Table of Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%            Model Predictive Control - Exercise 4
%              EPFL - Spring semester 2017 -
%
%            Huber Lukas - Zgraggen Jannik
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear variables; close all; clear variables;

addpath(genpath('../tbxmanager'));
```

Definition System & LQR Control

```
close all;

A = [0.9752, 1.4544; -0.0327, 0.9315];
B = [0.0248; 0.0327];
x0=[3;0];

dimX = size(A,1);
dimU = size(B,2);

Q = 10*[1 0; 0 1];
R = [1];

f_nat = 0.15; % [r/s] Natural frquency
discRate = 1.5; % [r/s] Discretization rate applied

dx_i = 0.1; % Dampin Ratio

N = 10; % Horizon length

sys = LTISystem('A',A,'B',B);

% Define limits
sys.x.min = [-5, -0.2]';
sys.x.max = [5, 0.2];

sys.u.min = -1.75;
sys.u.max = 1.75;
```

```
sys.x.penalty = QuadFunction(Q);
sys.u.penalty = QuadFunction(R);

LQRGarin = sys.LQRGain;
LQRPenalty = sys.LQRPenalty.weight;
LQRSet = sys.LQRSet;
Ff=LQRSet.A;
ff=LQRSet.b;
Qf=LQRPenalty;
```

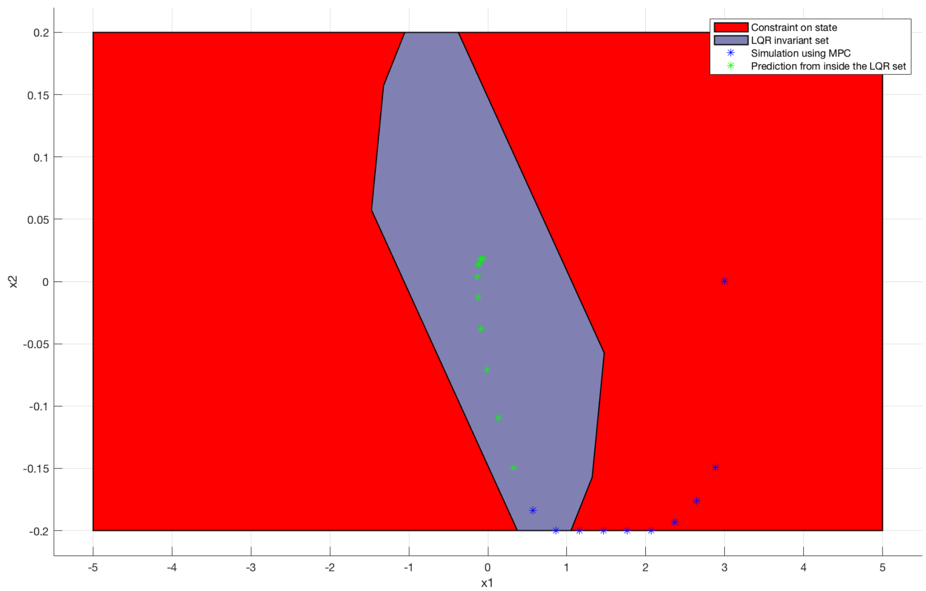
Iteration 1...
Iteration 2...
Iteration 3...

Exercise 1: Impelement MPC

```
close all;

% Optimization
H=blkdiag(kron(eye(N-1),Q),Qf,kron(eye(N),R));
h = zeros(N*(dimX+dimU),1);

% Define Matrizes for comparison restriction
g = [kron(ones(N-1,1),[5 5 0.2 0.2]');ff; kron(ones(N,1),[1.75
 1.75]')];
G = blkdiag(kron(eye(N-1),[1 0; -1 0; 0 1; 0 -1]),Ff, ...
            kron(eye(N),[1;-1]));

% Create Equality matrizes Aeq and beq
T = [eye(N*dimX) + kron(diag(ones(1,N-1),-1),-A),
 kron(diag(ones(1,N)),B)];
t = [A; zeros(dimX*(N-1),dimX)];

options=optimoptions('quadprog','ConstraintTolerance',1e-2);
x1=x0(1,1); x2=x0(2,1);

u_zopt=[];
time=1;

xi = x0;
i=1; maxIter = 100;
flag = 1; % default value to start loop

% No element violates conditions & flag= true
% Simulation is run until LQR set is reached, because in this region a
% controll is surely possible.
while(sum(Ff*xi>ff) > 0 && flag)
    tnew=t*xi;

    [Text, zopt, fval, flag] = evalc('quadprog(H, h, G, g, T, tnew,[],
[],xi,options);');

    x1=[x1;zopt(1)];
```

```
    x2=[x2;zopt(2)];
    u_zopt=[u_zopt,-zopt(2*N+1)];
    xi=[zopt(1);zopt(2)];
    time=[time,i+1];

    if(i > maxIter); fprintf('Maximum Iteration reached i=%d
 \n',i); break; end;
    i=i+1;

end

% Get and plot optimal trajectory data
x1_pred=zopt(3:2:2*(N));
x2_pred=zopt(4:2:2*(N));

% Create boundry polyhedron of the constraint set
h_bound=[sys.x.max;sys.x.max];
H_bound=[1 0;0 1;-1 0;0 -1 ];
P=Polyhedron([H_bound],[h_bound]);

% Create boundry polyhedron of the LQR set
h_bound2=[ff];
H_bound2=[Ff];
P2=Polyhedron([H_bound2],[h_bound2]);

% Check if state always lies inside boundaries
figure('Position',[0 0 1000 600]);
P.plot; hold on;
P2.plot('color',[0.5 0.5 0.7])
scatter(x1,x2,'b','*');
scatter(x1_pred,x2_pred,'g','*');
xlabel('x1'); ylabel('x2')
legend('Constraint on state','LQR invariant set','Simulation using
 MPC', ...
            'Prediction from inside the LQR set')

% The system respects the boundries for state constraints as well as
 input
% constraints. As the optimal solution is chosen, the optimal path is
 at
% the limit of the feasible set. This might lead to problem in the
 case of
% noise or even just numerical error.
% An option would be to introduce soft state (and input) constraints
 with a
% more conservative value to enhance the robustness of the control.

% We changed as well the matrices Q and R and zipped the corresponding
% figures to this Matlab file. We observed that raising the Q results
 in a
% smaller terminal set. This comes from the fact that a higher Q will
 raise
% the states cost in the LQR optimisation Problem. A higher state cost
 will
```

```
% result in a more agressive controller (higher gain K). Having a
 higher K
% decreases the the initial feasable set since the input constraints
 are going to
% be met at lower values of x. When we raise R than we are in the
 opposite
% case the states cost are smaller than the input cost in the LQR
% optimisation problem. This will result in a less agressive
 controller with
% a lower gain. The maximum control invariant set gets bigger when we
 raise
% R. The trajectories change only slightly in both cases.
```

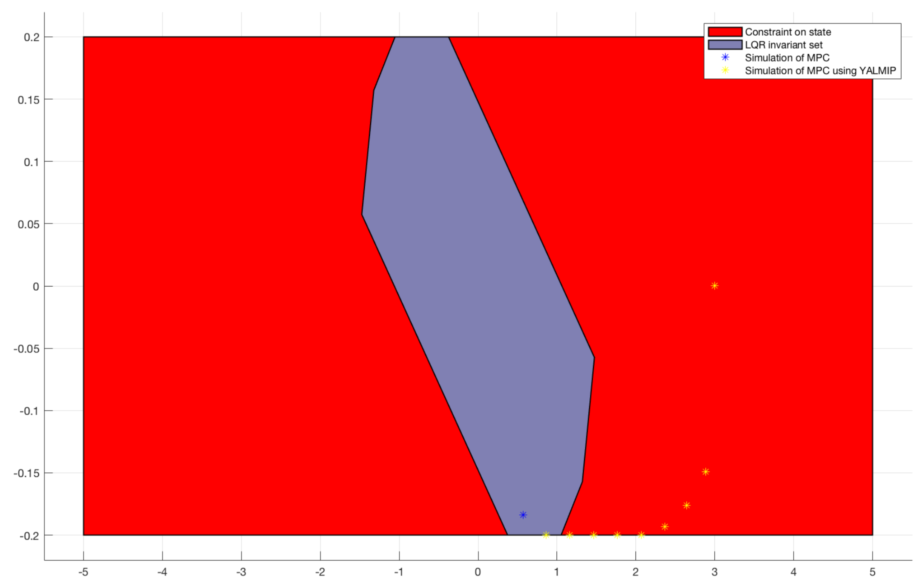


Exercise 2: Implement MPC using YALMIP

```
%close all;

% Parameter Definition
F = [ 1  0;               %State constraint
     -1  0;
      0  1;
      0 -1];
f = [5; 5; 0.2; 0.2];     %State constraint
m = [1.75; 1.75];         %Input constraint
M = [1; -1];              %Input constraint

% Define optimization variables
x = sdpvar(2,N,'full');
u = sdpvar(1,N,'full');

% Define constraints and objective
```

```
con = [];
obj = 0;

for i = 1:N-1
     con = [con, x(:,i+1) == A*x(:,i) + B*u(:,i)]; % System dynamics
     con = [con, F*x(:,i) <= f];                   % State constraints
     con = [con, M*u(:,i) <= m];                   % Input constraints
     obj = obj + x(:,i)'*Q*x(:,i) + u(:,i)'*R*u(:,i); % Cost function
end

con = [con, Ff*x(:,N) <= ff];        % Terminal constraint
obj = obj + x(:,N)'*Qf*x(:,N);       % Terminal weight

% Compile the matrices
%opt = sdpsettings('solver','sedumi','verbose',0); % choosing the
 solver

opt = sdpsettings;
opt.solver = 'quadprog';
opt.quadprog.TolCon = 1e-16;
ctrl = optimizer(con, obj, opt, x(:,1), u(:,1));


% Can now compute the optimal control input using
xi = x0;
x_yalm = [];
u_yalm = [];
t_yalm = [];

infeasible = 0; i = 1;

maxIter = maxIter; % 100
while(not(infeasible==1) && sum(Ff*xi>ff) > 0)
    [uOpt,infeasible] = ctrl{xi};

    x_yalm = [x_yalm, xi]; % save current values
    u_yalm = [u_yalm, uOpt];
    t_yalm = [t_yalm, i];

    xi = A*xi  + B*uOpt; % Update step

    if(i > maxIter); fprintf('Maximum Iteration reached i=%d
 \n',i); break; end;

    i = i + 1;
end
x_yalm = [x_yalm, xi]; % save current values
t_yalm = [t_yalm, i];

figure('Position',[0 0 1000 600]);
P.plot; hold on;
P2.plot('color',[0.5 0.5 0.7])
scatter(x1,x2,'b','*');
scatter(x_yalm(1,:), x_yalm(2,:), 'y','*');
```

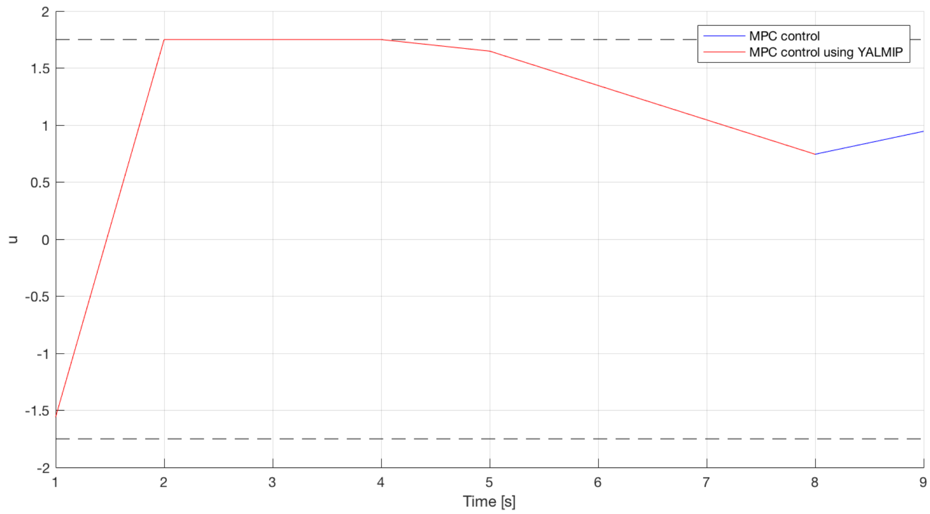
```
legend('Constraint on state','LQR invariant set','Simulation of
 MPC', ...
            'Simulation of MPC using YALMIP')

% BLue stars are hidden behind the yellow stars, because we get
 exactly the
% same values with YALMIP as with the manually implemented MPC
 controller.
% This is as expected.

% Suprisingly YALMIP stops one step earlier than the manual
 caclulation and
% accepts it as part of the LQR invariant set. This must be a result
 of
% numerical errors in matlab and the additional condition of the
 tolerance
% that was added in yalmip as a result of this.
```



MPC control

```
%close all;
% Check if input always lies inside boundaries
figure('Position',[0 0 800 400]); hold on;

plot(time(1:end-1),u_zopt,'b'); grid on;
plot(t_yalm(1:length(u_yalm)),u_yalm,'r'); grid on;
plot([1,max([time-1,t_yalm])],[1.75,1.75],'k--'); hold on; % upper
 boudnry
plot([1,max([time-1,t_yalm])],[-1.75,-1.75],'k--'); hold on; % lower
 bounrdry
plot(time(1:end-1),u_zopt,'b'); grid on;
```

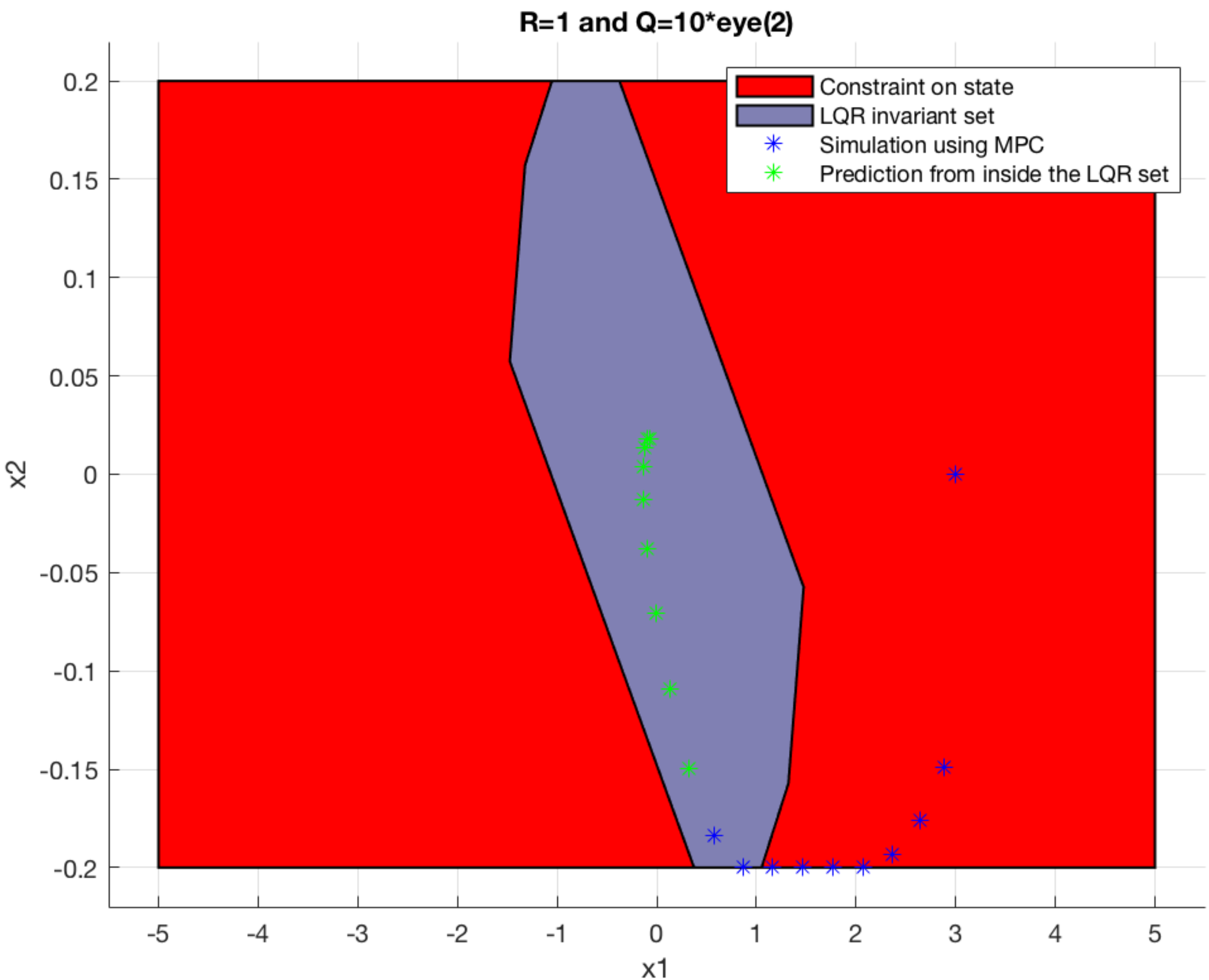
```
plot(t_yalm(1:length(u_yalm)),u_yalm,'r'); grid on;

xlabel('Time [s]'); ylabel('u')
xlim([min([time,t_yalm]), max([time-1,t_yalm])]);
legend('MPC control','MPC control using YALMIP')

% As the points on the trajectory, also the control u is the same for
 both
% calculation and simulation methods.
```

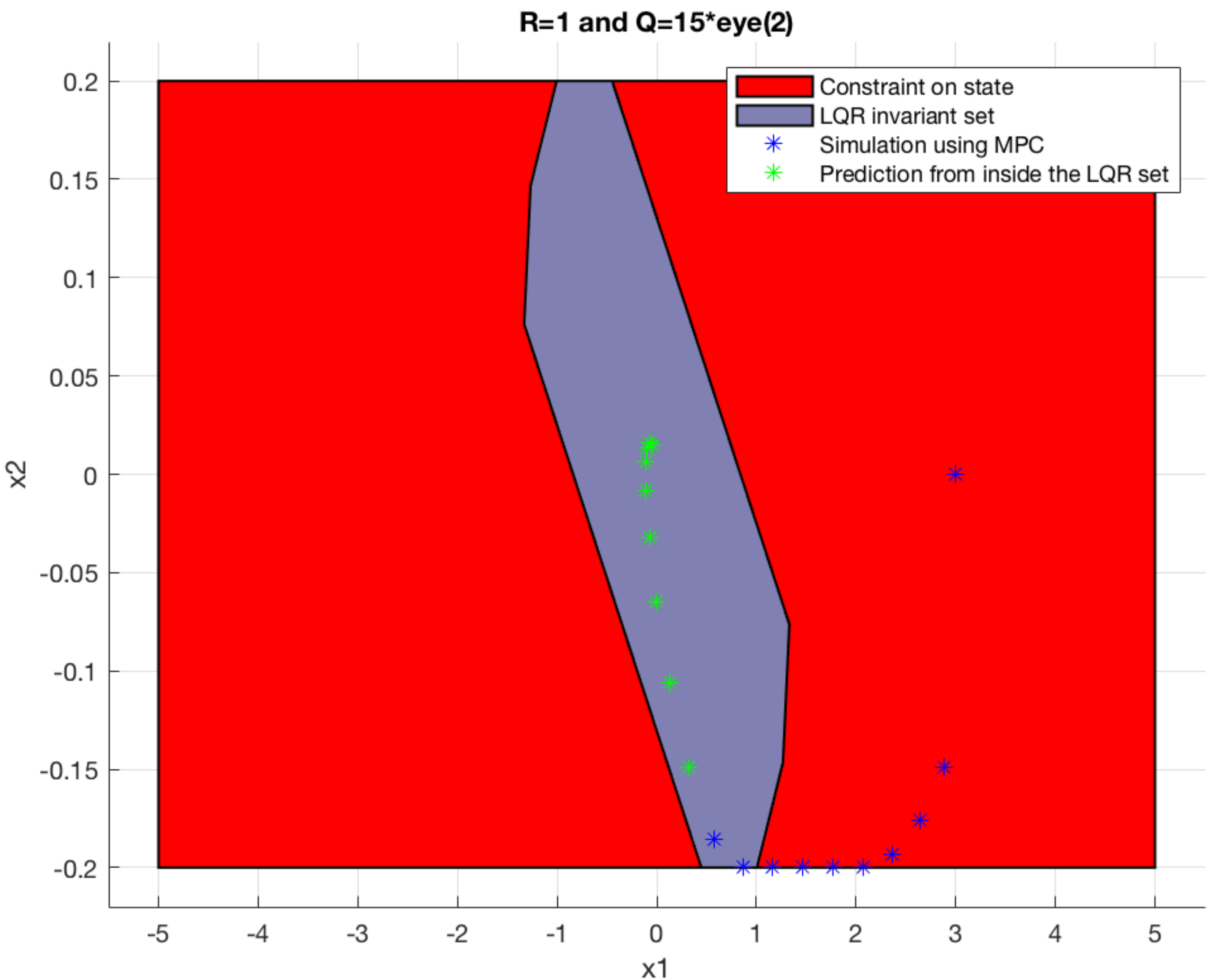


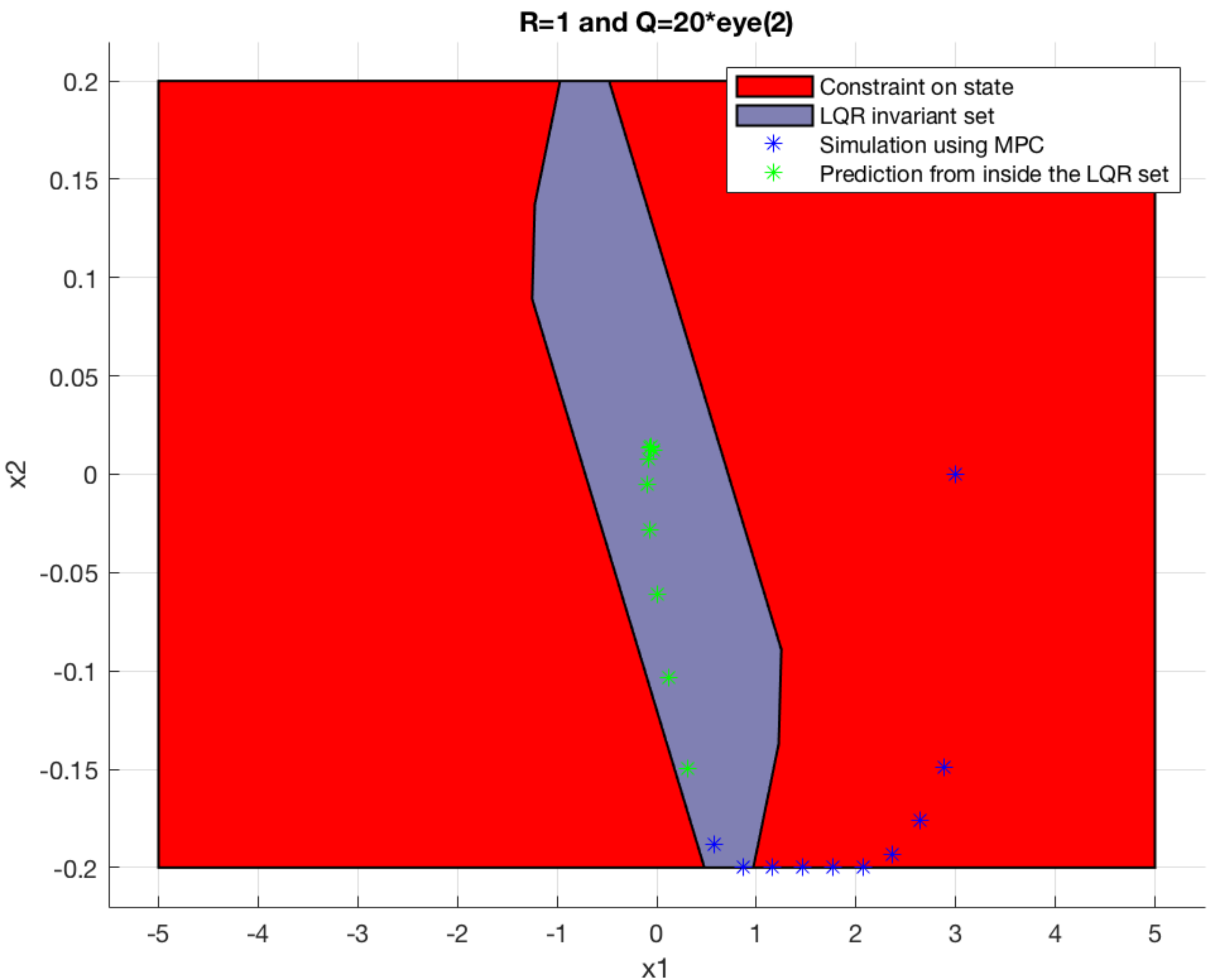

```
fprintf('Programm terminated. \n')
```

Programm terminated.

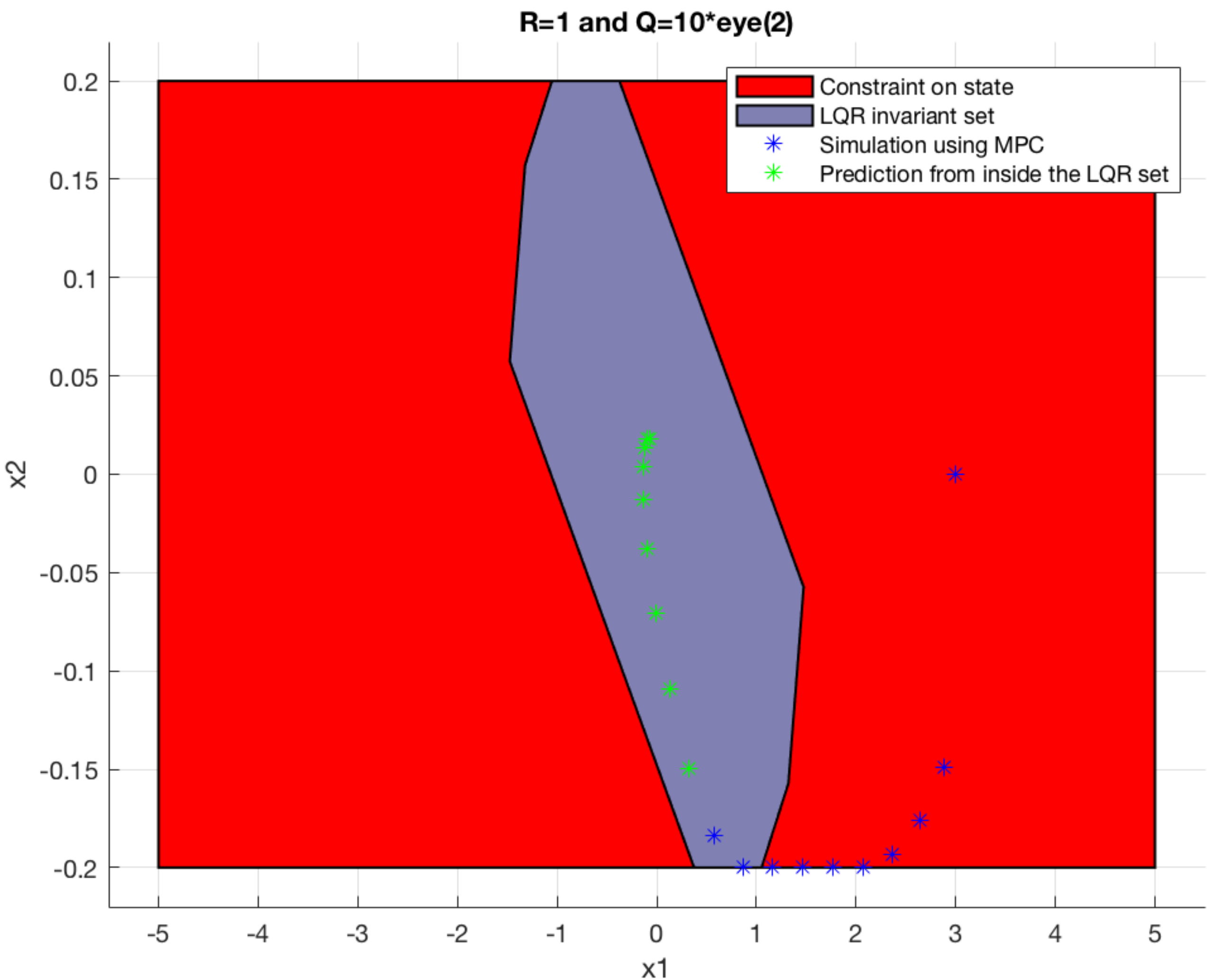
Published with MATLAB® R2016b


R=1 and Q=10*eye(2)
Constraint on state
LQR invariant set
Simulation using MPC
Prediction from inside the LQR set
x1
x2
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5

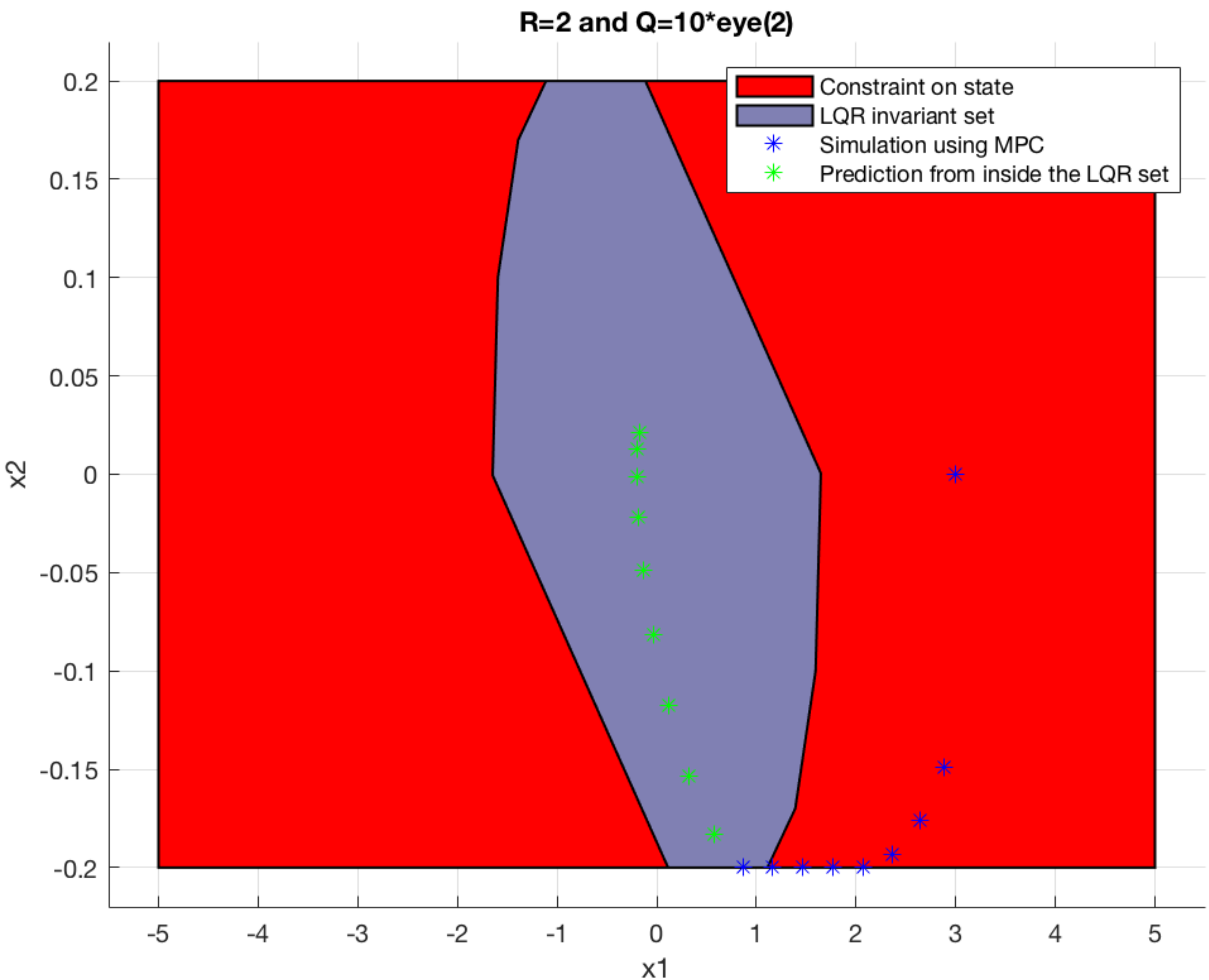

R=1 and Q=15*eye(2)
Constraint on state
LQR invariant set
Simulation using MPC
Prediction from inside the LQR set
x2
x1
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5

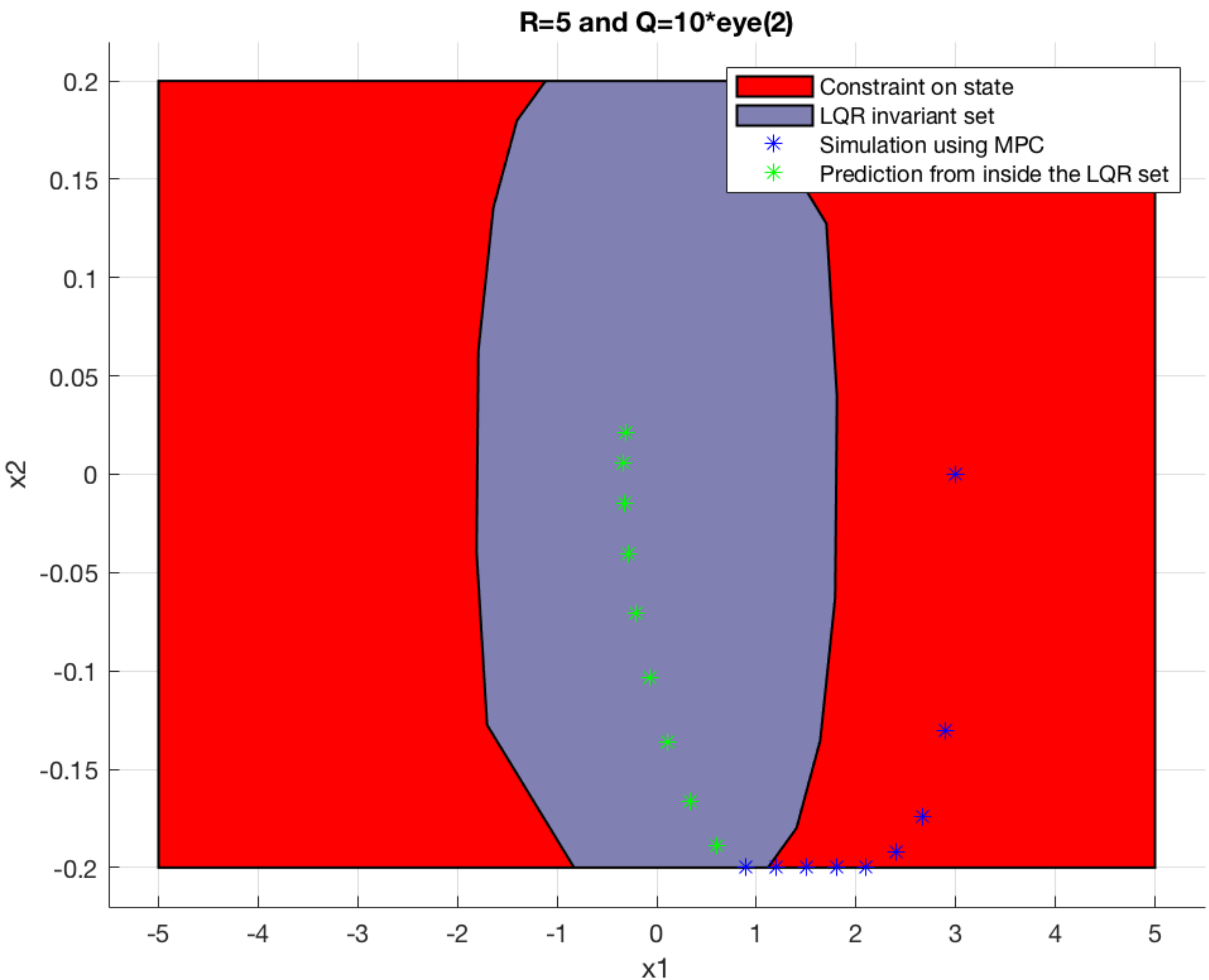


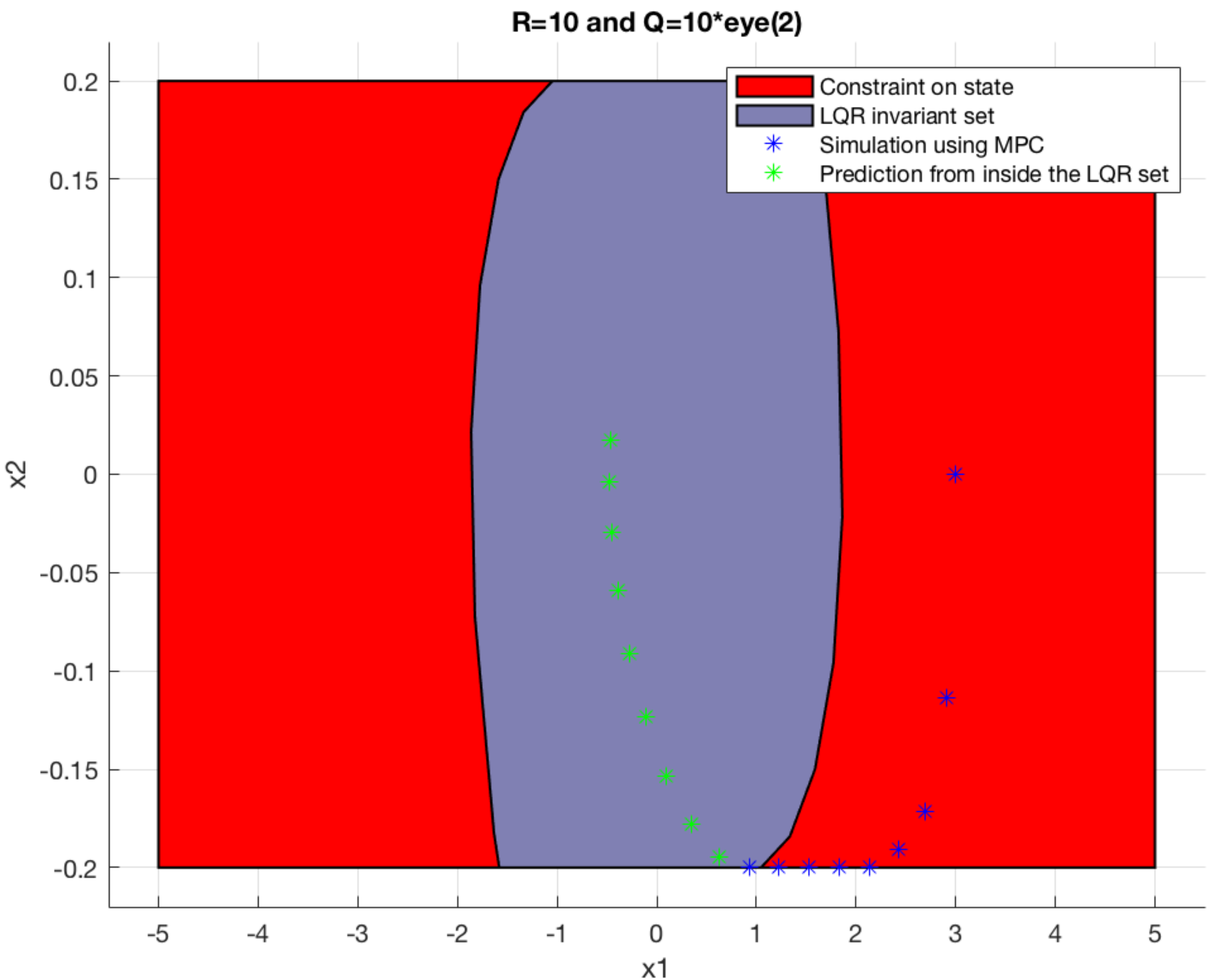
R=1 and Q=20*eye(2)
Constraint on state
LQR invariant set
Simulation using MPC
Prediction from inside the LQR set
x2
x1
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5



R=1 and Q=10*eye(2)
Constraint on state
LQR invariant set
Simulation using MPC
Prediction from inside the LQR set
x2
x1
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5


R=2 and Q=10*eye(2)
Constraint on state
LQR invariant set
Simulation using MPC
Prediction from inside the LQR set
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
x2
-5
-4
-3
-2
-1
0
1
2
3
4
5
x1


R=5 and Q=10*eye(2)
Constraint on state
LQR invariant set
Simulation using MPC
Prediction from inside the LQR set
x2
x1
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5


R=10 and Q=10*eye(2)
Constraint on state
LQR invariant set
Simulation using MPC
Prediction from inside the LQR set
x1
x2
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5