```
% <---- Main Program for Kalman Filter Demo using
%      Continuously Stirred Tank Reactor (CSTR) System  --->

 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  % Load discrete linear model obtained through linearization

  % 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 = 61 ;   % Number of samples in open loop simulation run

  % <---- Program Initialization  ---->
  fprintf( '\n\n Continuously Stirred Tank Reactor (CSTR): Kalman Filtering Demo Program' )
  fprintf( '\n\n\t'),
  mod_type = input('Plant Simulation using (0) : Linear (1) : Nonlinear model? ')  ;

%----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) ;
```

```
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) = [ 0.4 5 ]'  ;          % Initial deviation state in the plant (at k = 0)
yk(:,1) = C_mat * xk(:,1) + vk(1)   ;  %  Initial dev. Measurement (at k = 0)

%    Generation of Random Binary Input Sequences for System  Identification

 ip1 = idinput( N_samples, 'rbs', [0 0.05] ) ;
 ip2 = 0.1 * idinput( N_samples,'rbs', [0 0.05] ) ;
 uk = [ ip1' ; ip2' ]  ;

%  Matrices for saving Absolute variables
 Xk_abs = zeros(n_st, N_samples) ;
Uk_abs = zeros(n_ip, N_samples) ;
Yk_abs = zeros(n_op, N_samples) ;
Wk_abs = zeros(n_ud, N_samples) ;

Xk_abs(:,1) = Xs + xk(:,1)  ;               % Plant: Initial abs. state (at k = 0)
Yk_abs(:,1) = C_mat * Xs + yk(:,1) ;     %  Initial abs Measurement (at k = 0)
Uk_abs(:,1) = Us + uk(:,1) ;
Wk_abs(1) = Ws + wk(1) ;

% Observer initialization

Qd_mat = state_sigma.^2 ;
Q_mat =  gama_d * Qd_mat *  gama_d' ;  % Cov[w(k)]
R_mat = meas_sigma^2 ;                        % Cov[v(k)]

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

Pk_updt = 10 * Q_mat  ;                          % Initialization of P(0|0)
Pk_pred = 10 * Q_mat  ;
```

```
Xkhat_abs(:,1) = Xs + xkhat(:,1)  ;                          %  Observer: Initial states (at k = 0)

est_error = zeros( n_st, N_samples) ;
est_error(:,1) = xk(:,1)-xkhat(:,1) ;

Pk_norms = zeros( N_samples+1, 2) ;
Pk_norms(1,:) = [ norm(Pk_pred) norm(Pk_updt) ] ;

% <------------ Open 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) ---->
  %   Integrate the model equations for U(k-1) and W(k-1) to compute X(k) and Y(k) ---->

  if ( mod_type == 0 )     % Process simulation using discrete linear perturbation model

     xk(:,k) = phy * xk(:,k-1) + gama_u * uk(:,k-1) + gama_d * wk(:,k-1) ;
     yk(:,k) = C_mat * xk(:,k) + vk(:,k) ;

     Xk_abs(:,k) = Xs + xk(:,k) ;                 % Save simulation data in absolute varriables
     Yk_abs(:,k) = Ys + yk(:,k) ;

  else                          % Process simulation using nonlinear ODE model

     CSTR_mod.Fc  = Uk_abs(1,k-1)  ;    % Assign manipulated and disturbance inputs
     CSTR_mod.F  =  Uk_abs(2,k-1)  ;
     CSTR_mod.Cao = Wk_abs(k-1)  ;

      %  Runge Kutta integration  over interval [(k-1)T, kT] using MATLAB ODE solver

     [t,Xt] = ode45( 'CSTR_Dynamics', [0 samp_T] , Xk_abs(:,k-1) ) ;
     Xk_abs(:,k) = Xt( length(t),:)' ;               % State at instnat (k+1)
```

```
    Yk_abs(:,k) = C_mat * Xk_abs(:,k) + vk(:,k) ;   % Measured Output at insthat (k)
    xk(:,k) =  Xk_abs(:,k)  - Xs ;        % Generate Perturbation variables
    yk(:,k) = Yk_abs(:,k) - Ys ;

  end

  %  <---- Kalman Filter computations from instant (k-1) to k ---->

  % Prediction step : mean and covariance computations

  xkpred(:,k) = phy * xkhat(:,k-1) + gama_u * uk(:,k-1)  ;   % Compute xhat(k|k-1)
   Pk_pred = phy * Pk_updt * phy' +  Q_mat  ;            % Compute P(k|k-1)

   % Kalman gain computations

  Vk_mat = R_mat + C_mat * Pk_pred * C_mat'  ;      % Innovation covariance
  Lck_mat =  Pk_pred * C_mat' * inv( Vk_mat ) ;      % Kalman gain matrix Lc(k)

  % Correction step: mean and covariance update

  ek(:,k) =  yk(:,k) - C_mat *  xkpred(:,k)  ;              % Compute Innovation
  xkhat(:,k) = xkpred(:,k) + Lck_mat * ek(:,k)  ;           % Updated mean xhat(k|k)
  Pk_updt = (eye(n_st) - Lck_mat * C_mat) * Pk_pred ;  % Compute P(k|k)

  Xkhat_abs(:,1) = Xs + xkhat(:,1)  ;          % Create absolute variables for display

  % Save estimation error results at instant k

  est_error(:,k) = xk(:,k)-xkhat(:,k) ;
  Pk_norms(k,:) = [ norm(Pk_pred) norm(Pk_updt) ] ;

  % <-----Specify Inputs at k'th sampling instnat ------>

  Uk_abs(:,k) = Us + uk(:,k) ;
  Wk_abs(k) = Ws + wk(k) ;
end

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

```
Init_Graphics_Style ;     % Set parameters for graphics (Optional)

figure(1),subplot(211), plot( kT, xk(1,:), kT, xkhat(1,:),'xr' ) ;
xlabel('Sampling Instant'), ylabel('Conc.(mod/m3)'), title( 'State Perturbations') ;
figure(1),subplot(212), plot( kT , xk(2,:) , kT, xkhat(2,:),'xr' ) ;
xlabel('Sampling Instant'), ylabel('Temp.(K)') ;

figure(2),subplot(211), plot( kT, est_error(1,:),'xr-' ) ;
xlabel('Sampling Instant'), ylabel('Conc.(mod/m3)'), title( 'State Estimation Errors') ;
figure(2),subplot(212), plot( kT , est_error(2,:),'xr-' ) ;
xlabel('Sampling Instant'), ylabel('Temp.(K)') ;

 figure(3), plot( kT , ek,'xr-' ) ;
 xlabel('Sampling Instant'), ylabel('e(k)'), title('Innovation Sequence') ;

 figure(4),subplot(211), stairs( kT , uk(1,:) ) ;
 xlabel('Sampling Instant'), ylabel('Coolent Flow'), title( 'Man. Input Perturbations') ;
 figure(4),subplot(212), stairs(kT , uk(2,:) ) ;
 xlabel('Sampling Instant'), ylabel('Inflow')

 figure(5),stairs( kT, wk ) ;
 xlabel('Sampling Instant'), ylabel('Inlet Conc. (mol/m3)'), title( 'Unmeas. Dist.↙
Perturbations') ;
```