```
% <----  Main Program for quadratic optimal control implemented using Kalman predictor
%      on Continuously Stirred Tank Reactor (CSTR) System
%      Model: Discrete Linear model obtained through linearization of  mechanistic model
%      Plant Simulation : Discrete Linear model with arbitrary MPM
% ------------------------------------------------------------------------->

clear all ; clc ;  close all

global CSTR_mod ;      %  Global Data structure containing System related parameters

load CSTR_para         % Initialize CSTR_mod data structure and operating conditions
load CSTR_LinMod_I

% Nominal model used for used for designing observer and controller
% Following local variables are created only for improving readability of the program

n_st = dmod_lin.n_st ; n_op = dmod_lin.n_op  ;
n_ip = dmod_lin.n_ip ; n_ud = dmod_lin.n_ud ;
Xs = dmod_lin.Xs ; Ys =  dmod_lin.Ys  ;    % Steady state operating conditions
Us = dmod_lin.Us  ; Ws  = dmod_lin.Ws  ;
phy = dmod_lin.phy ; gama_u = dmod_lin.gama_u ;
gama_d = dmod_lin.gama_d ; C_mat = dmod_lin.C ;

% Note: It is possible to work directly with elements of dmod_lin object
% without requiring creation of above local variables

samp_T = dmod_lin.T ;       % Sampling interval
N_samples = 301 ;   % Number of samples in open loop simulation run

% Matrices for plant simulation: Model Plant Mismatch has been
% introduced deliberately here to demonstrate effectiveness of
% innovation bias approach. Multipliciation factors for introducing MPM
% (i.e. 0.8, 1.2 and 0.85) are chosen arbitrarily

phy_p = 0.8 * phy ;                  % Plant state transition matrix
gama_u_p = 1.2 * gama_u ;           % Plant input coupling matrix
gama_d_p = 0.85 * gama_d ;          % Plant disturbance coupling matrix

d_step = -0.2  ;       % Variable to introduce step disturbance
```

```
%----Initialization for absolute and dev state variables -------

% Create dummy arrays for simulation
% k'th column of these arrays corresponds to vector at k'th sampling instant
xk = zeros(n_st, N_samples) ;     % Matrices for saving deviation variables
uk = zeros(n_ip, N_samples) ;
yk = zeros(n_op, N_samples) ;
dk = zeros(n_ud, N_samples) ;

state_sigma = [ 0.01 ]' ;   % Generate state noise sequence for simulation
wk =  state_sigma * randn(n_ud, N_samples) ;
meas_sigma  = [ 0.1 ]';      % Generate state noise sequence for simulation
vk = meas_sigma * randn(1, N_samples) ;

xk(:,1) = [ 0.1 1 ]'  ;          % Initial deviation state in the plant (at k = 0)
yk(:,1) = C_mat * xk(:,1) + vk(1)   ;  %  Initial dev. Measurement (at k = 0)
dk(:,1) =  wk(:,1) ;

% Observer Initialization
xkpred = zeros(n_st, N_samples) ; %  Create Dummy matrices for estimated states
ek = zeros(n_op, N_samples) ;        % Innovation sequence
ek(:,1) = yk(:,1) - C_mat * xkpred(:,1) ;

% Steady State Kalman Estimator Design
Qd_mat = state_sigma.^2 ;
Q_mat =  gama_d * Qd_mat *  gama_d' ;  % Cov[w(k)]
R_mat = meas_sigma^2 ;                          % Cov[v(k)]
N_mat = zeros(n_st, n_op) ;             % Cross covariance E{wv'}

% Find Observer Gain matrix by solving steady state Riccati eqns.
dmod_CSTR = ss( phy,  gama_u, C_mat, zeros(n_op,n_ip), samp_T ) ;
[KEST, Lp_inf,Pk_pred_inf, Lc_inf, Pk_updt_inf] =  kalman(dmod_CSTR,Q_mat,R_mat,↙
N_mat) ;

% LQOC (Innovation Bias Formulation)  initialization
xsk = zeros(n_st, N_samples) ;     % Matrices for saving terget states
usk = zeros(n_ip, N_samples) ;
rk = zeros(n_op, N_samples) ;
```

```
ek_f = zeros(n_op, N_samples) ;          % Filtered Innovation sequence
phy_e = 0.9 * eye(n_op) ;                % Innovation filter parameter
phy_r = 0.85 * eye(n_op) ;               % Setpoint filter parameter

% Marices required in Target state Computation
Ku_mat = C_mat * inv(eye(n_st) - phy) * gama_u ;
Ke_mat = C_mat * inv(eye(n_st) - phy) * Lp_inf +eye(n_op) ;

% Compute Controller gain matrix by solving steady state Riccati eqns.
Wx = diag( [ 10 1 ] ) ;    % State weighting matrix
Wu =  diag( [0.1  1]) ;    % Error weighting matrix
[G_inf, S_inf, EigVal_CL] = dlqr(phy, gama_u, Wx, Wu) ;

% <------------- Closed  Loop Dynamic Simulation ----------------->

kT = zeros(N_samples,1) ;
kT(1) = 0 * samp_T ;   %  k = 1 corresponds to time = 0

for k = 2 : N_samples,
    k                              % Print sampling time on screen
    kT(k) = (k-1) * samp_T ;

    %   <---- Plant simulation form instnat (k-1) to (k) ---->
    %  Process simulation using discrete linear perturbation model with MPM ---->

    xk(:,k) =  phy_p * xk(:,k-1) + gama_u_p * uk(:,k-1) + gama_d_p * dk(:,k-1) ;
    yk(:,k) = C_mat * xk(:,k) + vk(:,k) ;

    % <-----Controller Calculations at k'th sampling instant ------>

    % Specify setpoint
    if ( k < 100 ), setpt = 0 ;
    elseif ( k >=101 ),  setpt = 5 ;    % Step chenge in setpoint
    end
    % Generate filtered setpoint trajectory
    rk(:,k) = phy_r * rk(:,k-1)+ (eye(n_op)-phy_r)* setpt ;

    %  Steady state Kalman Predictor computations from instant (k-1) to k
```

```
    xkpred(:,k) = phy * xkpred(:,k-1) + gama_u * uk(:,k-1) + Lp_inf * ek(:,k-1)  ;
    ek(:,k) =  yk(:,k) - C_mat *  xkpred(:,k)  ;    % Compute Innovation

   % Filtered innovation for innovation bias implementation
   ek_f(:,k) = phy_e * ek_f(:,k-1) + (eye(n_op) - phy_e) *  ek(:,k) ;

   % Compute target input and target states
   usk(:,k) = pinv(Ku_mat) * ( rk(:,k) - Ke_mat * ek_f(:,k) ) ;
   xsk(:,k) =  inv(eye(n_st) - phy) * ( gama_u * usk(:,k) +  Lp_inf * ek_f(:,k) ) ;

   % Innovation Bias controller implementation
   uk(:,k) = usk(:,k)  - G_inf * (xkpred(:,k) - xsk(:,k)  )  ;
   Uk_abs(:,k) = Us + uk(:,k) ;

   % <---- Specify distrbance input at k'th instant for plant simulation ---->
   if ( k < 200 ),   dk(:,k)=  wk(:,k-1)  ;
   elseif ( k >=201 ),  dk(:,k)= d_step + wk(:,k-1)  ;    % Step chenge in setpoint
   end
end     %  <---- End of Dynamic Simulation Loop ---->

 %  <---- Display simulation results graphically ---->

  Init_Graphics_Style ;      % Set parameters for graphics (Optional)
  figure(1),subplot(211), plot( kT, xk(1,:) ) ;
  xlabel('Sampling Instant'), ylabel('Conc.(mod/m3)'), title( 'Plant State') ;
  figure(1),subplot(212), plot( kT , xk(2,:), 'xr', kT, rk, 'g' ) ;
  xlabel('Sampling Instant'), ylabel('Temp.(K)'), title( 'Controlled Output') ;
   figure(2),subplot(311), stairs( kT , uk(1,:) ) ;
   xlabel('Sampling Instant'), ylabel('Coolent Flow'), title( 'Man. Input Perturbations') ;
   figure(2),subplot(312), stairs(kT , uk(2,:) ) ;
   xlabel('Sampling Instant'), ylabel('Inflow')
   figure(2),subplot(313), stairs( kT, dk ) ;
   xlabel('Sampling Instant'), ylabel('Inlet Conc. (mol/m3)'), title( 'Unmeas. Dist.↙
Perturbations') ;
```