Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report

On

Finite Volume Method to Solve Poisson Equation

[Code No: MCSC 202]

Submitted By:

Supriya Adhikari (05)

Pranaw Raj Kafle (25)

Aavash Lamichhane (29)

Karib Maharjan (31)

Anusha Rajlawat (46)

Submitted To:

Dr. Gokul K.C.

Department of Mathematics

Submission Date: 10/01/2024

Abstract

This project explores the Finite Volume Method (FVM) as a numerical approach to solving the Poisson equation, a second-order elliptic partial differential equation with wide-ranging applications in physics and engineering. The Finite Volume Method discretized the domain into control volumes, facilitating a robust and versatile approach to numerical solutions. Through the use of Python, employing libraries such as NumPy, SciPy and Matplotlib, this project demonstrates the implementation of FVM for the Poisson equation. By emphasizing the importance of control volumes, cell centers, and appropriate boundary conditions, the project sheds light on the practicality and effectiveness of the Finite Volume Method in capturing complex physical phenomena. This work contributes to the broader understanding of numerical methods and their application to solve intricate partial differential equations in various scientific and engineering domains.

Table of Content

Background

The Finite Volume Method (FVM) is a numerical technique widely used for solving partial differential equations (PDEs), including the Poisson equation. The Poisson equation in two dimensions is given by:

$$\frac{\mathrm{d}^2\mathrm{u}}{\mathrm{dx}^2}+\frac{\mathrm{d}^2\mathrm{u}}{\mathrm{dy}^2}+g=0$$

In the Finite Volume Method, the domain is discretized into control volumes, and the integral form of the PDE is applied over each control volume. This method is particularly suitable for problems with physical interpretations related to conservation laws, making it applicable in fluid dynamics, heat transfer, and other fields.

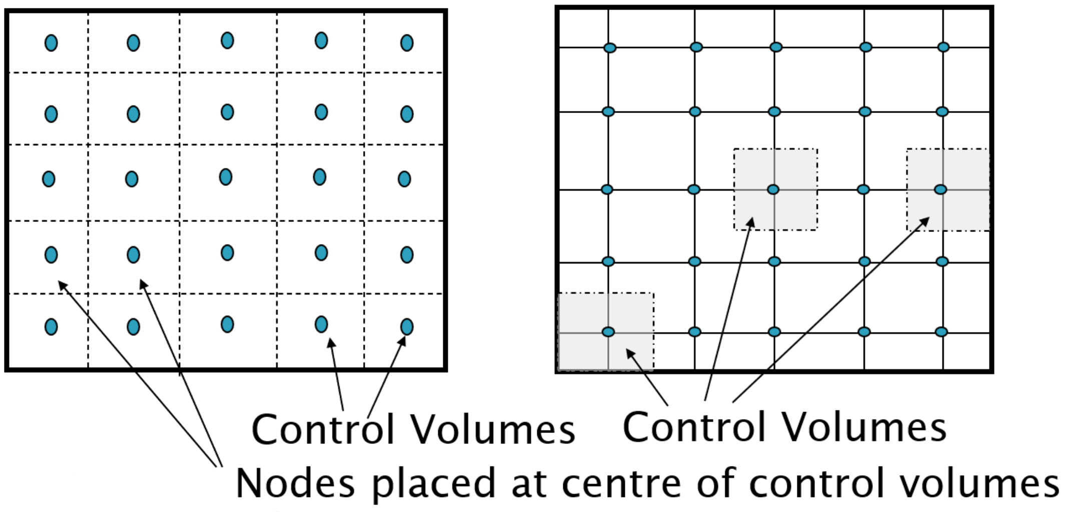
Objectives

- To implement the Finite Volume Method to discretize the Poisson equation, dividing the domain into control volumes and expressing the second derivative using volume averages,
- To apply appropriate boundary conditions to the problem,
- To visualize the numerical solution to understand the distribution of the physical quantity within the domain.

Mathematics

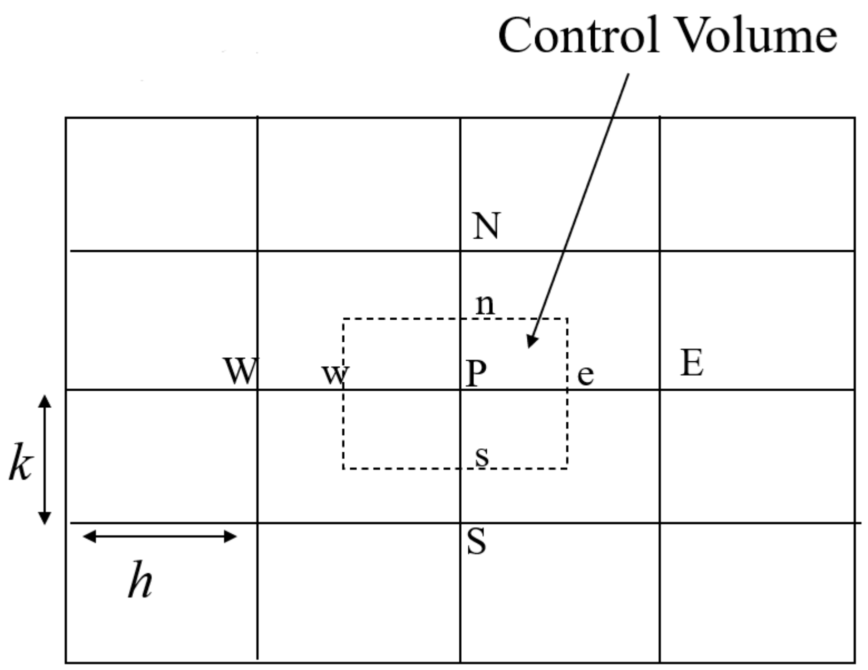
The Finite Volume Method involves the discretization of the domain into control volumes. The second derivative is expressed using volume averages, and the resulting system of algebraic equations is solved.

Domain divided into control volume as follows:



Consider poisson equation:

$$\frac{d^2u}{dx^2} + \frac{d^2u}{dy^2} + g = 0$$



For an area A enclosed by the surfaces S and vector field variable V, divergence theorem states

$$\int_A \nabla . \underline{v}\, dA = \int_S \underline{v} . \underline{n}\, dS$$

If we assume,

$$\underline{v} = \left[\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right] = \nabla u$$

The Divergence theorem can be used on the Laplace equation for the scalar field u.

$$\int_A \nabla .(\nabla u)\, dA = \int_S \nabla u . \underline{n}\, dS$$

Or

$$\int_A \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] dA = \int_S \left[\frac{\partial u}{\partial x} n_x + \frac{\partial u}{\partial y} n_y \right] dS$$

Using the divergence theorem it can be shown that:

$$\iint_{xy} \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + g \right] dxdy = \left[\left(\frac{\partial u}{\partial x} \right)_e - \left(\frac{\partial u}{\partial x} \right)_w \right] k + \left[\left(\frac{\partial u}{\partial y} \right)_n - \left(\frac{\partial u}{\partial y} \right)_s \right] h + ghk = 0$$

The first order derivatives can be approximated by:

$$\left(\frac{\partial u}{\partial x} \right)_e = \frac{U_E - U_P}{h} \qquad \left(\frac{\partial u}{\partial x} \right)_w = \frac{U_P - U_W}{h}$$

$$\left(\frac{\partial u}{\partial y} \right)_n = \frac{U_N - U_P}{k} \qquad \left(\frac{\partial u}{\partial y} \right)_s = \frac{U_P - U_S}{k}$$

Therefore at each node we have the following algebraic equations that represent the Poisson equation.

$$A_P U_P = A_E U_E + A_W U_W + A_N U_N + A_S U_S + S$$
$$A_E = A_W = \frac{k}{h}$$
$$A_N = A_S = \frac{h}{k}$$
$$A_P = A_E + A_W + A_N + A_S$$
$$S = ghk$$

The equation on the previous slide is in the standard form for finite volumes schemes

It is normal to write Ap in this way, rather than, say

$$A_P = 2\left(\frac{k}{h} + \frac{h}{k}\right)$$

This is because, if you use other volume shapes, such as triangles, rectangles or a mixture, the same sort of results (e.g. that Ap is the sum of the other coefficients) still hold

Boundary Conditions - Neumann

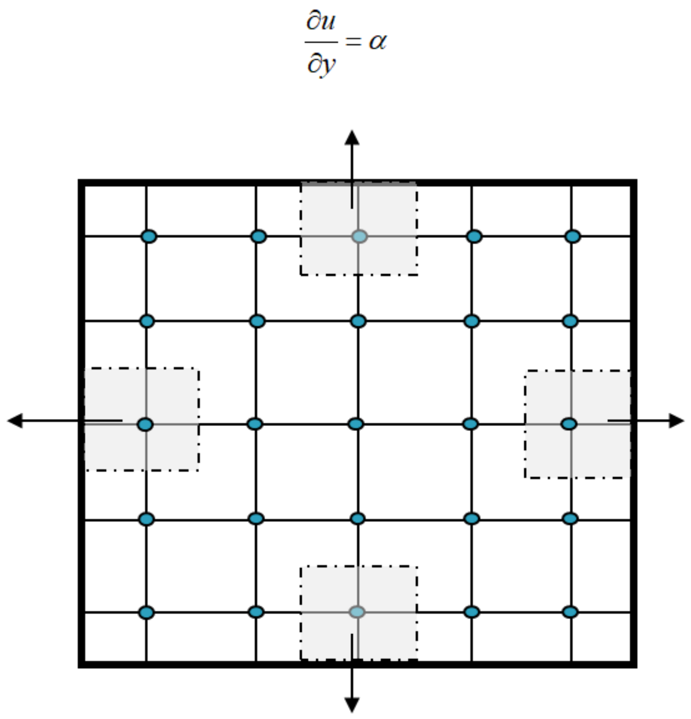
The divergence theorem gives:

$$\left[\left(\frac{\partial u}{\partial x}\right)_e - \left(\frac{\partial u}{\partial x}\right)_w\right]k + \left[\left(\frac{\partial u}{\partial y}\right)_n - \left(\frac{\partial u}{\partial y}\right)_s\right]h + ghk = 0$$

At north boundary

$$\left(\frac{\partial u}{\partial x}\right)_e = \frac{U_E - U_P}{h} \qquad \left(\frac{\partial u}{\partial x}\right)_w = \frac{U_P - U_W}{h}$$

$$\left(\frac{\partial u}{\partial y}\right)_n = \alpha \qquad \left(\frac{\partial u}{\partial y}\right)_s = \frac{U_P - U_S}{k}$$



Therefore at each node on the north boundary we have

$$A_P U_P = A_E U_E + A_W U_W + A_S U_S + S$$

$$A_E = A_W = \frac{k}{h}, \quad A_S = \frac{h}{k}$$

$$A_P = A_E + A_W + A_S$$

$$S = ghk + h\alpha$$

Similarly at each node on the east boundary, if $\left(\frac{\partial u}{\partial x}\right)_e = \beta$

$$A_P U_P = A_W U_W + A_N U_N + A_S U_S + S$$
$$A_W = \frac{k}{h}, \quad A_N = A_S = \frac{h}{k}$$
$$A_P = A_W + A_N + A_S$$
$$S = ghk + k\beta$$

Similar on other boundaries, but the signs are reversed

(south)

$$S = ghk - h\alpha$$

(west)

$$S = ghk - k\beta$$

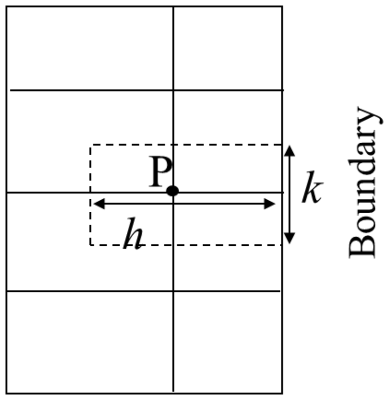
Boundary Conditions - Dirichlet

The divergence theorem gives:

$$\left[\left(\frac{\partial u}{\partial x}\right)_e - \left(\frac{\partial u}{\partial x}\right)_w\right]k + \left[\left(\frac{\partial u}{\partial y}\right)_n - \left(\frac{\partial u}{\partial y}\right)_s\right]h + ghk = 0$$

Example: at east boundary $u = \mu$

$$\left(\frac{\partial u}{\partial x}\right)_e = \frac{\mu - U_P}{0.5h} \qquad \left(\frac{\partial u}{\partial x}\right)_w = \frac{U_P - U_W}{h}$$
$$\left(\frac{\partial u}{\partial y}\right)_n = \frac{U_N - U_P}{k} \qquad \left(\frac{\partial u}{\partial y}\right)_s = \frac{U_P - U_S}{k}$$



Therefore at each node on the east boundary we have

$$A_P U_P = A_W U_W + A_N U_N + A_S U_S + S$$
$$A_W = \frac{k}{h}, \quad A_N = A_S = \frac{h}{k}$$
$$A_P = 2\frac{k}{h} + A_W + A_N + A_S$$
$$S = ghk + 2\frac{k}{h}\mu$$

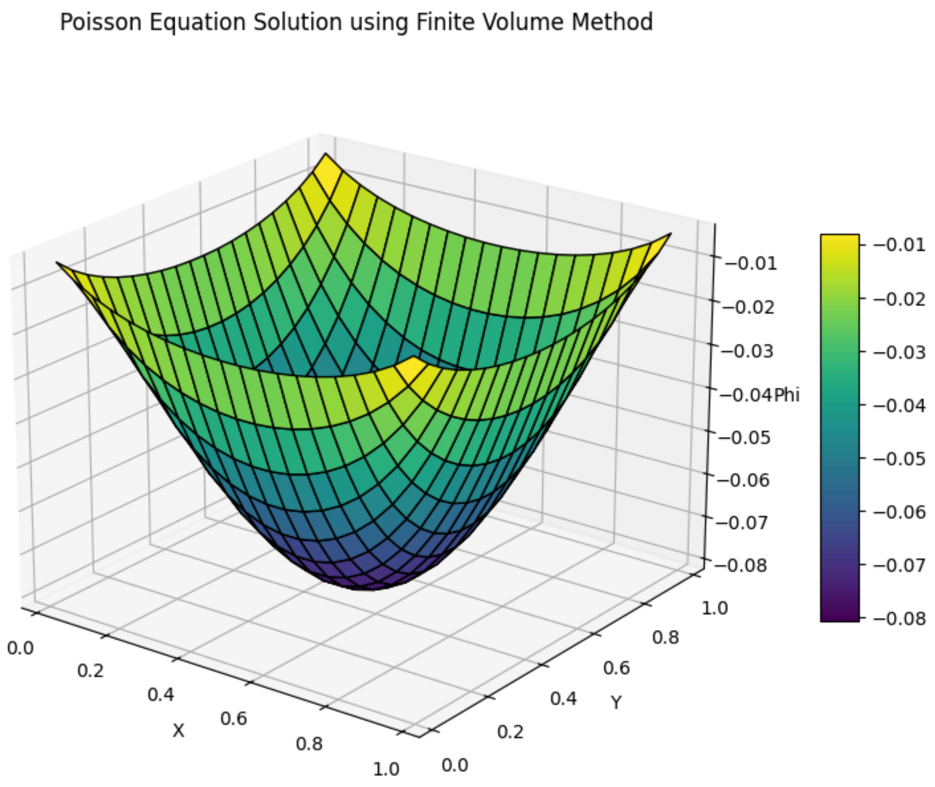
Source Code

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import spsolve

# Parameters
nx = 21 # Number of cell interfaces (grid points) in x-axis
ny = 21 # Number of cell interfaces (grid points) in x-axis
ncx = nx-1 # Numbers of cells
ncy = ny-1
Lx = 1.0 # Length of domain in x and y axes
Ly = 1.0
dx = Lx / (nx-1) # Size of a cell in x and y axes
dy = Ly / (ny-1)

# Grid and cell center calculation
x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)
cell_centers_x = 0.5 * (x[1:] + x[:-1])
cell_centers_y = 0.5 * (y[1:] + y[:-1])
X, Y = np.meshgrid(cell_centers_x,cell_centers_y)

def fn(x,y):
    return 1
    # return 100*(x**2+y**2)

# Source term
f = np.zeros((ncx, ncy))

# Set values for source term function
for i in range(ncx):
    for j in range(ncy):
        f[i,j] = fn(cell_centers_x[i],cell_centers_y[j])

# Discretization coefficients
a_w = dy/dx
a_e = dy/dx
a_n = dx/dy
a_s = dx/dy
a_p = -(a_w + a_s + a_e + a_n)
```

```
# Initialize solution
phi = np.zeros((ncx, ncy))

# Set Dirichlet boundary conditions
phi[:, 0] = 0.0
phi[:, -1] = 0.0
phi[0, :] = 0.0
phi[-1, :] = 0.0

# Assemble coefficient matrix A and right-hand side b
A = np.zeros((ncx * ncy, ncx * ncy))
for i in range(ncy):
    for j in range(ncx):
        idx = i * ncx + j
        if i > 0:
            A[idx, idx - ncx] = a_n
        if i < ncy - 1:
            A[idx, idx + ncx] = a_s
        if j > 0:
            A[idx, idx - 1] = a_w
        if j < ncx - 1:
            A[idx, idx + 1] = a_e
        A[idx, idx] = a_p
A=A/(dx*dy)

# Compressed Sparse Row (CSR), used for spsolve (Not necessary, but removes
warning)
A=csr_matrix(A)

b = f.flatten()

# Solve the linear system
phi = spsolve(A, b).reshape((ncy,ncx))

# 3D Plotting
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot the surface
surf = ax.plot_surface(X, Y, phi, cmap='viridis', edgecolor='k')
```

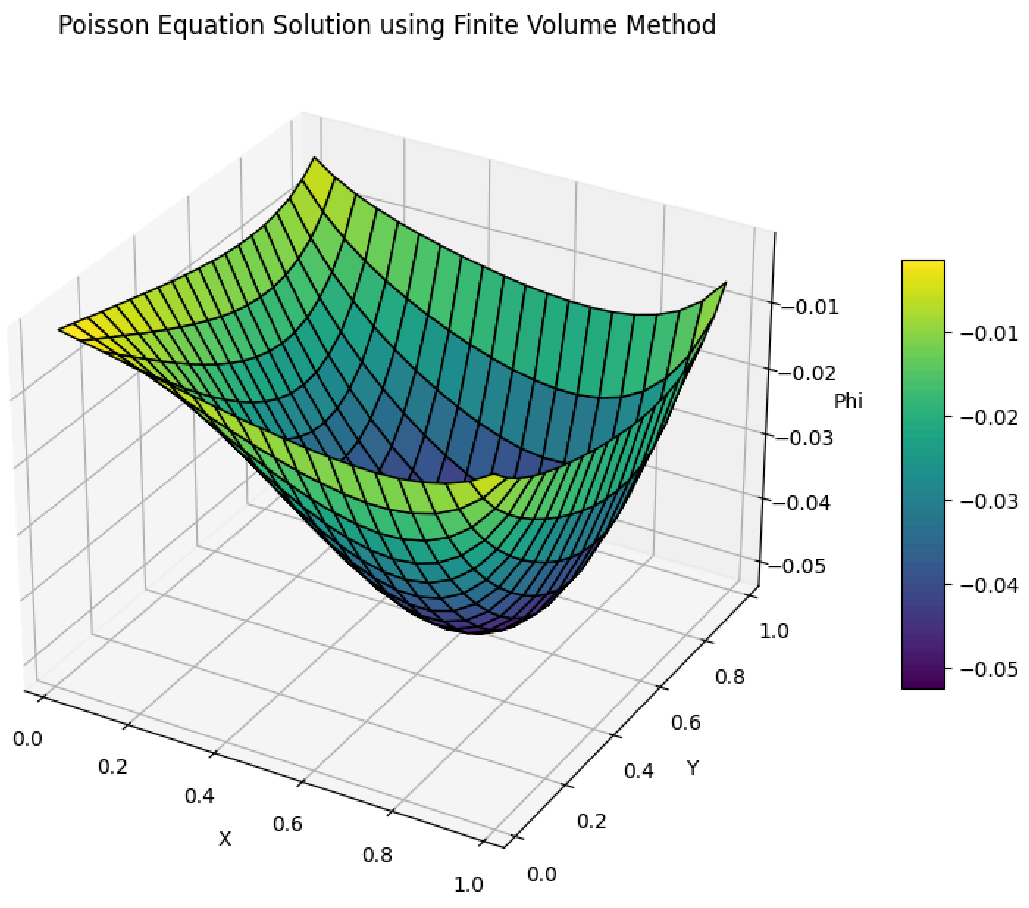
```
# Customize the plot
ax.set_title('Poisson Equation Solution using Finite Volume Method')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Phi')

# Uncomment below code to make fixed ratio for axes (Not Recommended)
# ax.set_box_aspect([np.ptp(coord) for coord in [X, Y, phi]])

# Add a colorbar
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10)

# Show the plot
plt.show()
```

Output:



When source term function fn is not 1, for example:

fn = x^2+y^2



Conclusion

In conclusion, this project successfully applied the Finite Volume Method to numerically solve the Poisson equation. The objectives were achieved, providing insights into the distribution of the physical quantity in the given domain. This work serves as a foundational exploration of numerical methods for solving PDEs using the Finite Volume Method, with potential applications in various scientific and engineering fields.