Numerical Solution for Diffusion and Transport System Using Control Volumes

Nasr Ghoniem

December 2024

We consider a system of coupled ordinary differential equations (ODEs) describing the evolution of concentration fields for n species across N control volumes. Each control volume corresponds to a point midway between the left and right boundaries. The governing equation for each species j is:

$$\frac{\partial C_j(x_k,t)}{\partial t} = \frac{J_{\text{left},k} - J_{\text{right},k}}{\Delta x_k} + P_j(x_k) - L_j(x_k)$$

Where:

- $C_j(x_k,t)$ is the concentration of species j at control volume k,
- $J_{\text{left},k}$ and $J_{\text{right},k}$ are the fluxes of species j at the midpoints between the control volumes $k-1/2$ and $k+1/2$, respectively,
- $P_j(x_k)$ is the spatially-dependent source term for species j at x_k,
- $L_j(x_k)$ is the spatially-dependent sink term for species j at x_k,
- Δx_k is the distance between control volumes k and $k+1$.

The fluxes are modeled using Fick's law:

$$J_{\text{left},k} = -D_j \frac{C_j(x_k) - C_j(x_{k-1})}{\Delta x_{k-1/2}}, \quad J_{\text{right},k} = -D_j \frac{C_j(x_{k+1}) - C_j(x_k)}{\Delta x_{k+1/2}}$$

For the boundary conditions, we have:

- At the left boundary ($k=1$), $J_{\text{left},0} = J_0(t)$, a given function of time,

- At the right boundary ($k = N$), $J_{\text{right},N+1} = 0$, assuming zero flux.

The system of ODEs for n species and N control volumes is then:

$$\frac{\partial C_j(x_k, t)}{\partial t} = \frac{J_{\text{left},k} - J_{\text{right},k}}{\Delta x_k} + P_j(x_k) - L_j(x_k)$$

Python Code for Numerical Solution

Below is the Python code used to solve the system of ODEs numerically using the '$solve_ivp$' method from the 'scipy' library:

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

# Define constants and parameters
D = [1.0, 1.0]  # Diffusion coefficients for species 1 and 2
N = 3  # Number of control volumes (adjustable)
n = 2  # Number of species
x = np.linspace(0, N, N+1)  # Spatial grid points (for simplicity, regular spacing

# Spatial grid step sizes, can be adjusted based on your needs
dx = np.diff(x)  # Calculate distances between adjacent control volumes (x)

# Functions for P(x) and L(x)
P = lambda x, j: 0.1 * x  # Spatial function P(x) for each species
L = lambda x, j: 0.05 * x  # Spatial function L(x) for each species

# Given flux at the left boundary as a function of time
J0 = lambda t: np.sin(t)

# Define the system of ODEs
def system_of_odes(t, C):
    dCdt = np.zeros_like(C)

    for j in range(n):  # Loop over each species
        for k in range(1, N-1):  # Loop over control volumes (1 to N-1)
```

```
            # Evaluate fluxes at the midpoints (k-1/2 and k+1/2)
            J_left = -D[j] * (C[j*N + k] - C[j*N + k-1]) / dx[k-1]  # Flux at k-1/
            J_right = -D[j] * (C[j*N + k+1] - C[j*N + k]) / dx[k]  # Flux at k+1/2

            # Apply the ODE for species j at control volume k
            dCdt[j*N + k] = (J_left - J_right) + P(x[k], j) - L(x[k], j)

        # Boundary conditions (left and right)
        # Left boundary (k=0) with a time-dependent flux J0(t)
        dCdt[j*N] = (J0(t) - (C[j*N + 1] - C[j*N]) / dx[0]) / D[j] + P(x[0], j) -

        # Right boundary (k=N-1) with zero flux
        dCdt[j*N + N-1] = (C[j*N + N-2] - 0) / dx[N-2] / D[j] + P(x[N-1], j) - L(x

    return dCdt

# Initial condition (uniform concentration across the grid)
C0 = np.zeros(N * n)

# Time span for the solution
t_span = (0, 10)  # From t=0 to t=10
t_eval = np.linspace(*t_span, 100)

# Solve the system of ODEs
sol = solve_ivp(system_of_odes, t_span, C0, t_eval=t_eval)

# Plot the solution
plt.figure(figsize=(8,6))
for j in range(n):
    for k in range(N):
        plt.plot(sol.t, sol.y[j*N + k], label=f'Species {j+1}, CV {k+1}')
plt.xlabel('Time')
plt.ylabel('Concentration')
plt.legend()
plt.title('Concentration Profile Over Time')
plt.show()
```