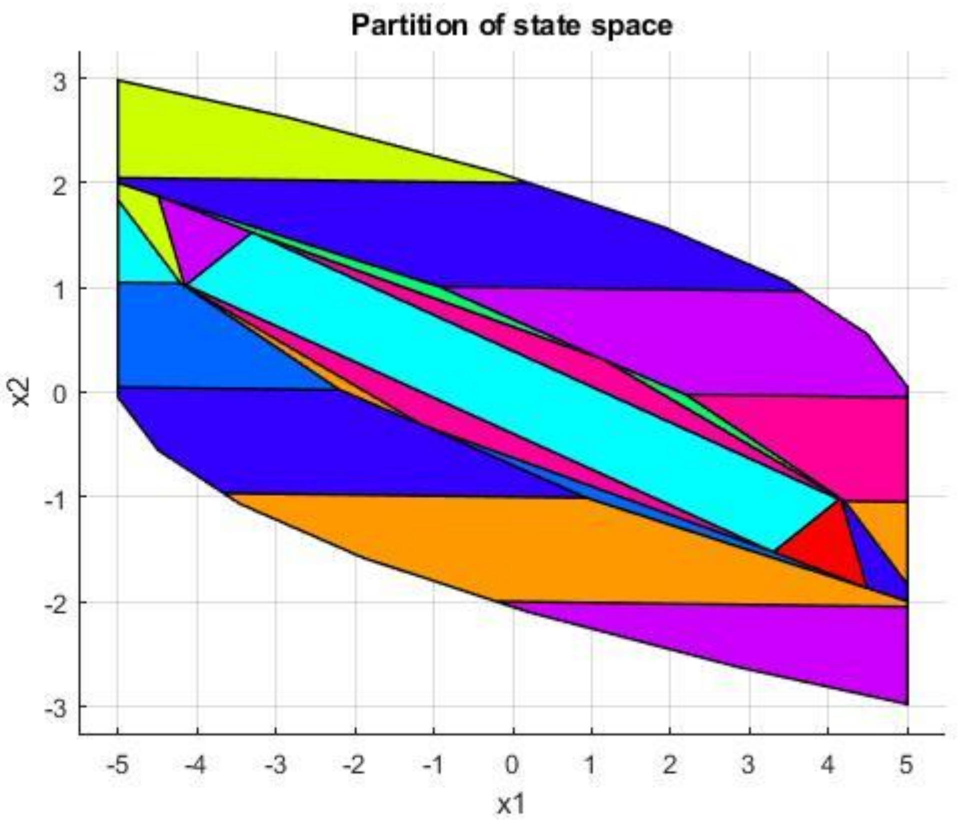
Question 4

```
      Controller            ExecutionTime
 _______________________    ______________

 'Online controller'          2.6152
 'Explicit controller'      0.85288
```

Note that this table includes the time to set up the LTI system, create the MPC controller, and time required to compute closed-loop trajectory. The explicit controller is clearly much faster at computing the closed-loop trajectory, as we would expect.

Question 5

This partition shows the regions of the state space where the optimal control law is affine.



Contents

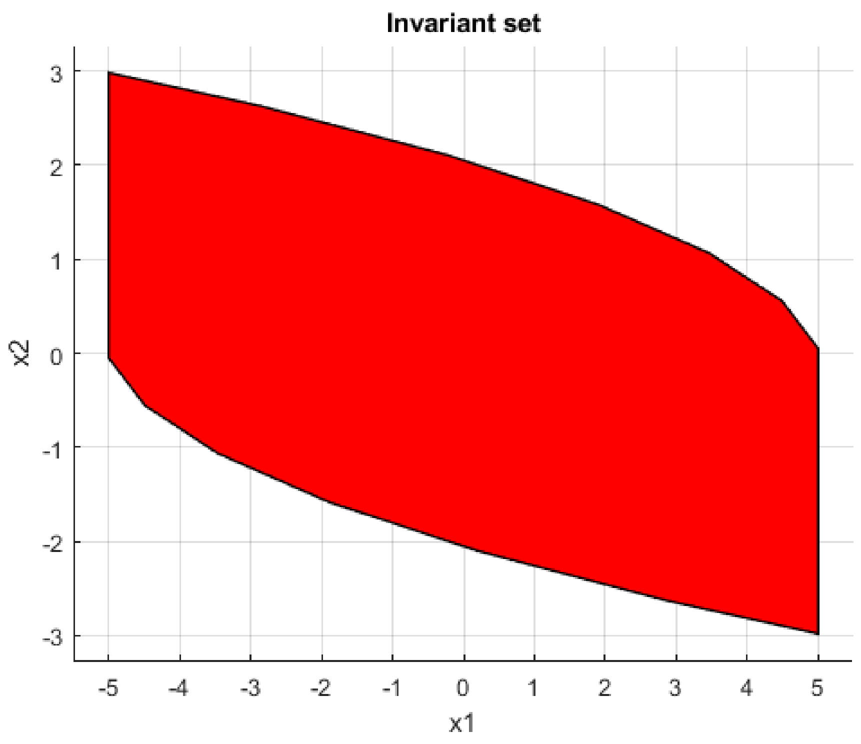
HW 7 Question 4

```
clc
clear all
close all
dbstop if error
```

Question 1

```
A = [0.99 1;
    0 0.99];
B = [0;1];

% computes a control invariant set for LTI system x^+ = A*x+B*u
system = LTISystem('A', A, 'B', B);
system.x.min = [-5; -5];
system.x.max = [5; 5];
system.u.min = -0.5;
system.u.max = 0.5;
Xf = system.invariantSet();
Xf.plot()
xlabel('x1')
ylabel('x2')
title('Invariant set')

% Compute a terminal cost from the Lyapunov equation
Q =[1 0;
    0 1];
R =1;
PN = dlyap(A',Q);
```

```
Iteration 1...
Iteration 2...
Iteration 3...
Iteration 4...
Iteration 5...
Iteration 6...
Iteration 7...
```

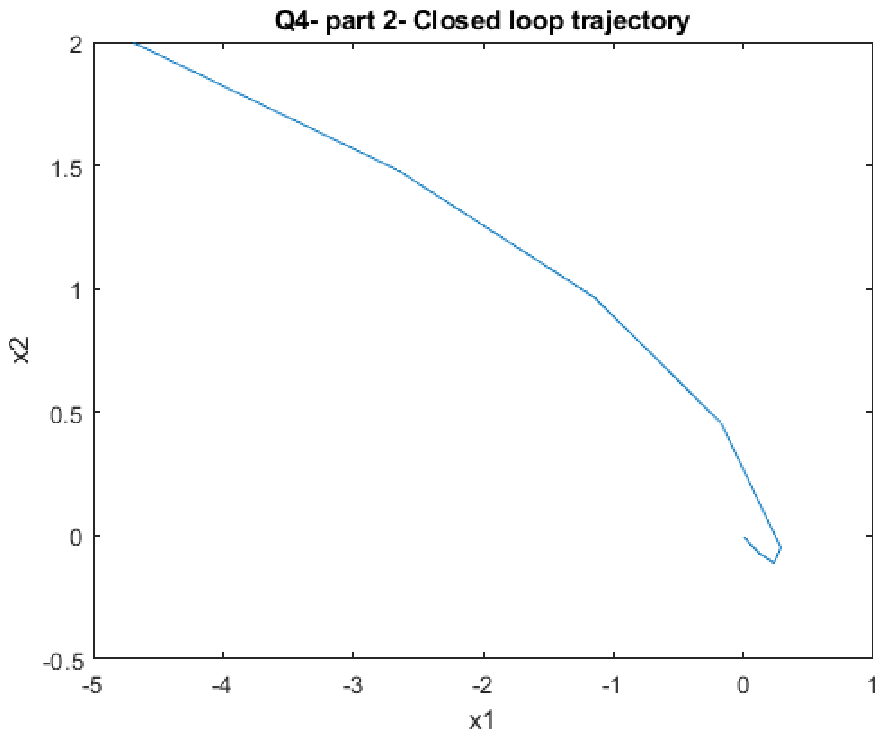


Question 2

Create an online MPC controller

```
clear system
tic
system = LTISystem('A', A, 'B', B);
system.x.min = [-5; -5];
system.x.max = [5; 5];
system.u.min = -0.5;
system.u.max = 0.5;
Xf = system.invariantSet();
system.x.penalty = QuadFunction(Q);
system.u.penalty = QuadFunction(R);

system.x.with('terminalSet');
system.x.terminalSet = Xf;
system.x.with('terminalPenalty');
system.x.terminalPenalty = QuadFunction(PN);

x0 = [-4.7;2];
N = 4;
mpc = MPCController(system, N);

[u, feasible, openloop] = mpc.evaluate(x0)
loop = ClosedLoop(mpc, system);
data = loop.simulate(x0, 30);
onlineTime = toc;

% Plot trajectory
figure
plot(data.X(1,:),data.X(2,:))
title('Q4- part 2- Closed loop trajectory')
xlabel('x1')
ylabel('x2')
```

```
Iteration 1...
Iteration 2...
Iteration 3...
Iteration 4...
Iteration 5...
Iteration 6...
Iteration 7...

u =

   -0.5000


feasible =

  logical

   1


openloop =

  struct with fields:

    cost: 40.8544
       U: [-0.5000 -0.5000 -0.5000 -0.4539]
       X: [2×5 double]
       Y: [0×4 double]
```

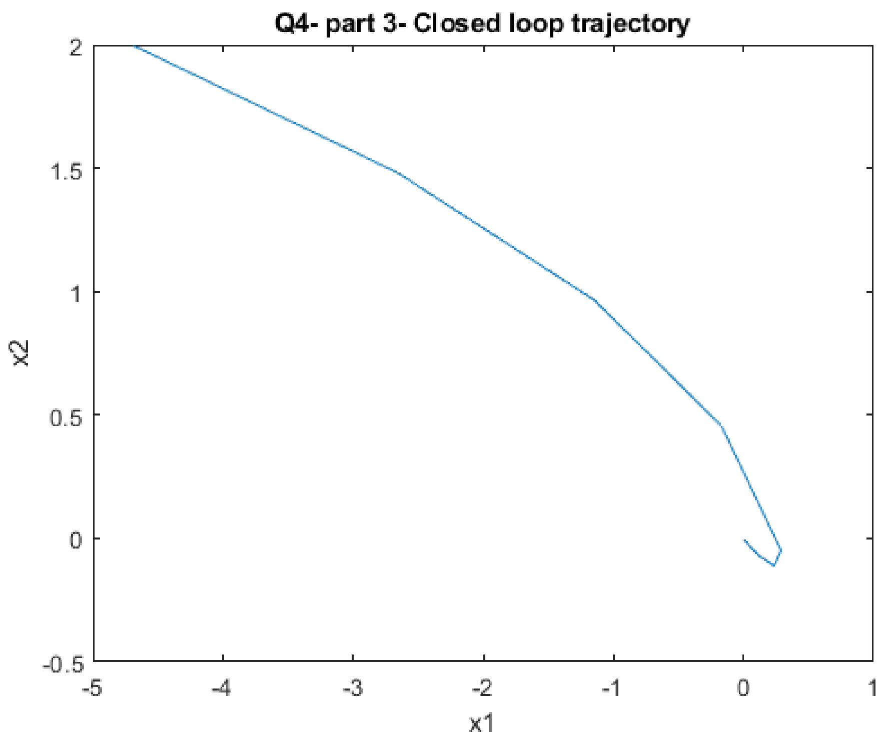


Question 3

```
clear system
clear mpc
tic
system = LTISystem('A', A, 'B', B);
system.x.min = [-5; -5];
system.x.max = [5; 5];
system.u.min = -0.5;
system.u.max = 0.5;
Xf = system.invariantSet();
```

```
system.x.penalty = QuadFunction(Q);
system.u.penalty = QuadFunction(R);
system.x.with('terminalSet');
system.x.terminalSet = Xf;
system.x.with('terminalPenalty');
system.x.terminalPenalty = QuadFunction(PN);
x0 = [-4.7;2];
N = 4;
mpc = MPCController(system, N);
expmpc = mpc.toExplicit();
[u, feasible, openloop] = expmpc.evaluate(x0)
loop = ClosedLoop(expmpc, system);
data2 = loop.simulate(x0, 30);
explicitTime = toc;

% Plot trajectory
figure
plot(data2.X(1,:),data2.X(2,:))
title('Q4- part 3- Closed loop trajectory')
xlabel('x1')
ylabel('x2')
```

```
Iteration 1...
Iteration 2...
Iteration 3...
Iteration 4...
Iteration 5...
Iteration 6...
Iteration 7...
mpt_plcp: 33 regions

u =

   -0.5000


feasible =

  logical

   1


openloop =

  struct with fields:

         cost: 40.8544
            U: [-0.5000 -0.5000 -0.5000 -0.4539]
            X: [2×5 double]
            Y: [0×4 double]
    partition: 1
       region: 8
```



Question 4

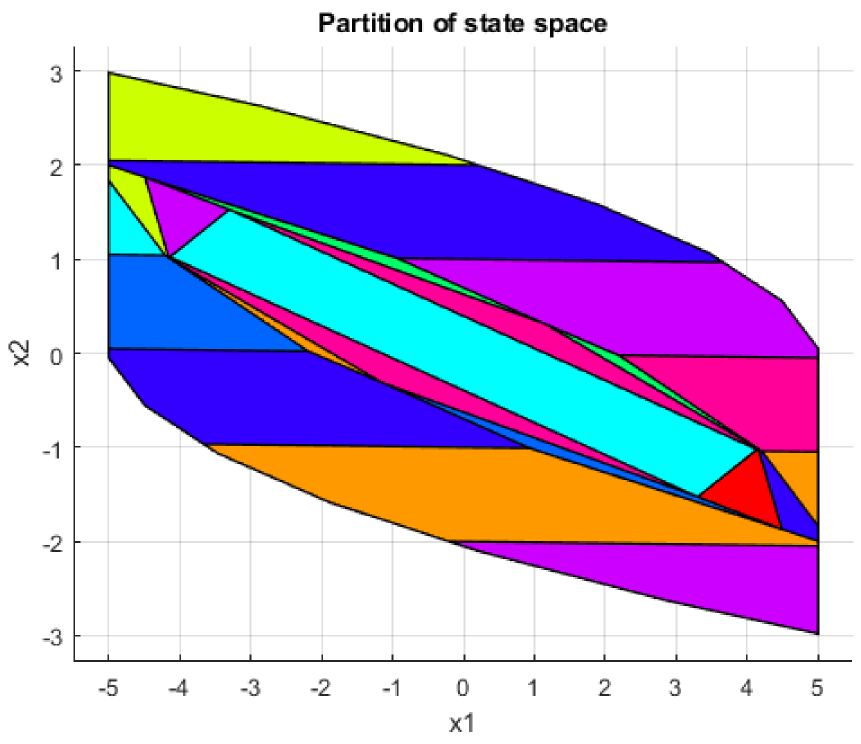
Table the two execution times

```
names = {'Online controller';'Explicit controller'};
times = {onlineTime; explicitTime};
T = cell2table([names, times],'VariableNames',{'Controller','ExecutionTime'})
```

```
T =

  2×2 table

         Controller          ExecutionTime
    _____________________    _____________

    'Online controller'         2.6131    
    'Explicit controller'      0.93478    
```

Question 5

```
figure
expmpc.partition.plot()
title('Partition of state space')
xlabel('x1')
ylabel('x2')
```


Partition of state space
x2
x1
3
2
1
0
-1
-2
-3
-5
-4
-3
-2
-1
0
1
2
3
4
5