**Solution of Ordinary Differential Equations (ODE)**

**Scilab**

*//Algorithm to compare the results of ODE function and Eulers results*

clc; clear;clf()

k = 0.05; CA0 = 1.0; t0 = 0; tf = 100; h = 5;

function **dC**=f(**t**, **C**)

**dC** = -k \* **C**^2

endfunction

t = t0:h:tf

n = length(t)-1

t\_euler = zeros(1,n+1)

CA\_euler = zeros(1,n+1)

t\_euler(1) = t0

CA\_euler(1) = CA0

for i = 1:n

t\_euler(i+1) = t\_euler(i) + h

CA\_euler(i+1) = CA\_euler(i) + h \* f(t\_euler(i), CA\_euler(i))

end

C\_num = ode(CA0, t0, t, f)

scf(0)

plot(t, C\_num, 'r-', 'LineWidth', 2)

plot(t, CA\_euler, 'b--', 'LineWidth', 2)

xlabel("Time (s)")

ylabel("Concentration C\_A (mol/L)")

title("Second-order Batch Reactor: Numerical vs Analytical Solution")

legend("Numerical (ODE solver)", "Euler Method")

xgrid()

**Python**

#Algorithm to compare the results of ODE function and Eulers results

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import odeint

k = 0.05; CA0 = 1.0; t0 = 0; tf = 100; h = 5;

def f(C, t):

return -k \* C\*\*2

t = np.arange(t0, tf + h, h)

n = len(t) - 1

CA\_euler = np.zeros(n+1)

CA\_euler[0] = CA0

for i in range(n):

CA\_euler[i+1] = CA\_euler[i] + h \* f(CA\_euler[i], t[i])

C\_num = odeint(f, CA0, t).flatten()

plt.figure(figsize=(8,6))

plt.plot(t, C\_num, 'r-', linewidth=2, label="Numerical (ODE solver)")

plt.plot(t, CA\_euler, 'b--', linewidth=2, label="Euler Method")

plt.xlim(0,100)

plt.ylim(0,1)

plt.xlabel("Time (s)")

plt.ylabel("Concentration C\_A (mol/L)")

plt.title("Second-order Batch Reactor: Numerical vs Euler Solution")

plt.legend()

plt.grid(True)

plt.show()