MECH 6327 - Homework 5

Jonas Wagner

2021, May 3

Contents

1 Control Design via SDP

Consider the linearized model of a Cessna Citation Aircraft (at 5000m, at speed 128.2 m/s)

$$\dot{x} = \begin{bmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} w y \quad = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{1}$$

with control input u = elevator angle, states x_1 = angle of at- tack, x_2 = pitch angle, x_3 = pitch rate, and x_4 = altitude (all relative to nominal), performance output y, and pitch rate disturbance input w.

Problem: Use the SDP formulations discussed in class to compute the optimal state feedback controller to minimize both the $\mathcal{H}_2$- and $\mathcal{H}_\infty$-norm from disturbance input w to output y. Also provide the optimal value for the respective closed-loop system norms.

1.1 $\mathcal{H}_2$-norm

The procedure outlined in class was implimented in CVX as seen in Appendix A.1, which resulted with the following results:

```
K_H2 =

        -44.6546   56.8903    1.7838    1.0000


Norm_H2 =

        0.5000
```

1.2 $\mathcal{H}_\infty$-norm

The procedure outlined in class was implimented in CVX as seen in Appendix A.1, which resulted with the following results:

```
K_Hinfty =

        1.0e+05 *

        -0.9457    1.1643    0.0299    0.0281


Norm_Hinfty =

        0.1330
```

2 Model Predictive Control (MPC)

Suppose the elevator angle input is limited to $\pm 0:262$rad (± 15 degrees), the elevator angle rate is limited to $\pm 0:524$rad/s (± 30 degrees/s), and we would like to limit the pitch angle to ± 0.349 rad (± 20 degrees). Consider the time discretized system with sampling period dt = 0:25s, and discrete-time dynamics matrices $A = e^{A_c \mathrm{d}t}$, $B = \mathrm{d}t\, B_c + \frac{1}{2}\,\mathrm{d}t\, A_c B_c + \frac{1}{6}\,\mathrm{d}t^2\, A_c^2 B_c$, where A_c and B_c are the above continuous time state space matrices.

2.1 LQR Implementation

Problem: Compute the optimal infnite-horizon LQR controller by solving the discrete-time algebraic Riccati equation with $Q = I$ and $R = 10$. Simulate the closed-loop system from initial state $x0 = \begin{bmatrix} 0 & 0 & 0 & 10 \end{bmatrix}^T$ but saturate the LQR controller to the input (magnitude and rate) constraints.

Solution: The calculation of an LQR controller gain was computed in MATLAB (Appendix A.2) and resulted in the following controller gain:

```
K_LQR =

        2.6795   -3.6639   -0.1890   -0.0447
```

This was then simulated with the required saturation constraints as can be seen in Figure 1.

It is clear that the LQR controller had difficulty stabilizing the system given the saturation requirements on the elevator position and rate. It is not only clear that the control signal was saturated and limited by the maximum rate requirements, but that state x_2 is well outside of the desired bounds. (It is also just obvious that the system is unstable and will likely continue with increasing magnitudes periodic movements)

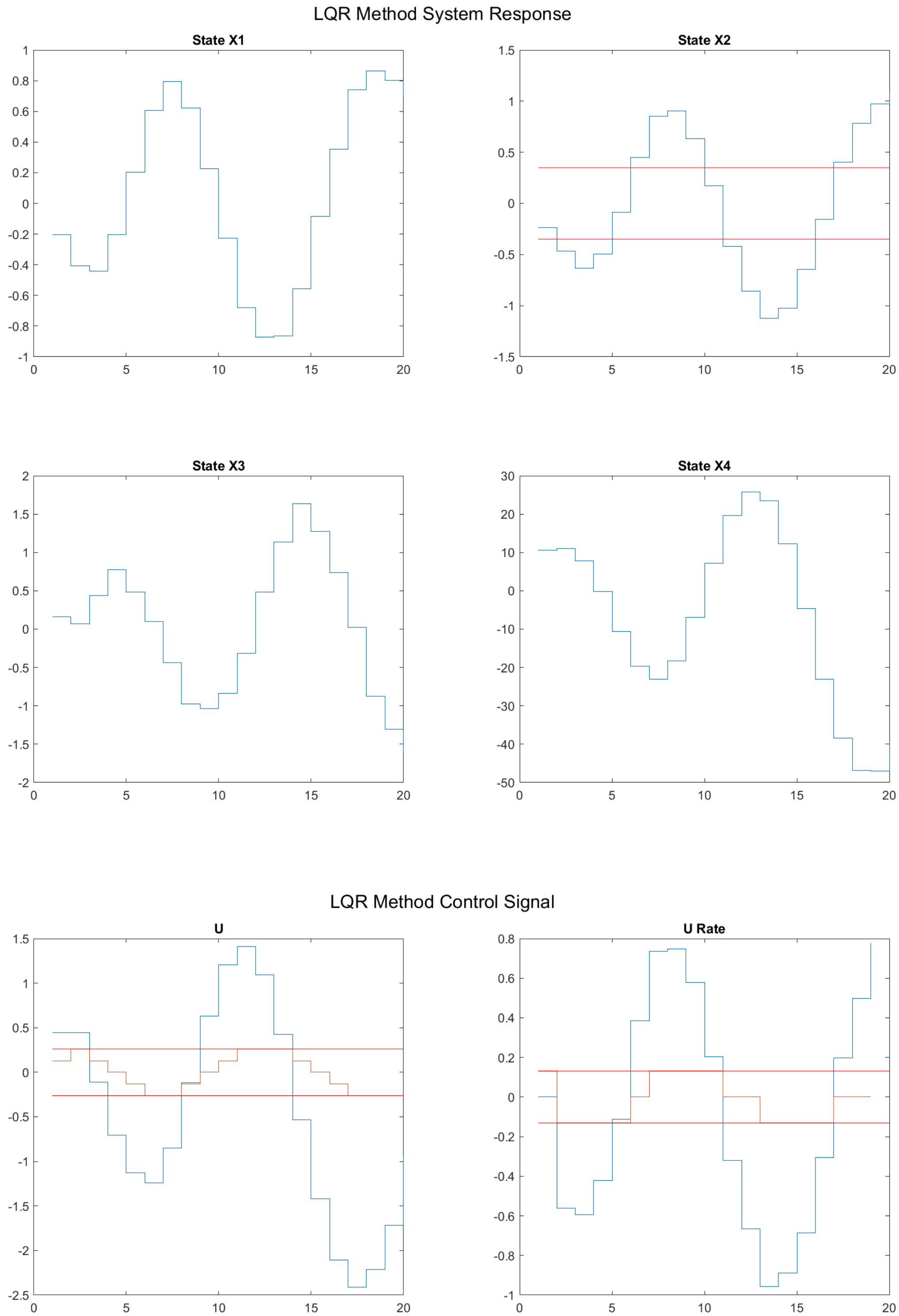

Figure 1: System response and control signal for the implementation of an infinite time LQR controller.

2.2 MPC Implementation

Problem: Implement a model predictive controller with horizon $T_h = 10$ sample periods and constant quadratic cost parameters $Q = I$ and $R = 10$ that explicitly accounts for the elevator angle and pitch constraints. Simulate the closed-loop system from initial state $x0 = \begin{bmatrix} 0 & 0 & 0 & 10 \end{bmatrix}^T$ and compare to the LQR controller. Determine using your implementation how short the MPC planning horizon can be reduced before stability problems arise.

Solution: The MPC was implimentated in MATLAB (Appendix A.2) and was then tested for multiple time horizons.

2.2.1 $T_h = 10$

In the original implementation (Figure 2), the controller is very effective (especially in comparison to the LQR controller). It is clear that the MPC implementation was able to intrinsically limit itself to the control limitations while also stabilizing the system effectively and within the desired timeframe while also staying within the ranges of x_2 dictated by the design goals.

2.2.2 $T_h = 5$

The MPC implementation for $T_h = 5$ (Figure 3) was also effective. Compared to the longer time-horizon implementations, the system appears to have slightly larger control-signal magnitudes as well as a faster, yet more volatile, response for the individual state responses.

2.2.3 $T - h = 3$

The MPC implementation for $T_h = 3$ (Figure 4) the system no longer stabalizable. Compared to the longer time-horizon implementations, the system has a troublesome appearance of uncontrolled ossilations that untimely cause state x_2 to reach outside of the desired bounds.

2.2.4 $T - h = 20$

A very different result occurs when the time-horizon is increased, as opposed to decreased. The MPC implementation for $T_h = 20$ (Figure 5) is even more effective then the original $T_h = 10$ implementation. All of the states quickly stabalized and are then maintained with very little control effort.

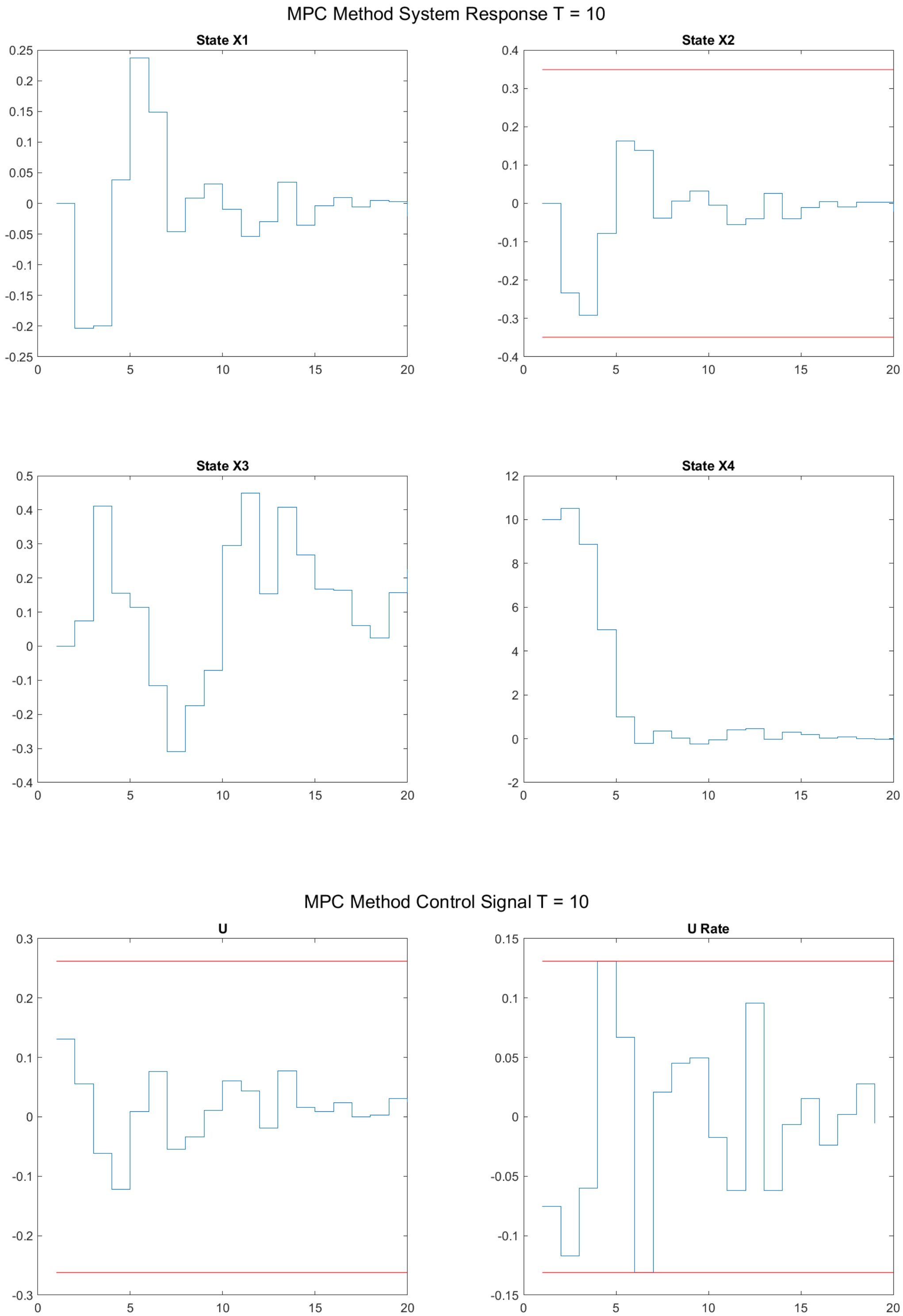

Figure 2: System response and control signal for the implementation of an MPC with $T_h = 10$.

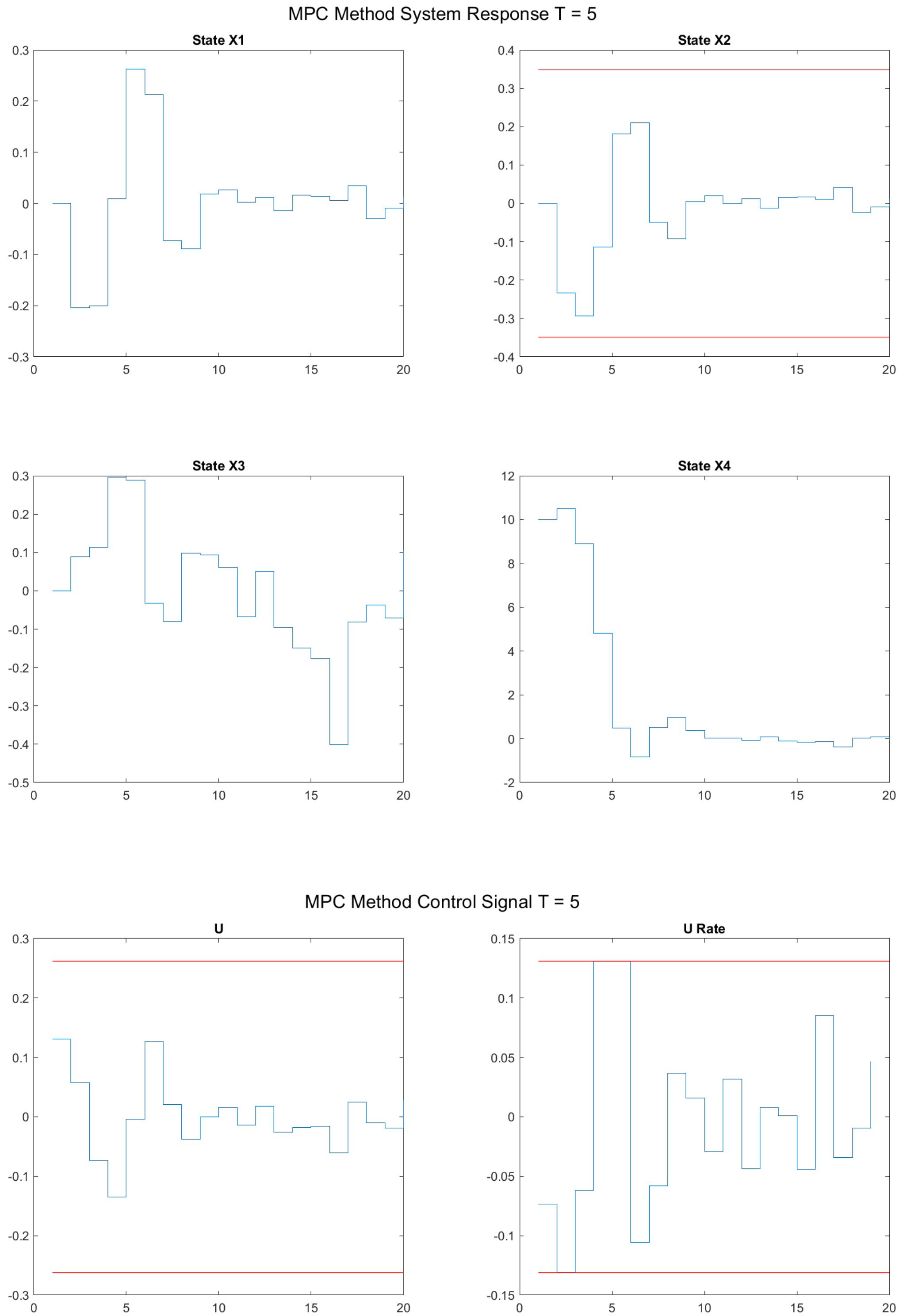

Figure 3: System response and control signal for the implementation of an MPC with $T_h = 5$.

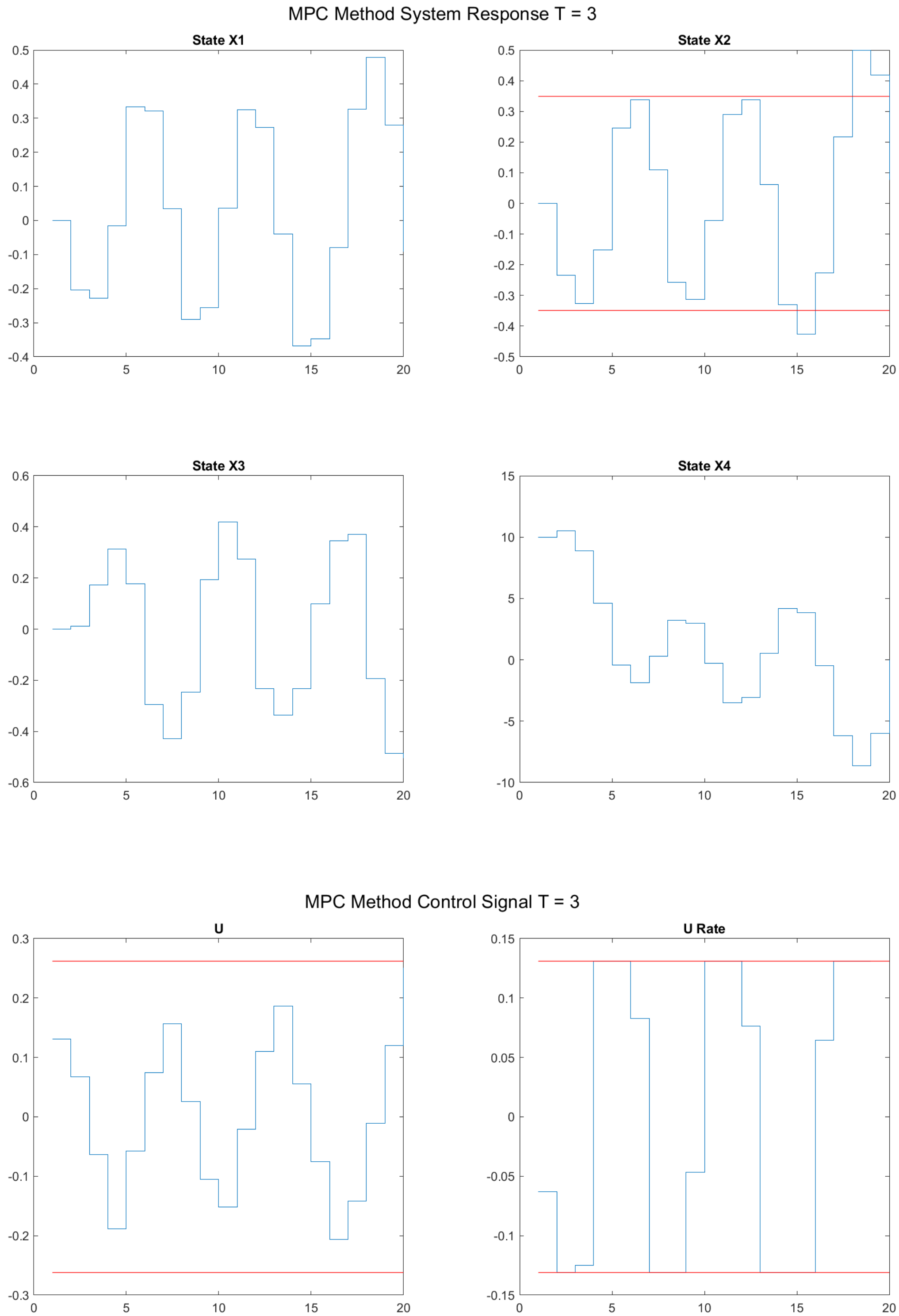

Figure 4: System response and control signal for the implementation of an MPC with $T_h = 3$.

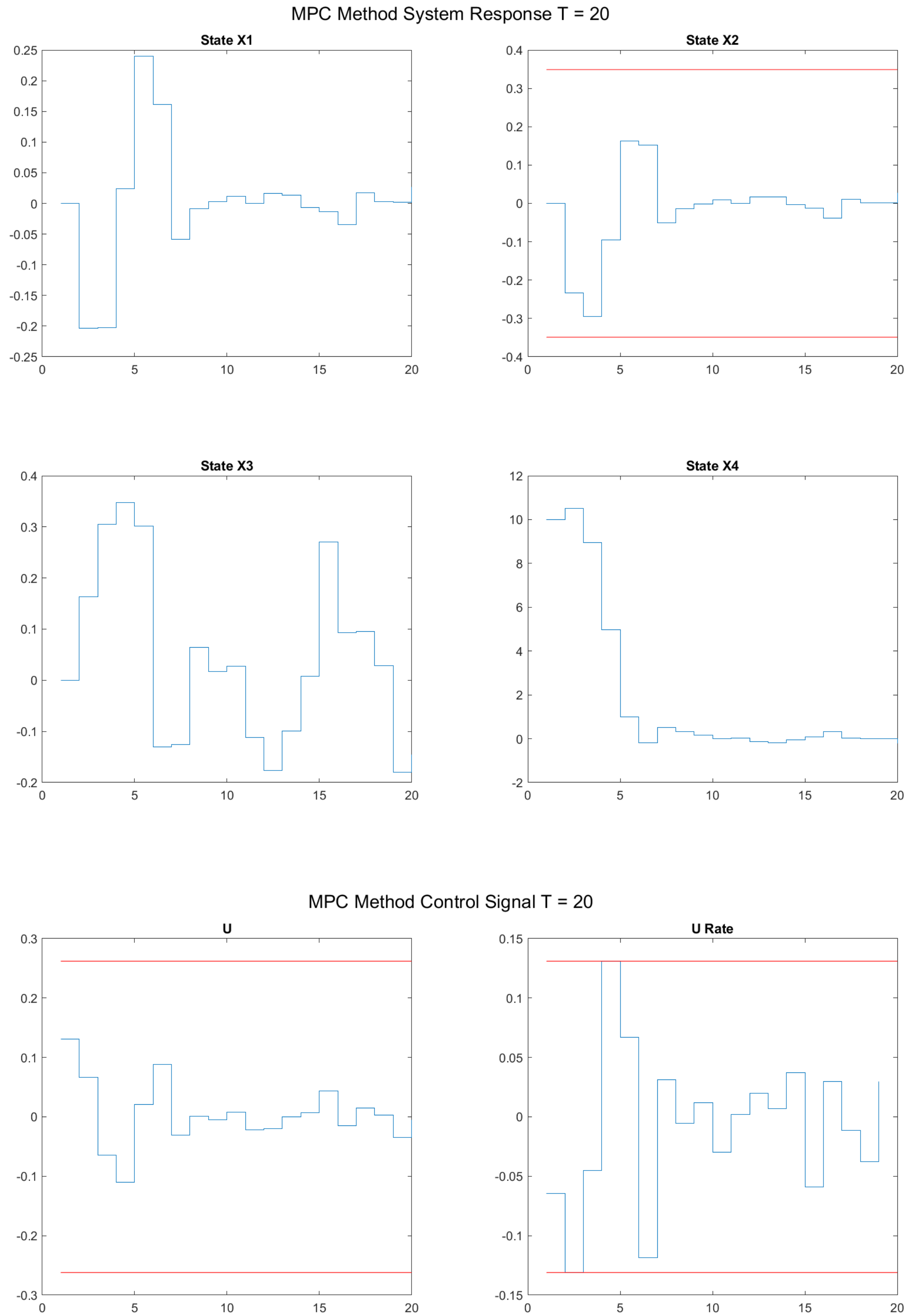

Figure 5: System response and control signal for the implementation of an MPC with $T_h = 20$.

A MATLAB Code:

All code I write in this course can be found on my GitHub repository:
https://github.com/jonaswagner2826/MECH6327

Script 1: MECH6327_HW5

```
% MECH 6327 - Homework 5
% Author: Jonas Wagner
% Date: 2020-05-02

clc;
clear
close all

% System Definition
MECH6327_HW5_sys_def

% Problem 1
MECH6327_HW5_pblm1

% Problem 2
MECH6327_HW5_pblm2

close all
```

A.1 Problem 1 Code

Script 2: MECH6327_HW5_pblm1

```
% MECH 6327 - Homework 5 - Problem 1
% Author: Jonas Wagner
% Date: 2020-05-02

clear;
close all;

h2norm = true;
hinftynorm = true;

%% System Setup
MECH6327_HW5_sys_def
n = size(A,1);
p = size(B,2);
q = size(C,1);

if h2norm
%% H-2 Norm Control
tol = 1e-6;
cvx_begin sdp
    variable X(n,n) symmetric
    variable W(q,q) symmetric
    variable L(1,n)
    variable Gamma
    minimize Gamma
    subject to
        [W, C*X+D*L;
         X*C'+L'*D', X] >= 0
        A*X + X*A' + B*L + L'*B' + F*F' <= 0
        trace(W) <= Gamma - tol
        X >= tol * eye(n)
        W >= tol * eye(q)
cvx_end

K_H2 = L*inv(X)
Norm_H2 = sqrt(Gamma)
end


if hinftynorm
```

```
%% H-infty Norm Control
tol = 1e-6;
cvx_begin sdp
    variable Q(n,n) symmetric
    variable L(p,n)
    variable eta

    minimize eta
    subject to
        [Q*A'+L'*B'+A*Q+B*L F Q*C'+L'*D';
         F' -eta zeros(p,q);
         C*Q+D*L zeros(q,p) -eye(q)] <= -tol*eye(n+q+1)
        Q >= tol*eye(n)
cvx_end


K_Hinfty = L*inv(Q)
Norm_Hinfty = sqrt(eta)
end
```

A.2 Problem 2 Code

Script 3: MECH6327_HW5_pblm2

```
% MECH 6327 - Homework 5 - Problem 1
% Author: Jonas Wagner
% Date: 2020-05-02

clear;
close all;


lqr = true;
mpc = true;
runModels = true;
plotResults = true;

if runModels
%% System Setup
MECH6327_HW5_sys_def
n = size(A,1);
p = size(B,2);
q = size(C,1);

A_c = A;
B_c = B;
C_c = C;
D_c = D;
F_c = F;


% Discritization
dt = 0.25;
A = expm(A_c * dt);
B = dt*B_c + (1/2)*dt*A_c*B_c + (1/6)*(dt^2)*(A_c^2)*B_c;
C = C_c;
D = D_c;
sys = ss(A,B,C,D,dt)

% System Limitations
u_limit = 0.262; % rad (input limit)
u_rate_limit = 0.524 * dt; % rad/s (Input Change Max
x2_limit = 0.349; % rad (objective)

```

```
%% Simulation Setup
N = 20;

w_power = 0.1;
W = w_power * randn(size(F,2),N);

x0 = [0;0;0;10];

if lqr
%% LQR Control Design
% LQR Design Parameters
Q = eye(n);
R = 10;

% Feedback gain calculation
[~,K_LQR,~] = idare(A,B,Q,R) % Prints out K

%% Simulate LQR Methods
% Feedback Gain
K = K_LQR;
% Simulation
X = zeros(n,N); % States
U = zeros(p,N); % Inputs
Y = zeros(q,N); % Outputs
U_sat = U;
% Initialization
x = x0;
U(:,1) = - K*x;
U_sat(:,1) = min(u_rate_limit, max(-u_rate_limit, U(:,1)));
%Assuming U(:,0) = 0
X(:,1) = A * x + B * U_sat(:,1) + F * W(:,1);
Y(:,1) = C * x + D * U_sat(:,1);
for i = 2:N
    % Control Calculation and saturation
    U(:,i) = - K*x;
    U_sat(:,i) = min(u_limit, max(-u_limit, U(:,i)));
    %Apparently this isn't what the limitation meant...
% % Applied Control and Limitation
% Bu = B * U_sat(:,i);
% Bu(3) = min(x3_limit, max(-x3_limit, Bu(3)));
    % Elevator Angle Rate Max
    if (U_sat(:,i) - U_sat(:,i-1) >= u_rate_limit)
        U_sat(:,i) = U_sat(:,i-1) + u_rate_limit;
```

```
    elseif (U_sat(:,i) - U_sat(:,i-1) <= -u_rate_limit)
        U_sat(:,i) = U_sat(:,i-1)-u_rate_limit;
    end
    % Time Update
    x = A * x + B * U_sat(:,i) + F * W(:,i);
    y = C * x + D * U_sat(:,i);
    % Save Values
    X(:,i) = x;
    Y(:,i) = y;
end

% Save to LQR Values
X_LQR = X;
U_LQR = U;
U_sat_LQR = U_sat;

end

if mpc
%% MPC Control Design
% MPC Parameters
Q = eye(n);
R = 10;
T = 10; % Time Horizon
umax = u_limit;

% History Matrices
Xhist = zeros(n,N); % States
Uhist = zeros(p,N); % Inputs
Yhist = zeros(q,N); % Outputs
Jhist = zeros(1,N); % Stage costs

% Simulation setup
X = zeros(n,T);
U = zeros(p,T);
x = x0;

% Dynamics Matrices
G = zeros(n*T,n);
for i=1:T
    G((i-1)*n+1:n*i,:) = A^i;
end

```

```
H = eye(n*T);
for i=1:T
    for j=1:T
        if i > j
            H((i-1)*n+1:n*i,(j-1)*n+1:n*j) = A^(i-j);
        end
    end
end
BB = kron(eye(T),B);
H = H * BB;

% Simulation
tic;
disp('------------ MPC Method: ------------')
for i = 1:N
    u_last = U(:,1); %assumes at 0 at start
    disp(['Iteraton: ', num2str(i)]);
    % solve open-loop optimization problem
    cvx_begin quiet
        variable X(n,T) % predicted state trajectory
        variable U(p,T) % planned control actions
        minimize ((vec(X)' * kron(eye(T), Q)...
                    * vec(X) + vec(U)'*kron(eye(T), R)...
                    *vec(U)))
        subject to
            vec(X) == G * x + H * vec(U); % System Dynamics
            norms(U,inf) <= umax; % Input Limitatiosn
            norm(U(:,1)- u_last ,inf) <= u_rate_limit;
            for j = 2:T
                norm(U(:,j)-U(:,j-1),inf) <= u_rate_limit;
            end
    cvx_end

    % Current Input and Output
    u = U(:,1);
    y = C * x + D * u;

    % Store Data
    Xhist(:,i) = x;
    Uhist(:,i) = u;
    Yhist(:,i) = y;
    Jhist(i) = x'*Q*x+u'*R*u;

```

```
    % State Update
    x = A*x + B*u + F * W(:,i);
end
MPC_runtime = toc

X_MPC = Xhist;
U_MPC = Uhist;
Y_MPC = Yhist;
J_MPC = Jhist;

end
end



if plotResults
%% Ploting

if lqr
% Plot LQR Plots
% Plot MPC Plots
figure('position',[0,0,1200,1000])
sgtitle('LQR Method System Response')
for i = 1:4
    subplot(2,2,i)
    stairs(X_LQR(i,:));
    hold on
    if i == 2
        plot(x2_limit*ones(N,1),'r')
        plot(-x2_limit*ones(N,1),'r')
    end
    title(['State X',num2str(i)])
end
saveas(gcf,[pwd,'\Homework\HW5\fig\pblm2_LQR_sys_response.png'])

figure('position',[0,0,1200,500])
sgtitle('LQR Method Control Signal')
subplot(1,2,1)
stairs(U_LQR')
hold on
stairs(U_sat_LQR')
plot(u_limit*ones(N,1),'r')
plot(-u_limit*ones(N,1),'r')
```

```
title('U')

subplot(1,2,2)
stairs(diff(U_LQR'))
hold on
stairs(diff(U_sat_LQR'))
plot(u_rate_limit*ones(N,1), 'r')
plot(-u_rate_limit*ones(N,1), 'r')
title('U Rate')
saveas(gcf,[pwd,'\Homework\HW5\fig\pblm2_LQR_ctrl_signal.png'])
end

if mpc
% Plot MPC Plots
figure('position',[0,0,1200,1000])
sgtitle(['MPC Method System Response T = ',num2str(T)])
for i = 1:4
    subplot(2,2,i)
    stairs(X_MPC(i,:));
    hold on
    if i == 2
        plot(x2_limit*ones(N,1),'r')
        plot(-x2_limit*ones(N,1),'r')
    end
    title(['State X',num2str(i)])
end
saveas(gcf,[pwd,'\Homework\HW5\fig\pblm2_MPC_T',num2str(T),...
    '_sys_response.png'])

figure('position',[0,0,1200,500])
sgtitle(['MPC Method Control Signal T = ',num2str(T)])
subplot(1,2,1)
stairs(U_MPC')
hold on
plot(u_limit*ones(N,1),'r')
plot(-u_limit*ones(N,1),'r')
title('U')


subplot(1,2,2)
stairs(diff(U_MPC'))
hold on
plot(u_rate_limit*ones(N,1), 'r')
```

```
plot(-u_rate_limit*ones(N,1), 'r')
title('U Rate')
saveas(gcf,[pwd,'\Homework\HW5\fig\pblm2_MPC_T',num2str(T),...
    '_ctrl_signal.png'])

end


end
```