Model Predictive Control: Theory and Applications

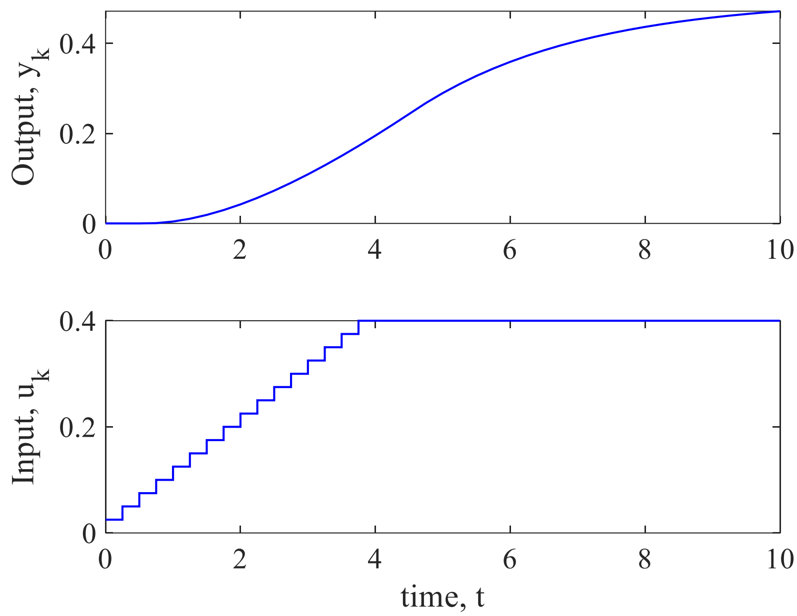
Assignment-11

Problem 1: No Measured Disturbance or Plant-Model Mismatch

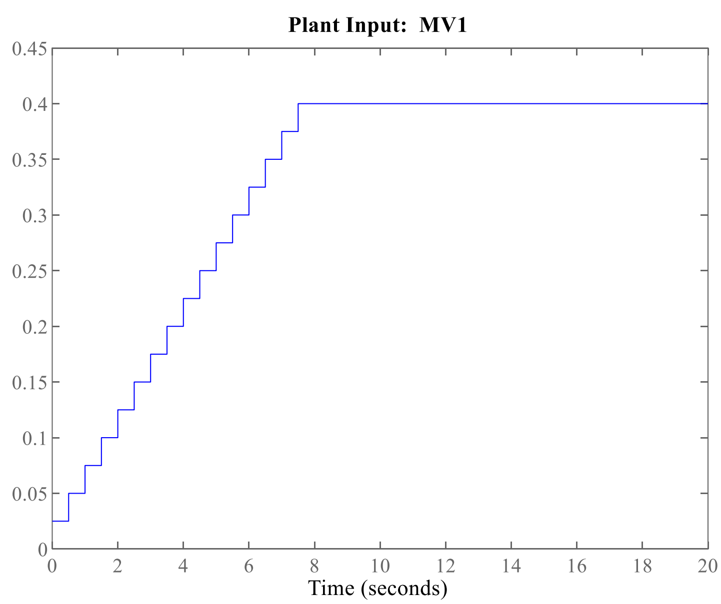
```
% 2 methods have been shown here. One using mpcmove() and other using sim()
%% Initialization
G1=tf(1.25,[5,1],'inputdelay',1.4);      % Transfer function
Ts=0.5;                                  % Sampling interval
Gc=ss(G1);                               % Continuous time SS conversion

% Discretizing partial time delay SS system
phi=expm(Gc.A*Ts);
tau=mod(1.4,Ts);
syms x;
gamma1=expm(Gc.A*(Ts-tau))*double(int(expm(Gc.A*x),0,tau))*Gc.B;
gamma0=double(int(expm(Gc.A*x),0,Ts-tau))*Gc.B;

A=[phi,gamma1,gamma0,0;0,0,1,0;0,0,0,1;0,0,0,0];
B=[0;0;0;1]
C=[Gc.C,0,0,0];
G=ss(A,B,C,0,-1);                        % Discrete SS model

m=mpc(G,Ts);                             % Creating model predictive
controller
m.C=5;                                   % Control horizon
m.P=12;                                  % Prediction horizon
m.W.OutputVariables=1;                   % Output weight
m.W.ManipulatedVariablesRate=0.1;        % Input-rate weight
m.MV.Min=-0.4;                           % Constraint limits
m.MV.Max=0.4;
m.MV.RateMin=-0.025;
m.MV.RateMax=0.025;

r=0.8;                                   % Set-point
maxTime=41;                              % Maximum number of iterations
Time=0.25*[0:maxTime-1];                 % Discrete time
x=mpcstate(m);                           % mpc state variable
y=0;
u=[];

%% Calculating input moves and output
for k=1:maxTime
    u=[u;mpcmove(m,x,y(k),r)];           % Calculating input moves using
mpcmove
    y=[y;G.C*x.Plant];
end
[y1,t1,u1]=sim(m,maxTime,r);             % Calculating input moves using sim

%% Comparing input moves and output from both methods
if u1==u & y1==y(1:end-1)
    disp('Input moves and output calculated from both methods match');
end

%% Plotting results
subplot(2,1,1);
```

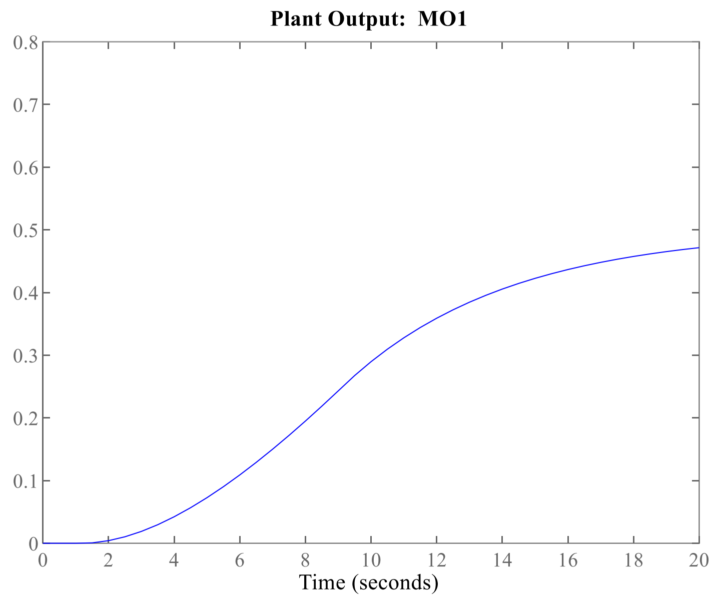
```
plot(Time,y(1:maxTime),'-b','linewidth',1);
ylabel('Output, y_k');
subplot(2,1,2);
stairs(Time,u(1:maxTime),'-b','linewidth',1);
ylabel('Input, u_k'); xlabel('time, t');

sim(m,maxTime,r);
```

The plot obtained is:



The plots obtained using sim() are:



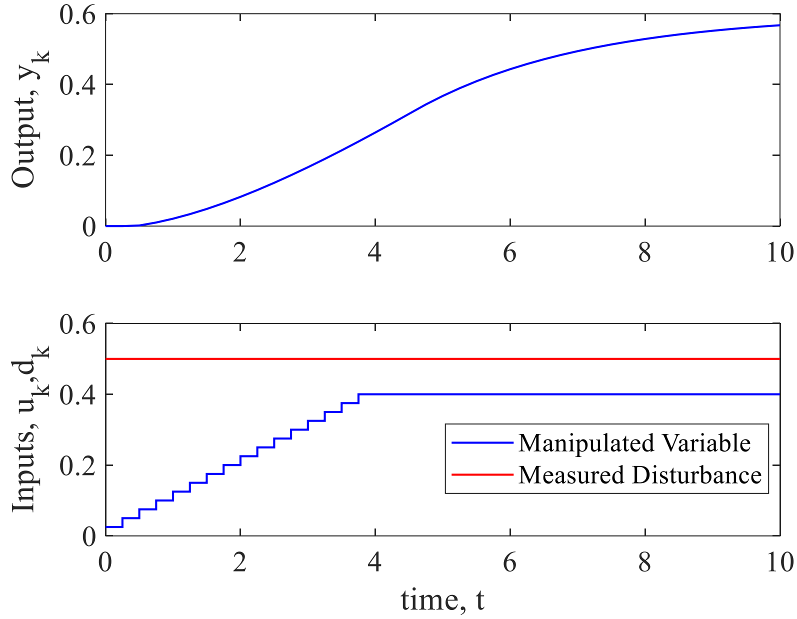


Problem 2: Measured Disturbance

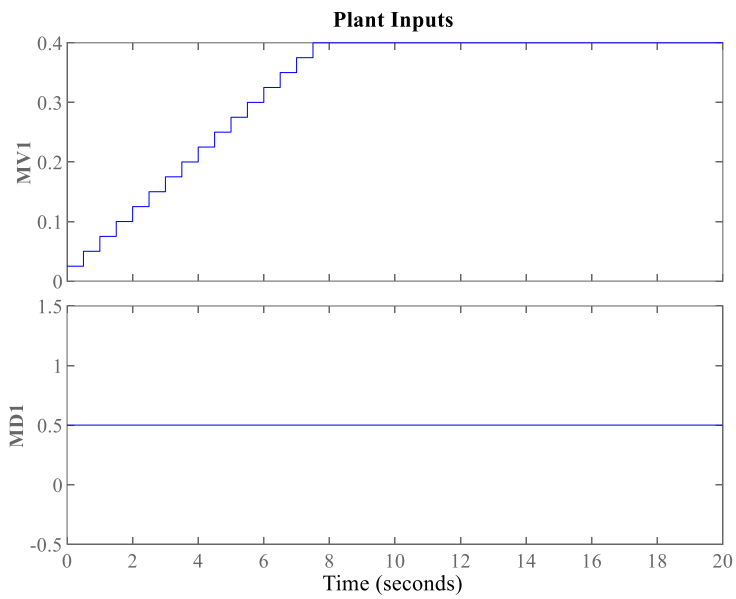
```
% 2 methods have been shown here. One using mpcmove() and other using sim()
%% Initialization
G1=tf(1.25,[5,1],'inputdelay',1.4);      % Transfer function for input
G2=tf(0.2,[6,1],'inputdelay',0.7);       % Transfer function for disturbance
Ts=0.5;                                  % Sampling interval
Gc1=ss(G1);                              % Continuous SS conversion
Gc2=ss(G2);

% Discretizing partial time delay SS system
% Input
phi1=expm(Gc1.A*Ts);
tau1=mod(1.4,Ts);
syms x;
gamma11=expm(Gc1.A*(Ts-tau1))*double(int(expm(Gc1.A*x),0,tau1))*Gc1.B;
gamma01=double(int(expm(Gc1.A*x),0,Ts-tau1))*Gc1.B;

A1=[phi1,gamma11,gamma01,0;0,0,1,0;0,0,0,1;0,0,0,0];
B1=[0;0;0;1]
C1=[Gc1.C,0,0,0];

% Disturbance
phi2=expm(Gc2.A*Ts);
tau2=mod(1.4,Ts);
gamma12=expm(Gc2.A*(Ts-tau2))*double(int(expm(Gc2.A*x),0,tau2))*Gc2.B;
gamma02=double(int(expm(Gc2.A*x),0,Ts-tau2))*Gc2.B;

A2=[phi2,gamma12,gamma02;0,0,1;0,0,0];
B2=[0;0;1]
C2=[Gc2.C,0,0];
```

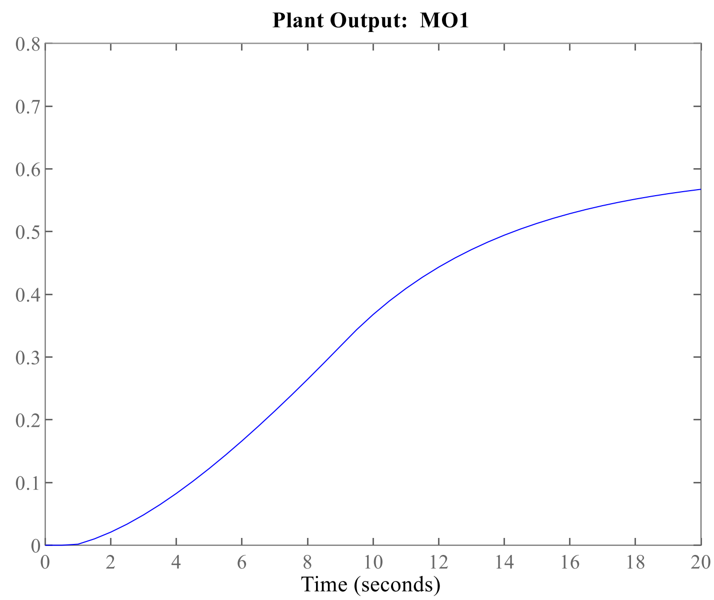
```
A=[A1,zeros(size(A1,1),size(A2,2));zeros(size(A2,1),size(A1,2)),A2];
B=[B1,zeros(size(B1,1),size(B2,2));zeros(size(B2,1),size(B1,2)),B2];
C=[C1,C2];
G=ss(A,B,C,0,-1);                       % Discrete SS model

G.InputGroup.MV = 1;                    % Distinguishing measured variable
G.InputGroup.MD = 2;                    % and measured disturbance

m=mpc(G,Ts);                            % Creating model predictive
controller
m.C=5;                                  % Control horizon
m.P=12;                                 % Prediction horizon
m.W.OutputVariables=1;                  % Output weight
m.W.ManipulatedVariablesRate=0.1;       % Input-rate weight
m.MV.Min=-0.4;                          % Constraint limits
m.MV.Max=0.4;
m.MV.RateMin=-0.025;
m.MV.RateMax=0.025;

r=0.8;                                  % Set-point
maxTime=41;                             % Maximum number of iterations
Time=0.25*[0:maxTime-1];                % Discrete time
x=mpcstate(m);                          % mpc state variable
y=0;
u=[];
d=0.5;                                  % Measured disturbance

%% Calculating input moves
for k=1:maxTime
    u=[u;mpcmove(m,x,y(k),r,d)];        % Calculating input moves using
mpcmove
    y=[y;G.C*x.Plant];
end
[y1,t1,u1]=sim(m,maxTime,r,d);          % Calculating input moves using sim

%% Comparing input moves and output from both methods
if u1==u & y1==y(1:end-1)
    disp('Input moves and output calculated from both methods match');
end

%% Plotting results
subplot(2,1,1);
plot(Time,y(1:maxTime),'-b','linewidth',1);
ylabel('Output, y_k');
subplot(2,1,2);
stairs(Time,u(1:maxTime),'-b','linewidth',1);
hold on;
stairs(Time,repmat(d,1,41),'-r','linewidth',1);
hold off;
ylabel('Inputs, u_k,d_k'); xlabel('time, t');
legend({'Manipulated Variable','Measured
Disturbance'},'Location','northeast');

sim(m,maxTime,r,d);
```

```
The plot obtained is:
```




```
The plots obtained using sim() are:
```



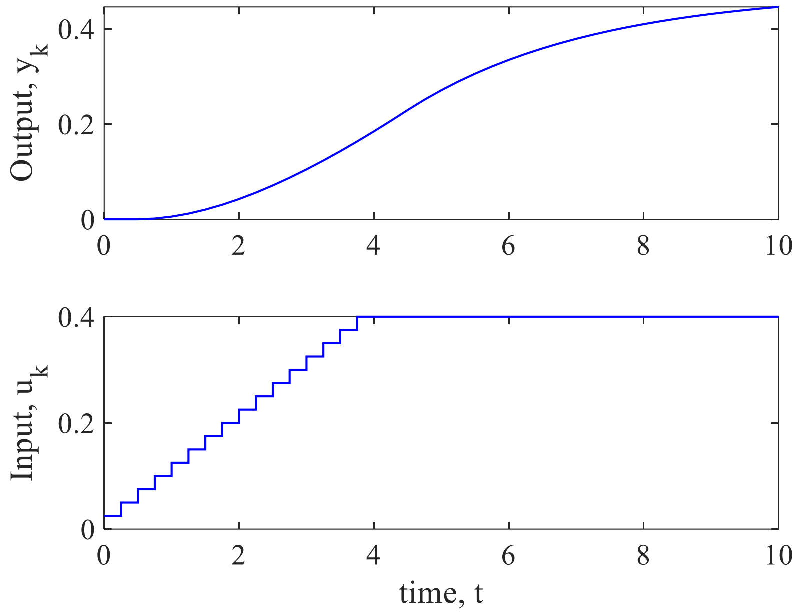


Problem3: Plant-Model Mismatch

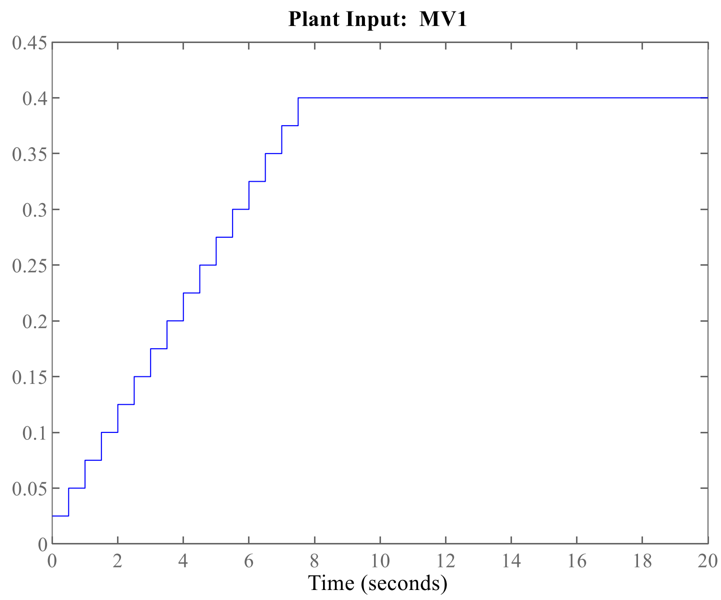
```
%% Initialization
G1=tf(1.25,[5,1],'inputdelay',1.4);      % Transfer function for model
G1t=tf(1.2,[5.5,1],'inputdelay',1.2);    % Transfer function for true plant
Ts=0.5;                                  % Sampling interval
Gc1=ss(G1);                              % Continuous SS conversion
Gc1t=ss(G1t);

% Discretizing partial time delay SS system
% Model
phi1=expm(Gc1.A*Ts);
tau1=mod(1.4,Ts);
syms x;
gamma11=expm(Gc1.A*(Ts-tau1))*double(int(expm(Gc1.A*x),0,tau1))*Gc1.B;
gamma01=double(int(expm(Gc1.A*x),0,Ts-tau1))*Gc1.B;

A1=[phi1,gamma11,gamma01,0;0,0,1,0;0,0,0,1;0,0,0,0];
B1=[0;0;0;1];
C1=[Gc1.C,0,0,0];

% Plant
phi1t=expm(Gc1t.A*Ts);
tau1t=mod(1.2,Ts);
syms x;
gamma1t=expm(Gc1t.A*(Ts-tau1t))*double(int(expm(Gc1t.A*x),0,tau1t))*Gc1t.B;
gamma0t=double(int(expm(Gc1t.A*x),0,Ts-tau1t))*Gc1t.B;

A1t=[phi1t,gamma1t,gamma0t,0;0,0,1,0;0,0,0,1;0,0,0,0];
B1t=[0;0;0;1];
C1t=[Gc1t.C,0,0,0];

G1=ss(A1,B1,C1,0,-1);                    % Discrete SS for model
G1t=ss(A1t,B1t,C1t,0,-1);                % Discrete SS for plant
```

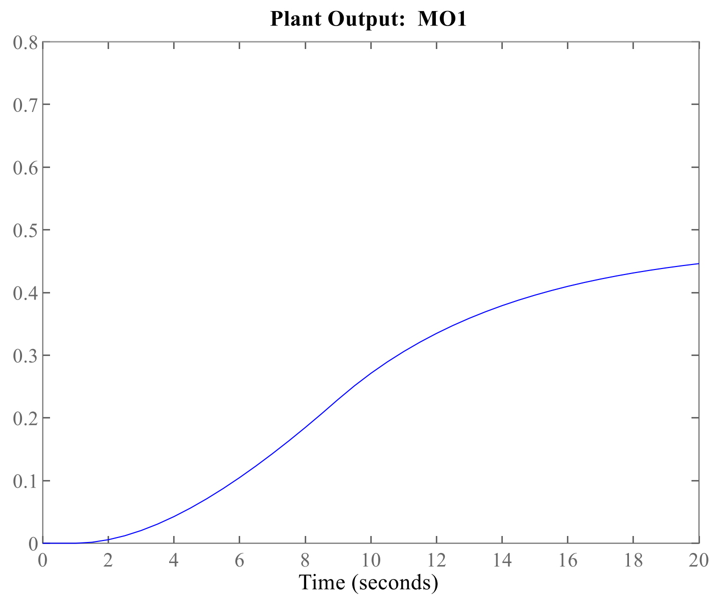
```
m=mpc(G1,Ts);                                     % Creating model predictive
controller
m.C=5;                                            % Control horizon
m.P=12;                                           % Prediction horizon
m.W.OutputVariables=1;                            % Output weight
m.W.ManipulatedVariablesRate=0.1;                 % Input-rate weight
m.MV.Min=-0.4;                                    % Constraint limits
m.MV.Max=0.4;
m.MV.RateMin=-0.025;
m.MV.RateMax=0.025;

r=0.8;                                            % Set-point
maxTime=41;                                       % Maximum number of iterations
Time=0.25*[0:maxTime-1];                          % Discrete time
x=mpcstate(m);                                    % mpc state variable
y=0;
u=[];
X=zeros(4,1);

%% Calculating input moves
for k=1:maxTime
    u=[u;mpcmove(m,x,y(k),r)];                            % Calculating input moves
using mpcmove
    X=G1t.A*X+G1t.B*u(k);
    y=[y;G1t.C*X];
end
mopts=mpcsimopt;
mopts.Model=G1t;
[y1,t1,u1]=sim(m,maxTime,r,mopts);                        % Calculating input moves
using sim

%% Comparing input moves and output from both methods
if u1==u & y1==y(1:end-1)
    disp('Input moves and output calculated from both methods match');
end

%% Plotting results
subplot(2,1,1);
plot(Time,y(1:maxTime),'-b','linewidth',1);
ylabel('Output, y_k');
subplot(2,1,2);
stairs(Time,u(1:maxTime),'-b','linewidth',1);
ylabel('Input, u_k'); xlabel('time, t');

sim(m,maxTime,r,mopts);
```

The plot obtained is:



The plots obtained using sim() are:





Observations:

- In all 3 cases the set point is not achieved because input is restricted at 0.4.
- The input moves reach maximum value at 0.4 and remain there in all the 3 cases.
- When disturbance of 0.5 acts, the output is closer to the set-point and greater than the other 2 cases by around 0.1 (0.1 is the gain of the disturbance transfer function times the disturbance)
- Plots made using data collected from mpcmove and sim are found to be the same.