```
% <---- Main Program for Development of black box linear perturbation
%      model for Continuously Stirred Tank Reactor (CSTR) System
%      In particular, 2,nd order ARMAX model is identified from dynamic
%      data generated by injecting PRBS perturbations into the system ---->

 clear all ; clc ;  close all
 Init_Graphics_Style ;

  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 for control purpose

  N_int = 100 ;              % Parameters of Euler integration
  int_T = samp_T / N_int ;  % Interval for Euler integration

  N_samples = 501 ;   % Number of samples in open loop simulation

  % <---- Program Initialization  ---->

 fprintf( '\n\n\t')
 mod_type = input('Process Simulation (0) : Linear (1) : Nonlinear ? ')  ;

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

% Create dummy working arrays for simulation
% k'th column of these arrays corresponds to vector at k'th sampling instant
```

```
% Generate dummy matrices for saving deviation variables
xk = zeros(n_st, N_samples) ;
uk = zeros(n_ip, N_samples) ;
yk = zeros(n_op, N_samples) ;

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

yk(:,1) = C_mat * xk(:,1) + vk(1)   ;  %  Initial dev. Measurement (at k = 0)

%    Generation of Random Binary Input Sequences for
%    open loop simulation and System  Identification

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

%  Dummy matrices for 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) ;

% ---------------- 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 ;
```

```
%  <---- Process simulation form instnat (k-1) to (k):
%  Integrate the model equations for Uc(k-1) and Wc(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) ;

   % Explicit Euler integration over interval [(k-1)T, kT]
   X0 = Xk_abs(:,k-1) ;  ti = kT(k-1) ;
   for i = 1 : N_int
      Xf = X0 + int_T * CSTR_Dynamics( ti, X0 ) ;
      X0 = Xf ;
      ti = ti + int_T ;
   end
  Xk_abs(:,k) = Xf ;

 %  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+1)
   xk(:,k) =  Xk_abs(:,k)  - Xs ;        % Generate Perturbation variables
   yk(:,k) = Yk_abs(:,k) - Ys ;

   CSTR_mod.Ca = Xk_abs(1) ;  CSTR_mod.T = Xk_abs(2) ;
end

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

Uk_abs(:,k) = Us + uk(:,k) ;
```

```
   Wk_abs(k) = Ws + wk(k) ;

end

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

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

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

figure(3),stairs( kT, wk ) ;
xlabel('Sampling Instant'), ylabel('Inlet Conc. (mol/m3)')
title( 'Unmeasured Disturbance Perturbations') ;

% <---- MISO ARMAX model identification ----->

% split data set into identification and validation sets

cstr_id_data = iddata( yk(:,1:400)', uk(:,1:400)', samp_T ) ;
cstr_val_data = iddata( yk(:,400:501)', uk(:,400:501)', samp_T ) ;

na = 2 ;          % order of A polynomial
nb = [ 2 2 ] ;    % order of B polynomials w.r.t. man. inputs
nc = 2 ;          % order of C polynomial
nd = [ 1 1 ] ;    % time delays w.r.t. man. inputs
cstr_armax_2 = armax( cstr_id_data, [ na nb nc nd ] )

% Analysis of model residuals / innovations {e(k)}
ek_obj = pe(  cstr_armax_2, cstr_id_data )  ;
ek = get( ek_obj, 'OutputData') ;
figure(4), plot( ek )
title('Model Residuals / Innovations'), xlabel('Sampling Instant'), ylabel('e(k)')

% Auto-correlation and cross correlation analysis
```

```
figure(5), cra( [ ek ek], 20, 0, 1 )
figure(6), subplot(211),cra( [ ek uk(1,1:400)'], 20, 0, 1 )
figure(6), subplot(212),cra( [ ek uk(2,1:400)'], 20, 0, 1 )

% Model validation
figure(7), compare( cstr_val_data, cstr_armax_2 )

% Generate observable cannonical state realization of
% the identified 2.nd order ARMAX model

dmod_id = ss( cstr_armax_2 )

save CSTR_IdMod.mat cstr_armax_2 dmod_id
```