International IT University
Faculty of Computer technologies and cyber security
Department: MCM

Report

In the discipline «Numerical Analysis»

Executed: Taldybayev B.A.

Group: IT3-2203

Lecturer: Шахан Н.Ш.

Almaty, 2025

Task 7: Simple reaction

1. $$\begin{cases} \frac{dC_{CH_4}}{dt} = -kC_{CH_4}(C_{O_2})^2 \\ \frac{dC_{O_2}}{dt} = -2kC_{CH_4}(C_{O_2})^2 \\ \frac{dC_{CO_2}}{dt} = kC_{CH_4}(C_{O_2})^2 \\ \frac{dC_{H_2O}}{dt} = 2kC_{CH_4}(C_{O_2})^2 \end{cases}$$, where CCH4, CO2, CCO2, CH2O concentrations of CH4, O2, CO2 and H2O respectively , and at t = 0 CCH4(0) = 1.0, CO2(0) = 0.1, CCO2(0) = 0, CH2O(0) = 0. Coefficient of speed of reaction k = 0.05
2. Euler’s method:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

3. Runge-Kutte 2-nd order:

$$\begin{gathered} k_1 = hf(t_n, y_n) \\ k_2 = hf(t_n + h/2, y_n + k_1/2) \\ y_{n+1} = y_n + k_2 \end{gathered}$$

4. Runge-Kutte 4-th order:

$$\begin{gathered} k_1 = hf(t_n, y_n) \\ k_2 = hf(t_n + h/2, y_n + k_1/2) \\ k_3 = hf(t_n + h/2, y_n + k_2/2) \\ k_4 = hf(t_n + h, y_n + k_3) \\ y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{gathered}$$

Code and graph:

```
import numpy as np
import matplotlib.pyplot as plt


def reaction_odes(t, C, k):
    C_CH4, C_O2, C_CO2, C_H2O = C
    dC_CH4 = -k * C_CH4 * (C_O2 ** 2)
    dC_O2 = -2 * k * C_CH4 * (C_O2 ** 2)
    dC_CO2 = k * C_CH4 * (C_O2 ** 2)
    dC_H2O = 2 * k * C_CH4 * (C_O2 ** 2)
    return np.array([dC_CH4, dC_O2, dC_CO2, dC_H2O])


def euler_method(f, C0, k, t_range, dt):
    t_values = np.arange(t_range[0], t_range[1] + dt, dt)
    C_values = np.zeros((len(t_values), len(C0)))
    C_values[0] = C0

    for i in range(1, len(t_values)):
        C_values[i] = C_values[i - 1] + dt * f(t_values[i - 1], C_values[i -
1], k)

    return t_values, C_values


def runge_kutta_2(f, C0, k, t_range, dt):
    t_values = np.arange(t_range[0], t_range[1] + dt, dt)
    C_values = np.zeros((len(t_values), len(C0)))
    C_values[0] = C0

    for i in range(1, len(t_values)):
        k1 = f(t_values[i - 1], C_values[i - 1], k)
        k2 = f(t_values[i - 1] + dt / 2, C_values[i - 1] + dt * k1 / 2, k)
        C_values[i] = C_values[i - 1] + dt * k2

    return t_values, C_values


def runge_kutta_4(f, C0, k, t_range, dt):
    t_values = np.arange(t_range[0], t_range[1] + dt, dt)
    C_values = np.zeros((len(t_values), len(C0)))
    C_values[0] = C0

    for i in range(1, len(t_values)):
        k1 = f(t_values[i - 1], C_values[i - 1], k)
        k2 = f(t_values[i - 1] + dt / 2, C_values[i - 1] + dt * k1 / 2, k)
        k3 = f(t_values[i - 1] + dt / 2, C_values[i - 1] + dt * k2 / 2, k)
        k4 = f(t_values[i - 1] + dt, C_values[i - 1] + dt * k3, k)
        C_values[i] = C_values[i - 1] + (dt / 6) * (k1 + 2 * k2 + 2 * k3 +
k4)

    return t_values, C_values


C0 = [1.0, 0.1, 0.0, 0.0]
k = 0.05
t_range = (0, 10)
dt = 0.1

t_euler, C_euler = euler_method(reaction_odes, C0, k, t_range, dt)
t_rk2, C_rk2 = runge_kutta_2(reaction_odes, C0, k, t_range, dt)
```

```
t_rk4, C_rk4 = runge_kutta_4(reaction_odes, C0, k, t_range, dt)

labels = ['CH4', 'O2', 'CO2', 'H2O']
plt.figure(figsize=(12, 6))

for i in range(4):
    plt.plot(t_euler, C_euler[:, i], '--', label=f'{labels[i]} (Euler)')
    plt.plot(t_rk2, C_rk2[:, i], '-.', label=f'{labels[i]} (RK2)')
    plt.plot(t_rk4, C_rk4[:, i], label=f'{labels[i]} (RK4)')

plt.xlabel('Time')
plt.ylabel('Concentration')
plt.title('Solution of ODE System using Euler, RK2, and RK4')
plt.legend()
plt.grid()
plt.show()
```