Finite Difference Solution for 2D Flow with Temperature-Dependent Viscosity

1 Governing Equations

We consider the laminar flow of a viscous fluid where momentum inertia is neglected, and velocity is primarily in the x-direction. The governing equations are:

1.1 Momentum Equation (Stokes Flow)

The x-momentum equation simplifies to:

$$0 = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial y}\left(\eta \frac{\partial u}{\partial y}\right). \tag{1}$$

where $\eta = \eta(T)$ is the temperature-dependent viscosity.

1.2 Energy Equation

The energy equation includes convection, diffusion, and viscous dissipation:

$$\rho C_p \left(\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x}\right) = k\frac{\partial^2 T}{\partial y^2} + \Phi, \tag{2}$$

where the viscous dissipation function is given by:

$$\Phi = \eta \left(\frac{\partial u}{\partial y}\right)^2. \tag{3}$$

2 Finite Difference Discretization

We discretize the equations using a finite difference approach.

2.1 Velocity Computation

Using a simple centered difference scheme, we approximate the velocity profile:

$$u_j = \left(\frac{\frac{\partial p}{\partial x}}{\eta}\right)\frac{\Delta y^2}{2} j(N_y - j). \tag{4}$$

2.2 Temperature Evolution

Using an explicit finite difference scheme, the temperature update equation is:

$$T_i^j = T_i^j + \Delta t \left[\alpha \left(\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} \right) - u_j \frac{T_{i,j} - T_{i-1,j}}{\Delta x} + \frac{\Phi}{\rho C_p} \right]. \tag{5}$$

3 Python Implementation

The following Python code implements the finite difference solution:

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

# Grid parameters
Nx = 50  # Number of x points
Ny = 50  # Number of y points
Lx = 1.0  # Length in x direction
Ly = 1.0  # Length in y direction
dx = Lx / (Nx - 1)
dy = Ly / (Ny - 1)

# Physical parameters
p0 = 1.0  # Pressure at x0
p1 = 0.0  # Pressure at x1
Tb = 300.0  # Boundary temperature at y0 and y1
rho = 1.0  # Density
Cp = 1.0  # Specific heat
k = 1.0  # Thermal conductivity
eta0 = 1.0  # Reference viscosity
b = 0.01  # Viscosity temperature dependence coefficient

# Initialize grids
u = np.zeros(Ny)  # Velocity profile (1D in y)
T = np.ones((Nx, Ny)) * Tb  # Temperature field (2D in x, y)

# Compute pressure gradient
dpdx = (p1 - p0) / Lx

# Solve for velocity u(y) using finite difference
for j in range(1, Ny - 1):
    eta_avg = (eta0 * np.exp(-b * Tb))  # Approximate viscosity
    u[j] = (dpdx / eta_avg) * (dy**2 / 2) * (j * (Ny - j))  # Parabolic
      profile

# Solve for temperature T(x, y) using finite difference
T_new = np.copy(T)
dt = 0.01  # Time step for stability
alpha = k / (rho * Cp)  # Thermal diffusivity

for it in range(1000):  # Time stepping
    T_new[:, 0] = Tb  # Boundary condition at y0
    T_new[:, -1] = Tb  # Boundary condition at y1

```

```
    for i in range(1, Nx - 1):
        for j in range(1, Ny - 1):
            # Compute viscosity at this location
            eta = eta0 * np.exp(-b * T[i, j])

            # Compute viscous dissipation
            Phi = eta * ( (u[j+1] - u[j]) / dy )**2

            # Discretized heat equation (explicit method)
            T_new[i, j] = T[i, j] + dt * (
                alpha * (
                    (T[i+1, j] - 2*T[i, j] + T[i-1, j]) / dx**2 +
                    (T[i, j+1] - 2*T[i, j] + T[i, j-1]) / dy**2
                ) - (u[j] * (T[i, j] - T[i-1, j]) / dx) + Phi / (rho * Cp)
            )

    # Convergence check
    if np.linalg.norm(T_new - T) < 1e-6:
        break

    T = np.copy(T_new)

# Plot results
plt.figure(figsize=(8, 6))
plt.contourf(np.linspace(0, Lx, Nx), np.linspace(0, Ly, Ny), T.T, 50, cmap='hot')
plt.colorbar(label='Temperature (K)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Temperature Field')
plt.show()
```