Table of Contents

MPC HW 2 - Problem 2

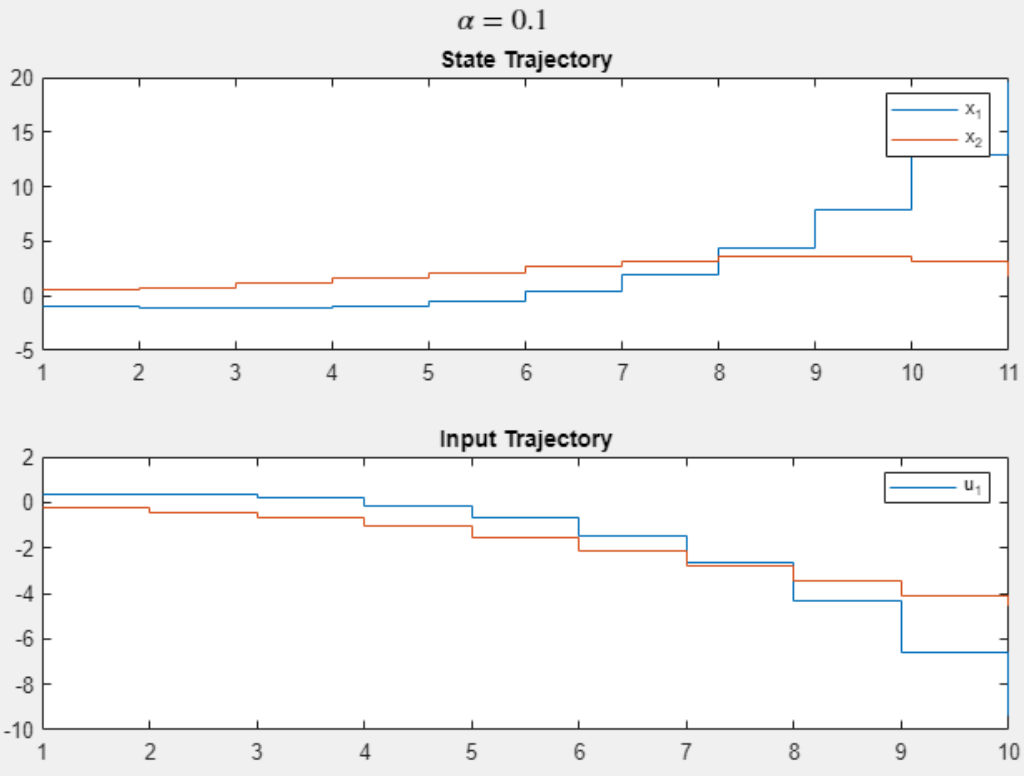
Author: Jonas Wagner Date: 2023-09-29

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

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

% State/input constraints
cons_x = @(x) x(1) <= 5;
cons_u = @(u) norm(u,'inf') <= 1;

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

% MPC Parameters
N = 3;
R = eye(nu);
P = 0;
```

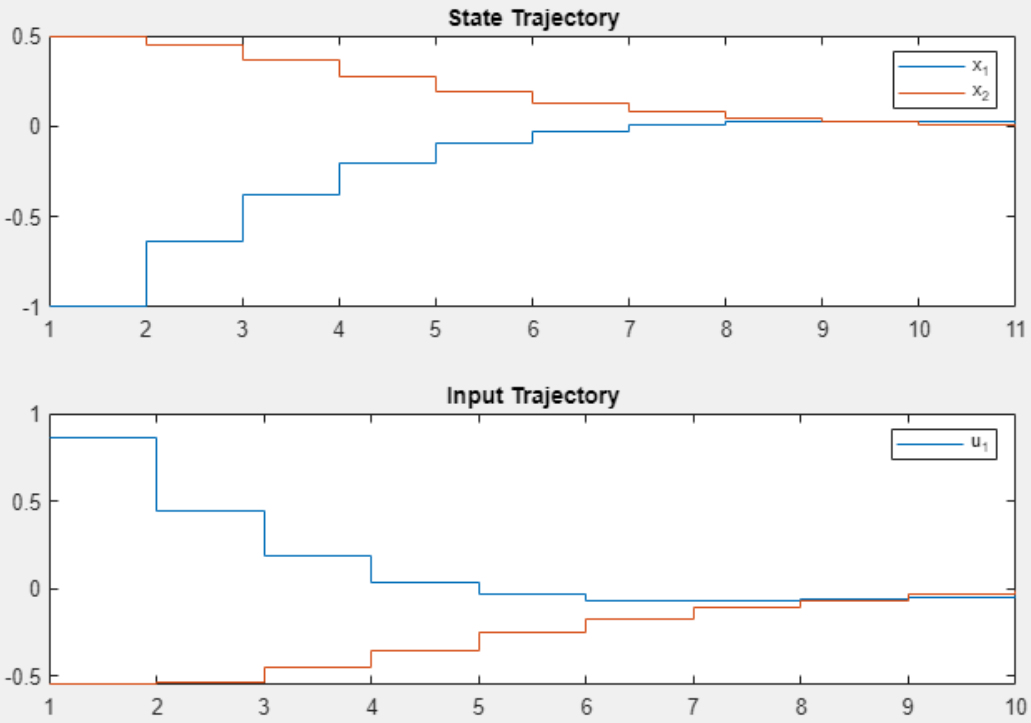
Part 2a

```
x0 = [-1; 0.5]; tf = 10;

Alpha = [0.1,0.5, 1, 2, 10];
for alpha = Alpha
    Q = alpha*eye(nx);
```

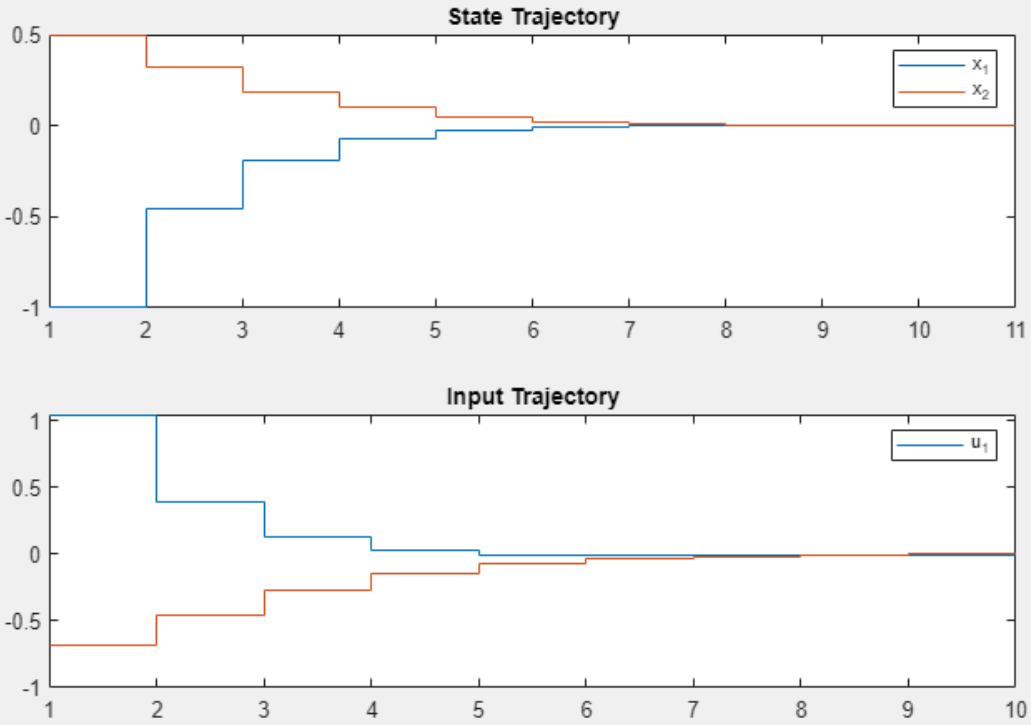
```
    % Controller setup
    cons = @(x,u,s) [];
    cons_f = @(x,u,s) [];
    controller = mpc_yalmip_controller(A, B, P, Q, R, N, cons, cons_f);

    % Simulation
    [X, U, ~] = run_sim(A,B,controller, x0, tf);

    % Results
    figName = sprintf('pblm2a_alpha=%.0e',alpha);
    fig = plot_trajectory(X, U);
    sgtitle(strcat('$\alpha = ',num2str(alpha),'$'),'Interpreter','latex');
    saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')
end
```



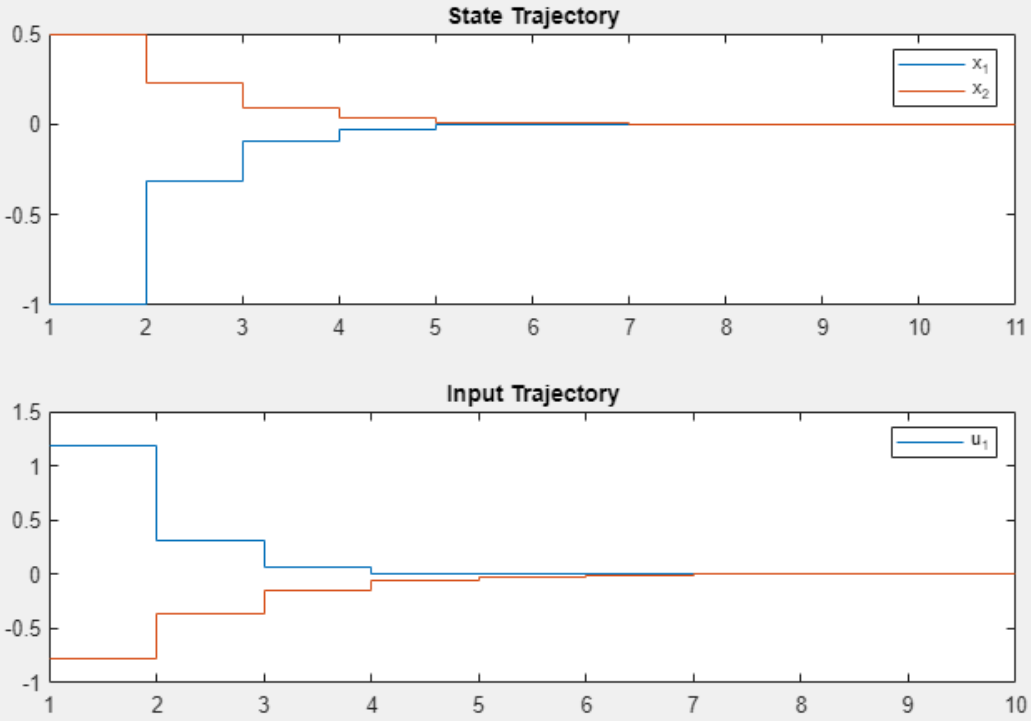
$\alpha = 0.5$



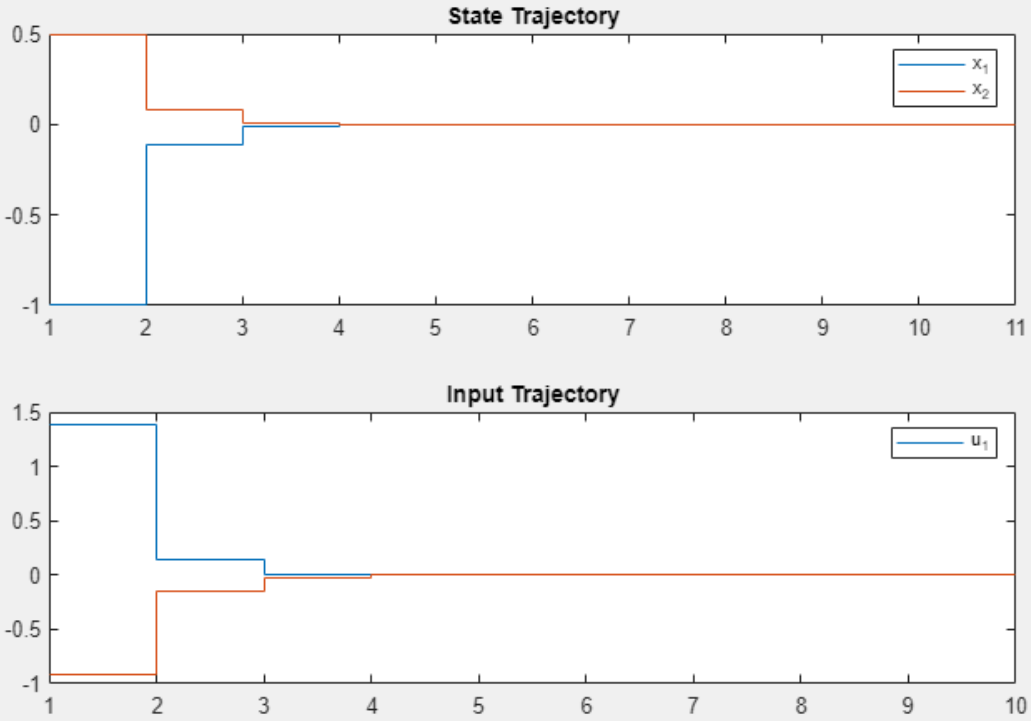
$\alpha = 1$



$\alpha = 2$



$\alpha = 10$

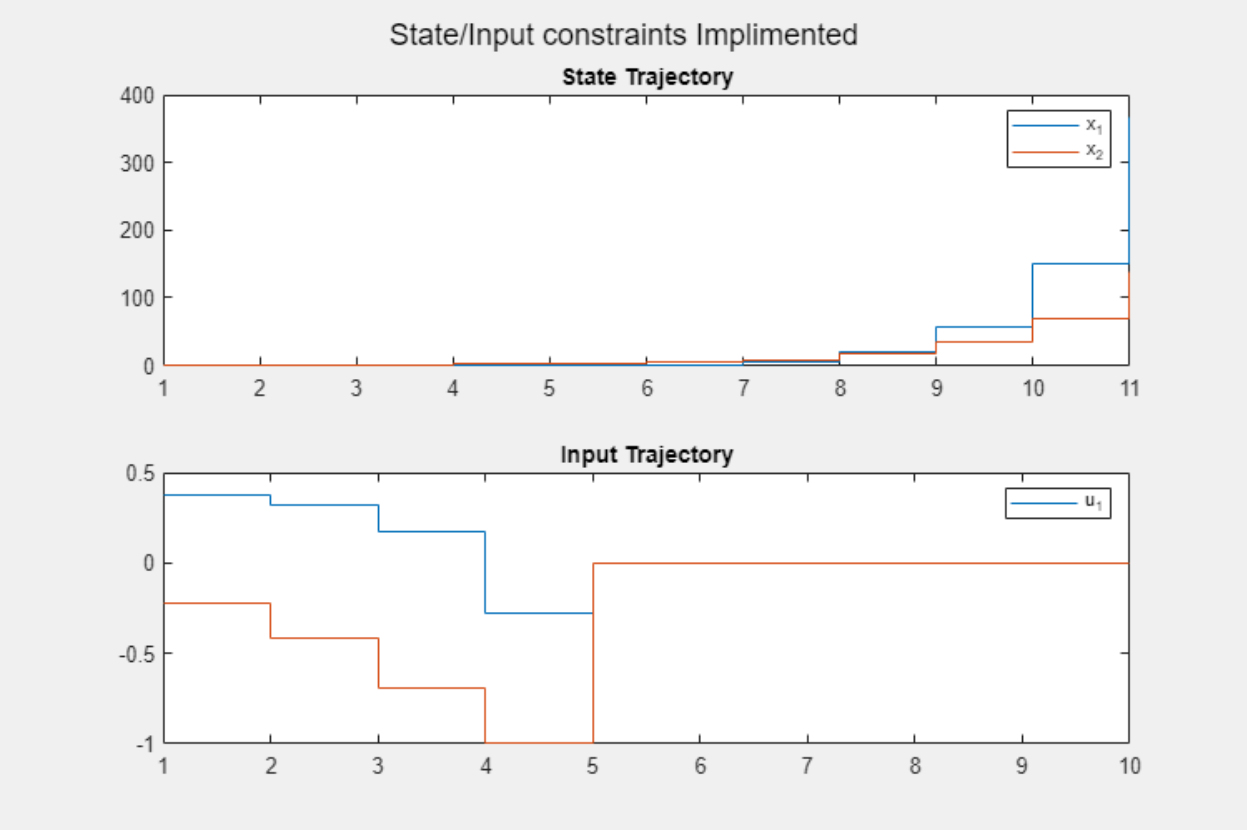


Part 2b

Setup and sim

```
alpha = 0.1; Q = alpha*eye(nx);
cons = @(x,u,s) [cons_x(x), cons_u(u)];
controller = mpc_yalmip_controller(A, B, P, Q, R, N, cons, cons_f);
[X, U, diagnostics_] = run_sim(A,B,controller, x0, tf);

% Results
figName = sprintf('pblm2b');
fig = plot_trajectory(X, U);
sgtitle('State/Input constraints Implimented')
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')

% U and Errors
for k = 1:tf
    fprintf('U_%d solution: %f\n',k,U(:,k))
    fprintf('Error (k=%d): %s\n',k,yalmiperror(diagnostics_{k}));
end
```

```
U_1 solution: 0.378958
U_-2.256091e-01 solution: Error (k=1): Successfully solved
U_2 solution: 0.324550
U_-4.187884e-01 solution: Error (k=2): Successfully solved
U_3 solution: 0.171374
U_-6.909988e-01 solution: Error (k=3): Successfully solved
U_4 solution: -0.277110
U_-1.000000e+00 solution: Error (k=4): Successfully solved
U_5 solution: 0.000000
U_0 solution: Error (k=5): Either infeasible or unbounded
U_6 solution: 0.000000
U_0 solution: Error (k=6): Either infeasible or unbounded
U_7 solution: 0.000000
U_0 solution: Error (k=7): Either infeasible or unbounded
U_8 solution: 0.000000
U_0 solution: Error (k=8): Either infeasible or unbounded
U_9 solution: 0.000000
U_0 solution: Error (k=9): Either infeasible or unbounded
U_10 solution: 0.000000
U_0 solution: Error (k=10): Either infeasible or unbounded
```



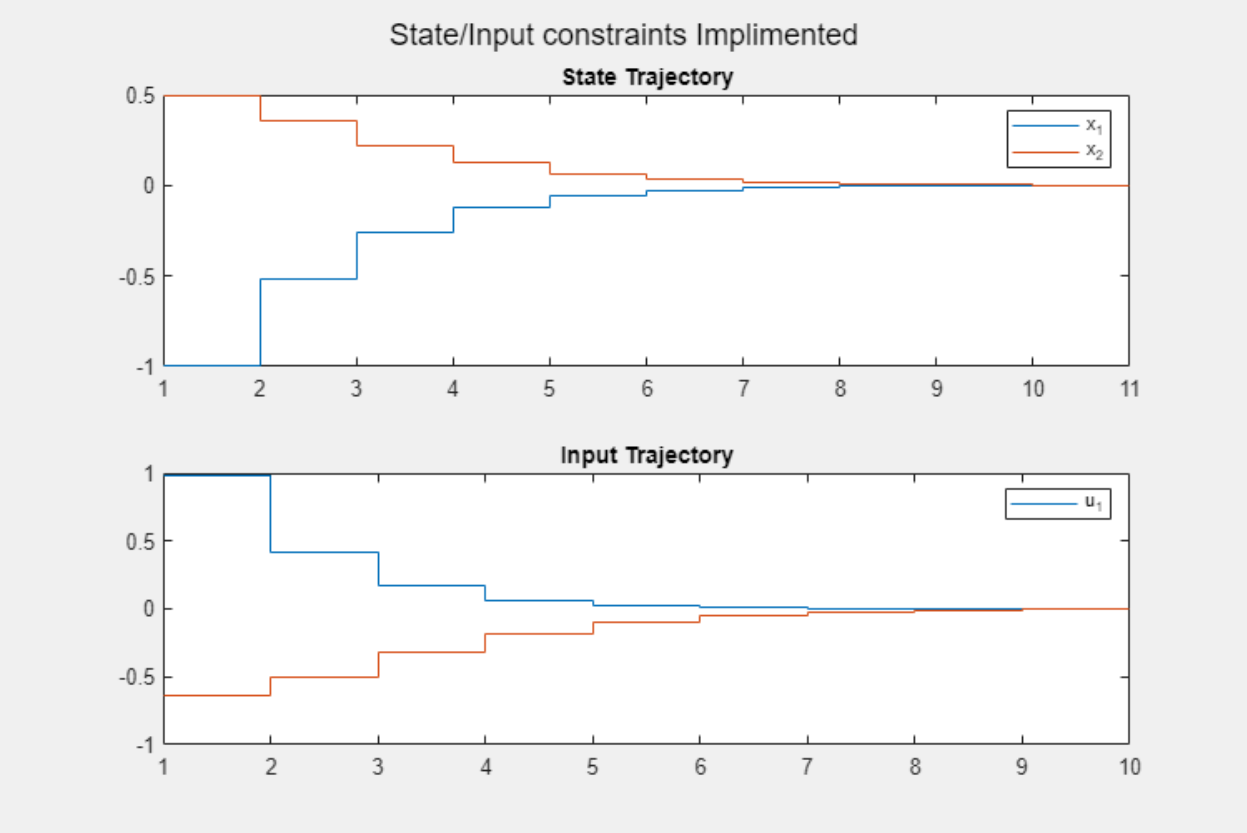
Part 2c

Setup and sim

```
alpha = 0.1; Q = alpha*eye(nx);
cons = @(x,u,s) [cons_x(x), cons_u(u)];
cons_f = @(x,u,s) x==0;
controller = mpc_yalmip_controller(A, B, P, Q, R, N, cons, cons_f);
[X, U, diagnostics_] = run_sim(A,B,controller, x0, tf);

% Results
figName = sprintf('pblm2c');
fig = plot_trajectory(X, U);
sgtitle('State/Input constraints Implimented')
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')

% U and Errors
for k = 1:tf
    fprintf('U_%d solution: %f\n',k,U(:,k))
    fprintf('Error (k=%d): %s\n',k,yalmiperror(diagnostics_{k}));
end
```

```
U_1 solution: 0.984663
U_-6.399073e-01 solution: Error (k=1): Successfully solved
U_2 solution: 0.414154
U_-4.999188e-01 solution: Error (k=2): Successfully solved
```

```
U_3 solution: 0.168661
U_-3.170114e-01 solution: Error (k=3): Successfully solved
U_4 solution: 0.065967
U_-1.815646e-01 solution: Error (k=4): Successfully solved
U_5 solution: 0.024420
U_-9.764496e-02 solution: Error (k=5): Successfully solved
U_6 solution: 0.008310
U_-5.022334e-02 solution: Error (k=6): Successfully solved
U_7 solution: 0.002421
U_-2.495200e-02 solution: Error (k=7): Successfully solved
U_8 solution: 0.000458
U_-1.204290e-02 solution: Error (k=8): Successfully solved
U_9 solution: -0.000089
U_-5.665221e-03 solution: Error (k=9): Successfully solved
U_10 solution: -0.000175
U_-2.601983e-03 solution: Error (k=10): Successfully solved
```



Part 2d

```
tic
% Batch approch
[S_x, S_u] = construct_Sx_Su(A,B,N);

% State/Input constraints
A_x = [1 0]; b_x = 5;
A_xf = [eye(nx);-eye(nx)]; b_xf = zeros(2*nx,1); %<--- origin
```

```
A_u = [eye(nu);-eye(nu)]; b_u = ones(2*nu,1);

for i = 1:N
    A_con_{i} = A_x;
    b_con_{i} = b_x;
end
A_con_{N+1} = A_xf;
b_con_{N+1} = b_xf;
for i = 1:N
    A_con_{i+N+1} = A_u;
    b_con_{i+N+1} = b_u;
end

A_con = blkdiag(A_con_{:}) * [S_x, S_u; zeros(N*nu,nx),eye(N*nu)];%
 zeros(size(S_u,2),size(S_x,2)), eye(size(S_u,2))];
b_con = vertcat(b_con_{:});

P = Polyhedron(A_con,b_con);
Q = P.projection(1:nx)
toc

Polyhedron in R^2 with representations:
    H-rep (redundant)   : Inequalities   9 | Equalities   0
    V-rep               : Unknown (call computeVRep() to compute)
Functions : none
Elapsed time is 0.385552 seconds.
```

Recursive Approch

```
tic
X_{N+1} = Polyhedron(A_xf,b_xf);
for k = N:-1:1
    P = Polyhedron( ...
        [X_{k+1}.A*A, X_{k+1}.A*B; zeros(size(A_u,1),nx),A_u],...
        [X_{k+1}.b; b_u]);
    X_{k} = P.projection(1:nx);
end

X_{1}
toc

% Ploting
figName = 'pblm2d';
fig = figure(WindowStyle="normal", Position=[0 0 750 500]);
X_{1}.plot
saveas(fig,[subfolder,filesep,'figs',filesep,figName],'png')

Polyhedron in R^2 with representations:
    H-rep (redundant)   : Inequalities   8 | Equalities   0
    V-rep               : Unknown (call computeVRep() to compute)
Functions : none
Elapsed time is 0.160652 seconds.
```

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}; %norm(Q*x_{k},2) + norm(R*u_{k},2);
        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,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};
    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

function [S_x, S_u] = construct_Sx_Su(A, B, N, augmentRow)
    arguments
        A
        B
        N
        augmentRow = true
    end
    nx = size(A,1);
    nu = size(B,2);
    % Construct S_x,and S_u
    for i = 1:N
        S_x_{i} = A^(i);

        for j = 1:N
            if j <= i
                S_u__{j} = A^(i-1+j-1)*B;
            else
                S_u__{j} = zeros(nx,nu);
            end
        end
        S_u_{i} = horzcat(S_u__{:});
    end
    S_x = vertcat(S_x_{:});
    S_u = vertcat(S_u_{:});

    % augment w/ extra row
    if augmentRow
```

```
        S_x = vertcat(eye(nx), S_x);
        S_u = vertcat(zeros(nx,N*nu), S_u);
    end
end
```

Published with MATLAB® R2023a