CTA Assignment

First Name: Mehmet Batu

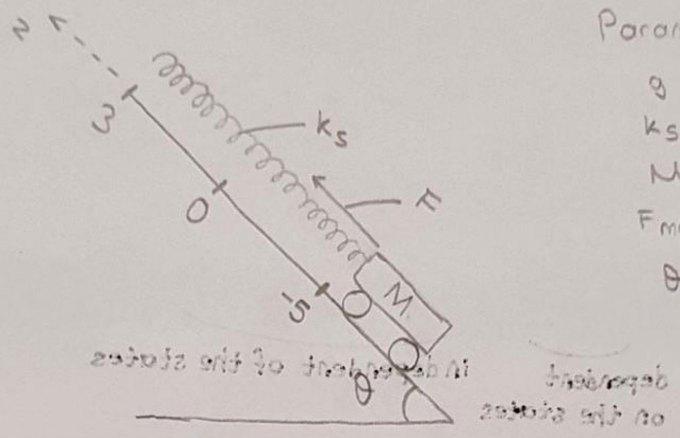
Last Name: Özmeteler

Matriculation Number: 230306

Q1

CTA Assignment WS20/21

1 -



Parameters

$g = 10\ m/s^2$

$k_s = 3\ N/m$

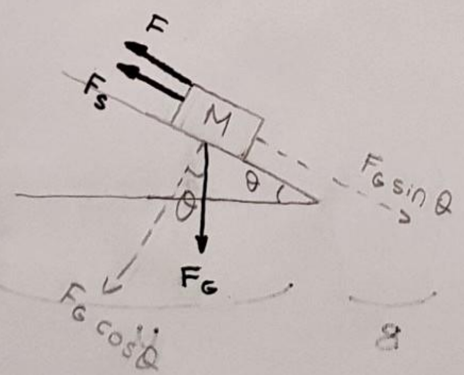
$M = 15\ Kg$

$F_{max} = 115\ N$

$\theta = 30°$

1)

- Free-body diagram of mass M



* First Law of Motion

$$\sum_{i=1}^{n} F_i = 0$$

Sum off all forces acting on a static object equals to zero.

* Second law of Motion

Linear Motion :

$$\sum_{i=1}^{n} F_i = m\,a \ , \quad a = \ddot{x} \ , \quad v = \dot{x}$$

- Equations and Force Balance

$$F \ , \quad F_s = k_s \Delta x \ , \quad F_G \sin\theta = M g \sin\theta$$

F is a constant input, a linear force provided by the motor. $F_G \sin\theta$ is also constant and doesn't depend on the state of the system.

* $M a = F - F_s - F_G \sin\theta$

* States of the system are the velocity and the displacement of the object since those are the dynamics which change over time.

$x_1 = z$, $x_2 = \dot{z}$, $\dot{x}_1 = x_2 = \dot{z}$, $\dot{x}_2 = \ddot{z}$

2) $M\ddot{z} = F - F_s - F_g \sin\theta$

$M\ddot{z} = F - k_s(z - (-5)) - Mg\sin\theta$

$$\ddot{z} = \underbrace{-\frac{k_s}{M}z}_{\text{dependent on the states}} \underbrace{- \frac{5k_s}{M} + \frac{F}{M} - g\sin\theta}_{\text{independent of the states}}$$

3) $\dot{x} = Ax + Bu \qquad y = Cx + Du$

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} z \\ \dot{z} \end{pmatrix} \qquad \dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \dot{z} \\ \ddot{z} \end{pmatrix}$$

$$\underbrace{\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix}}_{\dot{x}} = \underbrace{\begin{pmatrix} 0 & 1 \\ -\frac{k_s}{M} & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x} + \underbrace{\begin{pmatrix} 0 \\ 1/M \end{pmatrix}}_{B} \underbrace{(F - 5k_s - Mg\sin\theta)}_{u}$$

$$y = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x} + \underbrace{0}_{D} \underbrace{(F - 5k_s - Mg\sin\theta)}_{u}$$

4) $A = \begin{pmatrix} 0 & 1 \\ -1/5 & 0 \end{pmatrix}$ $B = \begin{pmatrix} 0 \\ 1/15 \end{pmatrix}$ $C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ $D = 0$

$u = F - 90$ $\qquad x = \begin{pmatrix} z \\ \dot{z} \end{pmatrix}$

5) (A, B) is controllable $\iff$ $\text{rank}\,[\overbrace{B \;\; AB \; \dots \; A^{n-1}B}^{\text{controllability matrix}}] = n$

$AB = \begin{pmatrix} 1/15 \\ 0 \end{pmatrix}$ $\qquad \text{rank}\begin{pmatrix} 0 & 1/15 \\ 15 & 0 \end{pmatrix} = 2 = n$

The system is controllable. ✓

6) $\lambda_1 = -0,8 \quad \lambda_2 = -1$

$(s+0,8)(s+1) = s^2 + 1,8s + 0.8$ **(desired characteristic poly.)**

$$\det(sI - (A+BK)) = \left| \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} - \underbrace{\left\{ \begin{pmatrix} 0 & 1 \\ -1/5 & 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1/15 \end{pmatrix} (k_1 \;\; k_2) \right\}}_{\begin{bmatrix} 0 & 1 \\ -1/5 + \frac{k_1}{15} & \frac{k_2}{15} \end{bmatrix}} \right|$$

$$= \begin{vmatrix} s & -1 \\ +1/5 - \frac{k_1}{15} & s - \frac{k_2}{15} \end{vmatrix}$$

$$= \left(s^2 - \frac{s k_2}{15}\right) - \left(-\frac{1}{5} + \frac{k_1}{15}\right) = s^2 - \frac{k_2}{15} s - \frac{k_1 - 3}{15}$$

$k_2 = -27, \quad k_1 = -9$

7) The closed-loop response of the system is plotted in MATLAB with closed-loop poles $\lambda_1 = -0{,}8$, $\lambda_2 = -1$ over a time period of 20 seconds.

8) From the closed-loop response plots of the states and the input, it can be clearly seen that the controlled robot reaches equilibrium position ($z = 0$) and velocity ($\dot{z} = 0$) at about 6-8 seconds. For the rest of the simulation, states are stable. The input force at the beginning is 135 N and reaches equilibrium at $F = 90$ N. However, the maximum amount of force that the motor can provide is $F_{max} = 115$ N meaning that due to the amount of force required to move the robot ($135\text{ N} > F_{max}$), with these chosen closed-loop poles, Design (I) is problematic.

9) To mitigate the problem that results from Design (I), we need to choose new eigen values such that input force is smaller than $F_{max} = 115$ N. The rate of convergence towards equilibrium point for Design (I) seemed to be decent. Therefore, we just rescale the Design (I) eigen values with a factor of 0.75 to reduce the input force required to move the robot. For faster convergence, more force is required hence we sacrifice convergence speed for less input force.

$$\lambda_1 = 0.75 \times -0{,}8 = -0{,}6 \quad , \quad \lambda_2 = 0{,}75 \times -1 = -0{,}75$$

$$(s + 0{,}75)(s + 0{,}6) = s^2 + 1{,}35\,s + 0{,}45$$

Using the calculated characteristic polynomial, with k_1 and k_2 plugged in from Task (6):

$$\det(sI - (A + BK)) = s^2 - \frac{k_2}{15}\,s - \frac{k_1 - 3}{15}$$

$$k_1 = -3{,}75 \quad , \quad k_2 = -20{,}25$$

10) The closed-loop response of the system is plotted in MATLAB with closed-loop poles $\lambda_1 = -0.6$, $\lambda_2 = -0.75$ over a time period of 20 seconds.

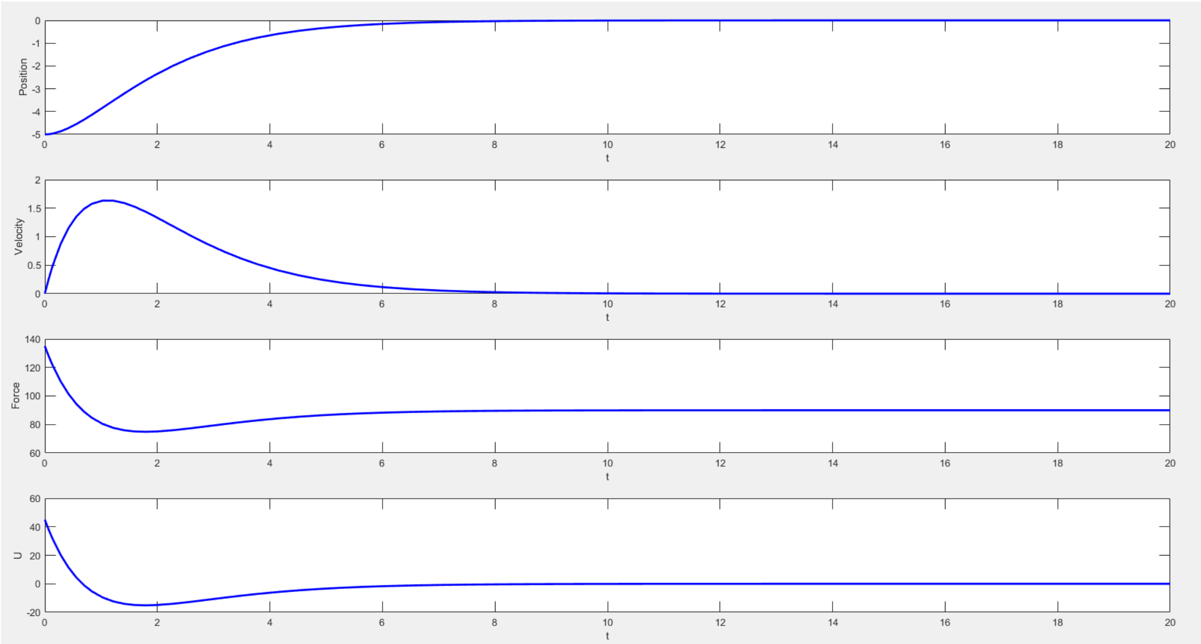
11) The design (II) should move the robot to position $z = 0$ on the inclined plane and the robot shall remain there afterwards. Therefore we plug $z = 0$ and $\ddot{z} = 0$ in the main differential equation:

$$M\ddot{z} = F_{ss} - k_s(z - (-5)) - Mg\sin\theta$$

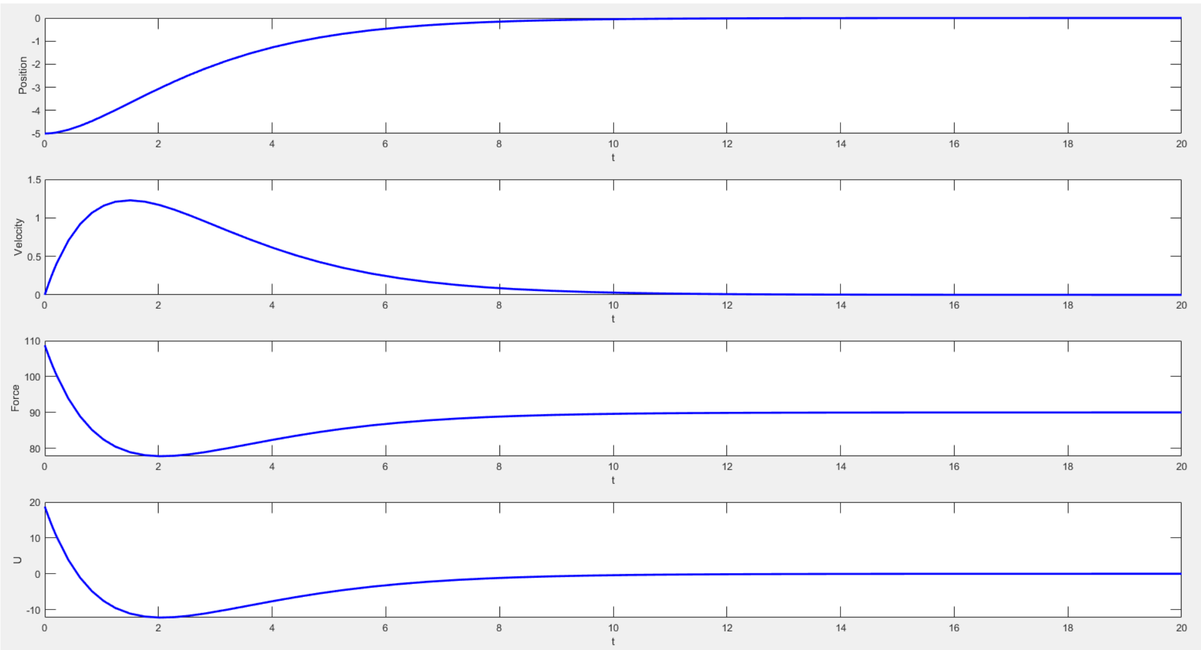
$$F_{ss} = k_s z + 5k_s + Mg\sin\theta$$

$$F_{ss} = 5k_s + Mg\sin\theta = 90N$$

When compared with the MATLAB plot, it can be seen that both values are equal. The input force (F) plot for design (II) in MATLAB reaches (F = 90 N) equilibrium point at about 8-10 seconds.



Closed-Loop Plots for Position, Velocity, Force and U respectively for Design (I)

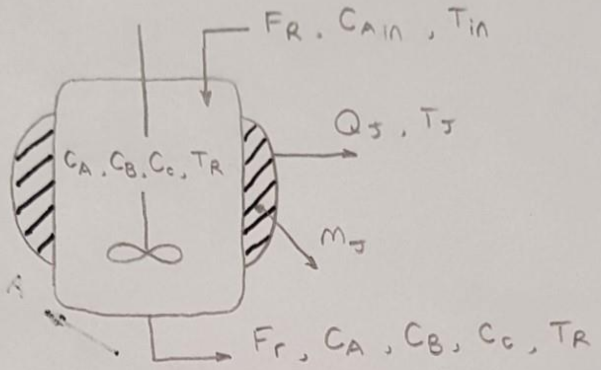


Closed-Loop Plots for Position, Velocity, Force and U respectively for Design (II)

Q2

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

2 -



- The feed inlet flow only contains component A.
  (A is the only input with flow rate F_R . concentration $C_{A,in}$ and temperature T_{in})
- The volume of the reactor is constant at all times.
- The reaction mixture is completely filled with liquid and ideally mixed. ($C_{A\,out} = C_A$, $C_{B\,out} = C_B$. $C_{C\,out} = C_C$. $T_{R\,out} = T_R$)

* V will be used with concentration to get the mole balances since it's constant at all times.

- The liquid is incompressible. (no change in temperature or concentration due to compression)
- The reactions are elementary and obey the Arrhenius relation:

  * $r_i = k(T_R)\, c_i = k_{0_1} \exp(-E_R / T_R)\, C_i$

* The amount of heat that is removed can be adjusted by Q_J

* The heat transfer coefficient between the reaction medium and the jacket (kA) is constant.

* The initial concentrations in the reactor are $C_{A,0} = 0$ $C_{B,0} = 0$ and the initial reactor and jacket temperatures are $T_{R,0} = T_{J,0} = 387.05\,[K]$

1) The required balances that describe the dynamic behavior of the system are energy and mole balances.

2) - The change in concentration can be obtained from mole balance by dividing by V since it's constant at all times.

$$m = \rho V \quad , \quad n = c V \quad \frac{dc}{dt} = \frac{1}{V}\frac{dn}{dt}$$

* For the substance A;

$$\dot{n}_A = \dot{n}_{A,in} - \dot{n}_{A,out} + \dot{n}_{R,A}$$

A becomes B

$$\dot{n}_{A,in} = F_R\, c_{A,in} \quad , \quad \dot{n}_{A,out} = F_R\, c_A$$

Substance A becomes B with the irreversible exothermic first order reaction 1:

$$n_{R,A} = -k_{01}\, e^{-\frac{E_{R,1}}{T_R}}\, c_A$$

$$\dot{n}_A = F_R\,(c_{A,in} - c_A) - k_{01}\, e^{-\frac{E_{R,1}}{T_R}}\, c_A\, V$$

$$\boxed{\frac{dc_A}{dt} = \frac{F_R}{V}(c_{A,in} - c_A) - k_{01}\, e^{-\frac{E_{R,1}}{T_R}}\, c_A}$$

* For the substance B;

$$\dot{n}_B = \cancel{\dot{n}_{B,in}} - \dot{n}_{B,out} + \dot{n}_{R,B}$$

A becomes B, B becomes C

there is no input of B

$$\dot{n}_{B,out} = F_R\, c_B \quad , \quad n_{R,B} = r_1 V - r_2 V$$

$$\dot{n}_{R,B} = \underbrace{\left(k_{01}\, e^{-\frac{E_{R,1}}{T_R}}\, c_A\, V\right)}_{A \rightarrow B} - \underbrace{\left(k_{02}\, e^{-\frac{E_{R,2}}{T_R}}\, c_B\, V\right)}_{B \rightarrow C}$$

$$\dot{n}_B = -F_R\, c_B + \left(k_{o_1}\, e^{\frac{-E_{R,1}}{T_R}}\, c_A\, V\right) - \left(k_{o_2}\, e^{\frac{-E_{R,2}}{T_R}}\, c_B\, V\right)$$

$$\boxed{\frac{d c_B}{dt} = -\frac{F_R}{V}\, c_B + \left(k_{o_1}\, e^{\frac{-E_{R,1}}{T_R}}\, c_A\right) - \left(k_{o_2}\, e^{\frac{-E_{R,2}}{T_R}}\, c_B\right)}$$

* For the substance C ;

$$\dot{n}_C = \cancel{n_{C,in}} - n_{C,out} + n_{R,B} \longrightarrow \text{B becomes C}$$

there is no input of C

$$n_{C,out} = F_R\, c_A \quad , \quad n_{R,C} = k_{o_2}\, e^{\frac{-E_{R,2}}{T_R}}\, c_B\, V$$

$$\dot{n}_C = -F_R\, c_A + k_{o_2}\, e^{\frac{-E_{R,2}}{T_R}}\, c_B\, V$$

$$\boxed{\frac{d c_C}{dt} = -\frac{F_R}{V}\, c_A + k_{o_2}\, e^{\frac{-E_{R,2}}{T_R}}\, c_B}$$

- The change in temperature can be obtained from energy balance by dividing Q into m and cp.

$$Q = m\, c_p\, T \quad , \quad m = \rho V$$

* For the energy balance ;

$$\frac{dQ}{dt} = \underbrace{\dot{Q}_{in}}_{\text{input}} + \underbrace{\dot{Q}_R}_{\text{energy flow generated by the exothermic reactions}} - \underbrace{\dot{Q}_{out}}_{\text{output}} - \underbrace{\dot{Q}_J}_{\text{consumption (heat exchange with the cooling jacket)}}$$

The only input component is A , thus :

$$\dot{Q}_{in} = \rho\, F_R\, c_p\, T_{in}$$

Inside the CSTR , everything is well-mixed, therefore :

$$\dot{Q}_{out} = \rho\, F_R\, c_p\, T_R$$

$$\dot{Q}_R = \underbrace{r_{R,A}\,\Delta H_{R,1}}_{\text{1st Exothermic Reaction}} + \underbrace{r_{R,B}\,\Delta H_{R,2}}_{\text{2nd Exothermic Reaction}}$$

$$= \left(-k_{0_1}\, e^{\frac{-E_{R,1}}{T_R}}\, C_A\,\Delta H_{R,1}\, V\right) + \left(-k_{0_2}\, e^{\frac{-E_{R,2}}{T_R}}\, C_B\,\Delta H_{R,2}\, V\right)$$

The amount of heat that is removed by the jacket can be written as:

$$\dot{Q}_J = kA\,(T_R - T_J)$$

Putting all equations together yields:

$$\frac{dQ}{dt} = \rho F_R c_p T_{in} - \rho F_R c_p T_R - k_{0_1}\, e^{\frac{-E_{R,1}}{T_R}}\, C_A\,\Delta H_{R,1}\, V$$
$$- k_{0_2}\, e^{\frac{-E_{R,2}}{T_R}}\, C_B\,\Delta H_{R,2}\, V - kA\,(T_R - T_J)$$

$$\boxed{\frac{dT_R}{dt} = \frac{F_R}{V}(T_{in} - T_R) - \frac{\left(k_{0_1}\, e^{\frac{-E_{R,1}}{T_R}} C_A \Delta H_{R,1}\right)}{\rho\, C_p} - \frac{\left(k_{0_2}\, e^{\frac{-E_{R,2}}{T_R}} C_B \Delta H_{R,2}\right)}{\rho\, C_p} - \frac{kA\,(T_R - T_J)}{\rho\, C_p\, V}}$$

* For the cooling jacket:

$$\boxed{\frac{dT_J}{dt} = \frac{kA\,(T_R - T_J) - Q_J}{m_J\, C_{p_J}}}$$

* For an exothermic reaction, the system loses energy to generate heat. $\dot{Q}_R$ term is positive because ΔH_R terms are negative.

– The system is non-linear because there exists exponential terms due to the Arrhenius equation for reactions.

$$r_i = k(T_R)\, c_i = k0_i \exp(-E_{R_i} / T_R)\, c_i$$

3) $$\frac{dC_A}{dt} = \frac{F_{R,ss}}{V}(C_{A,in} - C_{A,ss}) - k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}} C_{A,ss} = 0$$

$$\frac{F_{R,ss}}{V} C_{A,in} = \left(\frac{F_{R,ss}}{V} + k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}}\right) C_{A,ss}$$

$$\boxed{C_{A,ss} = \frac{\frac{F_{R,ss}}{V} C_{A,in}}{\left(\frac{F_{R,ss}}{V} + k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}}\right)}} \quad (1)$$

$$\frac{dC_B}{dt} = -\frac{F_{R,ss}}{V} C_{B,ss} + \left(k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}} C_{A,ss}\right) - \left(k_{02}\, e^{\frac{-E_{R,2}}{T_{R,ss}}} C_{B,ss}\right) = 0$$

- Plug (1) into this equation to get (2)

$$\frac{dT_J}{dt} = \frac{kA(T_{R,ss} - T_{J,ss}) - Q_{J,ss}}{m_J C_{P_J}} = 0$$

$$T_{J,ss} = \frac{kA\, T_{R,ss} - Q_{J,ss}}{kA} \quad (3)$$

$$\frac{dT_R}{dt} = \frac{F_{R,ss}}{V}(T_{in} - T_{R,ss}) - \frac{(k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}} C_{A,ss}\, \Delta H_{R,1})}{\rho C_p} - \frac{(k_{02}\, e^{\frac{-E_{R,2}}{T_{R,ss}}} C_{B,ss}\, \Delta H_{R,2})}{\rho C_p} \dots$$

$$\dots - \frac{kA(T_{R,ss} - T_{J,ss})}{\rho C_p V} \quad (4)$$

- Plug the solutions of (1), (2) and (3) to get rid of other terms and solve for $T_{R,ss}$. This problem is solved in MATLAB.

$T_{R,ss} = 398,6581$ K $C_{A,ss} = 1,6329$ kmol / m^3

$T_{J,ss} = 397,3736$ K $C_{B,ss} = 1,1101$ kmol / m^3

4) Taylor - Series Expansion:

$$f(x) = f(x_0) + \left.\frac{df}{dx}\right|_{x} (x - x_0) + \frac{1}{2}\frac{d^2 f}{dx^2}(x - x_0)^2 + \dots = 0$$

- To linearize the system of non-linear differential equations, we use Multi-variable Taylor-Series Expansion:

$\dot{x} = f(x,u)$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$, $f(x_s, u_s) = 0$

(x_s, u_s) is an equilibrium point

$x(t) = x_s + \Delta x(t)$, $u(t) = u_s + \Delta u(t)$

Multi-variable Taylor Series Expansion

$$\dot{x}(t) = f(x_s, u_s) + \underbrace{\begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{x_n} \end{pmatrix}}_{J_{fx}(x_s, u_s)} \Delta x(t) + \underbrace{\begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_p} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_p} \end{pmatrix}}_{J_{fu}(x_s, u_s)} \Delta u(t) + \dots$$

* Higher order terms are neglected for linear approximation and derivatives are evaluated at (x_s, u_s)

$$\Delta \dot{x}(t) = A\, \Delta x(t) + B\, \Delta u(t)$$

$x_1 = C_A$, $x_2 = C_B$, $x_3 = T_R$, $x_4 = T_J$

$\dot{x}_1 = \frac{dC_A}{dt}$, $\dot{x}_2 = \frac{dC_B}{dt}$, $\dot{x}_3 = \frac{dT_R}{dt}$, $\dot{x}_4 = \frac{dT_J}{dt}$

f_1 => the equation for $\dot{x}_1$

f_2 => // // // $\dot{x}_2$

f_3 => // // // $\dot{x}_3$

f_4 => // // // $\dot{x}_4$

$$\left.\frac{\partial f_1}{\partial x_1}\right|_{x_s,u_s} = -\frac{F_{R,ss}}{V} - k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}}$$

$$\left.\frac{\partial f_1}{\partial x_2}\right|_{x_s,u_s} = 0 \qquad \left.\frac{\partial f_1}{\partial x_3}\right|_{x_s,u_s} = \frac{-k_{01}\, C_{A,ss}\, E_{R,1}}{T_{R,ss}^{2}} \cdot e^{\frac{-E_{R,1}}{T_{R,ss}}}$$

$$\left.\frac{\partial f_1}{\partial x_4}\right|_{x_s,u_s} = 0 \qquad \left.\frac{\partial f_1}{\partial u_1}\right|_{x_s,u_s} = \frac{C_{A,in} - C_{A,ss}}{V}$$

$$\left.\frac{\partial f_1}{\partial u_2}\right|_{x_s,u_s} = 0 \qquad \left.\frac{\partial f_2}{\partial x_1}\right|_{x_s,u_s} = k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}}$$

$$\left.\frac{\partial f_2}{\partial x_2}\right|_{x_s,u_s} = -\frac{F_{R,ss}}{V} - k_{02}\, e^{\frac{-E_{R,2}}{T_{R,ss}}}$$

$$\left.\frac{\partial f_2}{\partial x_3}\right|_{x_s,u_s} = \frac{k_{01}\, e^{\frac{-E_{R,1}}{T_{R,ss}}}\, C_{A,ss}\, E_{R,1}}{T_{R,ss}^{2}} - \frac{k_{02}\, e^{\frac{-E_{R,2}}{T_{R,ss}}}\, C_{B,ss}\, E_{R,2}}{T_{R,ss}^{2}}$$

$$\left.\frac{\partial f_2}{\partial x_4}\right|_{x_s,u_s} = 0 \qquad \left.\frac{\partial f_2}{\partial u_1}\right|_{x_s,u_s} = \frac{-C_{B,ss}}{V} \qquad \left.\frac{\partial f_2}{\partial u_2}\right|_{x_s,u_s} = 0$$

$$\left.\frac{\partial f_3}{\partial x_1}\right|_{x_s,u_s} = -\frac{k_{01}\, e^{-\frac{E_{R,1}}{T_{R,ss}}}\, \Delta H_{R,1}}{\rho\, C_p} \qquad \left.\frac{\partial f_3}{\partial x_2}\right|_{x_s,u_s} = \frac{-k_{02}\, e^{-\frac{E_{R,2}}{T_{R,ss}}}\, \Delta H_{R,2}}{\rho\, C_p}$$

$$\left.\frac{\partial f_3}{\partial x_3}\right|_{x_s,u_s} = -\frac{F_{R,ss}}{V} - \frac{\left(k_{01}\, e^{-\frac{E_{R,1}}{T_{R,ss}}}\, C_{A,ss}\, \Delta H_{R,1}\, E_{R,1}\right)}{\rho\, C_p\, T_{R,ss}^{2}}$$

$$- \frac{\left(k_{02}\, e^{-\frac{E_{R,2}}{T_{R,ss}}}\, C_{B,ss}\, \Delta H_{R,2}\, E_{R,2}\right)}{\rho\, C_p\, T_{R,ss}^{2}} - \frac{kA}{\rho C_p V}$$

$$\left.\frac{\partial f_3}{\partial x_4}\right|_{x_s,u_s} = +\frac{kA}{\rho C_p V} \qquad \left.\frac{\partial f_3}{\partial u_1}\right|_{x_s,u_s} = \frac{(T_{in} - T_{R,ss})}{V}$$

$$\left.\frac{\partial f_3}{\partial u_2}\right|_{x_s,u_s} = 0 \qquad \left.\frac{\partial f_4}{\partial x_1}\right|_{x_s,u_s} = 0 \qquad \left.\frac{\partial f_4}{\partial x_2}\right|_{x_s,u_s} = 0$$

$$\left.\frac{\partial f_4}{\partial x_3}\right|_{x_s,u_s} = \frac{kA}{m_J C_{pJ}} \qquad \left.\frac{\partial f_4}{\partial x_4}\right|_{x_s,u_s} = \frac{-kA}{m_J C_{pJ}}$$

$$\left.\frac{\partial f_4}{\partial u_1}\right|_{x_s,u_s} = 0 \qquad \left.\frac{\partial f_4}{\partial u_2}\right|_{x_s,u_s} = -\frac{1}{m_J C_{pJ}}$$

After plugging the values of the parameters;

$$\begin{pmatrix} \Delta\dot{x}_1 \\ \Delta\dot{x}_2 \\ \Delta\dot{x}_3 \\ \Delta\dot{x}_4 \end{pmatrix} = \begin{pmatrix} -0{,}7386 & 0 & -0{,}0503 & 0 \\ 0{,}5021 & -0{,}7386 & 0{,}0161 & 0 \\ 0{,}75 & 1{,}9643 & -0{,}5412 & 0{,}5138 \\ 0 & 0 & 1{,}4448 & -1{,}4448 \end{pmatrix} \begin{pmatrix} \Delta x_1(t) \\ \Delta x_2(t) \\ \Delta x_3(t) \\ \Delta x_4(t) \end{pmatrix} +$$

$$\begin{pmatrix} 346{,}7 & 0 \\ -111{,}0 & 0 \\ -1160{,}8 & 0 \\ 0 & -0{,}1 \end{pmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \end{pmatrix}$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta x_1(t) \\ \Delta x_2(t) \\ \Delta x_3(t) \\ \Delta x_4(t) \end{pmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta u_1(t) \\ \Delta u_2(t) \end{pmatrix}$$

* After checking the eigen values of this system and seeing that they all have negative real parts, it can be concluded that the equilibrium point (x_s, u_s) is locally stable.

5) The validity of the linearization is checked by simulating the linearized system in MATLAB against the original model at the equilibrium point for $\Delta u = \pm 10\%$ of the steady-state input. Linearized model of the system seems to converge to the same steady-states approximately in 60 seconds which means that the linearization is valid.

6) The operability of the linearized system is checked in MATLAB. The operability matrix is calculated as
$M = -A^{-1}B$ $(Ax = -Bu \; , \; x = -A^{-1}B)$

* No eigen value of $M^T M$ is zero and matrix M is full-rank. This means that it is possible to operate the system at steady-state conditions. (rank(M) = 2)

* The condition number $\gamma = \sigma_1 / \sigma_p = 8{,}1944e+04$
$\gamma < 1e+05$ => Operability matrix is not ill-conditioned yet it's quite large.

* Operability matrix M establishes the relationship between inputs and outputs at stationary conditions.
The singular value decomposition of M is:
$M = U \Sigma V$ where the columns of U matrix are the eigen vectors of MM^T and the columns of V are the eigen vectors of $M^T M$.

* At stationary conditions, the states are related with the inputs as:
$u_s = V\alpha$, $x_s = U \Sigma V^T V \alpha = U \Sigma \alpha$
where α is a vector with information about the magnitude of the inputs along the directions of the columns vectors of V. The matrix U is the one which gives information of how the equilibrium point changes as a function of the direction and the amplification of the inputs. The maximum steady-state gain for the system can be obtained with the input in the direction of V_1 corresponding to the largest singular value. A higher gain means that less effort is required to move the system to a new operating point. (If σ_J is large, then input in the direction of V_J has a large effect on the output in direction u_1)

For our case, since $p < n$, the columns u_3 and u_4 describe the part of the state-space in which the steady-state cannot be moved. The dimension of steady state subspace is 2.

7) The condition needed to assign the closed-loop poles is decided by the Kalman Criterion for controllability.

(A, B) is controllable if $\operatorname{rank}[B \; AB \; \dots \; A^{n-1}B] = n$

Computing on MATLAB, we get rank(ctb-kalman) = 4 which shows the controllability matrix has full rank ($n = 4$). The linearized system's original eigen values are at $-0{,}1205$, $-0{,}4637$, $-0{,}9058$ and $-1{,}9734$.

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. The controller has been designed in MATLAB and the corresponding explanations can be found there as comments. The system behavior with a controller is simulated with 2 different initial conditions. The system seems to reach the equilibrium point under 10 seconds using this controller.

8) The observability analysis has been done in MATLAB and observability matrix (Kalman Observability Criterion) has full-rank. This means the L matrix can be chosen such that eig (A-LC) take arbitrary assigned values and the observer error converges to zero with the chosen dynamics.

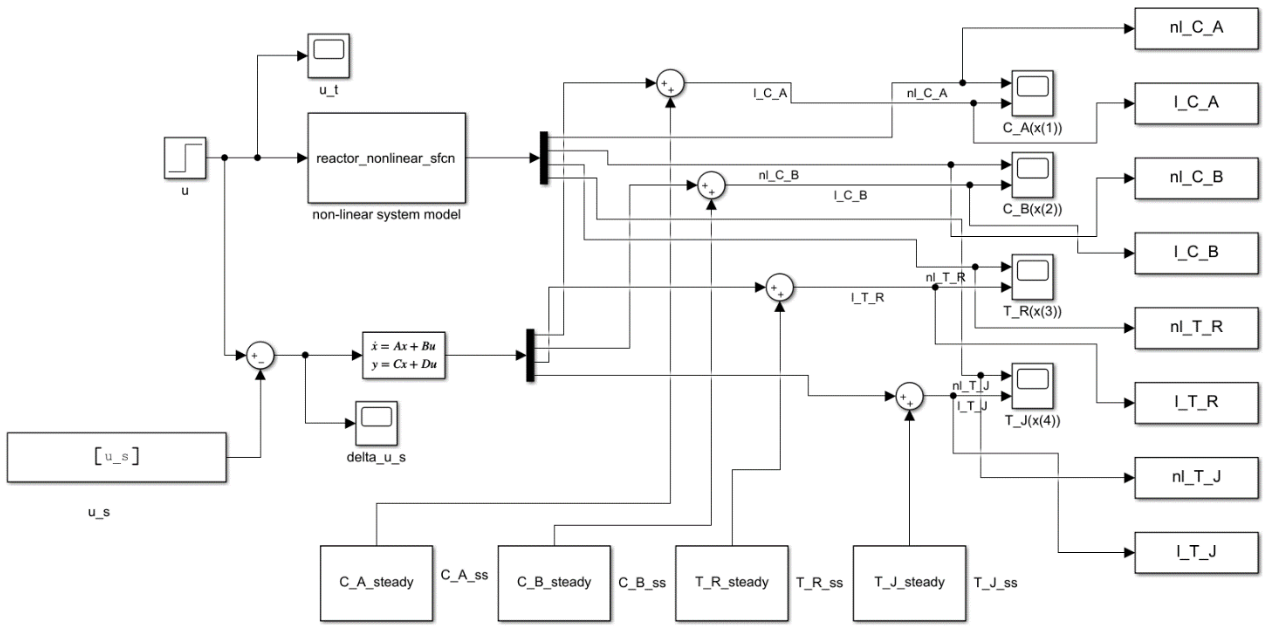
$$\dot{e} = (A - LC)(x - \hat{x}) = (A - LC)\,e$$

It is required that $\lim_{t \to \infty} e(t) = 0 \Rightarrow (A-LC)$ must be asymptotically stable!

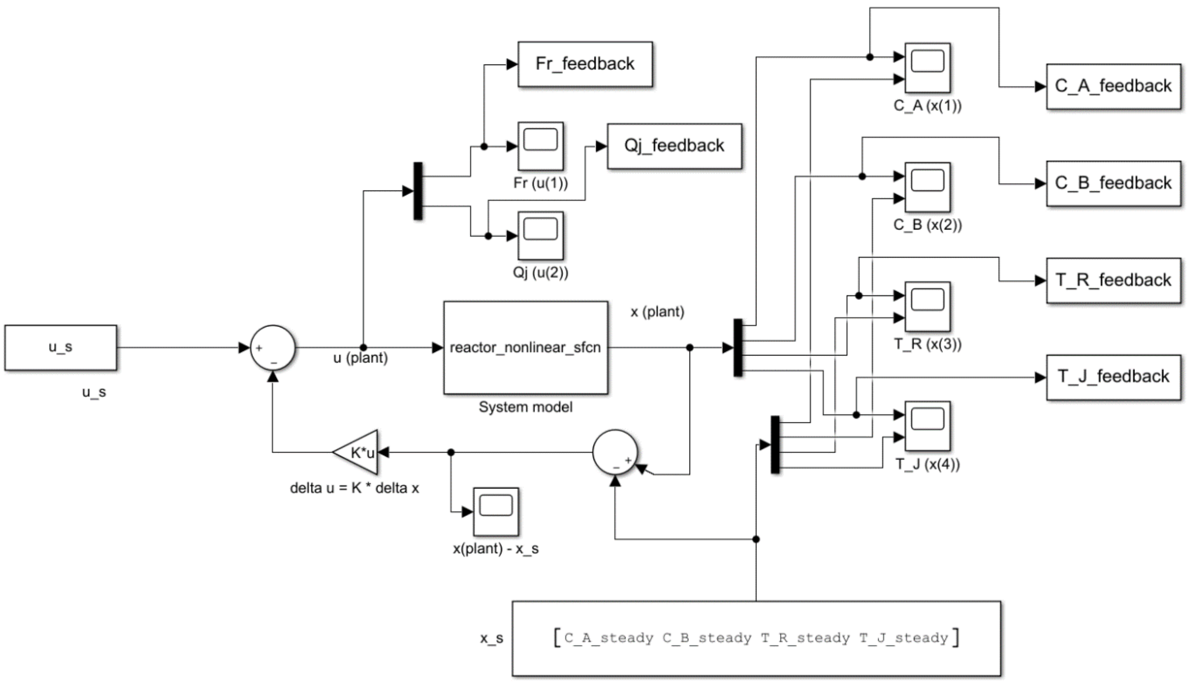
The observer has been designed in MATLAB and corresponding explanations can be found in comments.

9) The non-linear system with the observer-based feedback controller has been simulated in MATLAB. The comparison and the explanation can be found in comments.

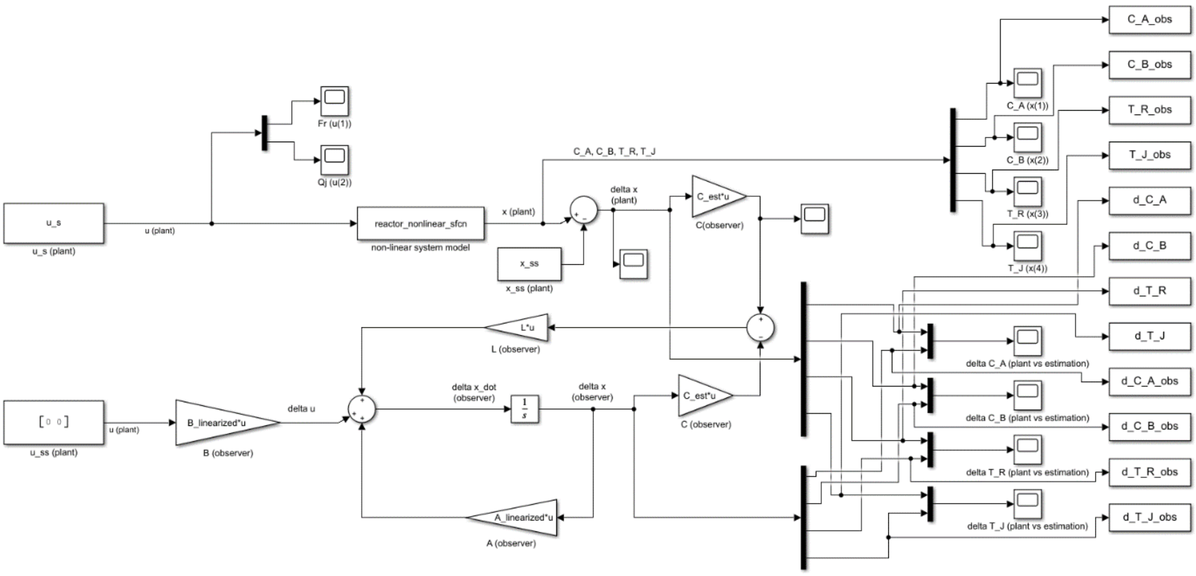
Q2 – Models



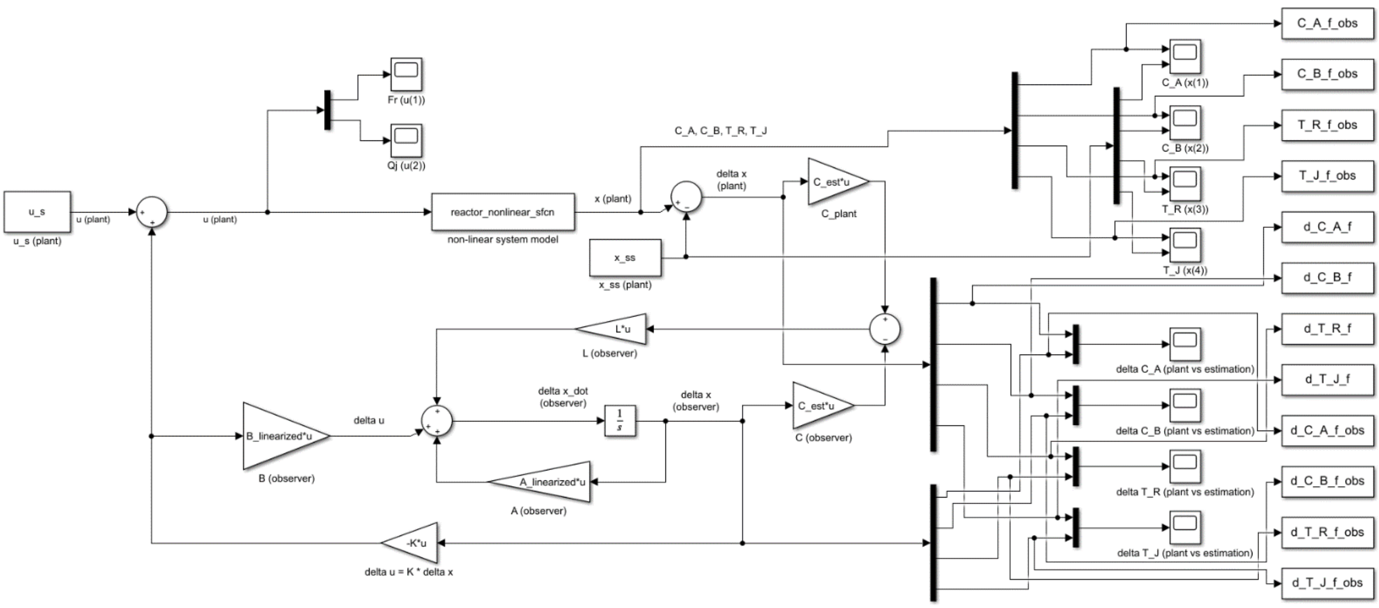
reactor_nonlinear_vs_linear_simulation.mdl



reactor_feedback_full_state.mdl

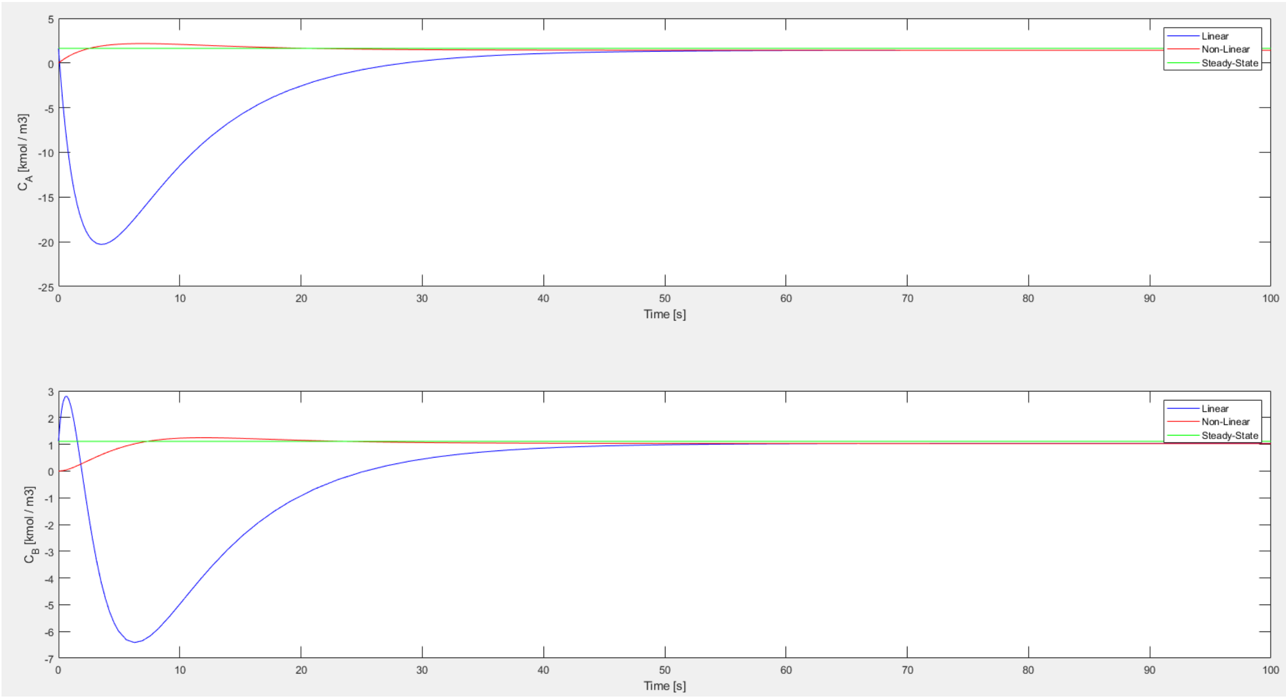


reactor_observer.mdl

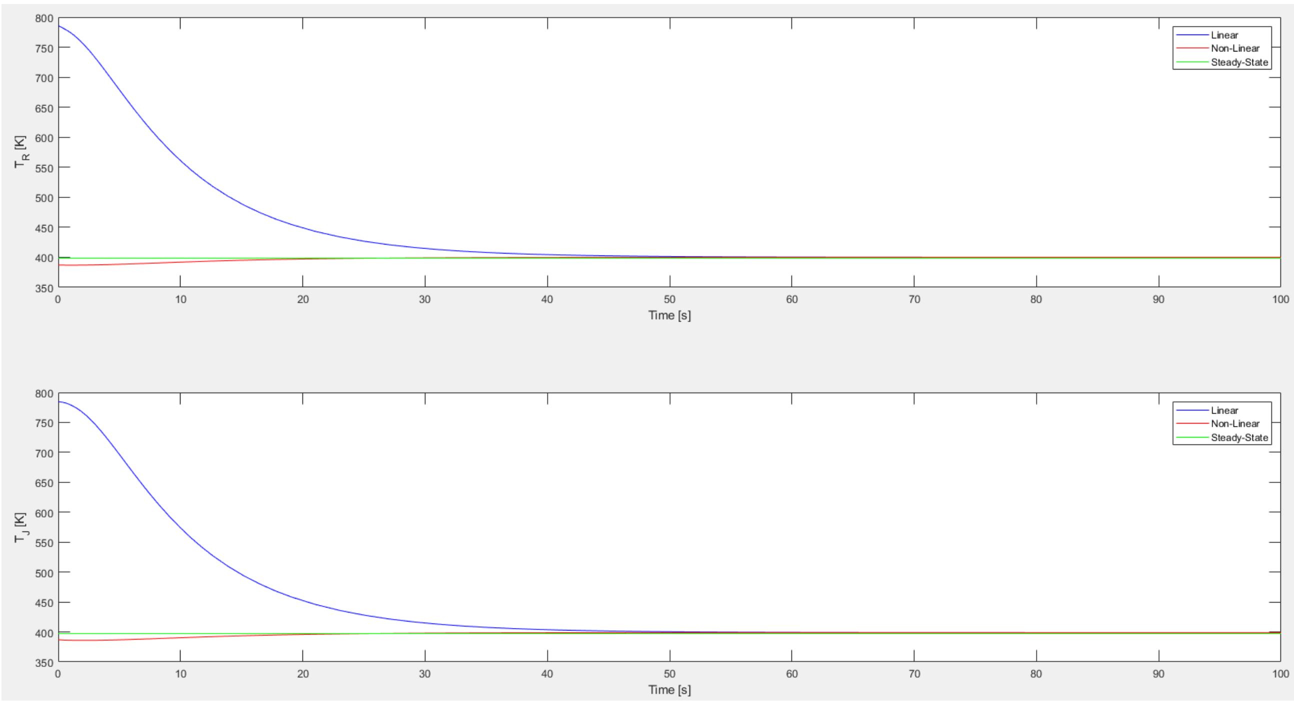


reactor_feedback_observer.mdl

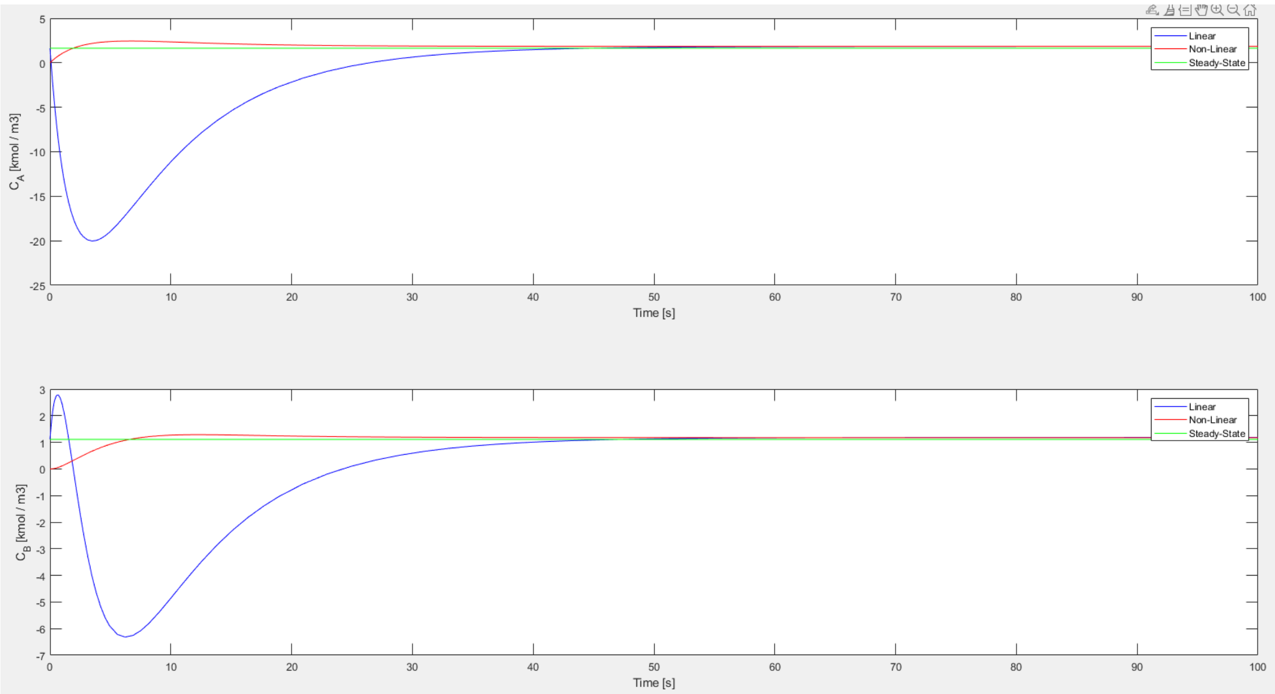
Q2 – Task5 – Plots



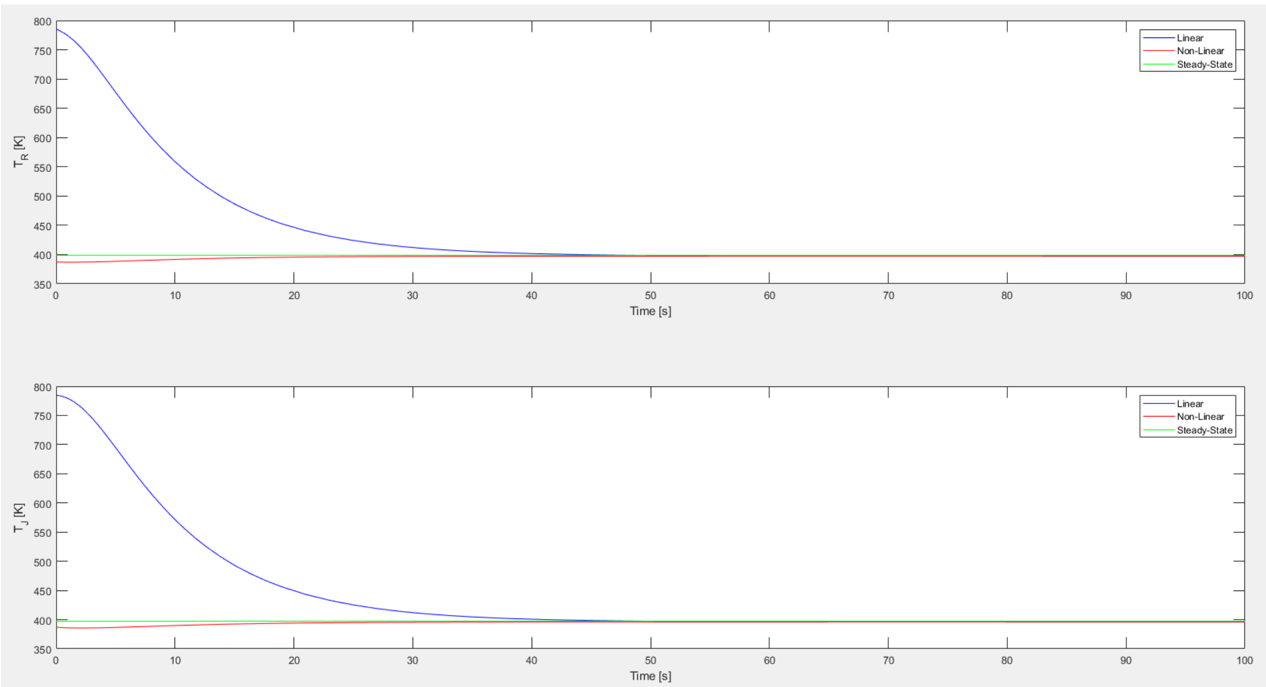
Nonlinear vs Linear Simulation for C_A, C_B respectively with Δu = -10%



Nonlinear vs Linear Simulation for T_R, T_J respectively with Δu = -10%

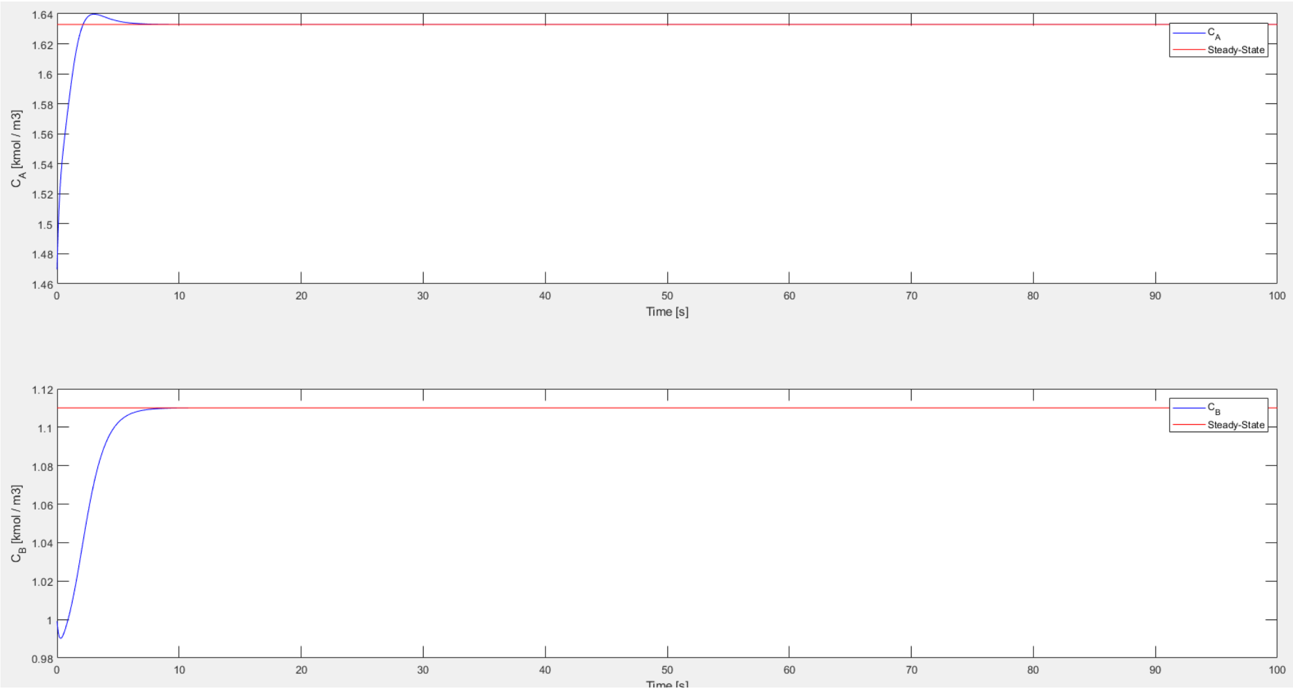


Nonlinear vs Linear Simulation for C_A, C_B respectively with Δu = +10%



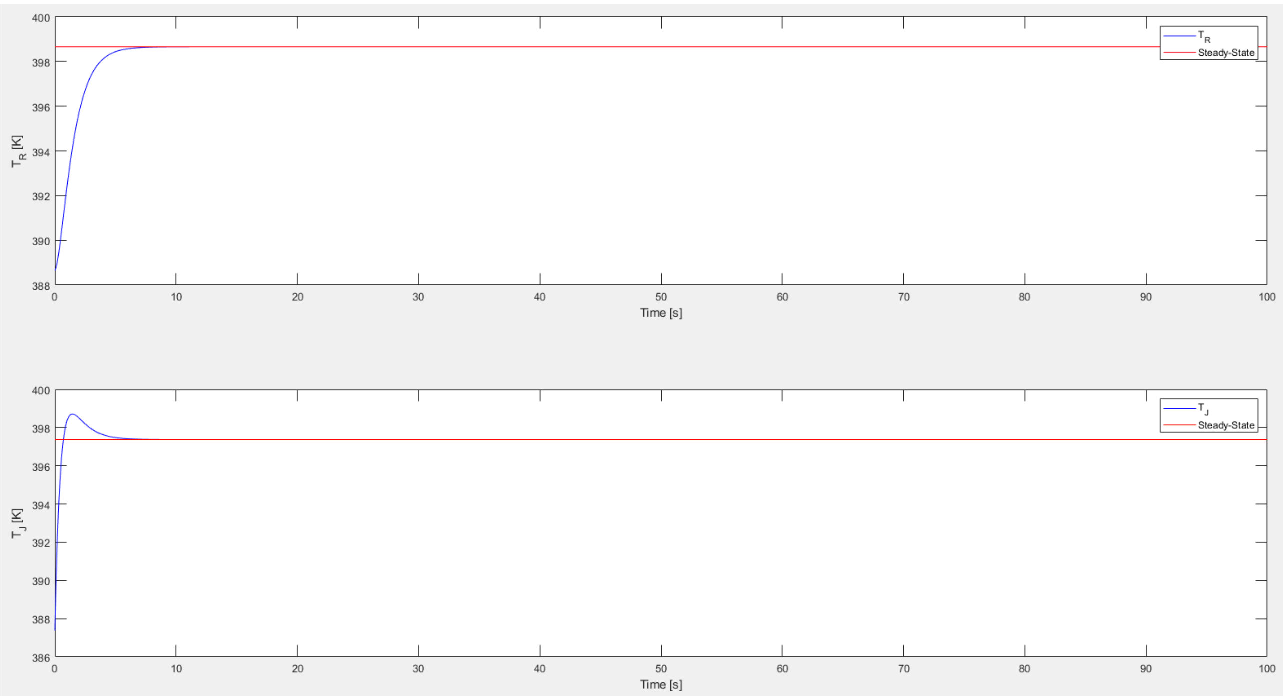
Nonlinear vs Linear Simulation for T_R, T_J respectively with Δu = +10%

Q2 – Task7 – Plots



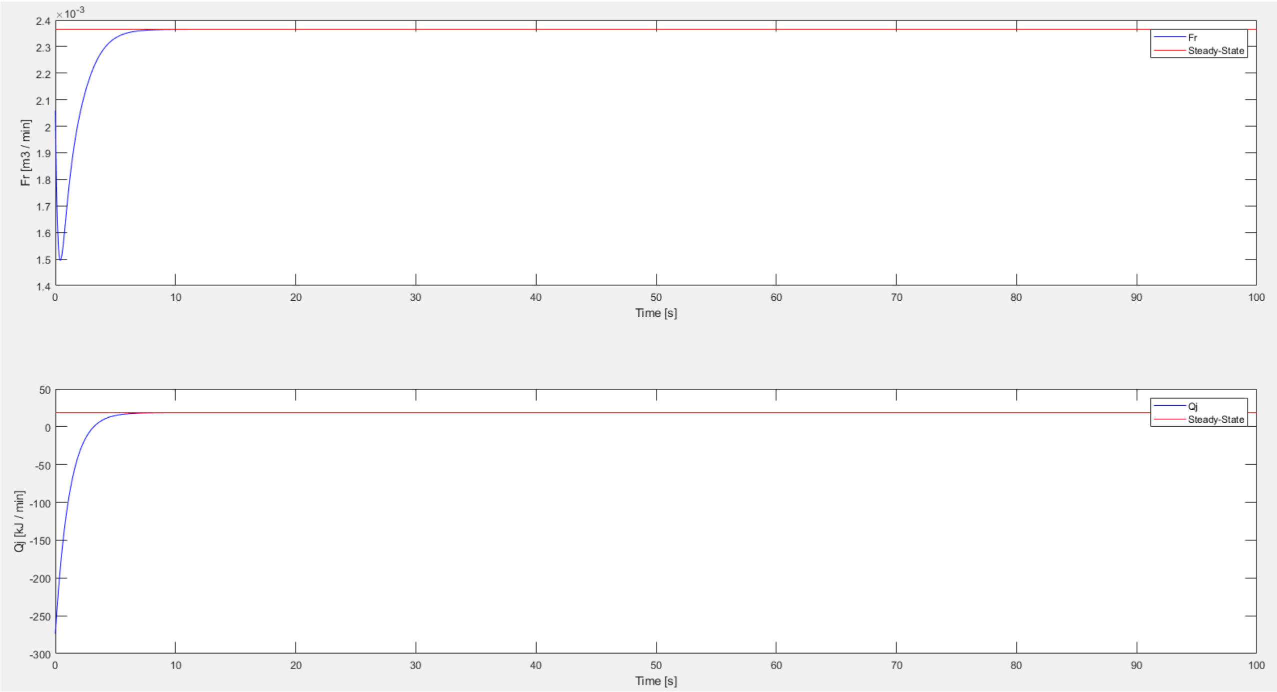
State-Feedback Controller Closed-Loop Simulation for C_A, C_B respectively

with initial condition x0,1



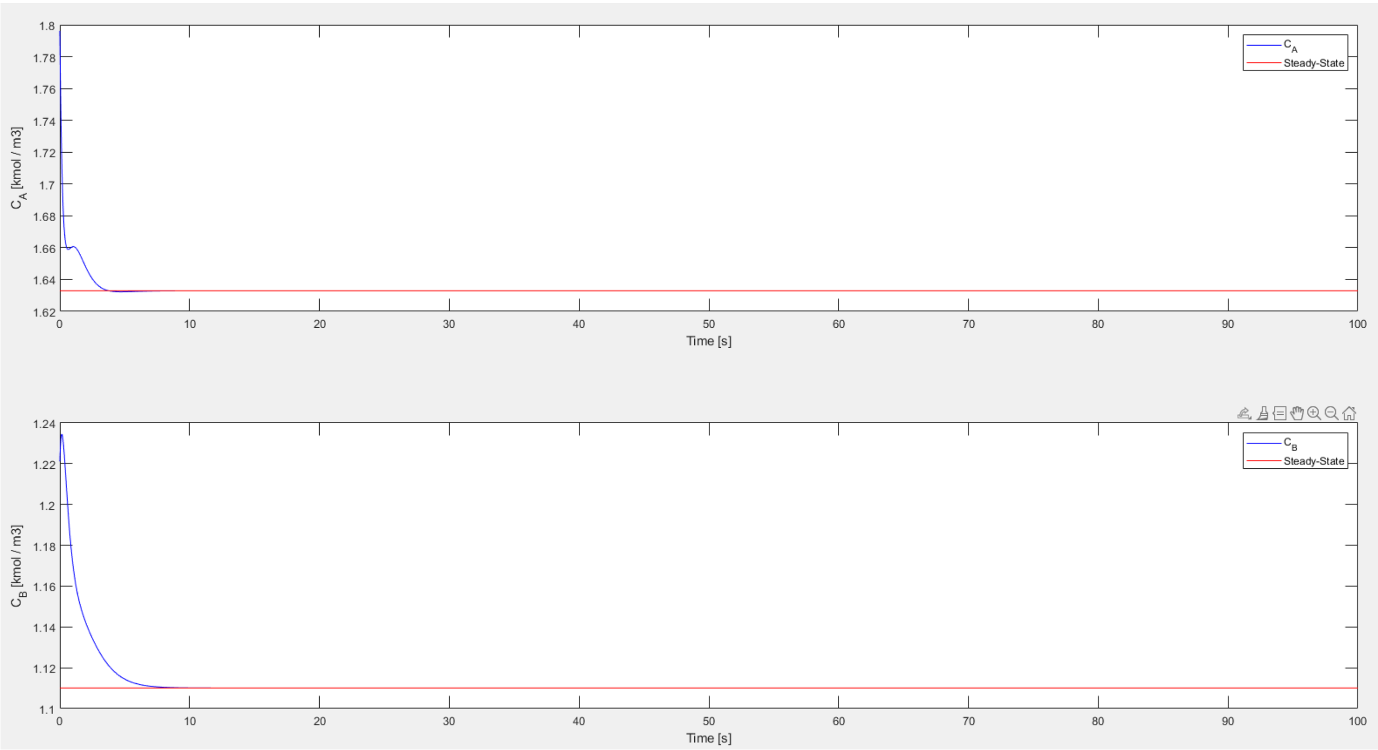
State-Feedback Controller Closed-Loop Simulation for T_R, T_J respectively

with initial condition x0,1



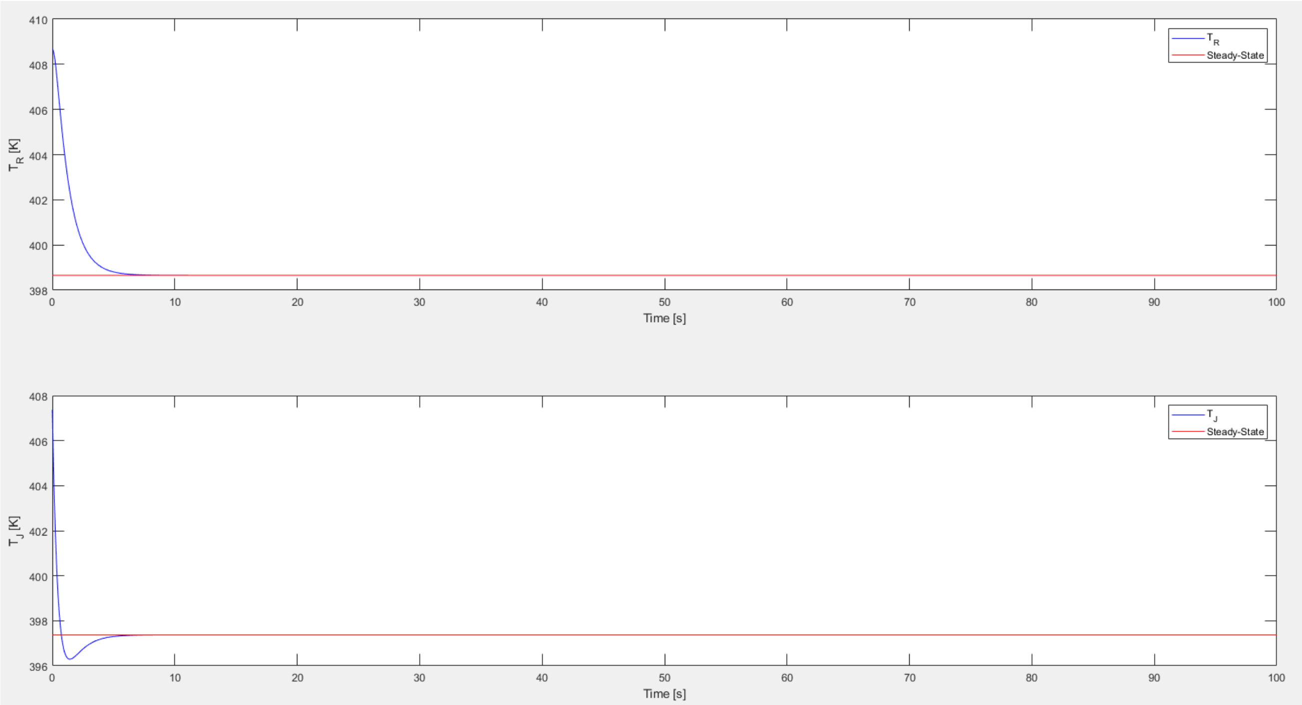
State-Feedback Controller Closed-Loop Simulation for Fr, Qj respectively

with initial condition x0,1



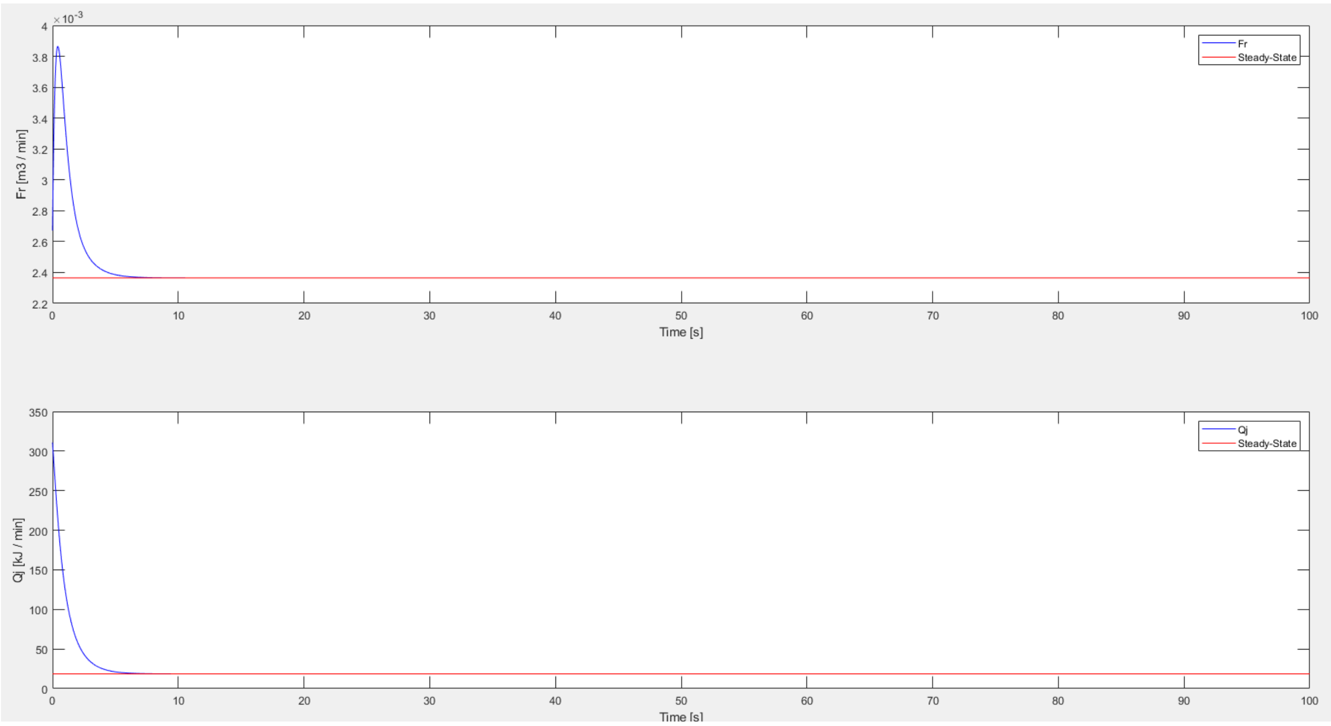
State-Feedback Controller Closed-Loop Simulation for C_A, C_B respectively

with initial condition x0,2



State-Feedback Controller Closed-Loop Simulation for T_R, T_J respectively

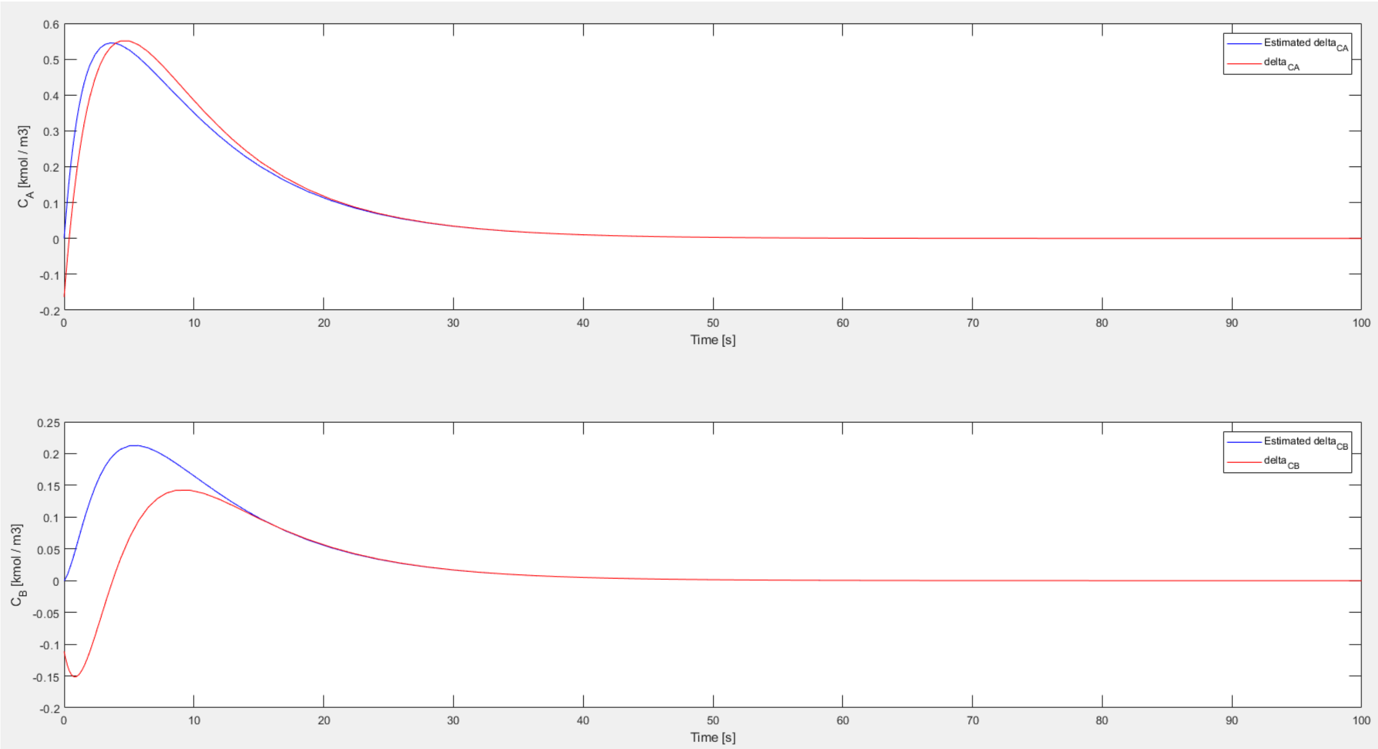
with initial condition x0,2



State-Feedback Controller Closed-Loop Simulation for Fr, Qj respectively

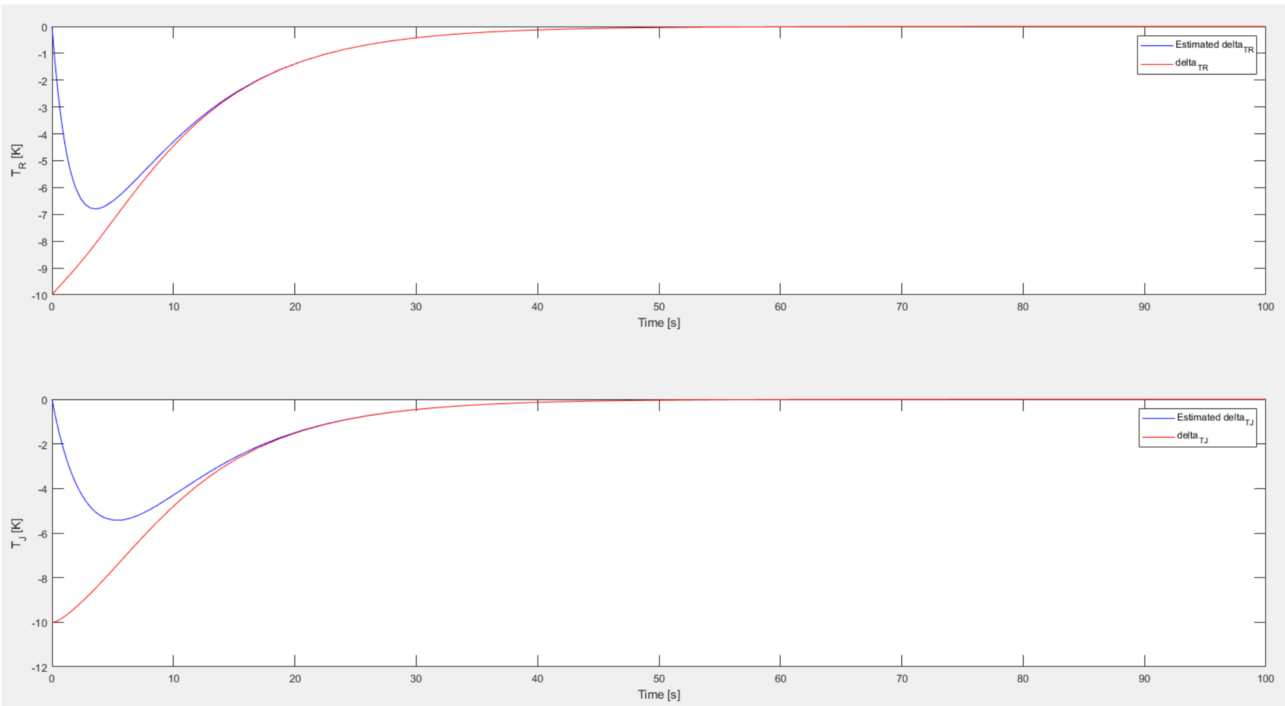
with initial condition x0,2

Q2 – Task8 – Plots



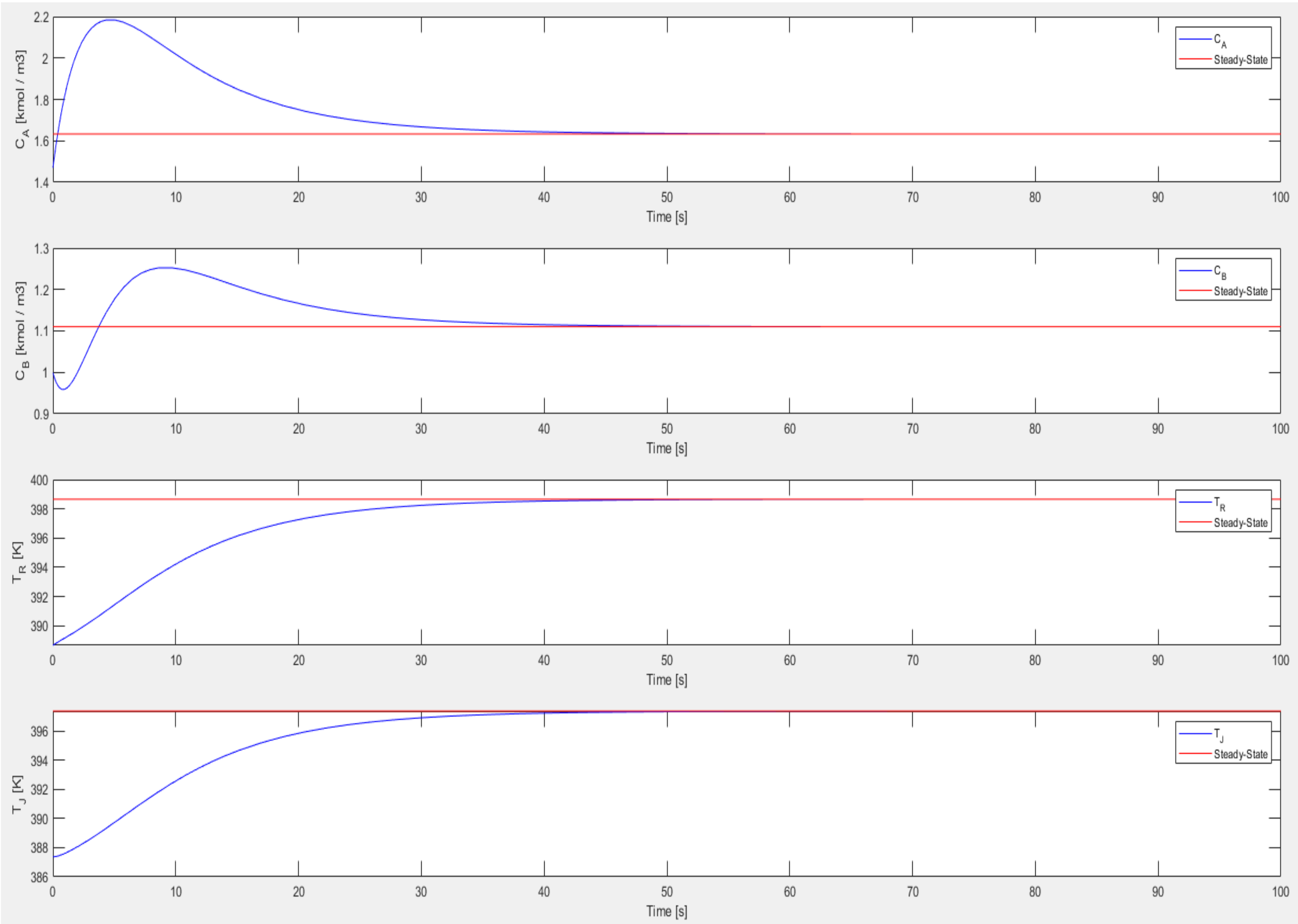
Observer Closed-Loop Simulation for ΔC_A, ΔC_B respectively

with initial condition x0,1



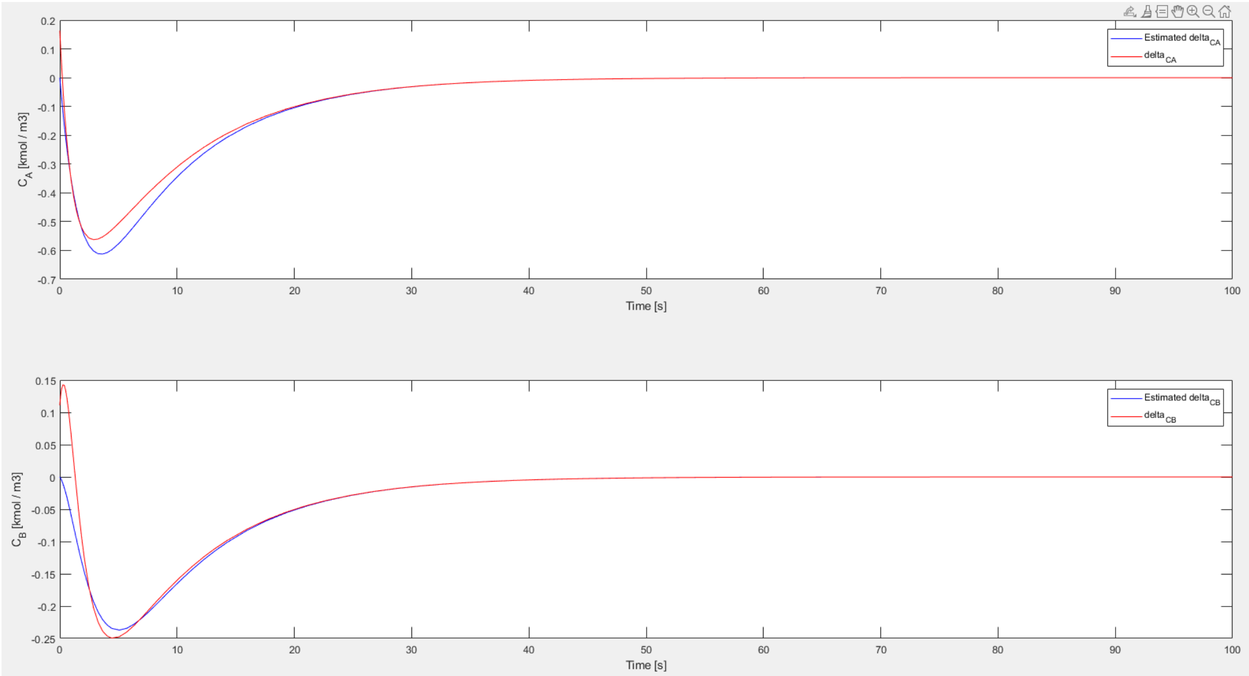
Observer Closed-Loop Simulation for ΔT_R, ΔT_J respectively

with initial condition x0,1



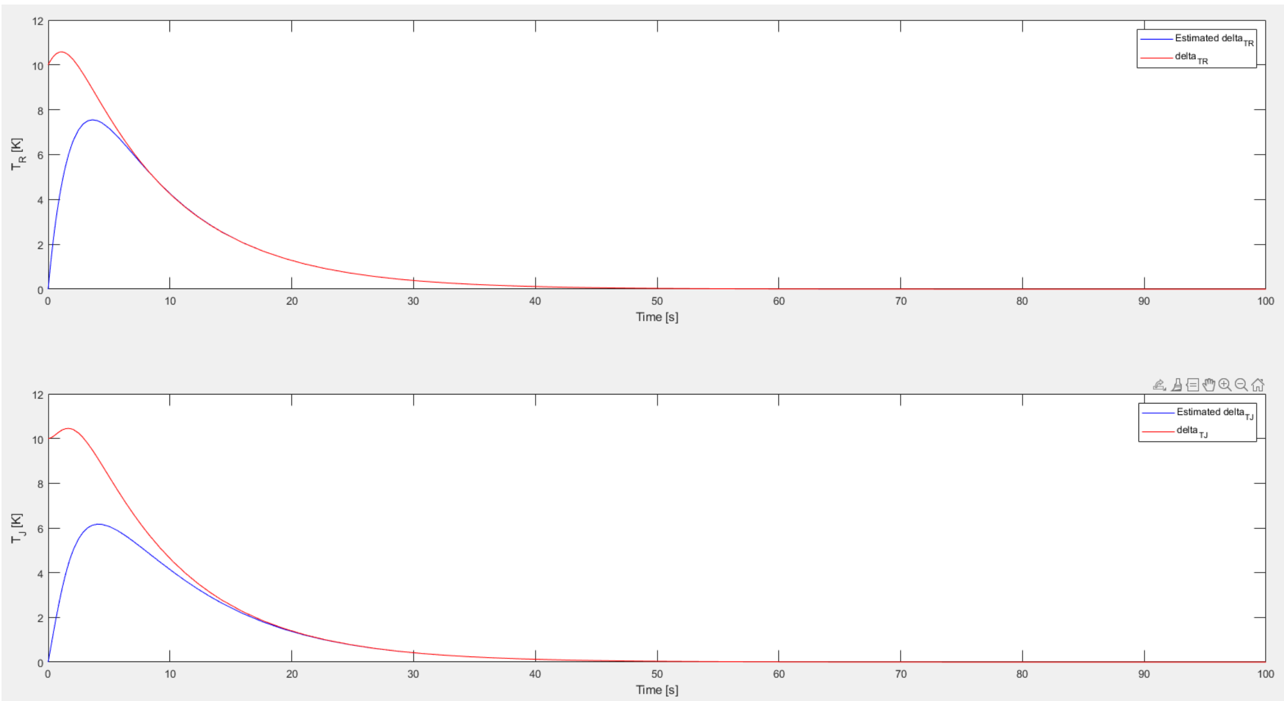
Observer Closed-Loop Simulation for C_A, C_B, T_R, T_J respectively

with initial condition x0,1



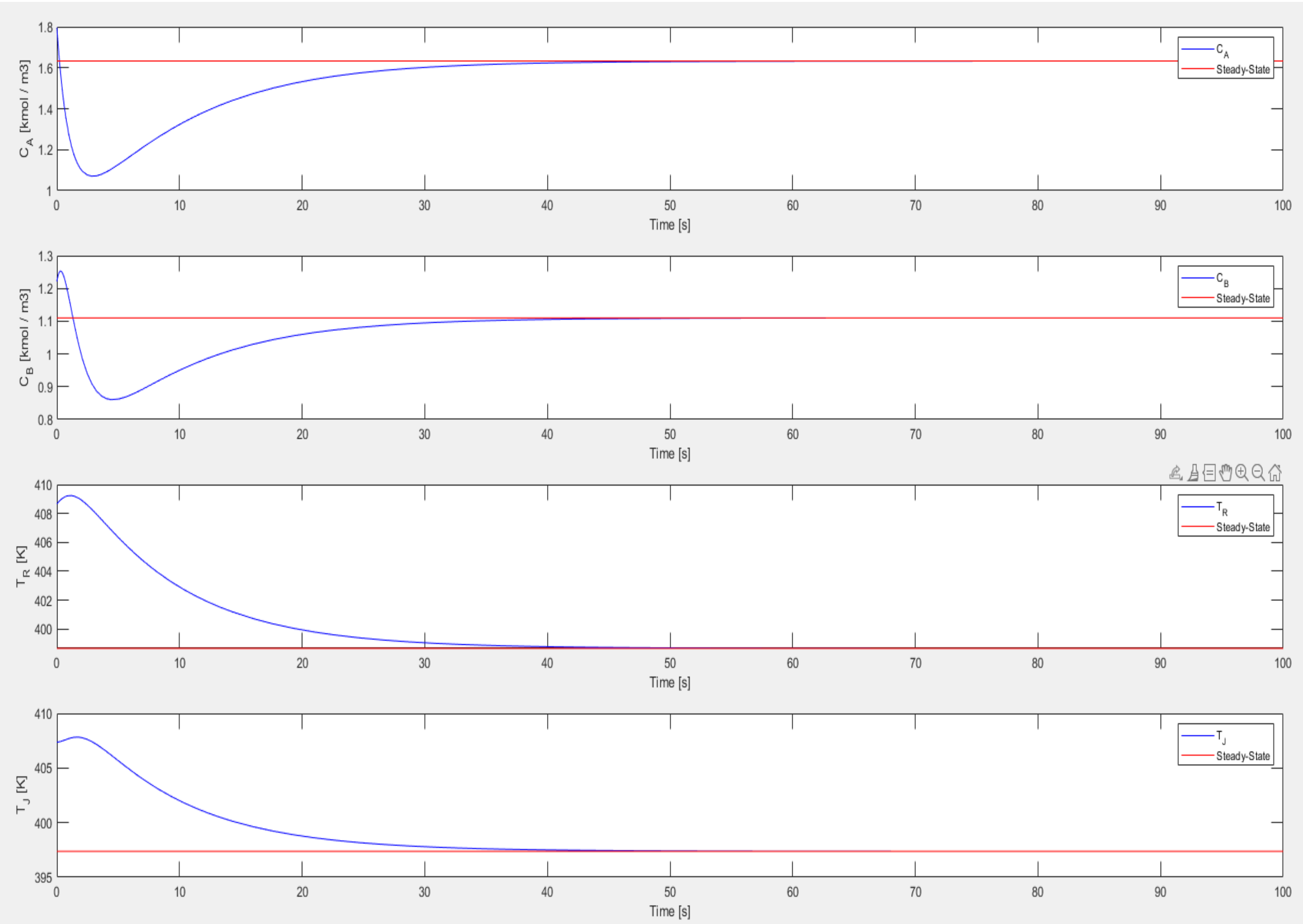
Observer Closed-Loop Simulation for ΔC_A, ΔC_B respectively

with initial condition x0,2



Observer Closed-Loop Simulation for ΔT_R, ΔT_J respectively

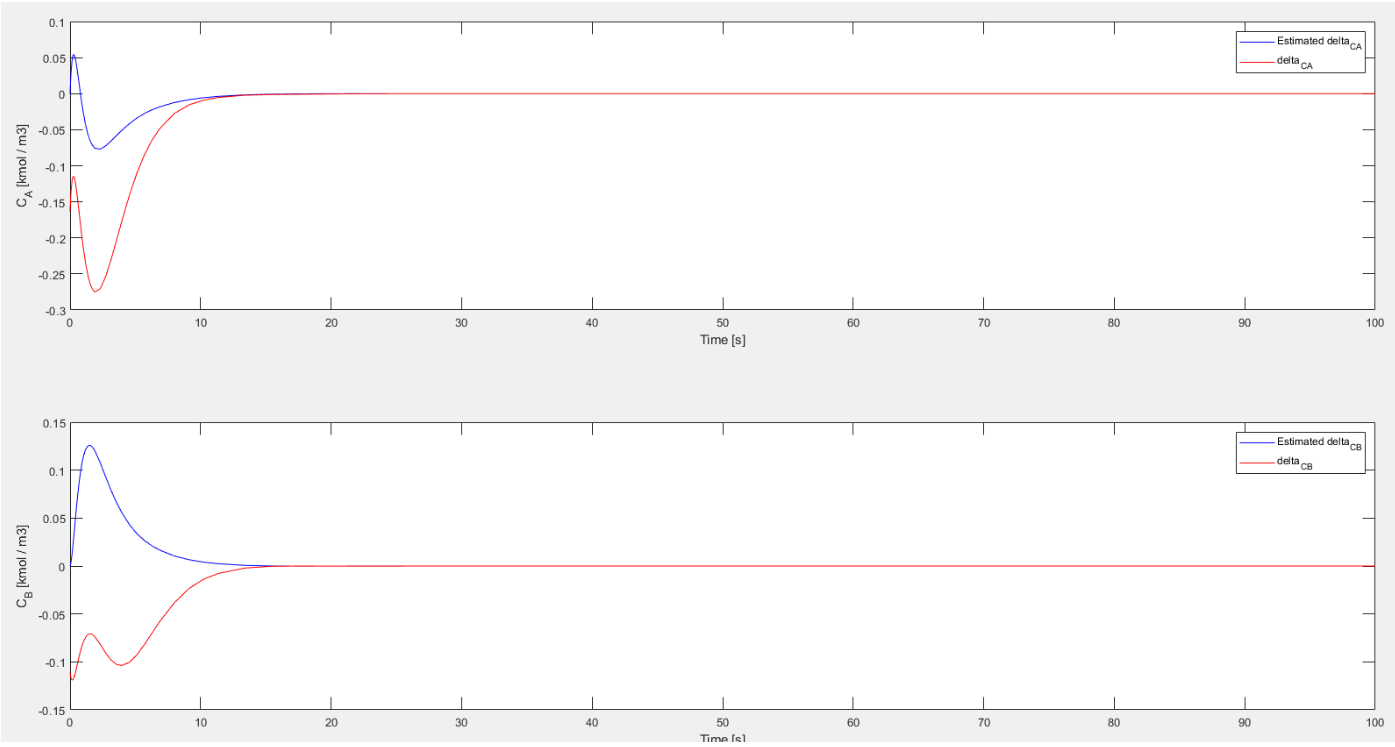
with initial condition x0,2



Observer Closed-Loop Simulation for C_A, C_B, T_R, T_J respectively

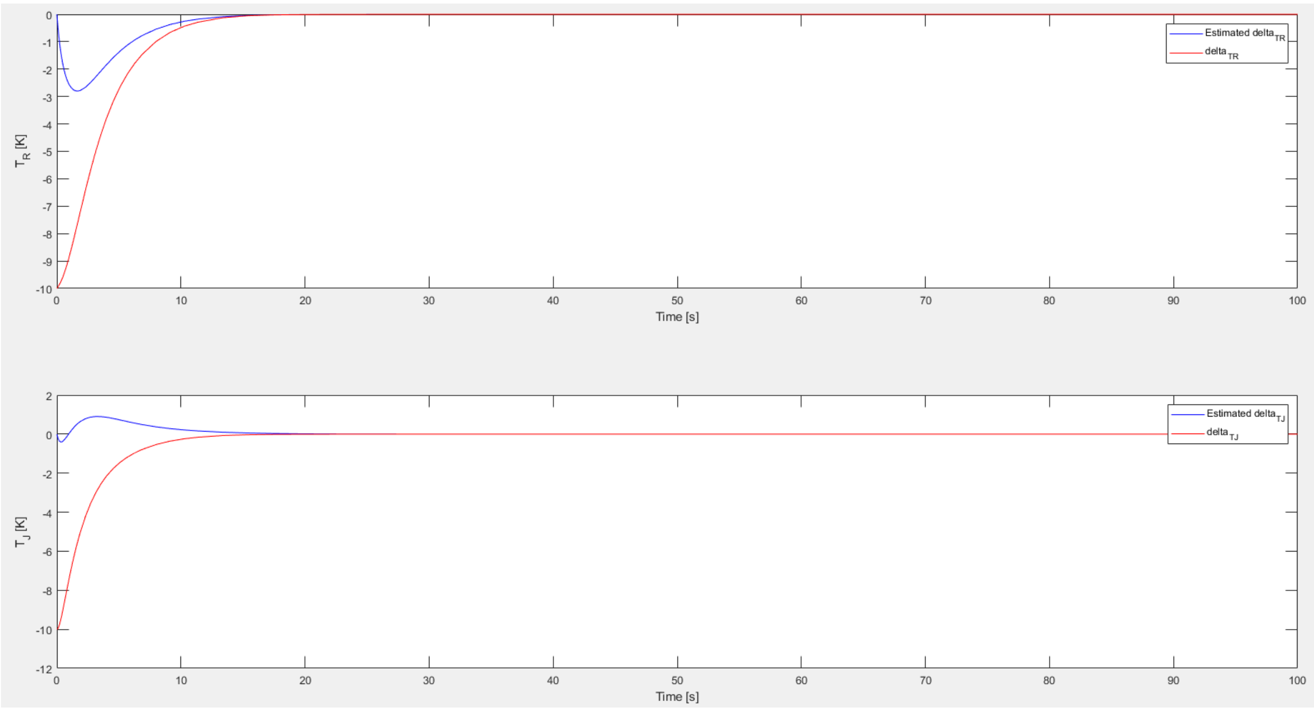
with initial condition x0,2

Q2 – Task9 – Plots

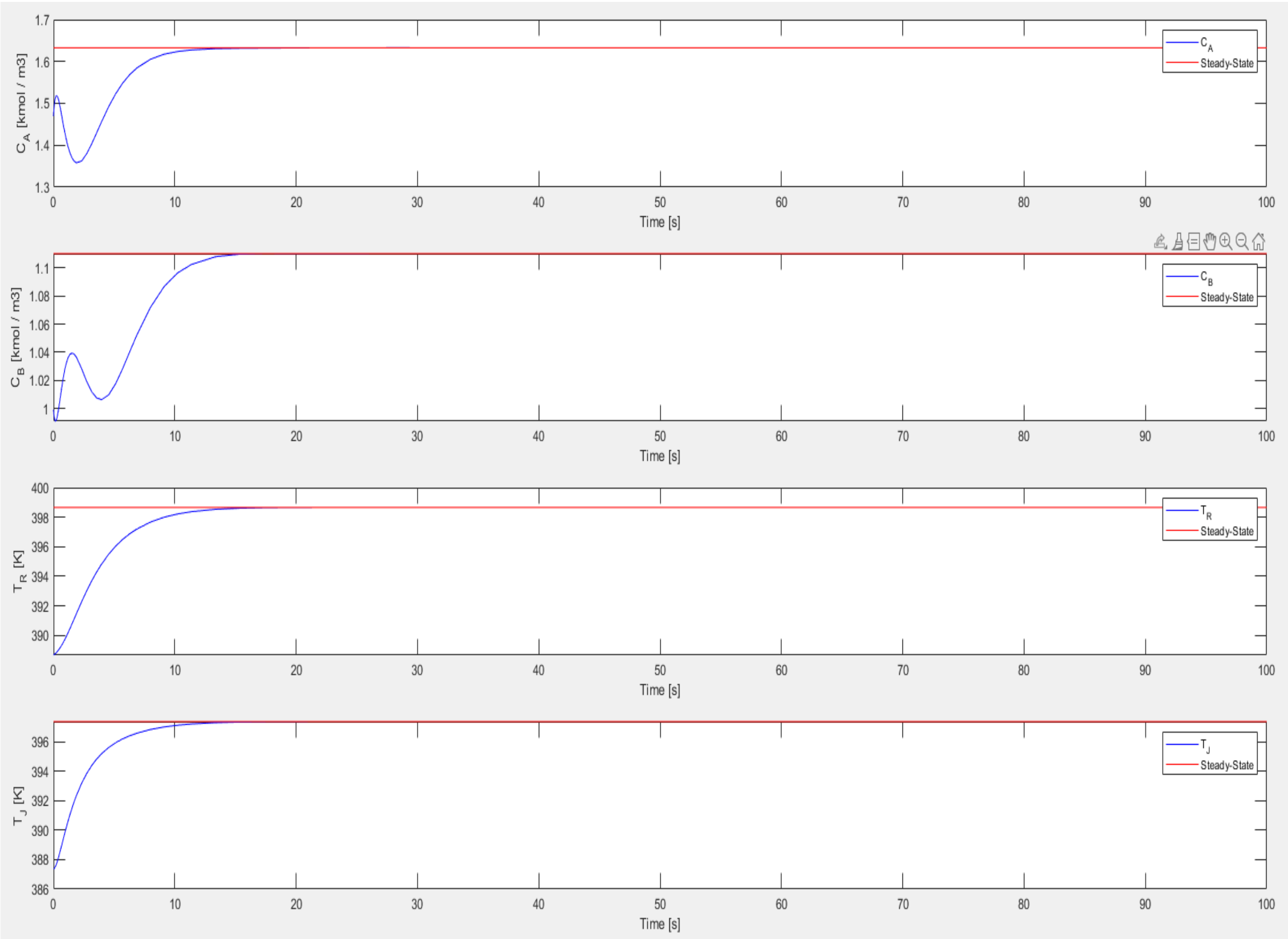


Observer State-Feedback Closed-Loop Simulation for ΔC_A, ΔC_B respectively

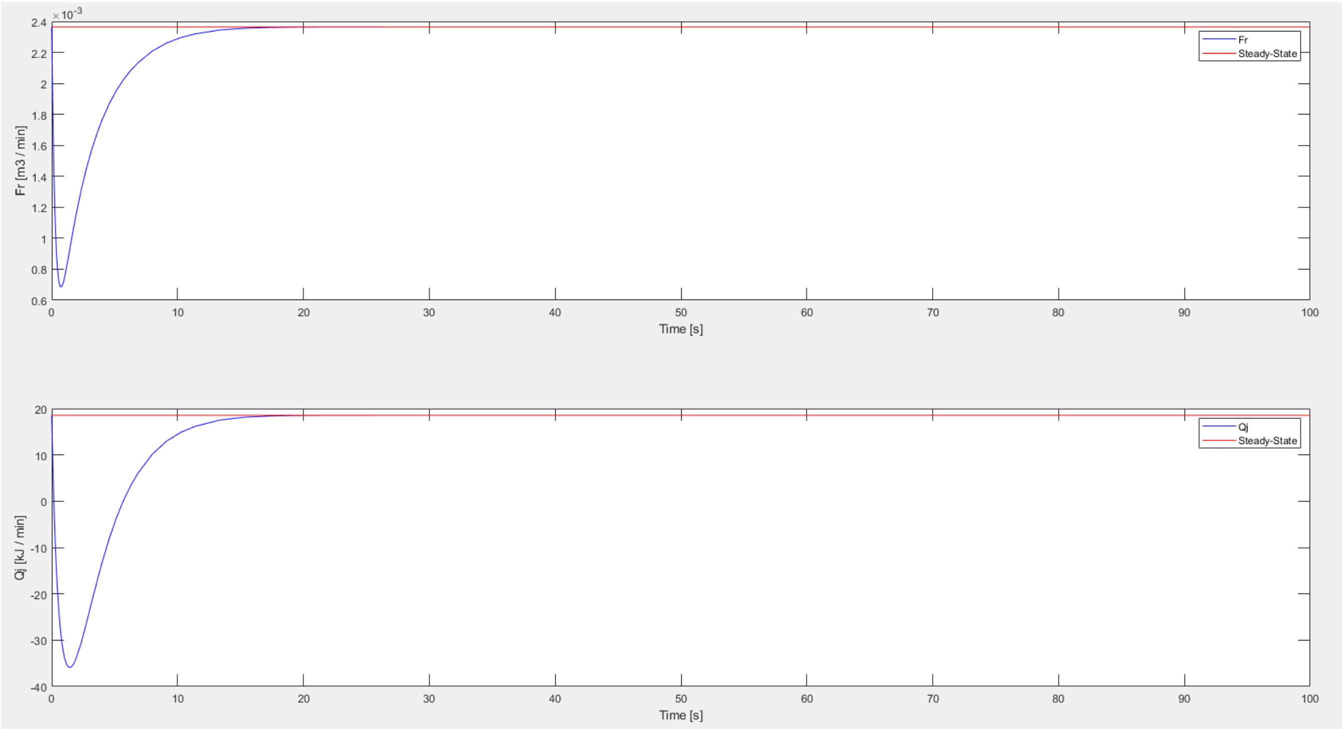
with initial condition x0,1



Observer State-Feedback Closed-Loop Simulation for ΔT_R, ΔT_J respectively with initial condition x0,1

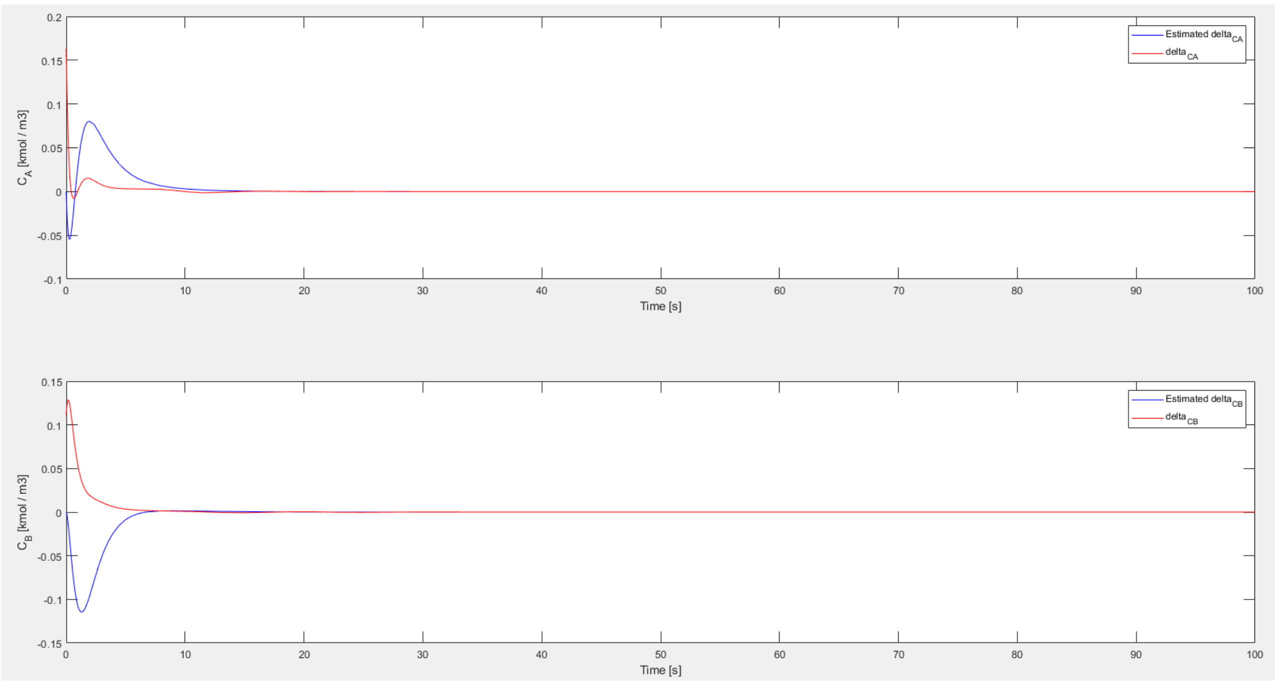


Observer State-Feedback Closed-Loop Simulation for C_A, C_B, T_R, T_J respectively with initial condition x0,1



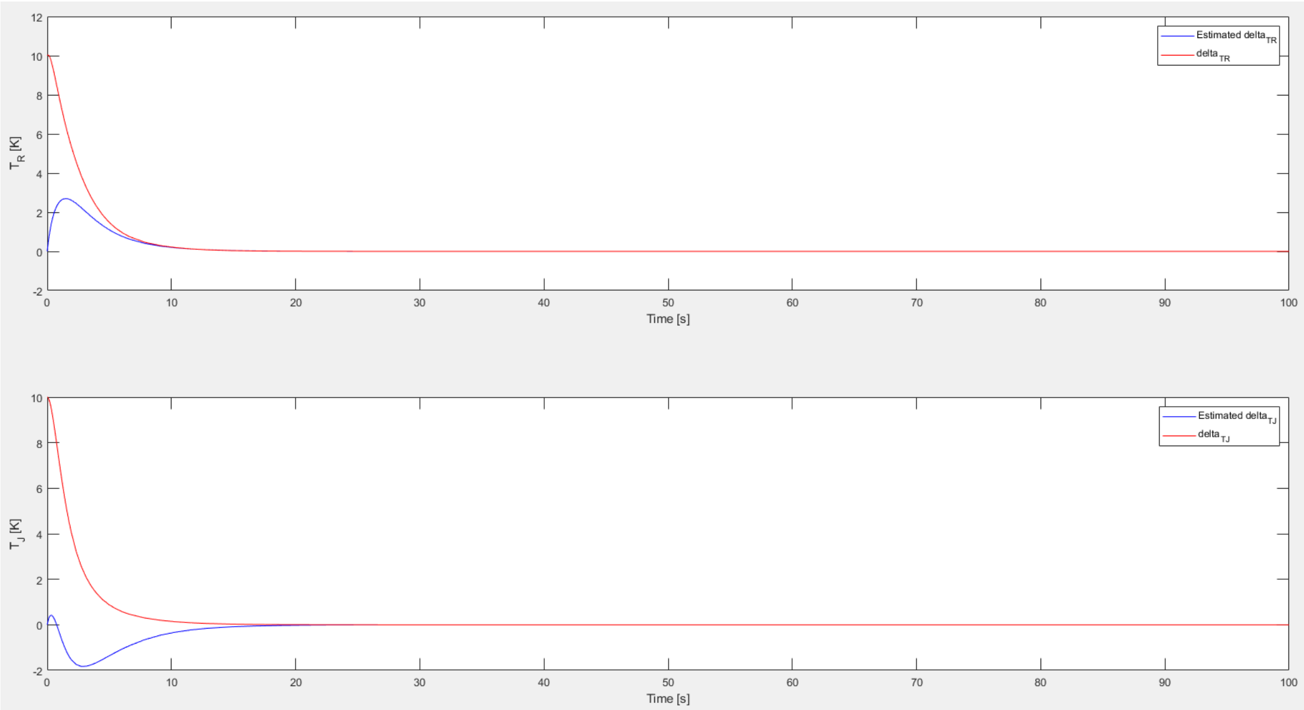
Observer State-Feedback Closed-Loop Simulation for Fr, Qj respectively

with initial condition x0,1



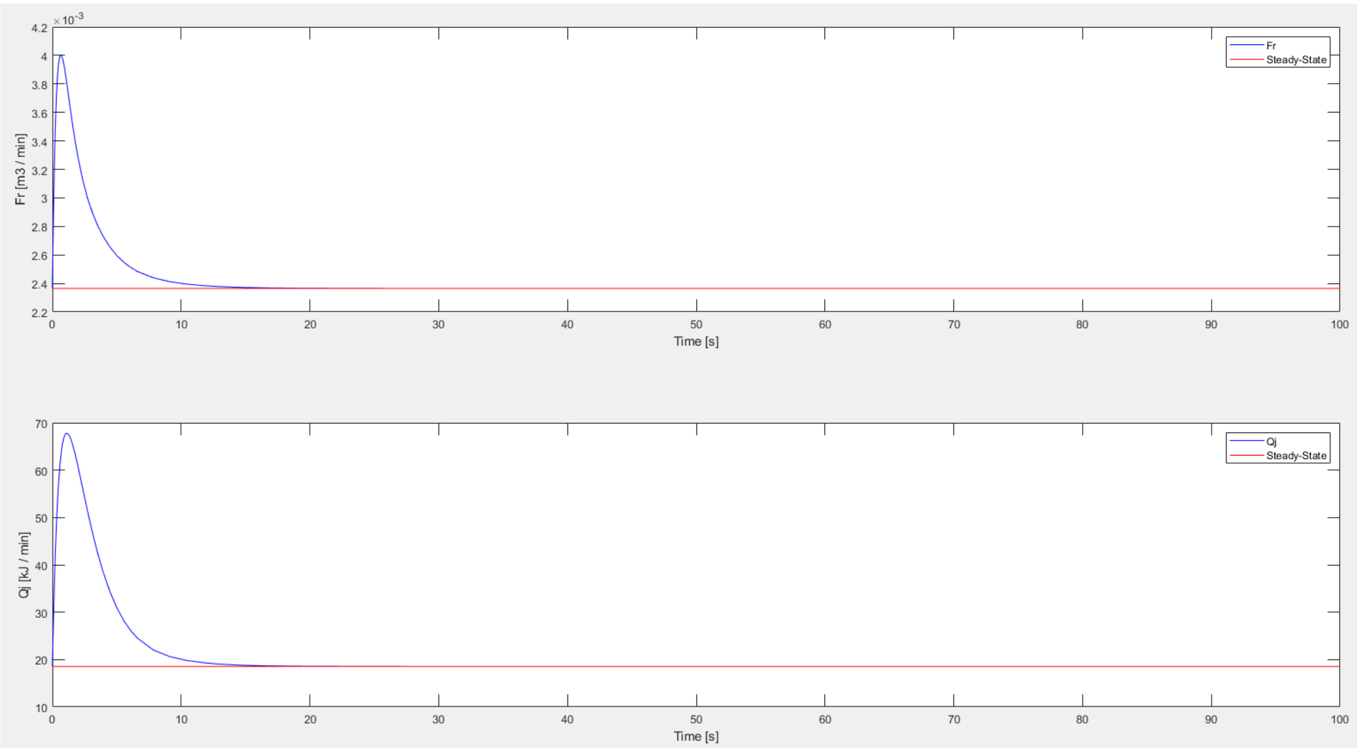
Observer State-Feedback Closed-Loop Simulation for ΔC_A, ΔC_B respectively

with initial condition x0,2



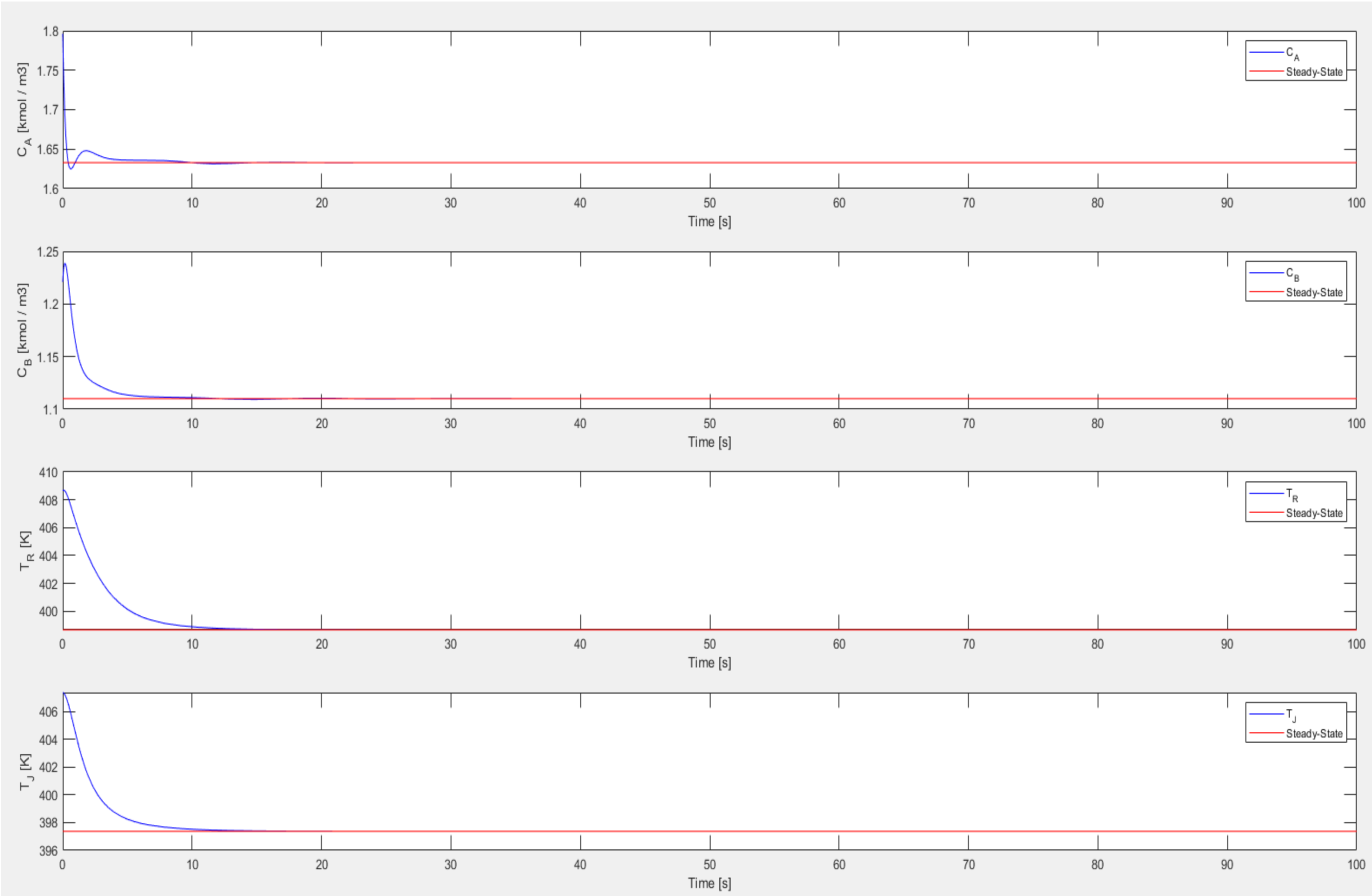
Observer State-Feedback Closed-Loop Simulation for ΔT_R, ΔT_J respectively

with initial condition x0,2



Observer State-Feedback Closed-Loop Simulation for Fr, Qj respectively

with initial condition x0,2



Observer State-Feedback Closed-Loop Simulation for C_A, C_B, T_R, T_J respectively with initial condition x0,2

```matlab
clear all
clc

% Q1-----------------------------------------------------------------

% Parameters
M = 15;     % Mass of the Robot [kg]
theta = 30; % Angle of Inclination of the Plane [degree]
ks = 3;     % Spring Constant [N / M]
g = 10;     % Gravitational Acceleration [m / s2]

% Inputs
% U = F - 5*ks - M*g*sin(theta)
% F                 => Linear Force Provided by the Motor [m3 / min]

% States
% x       (x1)      => Displacement of the Robot [m]
% v       (x2)      => Velocity of the Robot [m / s]

% State-space Matrices
A_mat = [0 1; -ks/M 0];
B_mat = [0; 1/M];
C_mat = eye(2);
D_mat = [0; 0];
sys_robot = ss(A_mat, B_mat, C_mat, D_mat);
poles_robot = eig(sys_robot);

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

% Q1 - 7) Use MATLAB to plot the closed-loop response.
% Plot the states and the force supplied by the motor
% over a time period of 20 seconds.

design1_eig = [-0.8, -1];

%K_d1 = -place(A_mat, B_mat, design1_eig);

K_d1 = [-9 -27];

poles_robot_d1 = eig(ss(A_mat + B_mat*K_d1, B_mat, C_mat, D_mat));
% New eigen values of the controlled robot are at -0.8 and -1.0

x0 = [-5; 0]; % initial conditions

t_span = [0 20];
[t, x_d1] = ode15s(@(t,x)ode_sys_controlled(t, x, K_d1), t_span, x0);
U_d1 = K_d1*x_d1';

% U = K*x = F - 5*ks - M*g*sin(theta)
F_d1 = U_d1 + 5*ks + M*g*sin(deg2rad(theta));
```

```
% figure(1);
% subplot(4,1,1);
% plot(t, x_d1(:,1),'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('Position');
%
% subplot(4,1,2);
% plot(t, x_d1(:,2),'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('Velocity');
%
% subplot(4,1,3);
% plot(t, F_d1, 'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('Force');
%
% subplot(4,1,4);
% plot(t, U_d1, 'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('U');

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

% Q1 - 9) Design(II): Design a controller for the system such that the
% problem that resulted from Design(I) is mitigated. Specify the eigen
% values of the controlled system (closed-loop poles). The controller
 shall
% move the robot to the position z = 0 on the inclined plane and the
 robot
% shall remain there afterwards. Determine the gain matrix K and show
 all
% the steps of your calculations.

design2_eig = 0.75*design1_eig;

K_d2 = [-3.75, -20.25];

%K_d2 = -place(A_mat, B_mat, design2_eig);

poles_robot_d2 = eig(ss(A_mat + B_mat*K_d2, B_mat, C_mat, D_mat));

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

% Q1 - 10) Use MATLAB to plot the closed-loop response.
% Plot the states and the force supplied by the motor
% over a time period of 20 seconds.

[t, x_d2] = ode15s(@(t,x)ode_sys_controlled(t, x, K_d2), t_span, x0);
U_d2 = K_d2*x_d2';

% U = K*x = F - 5*ks - M*g*sin(theta)
F_d2 = U_d2 + 5*ks + M*g*sin(deg2rad(theta));

% figure(2);
% subplot(4,1,1);
% plot(t, x_d2(:,1),'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('Position');
```

```
% subplot(4,1,2);
% plot(t, x_d2(:,2),'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('Velocity');
%
% subplot(4,1,3);
% plot(t, F_d2, 'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('Force');
%
% subplot(4,1,4);
% plot(t, U_d2, 'b-', 'LineWidth', 2);
% xlabel('t'); ylabel('U');
%--------------------------------------------------------------------

function f = ode_sys_controlled(t, x, K)

% Parameters

M = 15;      % Mass of the Robot [kg]
theta = 30; % Angle of Inclination of the Plane [degree]
ks = 3;      % Spring Constant [N / M]
g = 10;      % Gravitational Acceleration [m / s2]


% State space matrices
A_mat = [0 1; -ks/M 0];

B_mat = [0; 1/M];

% ODE Right Hand Sides
% x_dot = A*x + B*u
% u = K*x

f = A_mat*x + B_mat*K*x;
end
```

Published with MATLAB® R2020b

```
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

```matlab
% reactor_model_ode_rhs.m

function f = reactor_model_ode_rhs(x, u)

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

% States
C_A = x(1);             % Concentration of Component A [kmol / m3]
C_B = x(2);             % Concentration of Component B [kmol / m3]
T_R = x(3);             % Reactor Temperature [K]
T_J = x(4);             % Jacket Temperature  [K]

% Inputs
Fr = u(1);              % Feed Volumetric Flowrate [m3 / min]
Qj = u(2);              % Heat Removal by The Jacket [kJ / min]

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

% 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]

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

% Differential Equations

% dC_A / dt
f(1,1) = ((Fr / V) * (C_A_in - C_A)) - (k_01*exp(-E_R1 / T_R)*C_A);

% dC_B / dt
f(2,1) = ((-Fr / V)*C_B) + (k_01*exp(-E_R1 / T_R)*C_A) - (k_02*exp(-
E_R2 / T_R)*C_B);

% dT_R / dt
f(3,1) = ((Fr / V)*(T_in - T_R)) - ((k_01*exp(-E_R1 /
 T_R)*C_A*deltaH_R1) / (rho*cp)) ...
    - ((k_02*exp(-E_R2 / T_R)*C_B*deltaH_R2) / (rho*cp)) - ((kA*(T_R -
 T_J)) / (rho*cp*V));
```

```
% dT_J / dt
f(4,1) = ((kA*(T_R - T_J)) - Qj) / (m_j*cp_j);

end
```

```
Not enough input arguments.

Error in reactor_model_ode_rhs (line 9)
C_A = x(1);             % Concentration of Component A [kmol / m3]
```

Published with MATLAB® R2020b

```
% reactor_nonlinear_sfcn.m

% ---------------------------------------------------------------------
% Simulink Function for the Simulation of the Non-linear System
% ---------------------------------------------------------------------

function [sys, x0, str, ts] = reactor_nonlinear_sfcn(t, x, u, flag)


% Choose the function performed currently by the S-function
switch flag

    % Initialization
    case 0

        str = [];       % Empty (default behavior)
        ts  = [0 0];    % Default values for continuous systems

        % Dimensions of the system (states, inputs and outputs)
        sys_dims = simsizes;    % simsizes: MATLAB construct for
                                %           initialization purposes

        % Problem-specific dimensions
        % The names of the fields of sys_dims are expected by Simulink
        sys_dims.NumContStates  = 4;  % Num. continuous states
        sys_dims.NumDiscStates  = 0;  % Num. discontinuous states
        sys_dims.NumInputs      = 2;  % Num. of inputs (Fr, Qj)
        sys_dims.DirFeedthrough = 0;  % Num. of feedthroughs (matrix
 D)
        sys_dims.NumSampleTimes = 1;  % Default for continuous systems

        sys_dims.NumOutputs = 4; % Num. of outputs (C_A, C_B, T_R,
 T_J;
                                 % The measurements are specified in
                                 % the block diagram with the matrix
 C)

        % Output: structure with system dimensions
        sys = simsizes(sys_dims);

        % actual initial conditions
        x0_actual = [0; 0; 387.05; 387.05];

        % part7 initial conditions
        C_A_ss = 1.6329;
        C_B_ss = 1.1101;
        T_R_ss = 398.6581;
        T_J_ss = 397.3736;

        x0_7_1 = [C_A_ss*0.9; C_B_ss*0.9; T_R_ss - 10; T_J_ss - 10];
        x0_7_2 = [C_A_ss*1.1; C_B_ss*1.1; T_R_ss + 10; T_J_ss + 10];
```

```
        % USER: Output: initial conditions
        x0 = x0_7_1;


    % Evaluation of the derivatives
    case 1

        % Output: RHS of the ODE system
        sys = reactor_model_ode_rhs(x, u);


    % Evaluation of the outputs (y = C*x)
    case 3

        % System outputs
        % (the measurements are specified in
        % the block diagram with the matrix C)
        sys = x;


    % Additional flags (values = 2, 4 and 9)
    case {2 4 9}

        sys = [];

    otherwise

        error('Unknown flag');

end

% ---------------------------------------------------------------------
% EOF
% ---------------------------------------------------------------------
```

```
Not enough input arguments.

Error in reactor_nonlinear_sfcn (line 12)
switch flag
```

Published with MATLAB® R2020b

```
% plot_nonlinear_vs_linear.m

% This file plots the results from the simulation
% of reactor_nonlinear_vs_linear_model_simulation.mdl model

close all;
clc;

% C_A_ss = 1.6329*ones(length(tout),1);
% C_B_ss = 1.1101*ones(length(tout),1);
% T_R_ss = 398.6581*ones(length(tout),1);
% T_J_ss = 397.3736*ones(length(tout),1);

% figure(1);
% subplot(2,1,1)
% plot(tout, l_C_A, 'b', tout, nl_C_A, 'r', tout, C_A_ss, 'g');
% xlabel('Time [s]');
% ylabel('C_A [kmol / m3]');
% legend('Linear', 'Non-Linear', 'Steady-State');
%
% subplot(2,1,2)
% plot(tout, l_C_B, 'b', tout, nl_C_B, 'r', tout, C_B_ss, 'g');
% xlabel('Time [s]');
% ylabel('C_B [kmol / m3]');
% legend('Linear', 'Non-Linear', 'Steady-State');
%
% figure(2)
% subplot(2,1,1)
% plot(tout, l_T_R, 'b', tout, nl_T_R, 'r', tout, T_R_ss, 'g');
% xlabel('Time [s]');
% ylabel('T_R [K]');
% legend('Linear', 'Non-Linear', 'Steady-State');
%
% subplot(2,1,2)
% plot(tout, l_T_J, 'b', tout, nl_T_J, 'r',  tout, T_J_ss, 'g');
% xlabel('Time [s]');
% ylabel('T_J [K]');
% legend('Linear', 'Non-Linear', 'Steady-State');
```

Published with MATLAB® R2020b

```
% plot_feedback_feedback_full_state.m

% This file plots the results from the simulation
% of reactor_feedback_full_state.mdl model

close all;
clc;

% C_A_ss = 1.6329*ones(length(tout),1);
% C_B_ss = 1.1101*ones(length(tout),1);
% T_R_ss = 398.6581*ones(length(tout),1);
% T_J_ss = 397.3736*ones(length(tout),1);
%
% Fr_ss = 0.002365*ones(length(tout),1);
% Qj_ss = 18.5583*ones(length(tout),1);

% figure(1);
% subplot(2,1,1)
% plot(tout, C_A_feedback, 'b', tout, C_A_ss, 'r');
% xlabel('Time [s]');
% ylabel('C_A [kmol / m3]');
% legend('C_A', 'Steady-State');
%
% subplot(2,1,2)
% plot(tout, C_B_feedback, 'b', tout, C_B_ss, 'r');
% xlabel('Time [s]');
% ylabel('C_B [kmol / m3]');
% legend('C_B', 'Steady-State');
%
% figure(2)
% subplot(2,1,1)
% plot(tout, T_R_feedback, 'b', tout, T_R_ss, 'r');
% xlabel('Time [s]');
% ylabel('T_R [K]');
% legend('T_R', 'Steady-State');
%
% subplot(2,1,2)
% plot(tout, T_J_feedback, 'b', tout, T_J_ss, 'r');
% xlabel('Time [s]');
% ylabel('T_J [K]');
% legend('T_J', 'Steady-State');
%
% figure(3)
% subplot(2,1,1)
% plot(tout, Fr_feedback, 'b', tout, Fr_ss, 'r');
% xlabel('Time [s]');
% ylabel('Fr [m3 / min]');
% legend('Fr', 'Steady-State');
%
% subplot(2,1,2)
% plot(tout, Qj_feedback, 'b', tout, Qj_ss, 'r');
% xlabel('Time [s]');
```

```
% ylabel('Qj [kJ / min]');
% legend('Qj', 'Steady-State');
```

Published with MATLAB® R2020b

```
% plot_observer.m

% This file plots the results from the simulation
% of reactor_observer.mdl model

close all;
clc;

% C_A_ss = 1.6329*ones(length(tout),1);
% C_B_ss = 1.1101*ones(length(tout),1);
% T_R_ss = 398.6581*ones(length(tout),1);
% T_J_ss = 397.3736*ones(length(tout),1);

% figure(1);
% subplot(2,1,1)
% plot(tout, d_C_A_obs, 'b', tout, d_C_A, 'r');
% xlabel('Time [s]');
% ylabel('C_A [kmol / m3]');
% legend('Estimated delta_C_A', 'delta_C_A');
%
% subplot(2,1,2)
% plot(tout, d_C_B_obs, 'b', tout, d_C_B, 'r');
% xlabel('Time [s]');
% ylabel('C_B [kmol / m3]');
% legend('Estimated delta_C_B', 'delta_C_B');
%
% figure(2)
% subplot(2,1,1)
% plot(tout, d_T_R_obs, 'b', tout, d_T_R, 'r');
% xlabel('Time [s]');
% ylabel('T_R [K]');
% legend('Estimated delta_T_R', 'delta_T_R');
%
% subplot(2,1,2)
% plot(tout, d_T_J_obs, 'b', tout, d_T_J, 'r');
% xlabel('Time [s]');
% ylabel('T_J [K]');
% legend('Estimated delta_T_J', 'delta_T_J');
%
% figure(3)
% subplot(4,1,1)
% plot(tout, C_A_obs, 'b', tout, C_A_ss, 'r');
% xlabel('Time [s]');
% ylabel('C_A [kmol / m3]');
% legend('C_A', 'Steady-State');
%
% subplot(4,1,2)
% plot(tout, C_B_obs, 'b', tout, C_B_ss, 'r');
% xlabel('Time [s]');
% ylabel('C_B [kmol / m3]');
% legend('C_B', 'Steady-State');
%
```

```
% subplot(4,1,3)
% plot(tout, T_R_obs, 'b', tout, T_R_ss, 'r');
% xlabel('Time [s]');
% ylabel('T_R [K]');
% legend('T_R', 'Steady-State');
%
% subplot(4,1,4)
% plot(tout, T_J_obs, 'b', tout, T_J_ss, 'r');
% xlabel('Time [s]');
% ylabel('T_J [K]');
% legend('T_J', 'Steady-State');
```

Published with MATLAB® R2020b

```
% plot_feedback_observer.m

% This file plots the results from the simulation
% of reactor_feedback_observer.mdl model

close all;
clc;

% C_A_ss = 1.6329*ones(length(tout),1);
% C_B_ss = 1.1101*ones(length(tout),1);
% T_R_ss = 398.6581*ones(length(tout),1);
% T_J_ss = 397.3736*ones(length(tout),1);
%
% Fr_ss = 0.002365*ones(length(tout),1);
% Qj_ss = 18.5583*ones(length(tout),1);

% figure(1);
% subplot(2,1,1)
% plot(tout, d_C_A_f_obs, 'b', tout, d_C_A_f, 'r');
% xlabel('Time [s]');
% ylabel('C_A [kmol / m3]');
% legend('Estimated delta_C_A', 'delta_C_A');
%
% subplot(2,1,2)
% plot(tout, d_C_B_f_obs, 'b', tout, d_C_B_f, 'r');
% xlabel('Time [s]');
% ylabel('C_B [kmol / m3]');
% legend('Estimated delta_C_B', 'delta_C_B');
%
% figure(2)
% subplot(2,1,1)
% plot(tout, d_T_R_f_obs, 'b', tout, d_T_R_f, 'r');
% xlabel('Time [s]');
% ylabel('T_R [K]');
% legend('Estimated delta_T_R', 'delta_T_R');
%
% subplot(2,1,2)
% plot(tout, d_T_J_f_obs, 'b', tout, d_T_J_f, 'r');
% xlabel('Time [s]');
% ylabel('T_J [K]');
% legend('Estimated delta_T_J', 'delta_T_J');
%
% figure(3)
% subplot(4,1,1)
% plot(tout, C_A_f_obs, 'b', tout, C_A_ss, 'r');
% xlabel('Time [s]');
% ylabel('C_A [kmol / m3]');
% legend('C_A', 'Steady-State');
%
% subplot(4,1,2)
% plot(tout, C_B_f_obs, 'b', tout, C_B_ss, 'r');
% xlabel('Time [s]');
```

```
% ylabel('C_B [kmol / m3]');
% legend('C_B', 'Steady-State');
%
% subplot(4,1,3)
% plot(tout, T_R_f_obs, 'b', tout, T_R_ss, 'r');
% xlabel('Time [s]');
% ylabel('T_R [K]');
% legend('T_R', 'Steady-State');
%
% subplot(4,1,4)
% plot(tout, T_J_f_obs, 'b', tout, T_J_ss, 'r');
% xlabel('Time [s]');
% ylabel('T_J [K]');
% legend('T_J', 'Steady-State');
%
% figure(4)
% subplot(2,1,1)
% plot(tout, Fr_f_obs, 'b', tout, Fr_ss, 'r');
% xlabel('Time [s]');
% ylabel('Fr [m3 / min]');
% legend('Fr', 'Steady-State');
%
% subplot(2,1,2)
% plot(tout, Qj_f_obs, 'b', tout, Qj_ss, 'r');
% xlabel('Time [s]');
% ylabel('Qj [kJ / min]');
% legend('Qj', 'Steady-State');
```

Published with MATLAB® R2020b