**Model Predictive Control: Theory and Applications**

**Assignment-6**

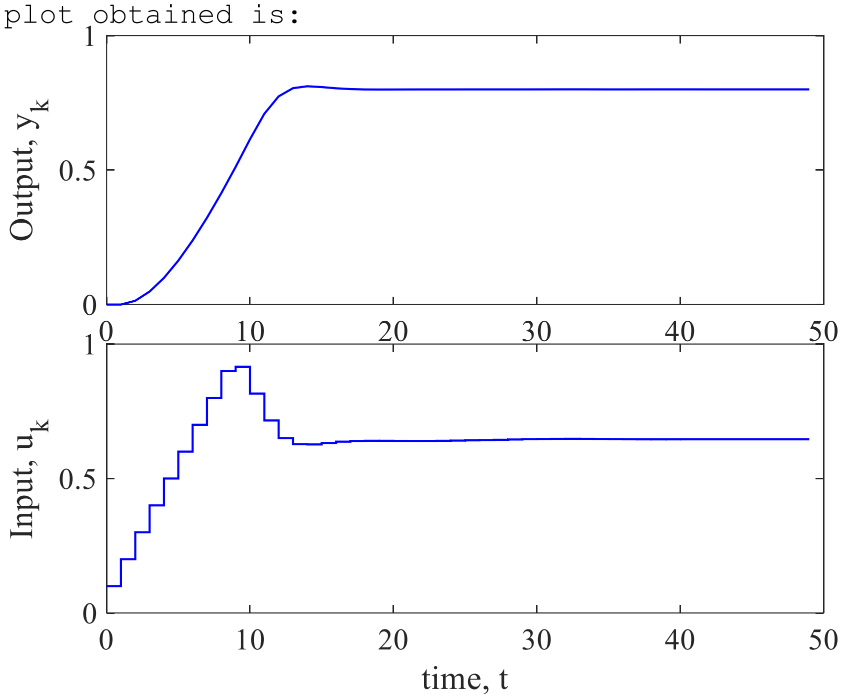
**Problem 1:** SISO DMC

**CODE**

```
%% Step response model
G=tf(1.25,[5 1],'inputdelay',1.4); h=1;
n=25;
[y,t]=step(G,0:h:n*h);
S=y(2:end);
clear G y t

%% Controller Parameters
ySP=0.8;          % Setpoint
m=4;             % Control horizon
p=10;            % Prediction horizon
Q=1;             % Output weight
R=0.1;          % Input weight
duMax=0.1;      % Input rate constraint
duMin=-duMax;
uMin=-1;       % Input constraint
uMax=1;

%% Initialization
uPrev=0.0;
Yk0=zeros(n,1);
maxTime=50;

Y_SAVE=zeros(maxTime+1,1);
T_SAVE=zeros(maxTime+1,1);
U_SAVE=zeros(maxTime,1);

%% Pre-compute Matrices
Su_col=S(1:p,:);
Im_col=ones(m,1);
bigSu=[];
bigIm=[];
for i=1:m
    bigSu=[bigSu,Su_col];
    bigIm=[bigIm,Im_col];
    Su_col=[0;Su_col(1:end-1,:)];
    Im_col=[0;Im_col(1:end-1)];
end
% For Objective
bigR=ones(p,1)*ySP;
GammaY=diag(ones(p,1)*Q);
GammaU=diag(ones(m,1)*R);
% For constraints
Im=eye(m);

% Pre-compute Hessian
Hess=bigSu'*GammaY*bigSu + GammaU;
% Pre-compute LHS of constraints
C_LHS=[Im;-Im;bigIm;-bigIm];
```

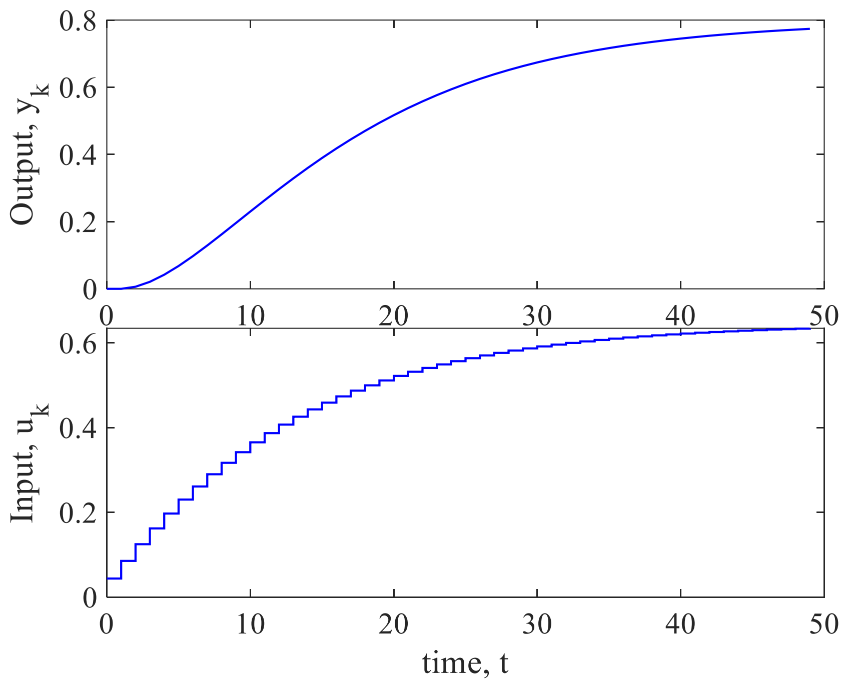
```
%% Implementation
for k=1:maxTime+1
    time=(k-1)*h;    % Current time

    % Calculate error
    if (k==1)
        YHAT=Yk0;
        err=0;
    else
        err=0;       % Replace this by "y_plant - YHat(1)"
    end

    % Calculate gradient
    predErr=YHAT(2:p+1)-bigR;
    grad=bigSu'*GammaY*predErr;
    % Calculate RHS of constraint
    cRHS=[repmat(duMax,m,1); repmat(-duMin,m,1)];
    cRHS=[cRHS; repmat( (uMax-uPrev),m,1)];
    cRHS=[cRHS; repmat(-(uMin-uPrev),m,1)];

    % Calculating current step
    % big_dU=-inv(Hess)*grad;
    big_dU=quadprog(Hess,grad,C_LHS,cRHS);
    du=big_dU(1);
    uk=uPrev+du;

    % Store results
    U_SAVE(k)=uk;
    uPrev=uk;

    % Implementing current step
    YHAT=[YHAT(2:end);YHAT(end)] + S*du;
    yk=YHAT(1);

    % Storing results
    Y_SAVE(k+1)=yk;
    T_SAVE(k+1)=k*h;
end

%% Plotting results
subplot('position',[0.12 0.58 0.8 0.4]);
plot(T_SAVE(1:maxTime),Y_SAVE(1:maxTime),'-b','linewidth',1);
ylabel('Output, y_k');
subplot('position',[0.12 0.12 0.8 0.4]);
stairs(T_SAVE(1:maxTime),U_SAVE(1:maxTime),'-b','linewidth',1);
ylabel('Input, u_k'); xlabel('time, t')

clear S n Y0 k du
```

The plot obtained is:



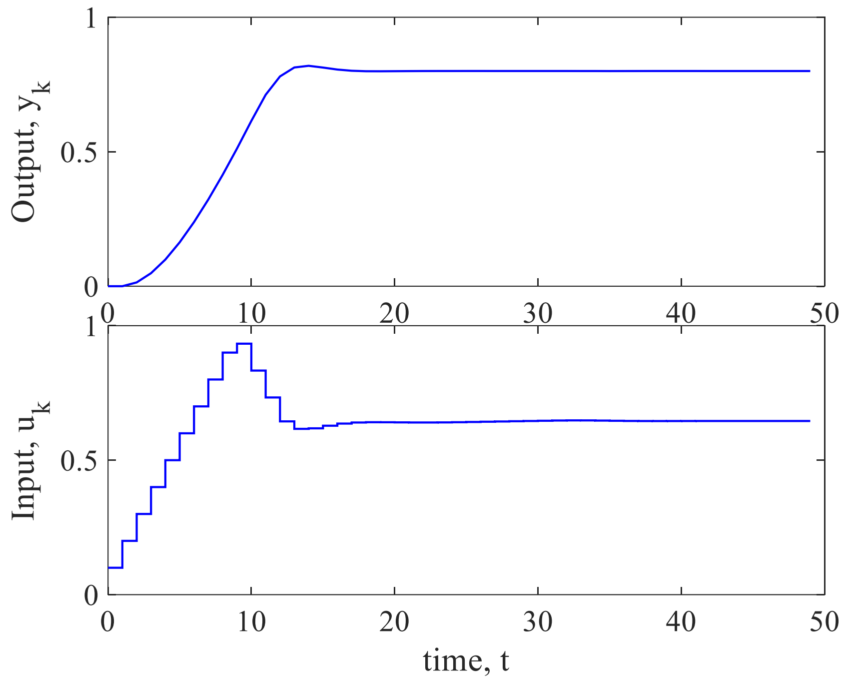

**Problem 2:** Effect of Tuning Parameters

2. R=100

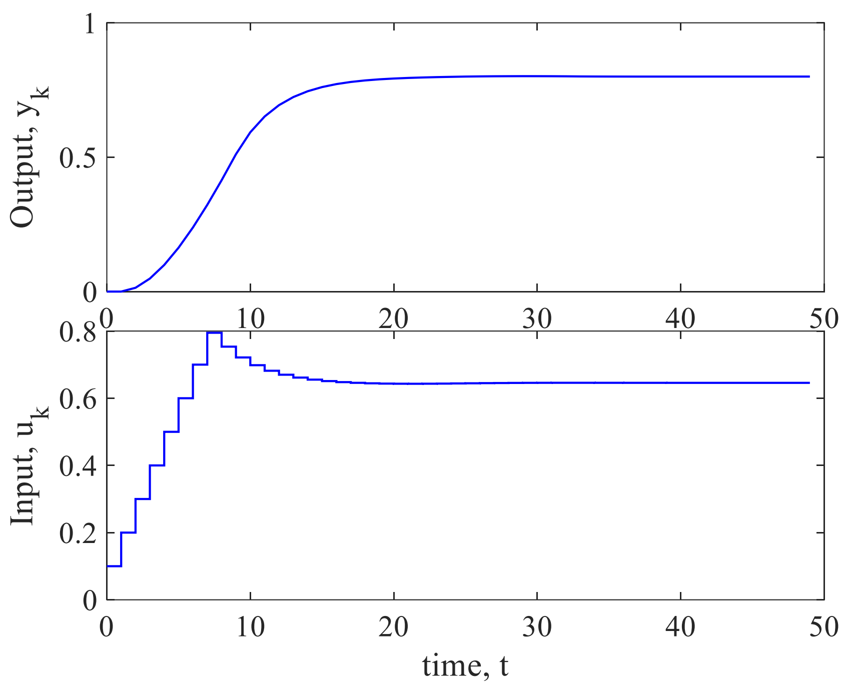



- Here the small input changes are made and due to more weight on the input changes in the objective function. This leads to a slow response compared to 1st case.

3. m=p=4




4. m=1,p=10




- In the case of m=1 only a single input move is considered each time. As a result the controller sees overshooting the set point undesirable. Thus it takes longer to attain set point compared to 1st case.
- In other words although there is a slight overshoot when m=4, but the response is faster compared to m=1. However, with m=1 the DMC algorithm is unable to infer this.

**Problem 3:** Extension to Measured Disturbance Case

**CODE**

```
%% Step response model
Gp=tf(1.25,[5 1],'inputdelay',1.4); h=1;      %Process transfer function
Gd=tf(0.2,[6 1],'inputdelay',0.7); h=1;      %Disturbance transfer function
n=25;
[yp,t]=step(Gp,0:h:n*h);
[yd,t]=step(Gd,0:h:n*h);
Sp=yp(2:end);
Sd=yd(2:end);

clear Gp Gd y t

%% Controller Parameters
ySP=0.8;          % Setpoint
m=4;             % Control horizon
p=10;            % Prediction horizon
Q=1;             % Output weight
R=0.1;           % Input weight
duMax=0.1;       % Input rate constraint
duMin=-duMax;
uMin=-1;         % Input constraint
uMax=1;

%% Initialization
uPrev=0.0;
Yk0=zeros(n,1);
maxTime=50;
D=[0.5*ones(maxTime+1,1)];
dD=D-[0; D(1:end-1)];

Y_SAVE=zeros(maxTime+1,1);
T_SAVE=zeros(maxTime+1,1);
U_SAVE=zeros(maxTime,1);

%% Pre-compute Matrices
Su_col=Sp(1:p,:);
Im_col=ones(m,1);
bigSu=[];
bigSd=Sd(1:p,:);
bigIm=[];
for i=1:m
    bigSu=[bigSu,Su_col];
    bigIm=[bigIm,Im_col];
    Su_col=[0;Su_col(1:end-1,:)];
    Im_col=[0;Im_col(1:end-1)];
end
% For Objective
bigR=ones(p,1)*ySP;
GammaY=diag(ones(p,1)*Q);
GammaU=diag(ones(m,1)*R);
% For constraints
Im=eye(m);

% Pre-compute Hessian
```

```
Hess=bigSu'*GammaY*bigSu + GammaU;
% Pre-compute LHS of constraints
C_LHS=[Im;-Im;bigIm;-bigIm];


%% Implementation
for k=1:maxTime
    time=(k-1)*h;   % Current time

    % Calculate error
    if (k==1)
        YHAT=Yk0;
        err=0;
    else
        err=0;      % Replace this by "y_plant - YHat(1)"
    end

    % Calculate gradient
    predErr=YHAT(2:p+1)+bigSd*dD(k)-bigR;
    grad=bigSu'*GammaY*predErr;
    % Calculate RHS of constraint
    cRHS=[repmat(duMax,m,1); repmat(-duMin,m,1)];
    cRHS=[cRHS; repmat( (uMax-uPrev),m,1)];
    cRHS=[cRHS; repmat(-(uMin-uPrev),m,1)];

    % Calculating current step
    % big_dU=-inv(Hess)*grad;
    big_dU=quadprog(Hess,grad,C_LHS,cRHS);
    du=big_dU(1);
    uk=uPrev+du;

    % Store results
    U_SAVE(k)=uk;
    uPrev=uk;

    % Implementing current step
    YHAT=[YHAT(2:end);YHAT(end)] + Sp*du + Sd*dD(k);
    yk=YHAT(1);

    % Storing results
    Y_SAVE(k+1)=yk;
    T_SAVE(k+1)=k*h;
end

%% Plotting results
subplot('position',[0.12 0.58 0.8 0.4]);
plot(T_SAVE(1:maxTime),Y_SAVE(1:maxTime),'-b','linewidth',1);
ylabel('Output, y_k');
subplot('position',[0.12 0.12 0.8 0.4]);
stairs(T_SAVE(1:maxTime),U_SAVE(1:maxTime),'-b','linewidth',1);
ylabel('Input, u_k'); xlabel('time, t')

clear Sp Sd n Y0 k du dD
```

The plot obtained is: