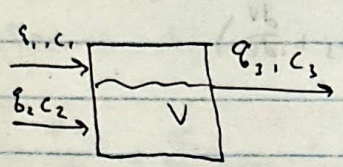
CHE 341

CHE 341

Assignment 1

① 01/14/24

1.



1. Objective

- Derive a dynamic balance for outlet conc ; vessel vol. as a $f(\dot{v}, C_1, C_2)$

2. Schematic → above.

3. Assumptions:

- well mixed vessel
- no rxn
- no heat added
- $P_1 = P_2 = P_3 =$ constant
- C_{P_i} = constant
- no shaft work.

4. Spatial dependence

- no vars as a $f(x,y,z)$ ∴ no spatial dependence

5. Dynamic balances:

$$\frac{dm}{dt} = \sum \dot{m}_{in} - \sum \dot{m}_{out}$$

∵ $m_i = \rho_i V$; $\dot{m}_i = \rho_i \dot{v}_i$; $\rho_1 = \rho_2 = \rho_3$ (well mixed)

$$\frac{d(\rho V)}{dt} = \dot{v}_1 \rho + \dot{v}_2 \rho - \dot{v}_3 \rho_3$$

∵ ρ = constant

$$\rho \frac{dV}{dt} = \dot{v}_1 \rho + \dot{v}_2 \rho - \dot{v}_3 \rho \quad \div \rho$$

$$\frac{dV}{dt} = \dot{v}_1 + \dot{v}_2 - \dot{v}_3$$

Species

$$\frac{dN_i}{dt} = \sum \dot{n}_{i,in} - \sum \dot{n}_{i,out} + r_i V \quad \text{(no rxn.)}$$

∵ $N_i = C_i V_i$; $\dot{n}_i = C_i \dot{v}_i$

$$\frac{d(C_3 V)}{dt} = C_1 \dot{v}_1 + C_2 \dot{v}_2 - C_3 \dot{v}_3$$

$$V \frac{dC_3}{dt} + C_3 \frac{dV}{dt} = C_1 \dot{v}_1 + C_2 \dot{v}_2 - C_3 \dot{v}_3$$

$$V \frac{dC_3}{dt} = C_1 \dot{v}_1 + C_2 \dot{v}_2 - C_3 \dot{v}_3 - C_3 \frac{dV}{dt}$$

$$\frac{dC_3}{dt} = \frac{C_1 q_1 + C_2 q_2}{V} - \frac{C_3}{V}\left(q_3 + \frac{dV}{dt}\right)$$

6. Other rel^{n}s:

$m = \rho V \qquad \dot{n} = cq$

~~xxx~~

7. DOF:

7 vars ; 4 eqns => 3 DOF

8. Inputs:

q_1, C_1, q_2, C_2

9. ~~Inputs~~ Outputs V (Stat^{n}) ; C_3 (process var) ; q_3 (state)

10. Simplify → DONE.

11. Steady Stat^{n}.

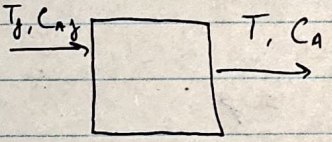
V = constant ; $\frac{dV}{dt} = 0$.

$q_3 = q_1 + q_2$

$$V\frac{dC_3}{dt} = C_1 q_1 + C_2 q_2 - C_3 q_3$$

2. Python script.

3. ~~q_2 Obj~~



1. objective.

- Develop the concentration response ; T response

2. schematic → Above ↑

3. Assumptions

- constant V ; $\frac{dV}{dt} = 0$
- $C_{p,A}$ = constant
- $\rho = 100 \frac{m^3}{hr}$
- well mixed
- P_A = constant
- no rxn ; no shaft work

4. Spatial dependence => no var is $f(x, y, z)$ → NONE.

(3)

Species balance.

$$\frac{dn_A}{dt} = \sum \dot{n}_{A,in} - \sum \dot{n}_{A,out} + \sum \dot{n}_{A,gen} - \sum \dot{n}_{A,cons}$$ → no rxn.

$$\frac{dn_A}{dt} = \dot{n}_{Af} - \dot{n}_A$$

$$\dot{n}_A = C_A \dot{v}$$

$$\frac{dC_A V}{dt} = C_{Af} \dot{v}_f - C_A \dot{v}$$

V, constant, $\therefore \dot{v}_f = \dot{v}$

$$V\left(\frac{dC_A}{dt}\right) = \dot{v}(C_{Af} - C_A)$$

$$\frac{dC_A}{dt} = \frac{\dot{v}}{V}(C_{Af} - C_A)$$

Energy Balance

$$mC_p \frac{dT}{dt} = \sum \dot{m}_{in} C_p (T_{in} - T_{ref}) - \sum \dot{m}_{out} C_p (T_{out} - T_{ref}) + Q + W_s$$

- $Q = 0$
- $W_s = 0$

$$mC_p \frac{dT}{dt} = \sum \dot{m}_{in} C_p (T_{in} - T_{ref}) - \sum \dot{m}_{out} C_p (T_{out} - T_{ref})$$

C_p = constant; cancel out.

$$m \frac{dT}{dt} = \sum \dot{m}_{in} (T_{in} - T_{ref}) - \sum \dot{m}_{out} (T_{out} - T_{ref})$$

$m = \rho V$; $\dot{m} = \rho \dot{v}$

$$\rho V \frac{dT}{dt} = \rho \dot{v}_f (T_{in} - T_{ref}) - \rho \dot{v} (T_{out} - T_{ref})$$

ρ at all same.

$$V \frac{dT}{dt} = \dot{v}_f (T_{in} - T_{ref}) - \dot{v} (T_{out} - T_{ref})$$

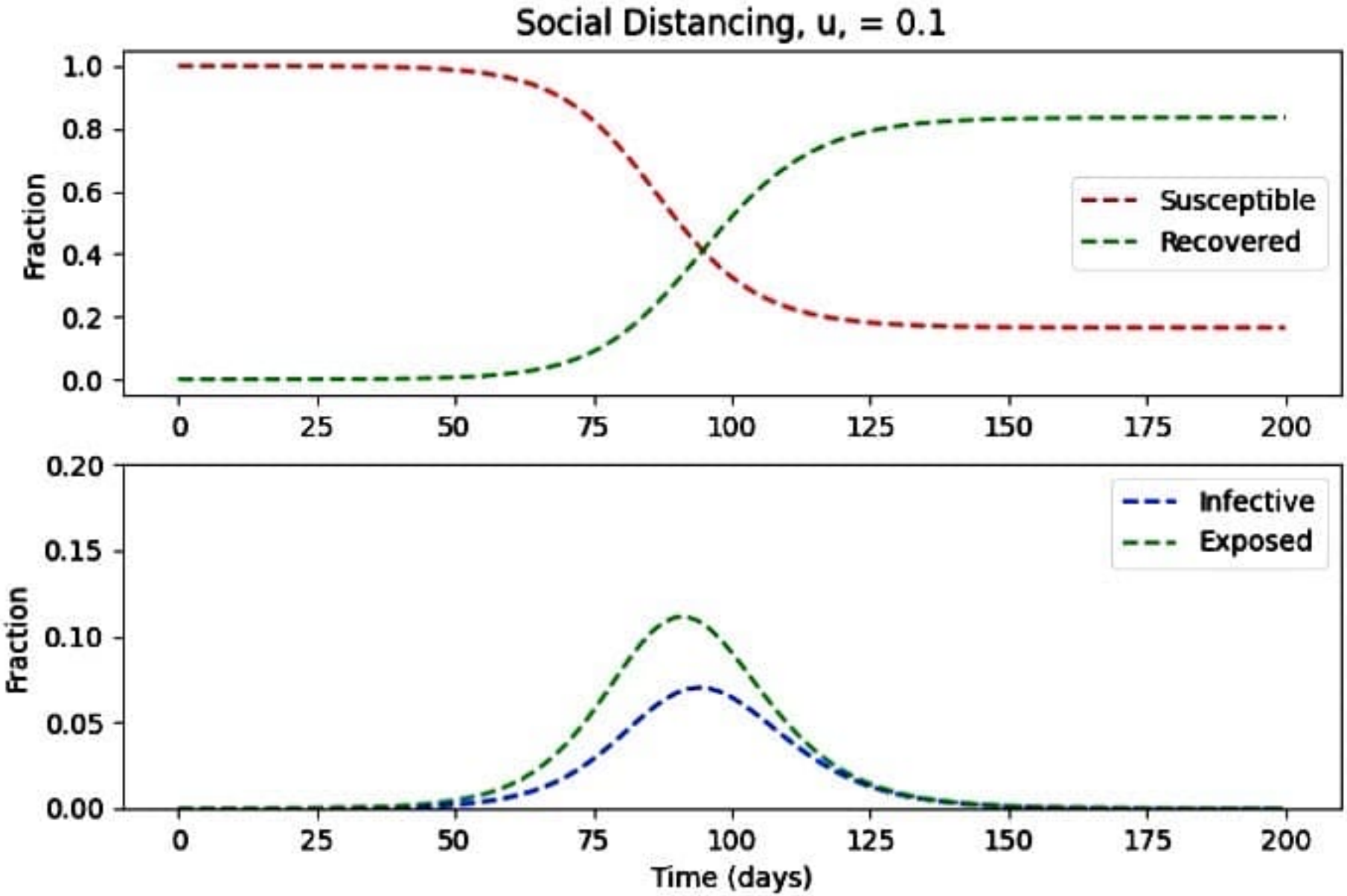
since V = constant → $\dot{v}_f = \dot{v}$.

$$V \frac{dT}{dt} = \dot{v}(T_{in} - T_{ref}) - \dot{v}(T_{out} - T_{ref})$$

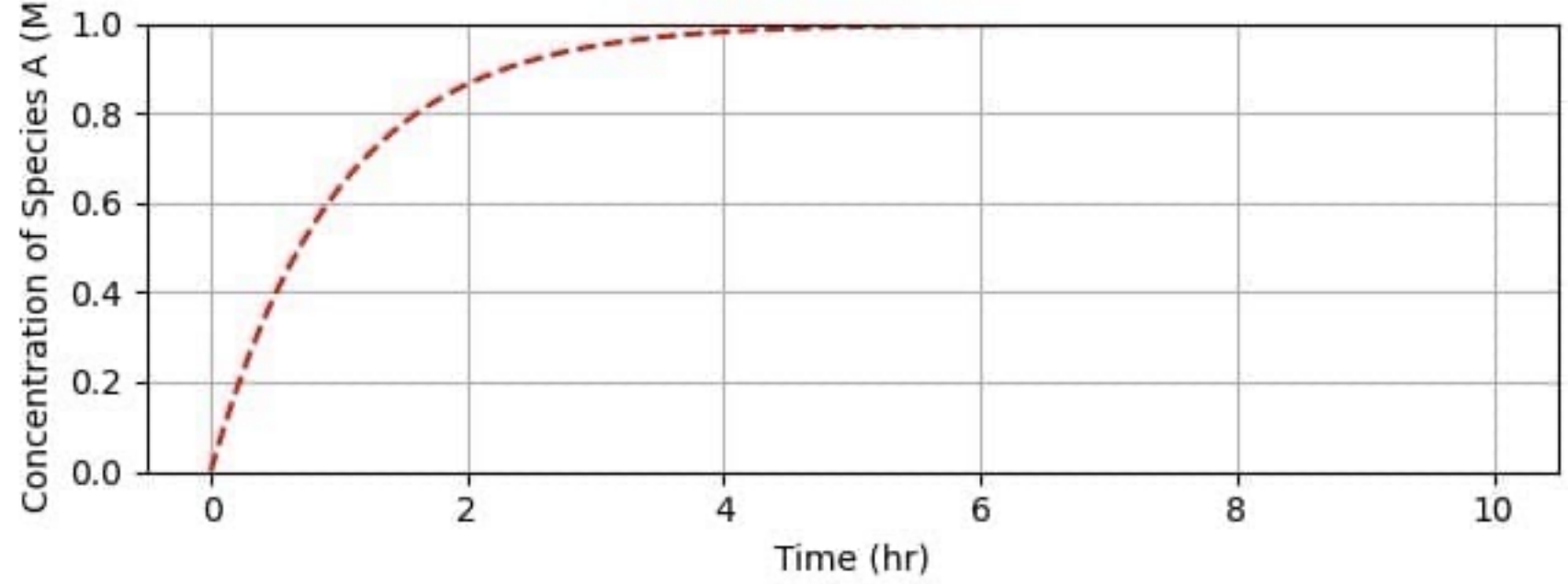
$$V \frac{dT}{dt} = \dot{v}\left[(T_{in} - T_{ref}) - (T_{out} - T_{ref})\right]$$

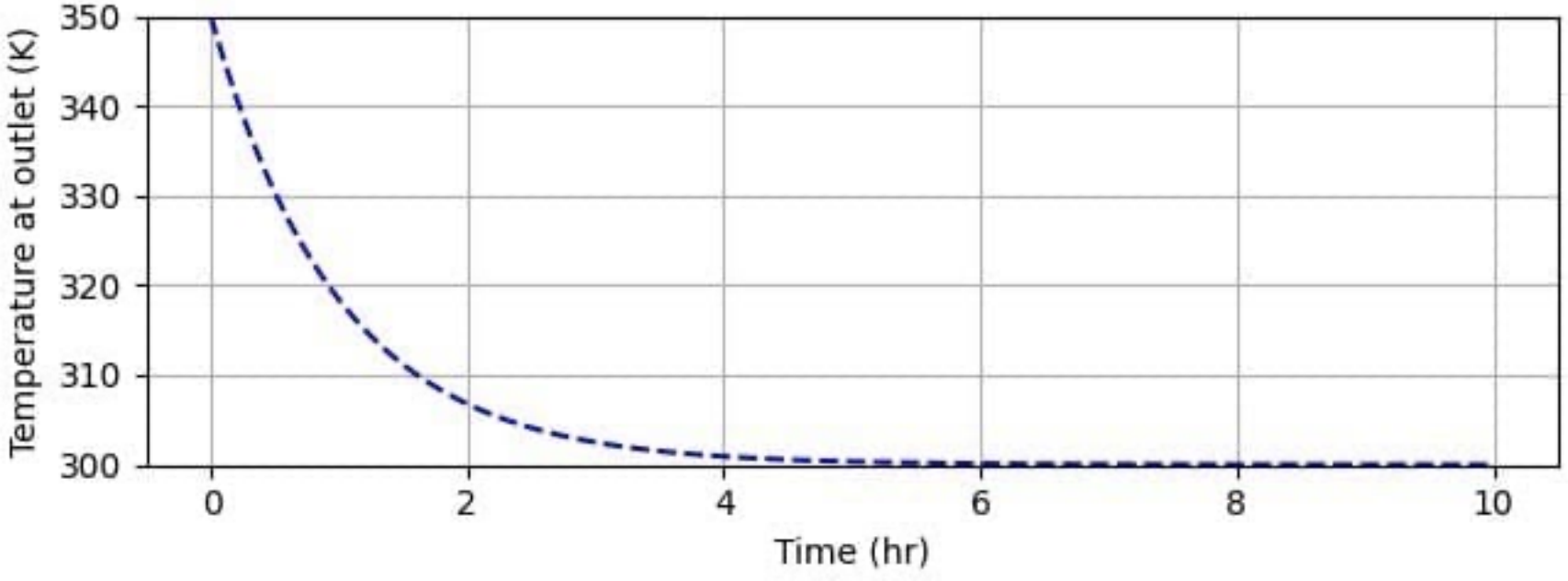
$$\frac{dT}{dt} = \frac{\dot{v}}{V}\left[(T_{in} - T_{ref}) - (T_{out} - T_{ref})\right]$$

$$\frac{dT}{dt} = \frac{\dot{v}}{V}\left[T_{in} - T_{ref} - T_{out} + T_{ref}\right] = \frac{\dot{v}}{V}\left[T_{in} - T_{out}\right]$$


Social Distancing, u, = 0.1
Fraction
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200
Susceptible
Recovered
Fraction
0.20
0.15
0.10
0.05
0.00
Infective
Exposed
Time (days)

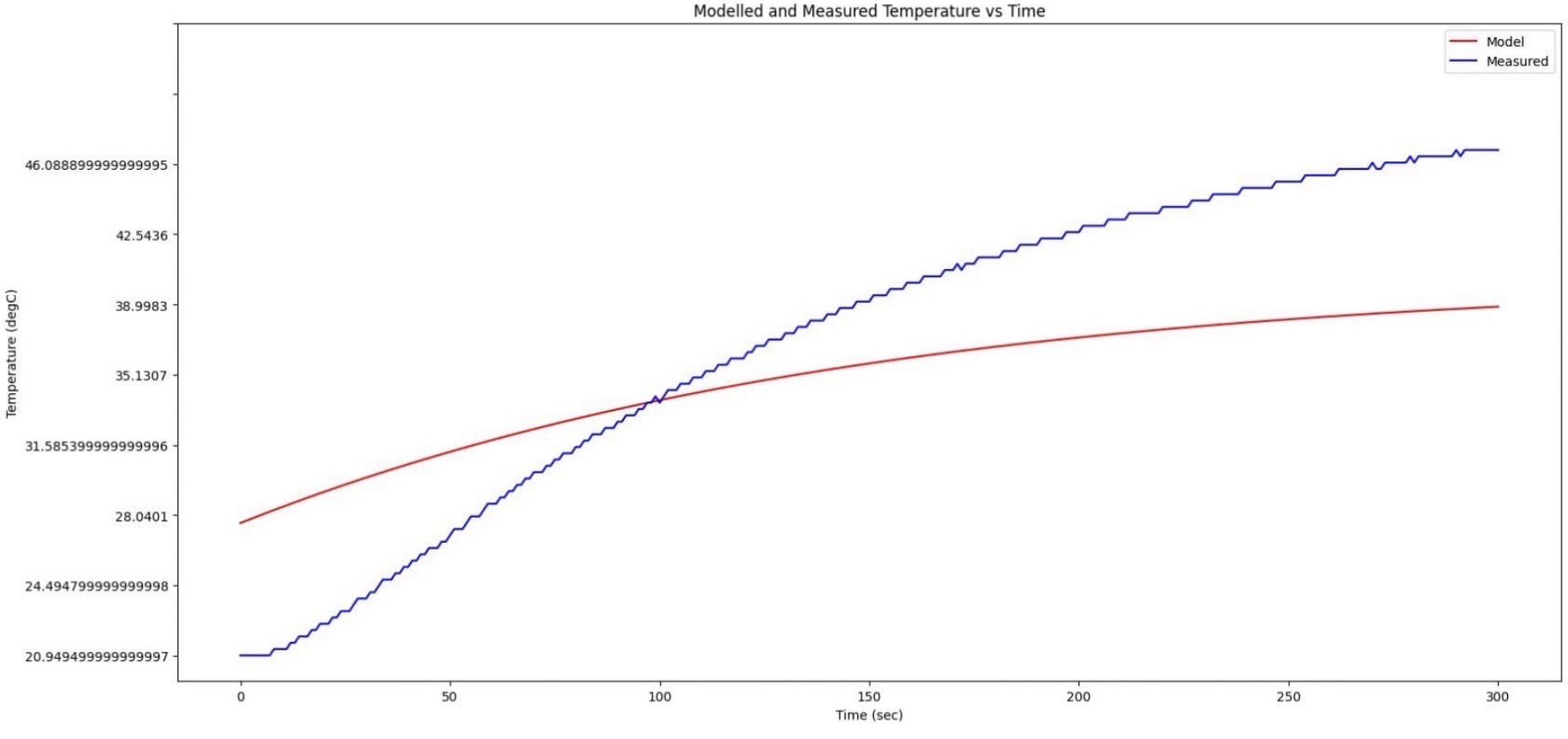
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint


V = 100 # m3
q = 100 # m3/hr

# z = [Ca, T]
# dz = [dCa_dt, dT_dt]
def func(z, t):
    q = 100 # volumetric flow rate -- 100 m^3/hr
    V = 100 # volume of vessel, 100 m^3
    Ca = z[0] # concentration
    Ca_f = 1 # inlet concentration
    T = z[1] # temperature at outlet
    Tf = 300 # inlet temperature
    dCa_dt = q/V * (Ca_f - Ca)
    dT_dt = q/V * (Tf - T)

    return [dCa_dt, dT_dt]

t = np.linspace(0, 10, 100)


z0 = [0, 350] # z[0] = Ca0, z[1] = T0


z = odeint(func, z0, t)
Ca = z[:, 0]
T = z[:, 1]

fig, axes = plt.subplots(2, 1, layout="constrained")
axes[0].plot(t, Ca, "r--")
axes[0].set_ylim(0, 1)
axes[0].set_xlabel("Time (hr)")
axes[0].set_ylabel("Concentration of Species A (M)")
axes[0].grid(True)

axes[1].plot(t, T, "b--")
axes[1].set_ylim(300, 350)
axes[1].set_xlabel("Time (hr)")
axes[1].set_ylabel("Temperature at outlet (K)")
axes[1].grid(True)

plt.show()
```

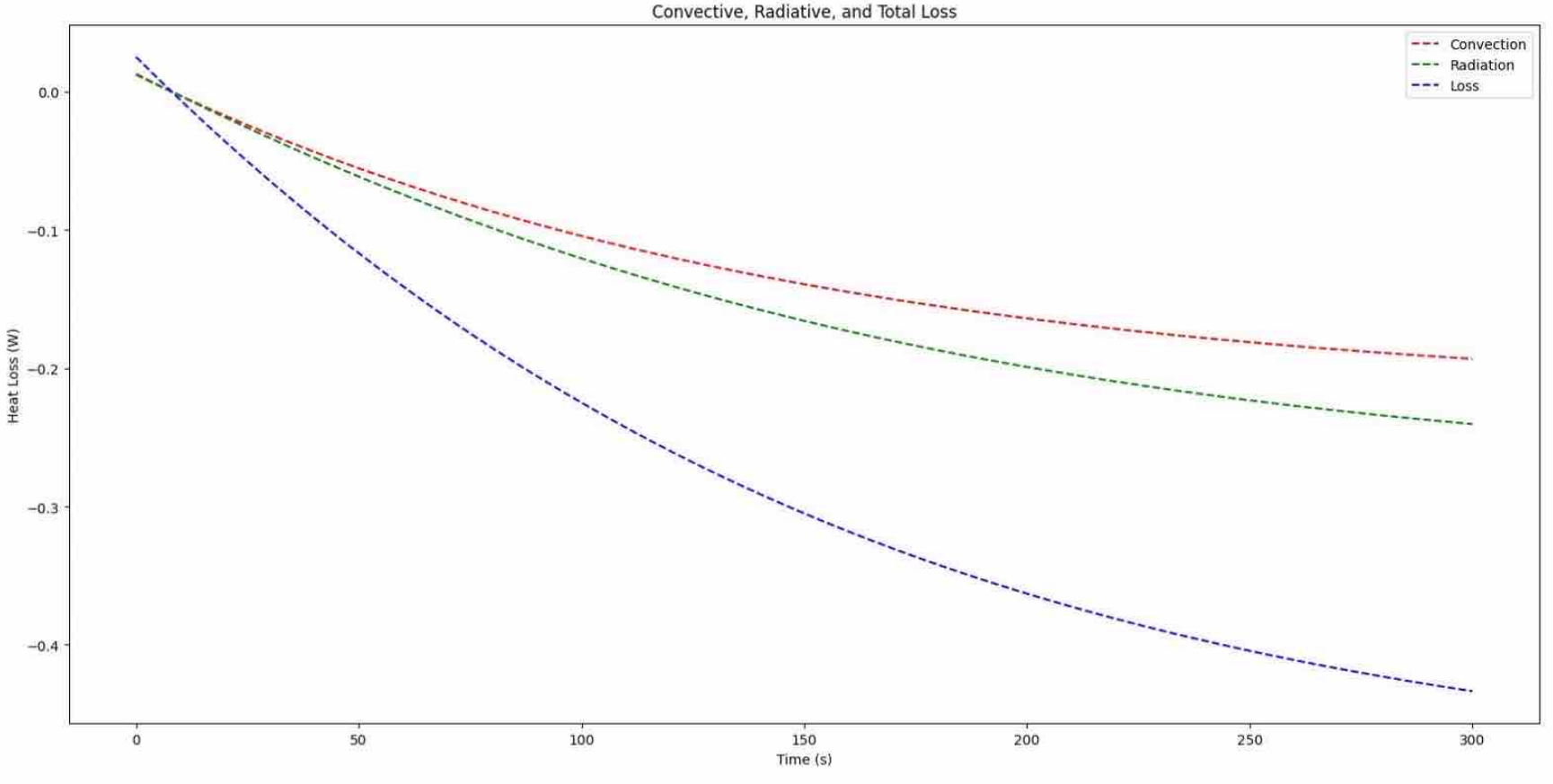

Concentration of Species A (M
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Time (hr)


Temperature at outlet (K)
350
340
330
320
310
300
0
2
4
6
8
10
Time (hr)

```python
import csv
import time
import tclab
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

n = 300 # number of second time points (5 min)
m = 0.004 # kg
tm = np.linspace(0, n, n+1)

# lab = tclab.setup(connected=False, speedup=5)
skip = True

# data
lab = tclab.TCLabModel()
if skip is False:
    # simulate data if no data has been written out yet
    t1 = {"data" : [{"time (s)" : 0, "Temperature (degC)" : lab.T1}]}
    lab.Q1(50) # set heater to 50%

    for i in range(n):
        data = {}
        print(i)
        time.sleep(1)
        print(lab.T1)
        data = {"time (s)" : i, "Temperature (degC)" : lab.T1}
        t1["data"].append(data)
    lab.close()


    with open("tclab_temps.csv", "w", encoding = "utf-8") as outfile:
        writer = csv.DictWriter(outfile, fieldnames = ["time (s)", "Temperature (degC)"])
        writer.writerows(t1["data"])
        outfile.close()

U = 5 # heat transfer coefficient, W/m2-k
m = 0.004 # mass, kg
A = 0.0012 # cross-sectional area, m2
Cp = 500 # heat capacity J/kg-k
T_inf = 23 + 273
T_a = 23 + 273
sigma = 5.67e-8
epsilon = 0.9
alpha = 0.01
def tclab_model(T, t):
    T_k = T + 273

    dT_dt = (U * A * (T_a - T_k) + epsilon * sigma * A * (T_inf**4 - T_k**4) + alpha * 50)/(m * Cp)

    return dT_dt

T0 = lab.T1
print(f"T0 => [{T0}]")
```

```
32
31 T_model = odeint(tclab_model, T0, tm)
30 T_actual = []
29 with open("tclab_temps.csv", "r") as infile:
28 |    reader = csv.reader(infile)
27 |    for row in reader:
26 |    |    T_actual.append(row[1])
25
24 T_model_K = T_model + 273
23 convection = U * A * (T_a - T_model_K)
22 radiation = epsilon * sigma * A * (T_inf**4 - T_model_K**4)
21 loss = convection + radiation
20
19
18 plt.figure(1)
17 plt.title("Modelled and Measured Temperature vs Time")
16 plt.plot(tm, T_model, "r-", label="Model")
15 plt.plot(tm, T_actual, "b-", label="Measured")
14 plt.ylabel("Temperature (degC)")
13 plt.xlabel("Time (sec)")
12 plt.yticks(np.linspace(0, 100, 10))
11 plt.legend()
10 plt.show()
 9
 8 plt.title("Convective, Radiative, and Total Loss")
 7 plt.plot(tm, convection, "r--", label="Convection")
 6 plt.plot(tm, radiation, "g--", label="Radiation")
 5 plt.plot(tm, loss, "g--", label="Loss")
 4 plt.ylabel("Heat Loss (W)")
 3 plt.xlabel("Time (s)")
 2 plt.legend()
 1 plt.show()
```


Modelled and Measured Temperature vs Time
Model
Measured
Temperature (degC)
46.088899999999995
42.5436
38.9983
35.1307
31.585399999999996
28.0401
24.494799999999998
20.949499999999997
0
50
100
150
200
250
300
Time (sec)


Convective, Radiative, and Total Loss
Convection
Radiation
Loss
Heat Loss (W)
0.0
−0.1
−0.2
−0.3
−0.4
0
50
100
150
200
250
300
Time (s)