```
clear all
clc

% Q2-------------------------------------------------------------------

% reactor_model_ode_rhs.m is used to represent the right-hand side of
% the non-linear differential equations.

% reactor_nonlinear_sfcn.m is used to represent the non-linear model
% as an S-Function in Simulink for simulation purposes.

% reactor_nonlinear_vs_linear_model_simulation.mdl is used to simulate
% the non-linear vs linear model system behavior.

% reactor_observer.mdl is used to simulate the closed-loop response
% with the designed Luenberger observer.

% reactor_feedback_full_state.mdl is used to simulate the closed-loop
% response with the designed state-feedback controller.

% reactor_feedback_observer.mdl is used to simulate the closed-loop
% response with the designed state-feedback controller coupled with
% the designed Luenberger observer.

%-------------------------------------------------------------------

% Parameters
C_A_in = 5.1;       % Component A Inlet Concentration [kmol / m3]
V = 0.01;           % Reactor Volume [m3]
k_01 = 2.145e10;    % Pre-exponential Factor - First Reaction [1/min]
k_02 = 2.145e10;    % Pre-exponential Factor - Second Reaction [1/min]
E_R1 = 9758.3;      % Reaction Activation Energy - First Reaction [K]
E_R2 = 9758.3;      % Reaction Activation Energy - Second Reaction [K]
deltaH_R1 = -4200;  % Heat of Reaction - First Reaction [kJ / kmol]
deltaH_R2 = -11000; % Heat of Reaction - Second Reaction [kJ / kmol]
T_in = 387.05;      % Inlet Temperature [K]
rho = 934.2;        % Liquid Density [kg / m3]
cp = 3.01;          % Heat Capacity of the Reaction Medium [kJ / kg*K]
cp_j = 2.0;         % Heat Capacity of the Jacket Medium [kJ / kg*K]
m_j = 5.0;          % Coolant Mass [kg]
kA = 14.448;        % Heat Transfer Coefficient [kJ / min*K]

%-------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% Q2 - 3)  Calculate the equilibrium points of the system for the
% given steady-state inputs. Fr_ss = 0.002365 [m3 / min]
% Qj_ss = 18.5583 [kJ / min]

Fr_ss = 0.002365; Qj_ss = 18.5583;

x0_guess = [1.0 1.0 350 350]; % initial guess for the equilibrium
% points of the state variables
u_s = [Fr_ss Qj_ss]; % steady-state inputs

% Configure the non-linear solver for a high accuracy
% (i.e. low tolerance)

options = optimset('TolFun', sqrt(eps), ...
                   'MaxFunEvals',1e6, 'MaxIter', 10000);

[x_ss, fun_val, flag_res, o, j] = fsolve(@reactor_model_ode_rhs,
 x0_guess, options, u_s);

% Equilibrium Points
C_A_steady = x_ss(1); % 1.6329 [kmol / m3]
C_B_steady = x_ss(2); % 1.1101 [kmol / m3]
T_R_steady = x_ss(3); % 398.6581 [K]
T_J_steady = x_ss(4); % 397.3736 [K]

%-------------------------------------------------------------------------

% Q2 - 4) Linearize the system around the computed equilibrium
% point(s) in part(2). Put the linearized system in the standard
% state space representation, assume that all the states are measured.
% Check the local stability of the computed equilibrium point(s).

% Calculation of Partial Derivatives to Plug in Steady-States
% for the Corresponding Jacobians

df1_dx1 = (-Fr_ss / V) - (k_01*exp(-E_R1 / T_R_steady));
df1_dx2 = 0;
df1_dx3 = ((-k_01*C_A_steady*E_R1) / (T_R_steady^2)) * exp(-E_R1 /
 T_R_steady);
df1_dx4 = 0;
df1_du1 = (C_A_in - C_A_steady) / V;
df1_du2 = 0;

df2_dx1 = k_01*exp(-E_R1 / T_R_steady);
df2_dx2 = (-Fr_ss / V) - (k_02*exp(-E_R2 / T_R_steady));
df2_dx3 = ((k_01*exp(-E_R1 / T_R_steady)*C_A_steady*E_R1) / ...
    (T_R_steady^2)) - ((k_02*exp(-E_R2 /
 T_R_steady)*C_B_steady*E_R2)...
    / (T_R_steady^2));
df2_dx4 = 0;
df2_du1 = -C_B_steady / V;
df2_du2 = 0;
```

```
df3_dx1 = -(k_01*exp(-E_R1 / T_R_steady)*deltaH_R1) / (rho*cp);
df3_dx2 = -(k_02*exp(-E_R2 / T_R_steady)*deltaH_R2) / (rho*cp);
df3_dx3 = (-Fr_ss / V) + (-(k_01*exp(-E_R1 / T_R_steady)*...
    C_A_steady*deltaH_R1*E_R1) / (rho*cp*(T_R_steady^2))) +...
    (-(k_02*exp(-E_R2 / T_R_steady)*C_B_steady*deltaH_R2*E_R2) /...
    (rho*cp*(T_R_steady^2))) - (kA / (rho*cp*V));
df3_dx4 = kA / (rho*cp*V);
df3_du1 = (T_in - T_R_steady) / V;
df3_du2 = 0;

df4_dx1 = 0;
df4_dx2 = 0;
df4_dx3 = kA / (m_j*cp_j);
df4_dx4 = -kA / (m_j*cp_j);
df4_du1 = 0;
df4_du2 = -1 / (m_j*cp_j);

% Linearized State Space Representation

% x(t) = x_ss + delta_x(t);
% u(t) = u_ss + delta_u(t);

% A = J_fx(xs, us); B = J_fu(xs, us);

% delta_x_dot(t) = A*delta_x(t) + B*delta_u(t);
% y = C*delta_x(t) + D*delta_u(t);

A_linearized = [df1_dx1 df1_dx2 df1_dx3 df1_dx4;...
    df2_dx1 df2_dx2 df2_dx3 df2_dx4;...
    df3_dx1 df3_dx2 df3_dx3 df3_dx4;...
    df4_dx1 df4_dx2 df4_dx3 df4_dx4];

B_linearized = [df1_du1 df1_du2;...
    df2_du1 df2_du2;...
    df3_du1 df3_du2;...
    df4_du1 df4_du2];

C_linearized = eye(4); % all states are measured
D_linearized = 0;

sys_ss = ss(A_linearized, B_linearized, C_linearized, D_linearized);
poles_linearized = eig(sys_ss);
% All poles have negative real parts, hence the all the
% equilibrium points are stable.

%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------

% Q2 - 5) Check the validity of the linearization by simulating the
% linearized system against the original model at the equilibrium
% point(s) for a delta_u = +- 10% of the steady-state input.

x0 = [0; 0; 387.05; 387.05]; % initial conditions for C_A,C_B,T_R,T_J

% Simulate the system to check the validity of the linearized
% model with 2 different inputs.
% Use Nonlinear_vs_Linear_Model_Simulation.mdl

u_s_part4_1 = 0.9*u_s;
u_s_part4_2 = 1.1*u_s;

%-------------------------------------------------------------------

% Q2 - 6) Check the operability of the linearized system(s). What is
% the dimension of the steady-state subspace? What do you infer from
% the matrices V and U (in terms of the input and output directions)?
% And how to explain this physically?

M = -inv(A_linearized) * B_linearized ;
% A*x = B*u => x = -inv(A)*B*u => x = M*u;
operability = rank(M); % rank(M) = 2

[U,S,V] = svd(M);
eig_MT_M = [S(1,1)^2 S(2,2)^2]; % [ 2.8971e+07 0.0043 ]

% No eigen value of M' * M is zero so the matrix M is
% full-rank. This means that it is possible to operate
% the system at steady-state conditions.

gamma  = S(1,1) / S(2,2);
% Condition number = 8.1944e+04 < 1e+5 (operability matrix M is not
% ill-conditioned)

%-------------------------------------------------------------------

% Q2 - 7) Assuming that all of the states are measured, design a state
% feedback controller to regulate the non-linear system around
% the selected equilibrium point. Check the condition needed to
% assign the closed loop poles freely. Give a reason for your
% selection of the closed-loop poles. Simulate the closed-loop
% with the non-linear system at the following 2 initial conditions
% x01 = [ 0.9*C_A_ss, 0.9*C_B_ss, T_R_ss - 10, T_J_ss -10]
% x02 = [ 1.1*C_A_ss, 1.1*C_B_ss, T_R_ss + 10, T_J_ss + 10]

% The condition needed to assign the closed-loop poles freely
% is decided by the Kalman Criterion for controllability.
% (A, B) is controllable if rank [B AB ... A^n-1 * B] = n
```

```
ctb_kalman = [B_linearized, A_linearized*B_linearized,...
    (A_linearized^2)*B_linearized, (A_linearized^3)*B_linearized];
controllability = rank(ctb_kalman); % rank(ctb_kalman) = 4

% Controllability matrix has full-rank (= n) which means the system
% is controllable.

% Linearized system's original eigen values are at -1.9734, -0.9058,
% -0.4637, -0.1205

% For controller design, we want fast converging, well-damped,
% not too fast eigen values. Also, they shouldn't be placed at
% the same spot since it causes sensitivity to errors.
% Therefore, we are going to choose our eigen values to be
% somewhat faster than the original ones. The decision for
% which eigen values to use was a process of trial and error.

new_eig_values_K = [-0.9; -1.5; -2; -3];

K = place(A_linearized, B_linearized, new_eig_values_K);
% Computes the K matrix required to place the eigen values of the
% controlled system (A-BK) at desired eigen values

A_c = [(A_linearized - B_linearized*K)]; % n x n
B_c = [B_linearized]; % n x p
C_c = eye(4); % y x n, all states are measured
D_c = 0; % y x p
sys_c = ss(A_c, B_c, C_c, D_c);
poles_controller = eig(sys_c);

% Simulate the system to check the non-linear system behavior
% with the controller with 2 initial conditions.
% Use reactor_feedback_full_state.mdl

% Using this state-feedback controller, the rate of convergence
% has been increased and the system reaches the equilibrium point
% under 10-15 seconds.

%-----------------------------------------------------------------------

% Q2 - 8) Now assume that the concentrations C_A and C_B are
% not measured, and the temperatures of the jacket are the only
% available measurements. Design a Luenberger observer to estimate
% the unmeasured states. Show the convergence of the estimated
% concentrations to the true states of the non-linear system by
% simulation from different initial conditions. Show the results
% using same initial conditions as task (7).

C_est = [0 0 0 0; 0 0 0 0; 0 0 1 0; 0 0 0 1]; % only T_R and T_J are
 measured

obs_kalman = [C_est; C_est*A_linearized;...
    C_est*(A_linearized^2); C_est*(A_linearized^3)];
```

```
observability = rank(obs_kalman); % rank(M) = 4

% Observability matrix has full-rank (= n) which means the L
% matrix can be chosen such that eig(A - L*C) take arbitrary
% assigned values and the observer error converges to zero
% with the chosen dynamics.((A-LC) => asymptotically stable)

% In practice, we want the convergence to be faster than the
% evolution of true states. Thus, we are going to choose our
% new eigen values for the (A-LC) matrix to be faster then the
% original ones. The decision for which eigen values to use was
% a process of trial and error.

new_eig_values_L = [-0.3; -0.5; -0.7; -0.9];

L = place(A_linearized', C_est', new_eig_values_L)';

% (A_linearized - L*C_est);

poles_observer = eig(A_linearized - L*C_est);

% Simulate the system to check the non-linear system behavior
% with the observer with 2 initial conditions.
% Use reactor_observer.mdl

% Since the system is observable, unmeasured states can be
% derived from the output using a Luenberger observer. Using
% this observer, unmeasured states can be estimated correctly
% within 10-15 seconds.(estimation error goes to zero in 10-15
 seconds).

%-----------------------------------------------------------------------

% Q2 - 9) Simulate the non-linear system with the observer-based
% feedback controller. Test the closed-loop with the non-linear
% system at the same 2 initial conditions from task (7) and compare
% between the simulation results with the state feedback and with
% observer-based state feedback in terms of the closed-loop
 performance.

% Simulate the system to check the non-linear system behavior
% with the controller coupled with the observer with 2 initial
% conditions. Use reactor_feedback_observer.mdl

% When we compare the state-feedback controller's separate closed-loop
% performance with our coupled system, it can be seen that the rate of
% convergence is approximately the same (maybe slightly worse).
% However, the state estimation of the observer alone is slightly
% worse than our coupled system. In conclusion, using this
% state-feedback controller coupled with this observer, the rate of
% convergence has been increased and the system states are estimated
% correctly under 10-15 seconds. The system reaches the equilibrium
% point also under 10-15 seconds.
```

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the value of the function tolerance, and
the problem appears regular as measured by the gradient.
```

Published with MATLAB® R2020b