Finite Volume Method for Poisson Equation

Supriya Adhikari
Pranaw Raj Kafle
Aavash Lamichhane
Karib Maharjan
Anusha Rajlawat

Poisson Equation

$$\nabla^2 u = f(x)$$

- ∇^2 represents Laplacian
- u – scalar field
- $f(x)$ – source term

Poisson Equation

Importances

- Widely used in physics, engineering, and applied mathematics.
- Describes various physical phenomena, such as heat conduction, fluid flow, and electrostatics.

Poisson Equation

Applications

- Describes electric potential in the absence of charge, modeling electric fields.
- Governs temperature distribution in materials and predicts heat transfer.
- Models pressure distributions in fluid flow for steady-state and transient scenarios.

Finite Volume Method

Integration Over Cells

Numerical Approximation of source term function

Finite Volume Method

- Numerical method for solving partial differential equations.
- Divides the domain into small control volumes.
- Integrates the governing equations over each control volume.

Finite Volume Method

Applications

- **Heat Transfer**
  - Modeling temperature distributions in materials.
  - Transient heat transfer simulations.
- **Fluid Dynamics (CFD)**
  - Simulating fluid flow in steady-state and transient conditions.
- **Biomedical Simulations**
  - Modeling diffusion processes in tissues.

Process

01

Discretize the Domain

02

Initialize matrices for the linear system

03

Evaluate Source term function

Process

04

Apply Boundary Conditions

05

Solve the Linear System

06

Plot the results

Problem Parameters

- L = Length of Domain
- Nx = Number of cells
- dx = Size of cell and distance between cell averages (mid-point of cells)
- Boundary Conditions
- Source Function i.e. f(x)

```python
# Define the problem parameters
L = 1.0  # Length of the domain
Nx = 50  # Number of cells
dx = L / Nx  # Cell center spacing
left_boundary = 0.0
right_boundary = 0.0
def source_term(x): #Can be modified for specific problem
    return 1
```

Discretization of Domain

The domain is divided into a set of Nx+1 cell interfaces, which represent Nx cells. The mid-points of the cells were also calculated.

```
# Step 1: Discretize the domain
x_values = np.linspace(0, L, Nx+1)
cell_centers = 0.5 * (x_values[1:] + x_values[:-1])
```

Initialize matrices for the linear system

The Laplacian operation is represented by the tridiagonal matrix using finite difference approximation since the distance between the cell centers is constant. ‘b’ represents the solution matrix for the function f(x) which is initially set to 0.

```
# Step 2: Initialize matrices for the linear system
# Using a tri-diagonal matrix to represent the Laplacian
A = diags([-1, 2, -1], [-1, 0, 1], shape=(Nx, Nx),format='csr') / dx**2
b = np.zeros(Nx)
```

Evaluate Source term function

Then the source term function is evaluated over each cell. It is assumed to be constant within the cell boundaries, so we approximate the function with the value at its center.

```
# Step 3: Evaluating Source term function
for i in range(1, Nx-1):
    b[i] = source_term(cell_centers[i])
```

Apply Boundary Conditions

The boundary conditions for the source term function are generally different. So, in this case Dirichlet boundary condition was used. Dirichlet boundary conditions tell us the exact values of the solution at specific points on the edge of the problem's area. This essentially locks in the solution's behavior at the boundaries by specifying what the solution should be at those particular points.

```
# Step 4: Apply boundary conditions
# In this example, using Dirichlet boundary conditions (u=0 at both ends)
# Modify based on your specific boundary conditions
b[0] = left_boundary
b[-1] = right_boundary
```

Solve the Linear System

Here, with the Laplacian discretized into a matrix and the functional value at each cell calculated, a linear system of the form Au = b is created and solved for u.

```
# Step 5: Solve the linear system
u = spsolve(A, b)
```

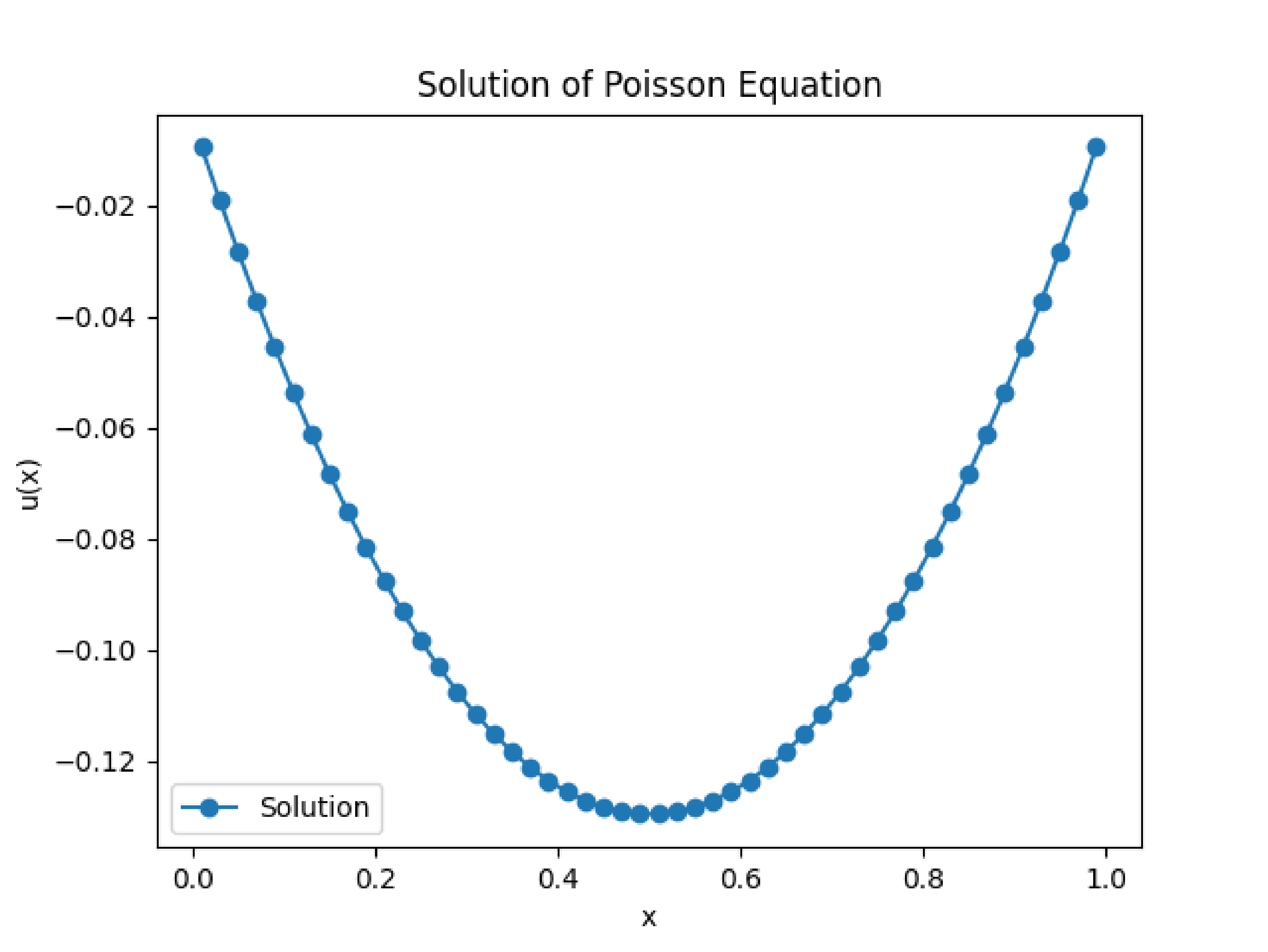
Plot the results

The results in u were plotted in the Y-axis against the respective values of x, plotted in the x-axis to obtain the solution of the poisson equation.

```
# Step 6: Plot the results
plt.plot(cell_centers, u, marker='o', label='Numerical Solution')
plt.xlabel('x')
plt.ylabel('u(x)')
plt.title('Finite Volume Method Example')
plt.legend()
plt.show()
```

Solution

The solution for the poisson equation as per the given parameters above i.e. L=1, Nx=50, f(x)=1 and left boundary and right boundary as 0 is:



Conclusion

- Finite Volume Method is a powerful tool for solving the Poisson Equation.
- Widely applicable in various scientific and engineering fields.
- Continuous research for improving efficiency and accuracy.

Thank you very much!