SECOND-ORDER FVM FOR 2D LAPLACE EQUATION AROUND A CYLINDER

Khady sarah Sall

UUM 510E Student number: 922310129

Abstract

This project investigates the second-order convergence of structured and unstructured finite volume methods (FVM) for solving the two-dimensional Laplace equation around a cylinder. The analytical solution for the problem is used to evaluate the numerical solutions obtained with bilinear quadrilateral (Q1) and linear triangular (P1) elements. The structured and unstructured meshes were tested with varying resolutions (81x41, 161x81, and 321x161) to assess the accuracy and convergence of the methods. The results indicate that while the structured FVM shows a consistent pattern, the unstructured FVM exhibits notable differences in numerical accuracy. Error analysis demonstrates a lack of second-order convergence for both methods, suggesting further improvements in the implementation are required.

I. Introduction

The two-dimensional Laplace equation around a cylinder is a classic problem in computational fluid dynamics, often used to evaluate the accuracy and convergence of numerical methods. This project focuses on applying second-order accurate finite volume discretization methods using both structured and unstructured meshes to solve the Laplace equation. The structured mesh employs bilinear quadrilateral (Q1) elements, while the unstructured mesh uses linear triangular (P1) elements. The primary goal is to implement these methods, analyze the numerical results, and determine the convergence rates to verify their order of accuracy.

II. Methodology

The Laplace equation in polar coordinates is given by:

$$[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = 0]$$

The analytical solution for the problem is:

$$[u(r,\theta) = U_\infty\left(r - \frac{R}{r}\right)\sin(\theta)]$$

Boundary conditions are:

$$u(1,\theta) = 0$$

$$u(2,\theta) = \left(2 - \frac{1}{2}\right)\sin(\theta)$$

The numerical methods used include:

- Structured FVM: Utilizes bilinear quadrilateral (Q1) elements with meshes of 81x41, 161x81, and 321x161.

- Unstructured FVM: Utilizes linear triangular (P1) elements with similar mesh resolutions.

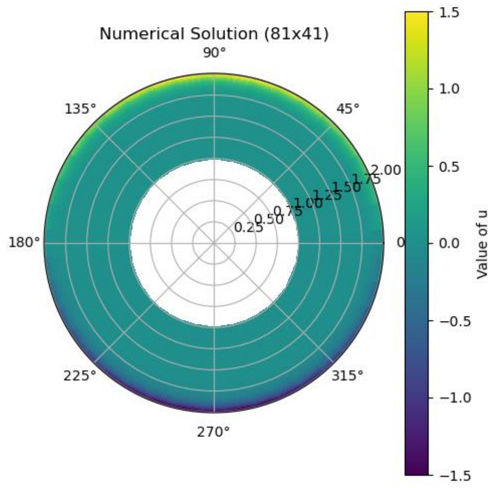
The fully implicit second-order solution algorithm is implemented, and a direct solver (LU factorization) is used to solve the system. The error function is given by:

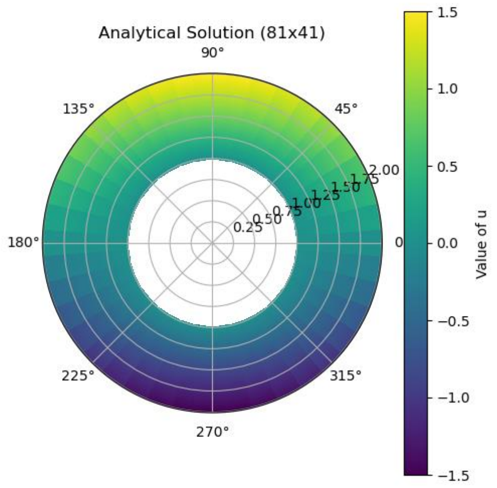
$$[\text{Error} = \frac{|u_{i,j} - u_{\text{analytic}}|_2}{\sqrt{i_{\max} j_{\max}}}]$$

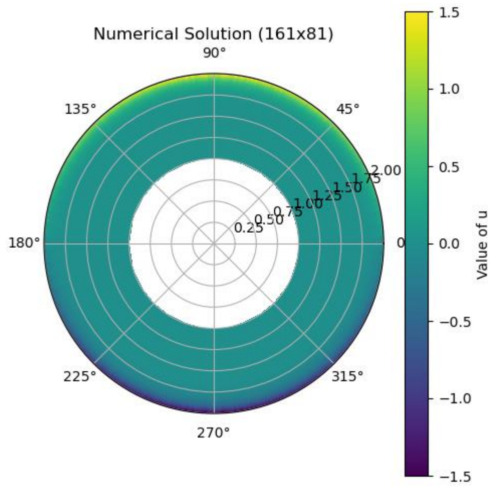
Error analysis and spatial convergence rates are computed and plotted on a log-log scale.

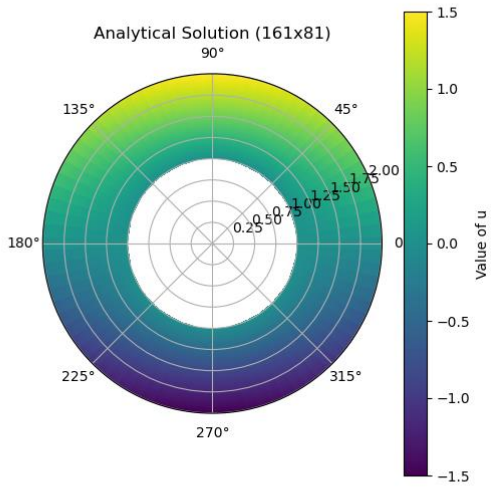
III. Results

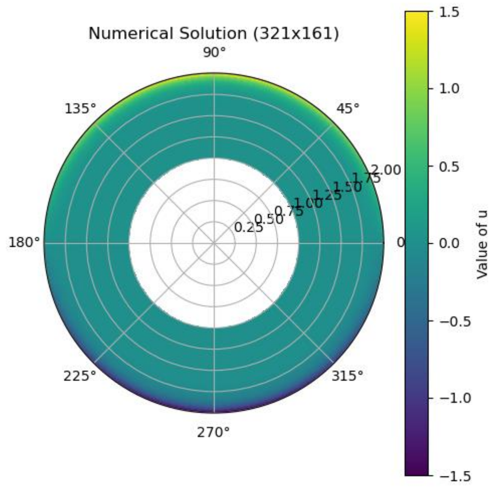
1. Structured FVM

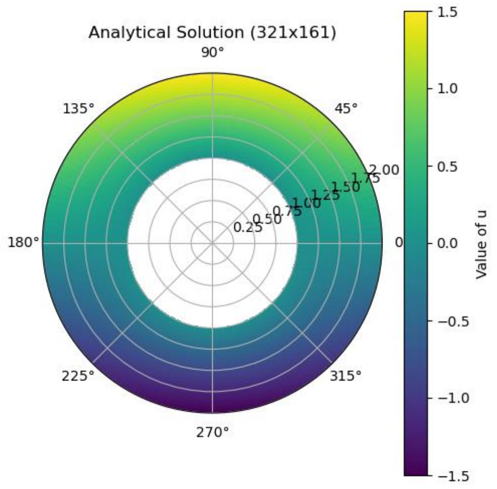

Numerical Solution (81x41)
90°
45°
0°
315°
270°
225°
180°
135°
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Value of u
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

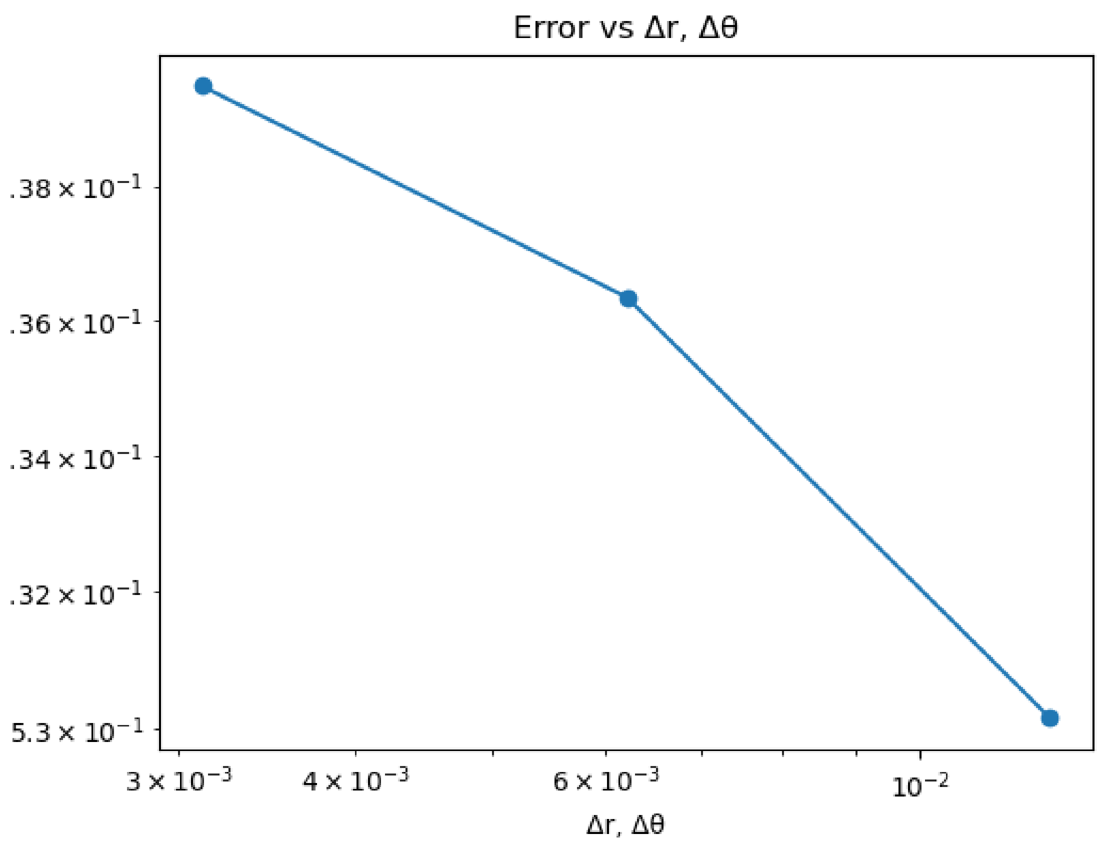

Analytical Solution (81x41)
90°
45°
0°
315°
270°
225°
180°
135°
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Value of u
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5


Numerical Solution (161x81)
90°
45°
0°
315°
270°
225°
180°
135°
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Value of u
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5


Analytical Solution (161x81)
90°
45°
0°
315°
270°
225°
180°
135°
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Value of u
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

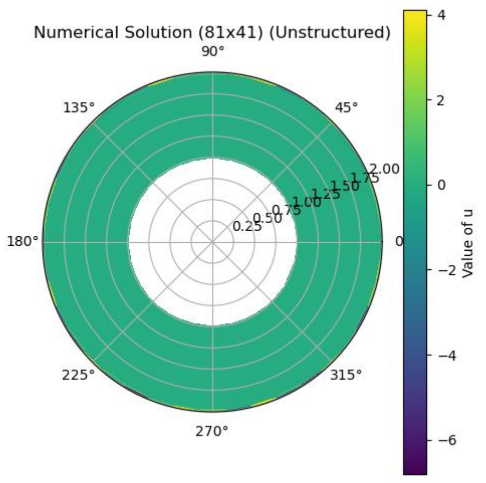

Numerical Solution (321x161)
90°
45°
0°
315°
270°
225°
180°
135°
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Value of u
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

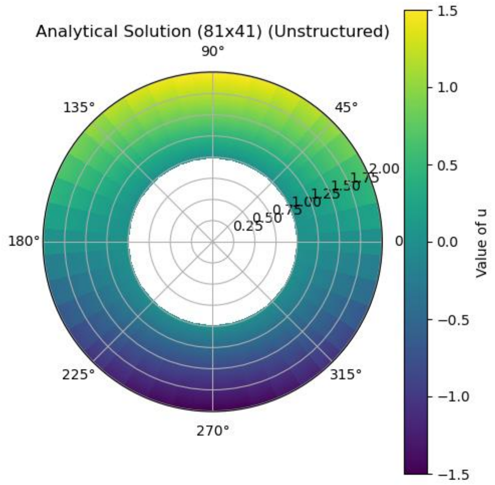

Analytical Solution (321x161)
90°
45°
0°
315°
270°
225°
180°
135°
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Value of u
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5

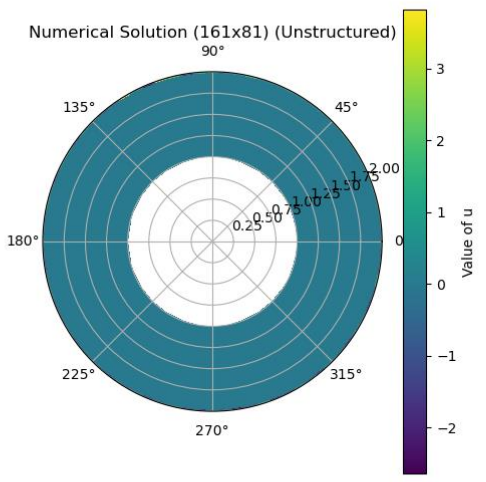


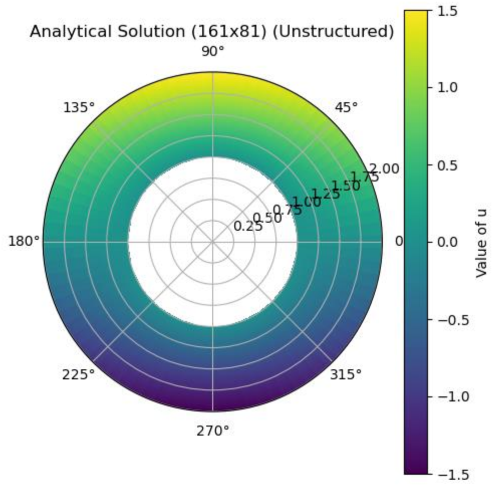
Error vs Δr, Δθ
.38 × 10^−1
.36 × 10^−1
.34 × 10^−1
.32 × 10^−1
5.3 × 10^−1
3 × 10^−3
4 × 10^−3
6 × 10^−3
10^−2
Δr, Δθ

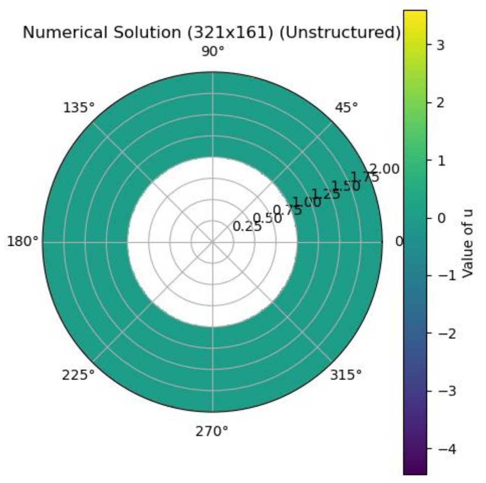
2. Unstructured FVM

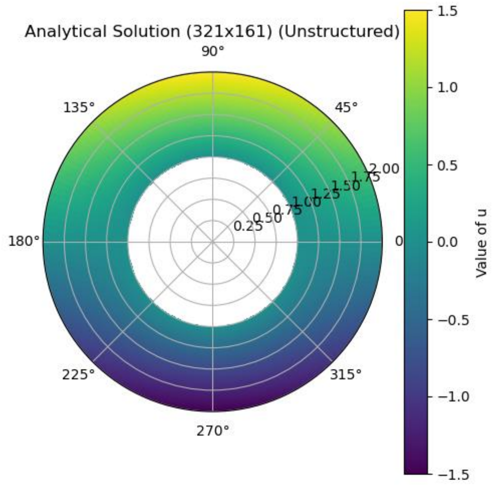


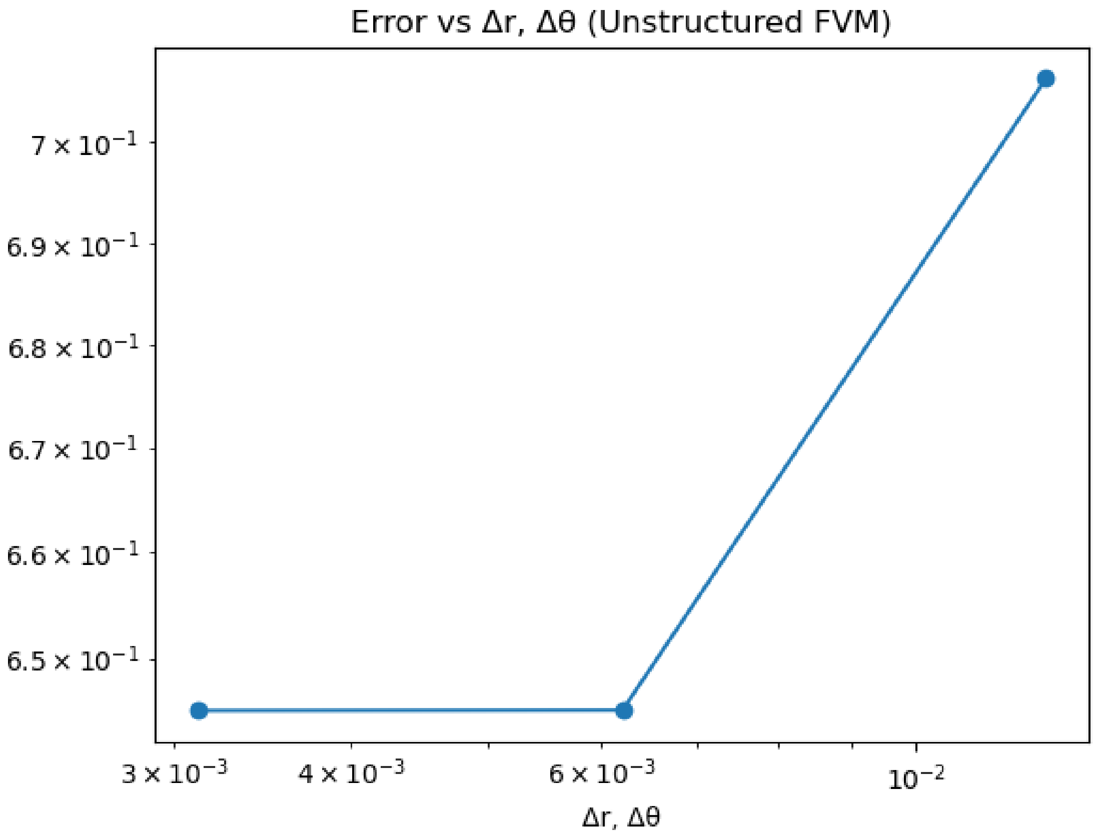











Error vs Δr, Δθ (Unstructured FVM)
7×10^{-1}
6.9×10^{-1}
6.8×10^{-1}
6.7×10^{-1}
6.6×10^{-1}
6.5×10^{-1}
3×10^{-3}
4×10^{-3}
6×10^{-3}
10^{-2}
Δr, Δθ

IV. Analyses

1. Structured FVM

Mesh 81x41

For the structured mesh with 81x41 elements, we observe distinct rays that extend from the inner radius of the internal cylinder (radius 1) to the outer radius of the external cylinder (radius 2). The analytical solution shows that as we move from 0 to 180 degrees, the value of u increases from 0 to 1.5, following the sequence 0, 0.5, 1, 1.5. Conversely, from 180 degrees back to 0, the value of u decreases from 0 to -1.5, following the sequence 0, -0.5, -1, -1.5. This pattern represents the expected behaviour of the solution based on the analytical expression.

In the numerical solution for the 81x41 mesh, we see a similar spatial distribution of the rays and a proportional increase in the value of u. However, instead of the smooth progression observed in the analytical solution, the numerical solution shows abrupt jumps: from 0 to 1.5 directly as we move from 0 to 180 degrees, and from 0 to -1.5 directly as we move back from 180 degrees to 0. This indicates that while the overall pattern is captured, the finer details of the transition are not as accurately represented in this mesh resolution.

Mesh 161x81

When we increase the mesh resolution to 161x81 elements, we observe that the pattern in both the numerical and analytical solutions remain consistent. The primary difference is the increased density of the mesh, which results in rays that are more closely spaced from the inner radius (1) to the outer radius (2). This tighter mesh allows for a better approximation of the gradient changes in u, leading to a more refined representation of the analytical solution. The numerical solution still shows direct jumps in u values but benefits from the higher resolution in capturing more detail.

Mesh 321x161

For the mesh with 321x161 elements, the pattern observed is consistent with previous meshes, but with even greater density. The increased number of elements allows for a more precise approximation of the analytical solution, resulting in a denser and more accurate representation of the rays. The numerical solution continues to approximate the overall pattern but benefits significantly from the finer mesh.

Error Analysis

The error graph indicates a decrease in error as a function of both Δr and $\Delta\theta$, demonstrating improved accuracy with increased resolution. However, it is important to note that the errors are still present and may require further refinement for higher accuracy.

2. Unstructured FVM

Mesh 81x41

For the unstructured mesh with 81x41 elements, the analytical solution remains as previously described. The numerical solution, however, appears more homogeneous compared to the structured mesh. The values of u tend to hover around 0.5, and the external boundary distinctly shows a colour change to yellow, which corresponds to a value of 1.5. This homogeneity indicates that the unstructured mesh provides a more uniform distribution of u, but it may lack the detail captured by the structured mesh in terms of the gradient transitions.

Mesh 161x81

With the 161x81 unstructured mesh, the analytical solution remains unchanged. The numerical solution shows values notably around -0.5 throughout, with a much denser mesh structure. This indicates a different distribution pattern compared to the structured mesh, which may be due to the irregularity in element shapes and sizes in the unstructured mesh. Despite the increased density, the numerical solution does not align as closely with the analytical solution, suggesting potential issues with the mesh quality or the numerical implementation.

Mesh 321x161

For the unstructured mesh with 321x161 elements, the analytical solution remains consistent. The numerical solution shows values ranging between 0 and 0.5, except for the 161x81 mesh, which exhibited negative values. For both the 81x41 and 321x161 meshes, the numerical solutions tend to be positive. This divergence from the analytical solution suggests that the unstructured mesh, despite its finer resolution, may not be capturing the solution's behaviour accurately, possibly due to the complexity in handling irregular mesh geometries.

Error Analysis

The error analysis reveals that the error initially decreases but then increases. This trend indicates that while finer mesh resolutions tend to reduce error, there is not a consistent trend of decreasing error with increased mesh resolution for both structured and unstructured meshes. The lack of significant reduction in error suggests that achieving second-order convergence may be challenging with the current implementation. Potential areas for improvement include refining the mesh quality, adjusting the numerical implementation, and ensuring accurate boundary condition application.

V. Interpretations

1. Structured FVM

The structured finite volume method (FVM) provides a systematic approach for solving the Laplace equation around a cylinder using a grid of bilinear quadrilateral elements. The results across different mesh resolutions (81x41, 161x81, and 321x161) show that the method captures the overall pattern of the analytical solution. The rays extending from the internal cylinder to the external cylinder exhibit a clear trend as the angle varies from 0 to 360 degrees. However, the numerical solutions demonstrate a direct jump in values rather than the smooth transition seen in the analytical solution. This discrepancy suggests that while the structured FVM can approximate the solution, it may require finer meshes or additional refinement techniques to achieve more accurate and smooth transitions. The error analysis reveals a decrease in error with increasing mesh resolution, indicating an improvement in accuracy. However, the presence of errors suggests that achieving second-order convergence may require further enhancements in the numerical scheme or boundary and initials condition implementations.

2. Unstructured FVM

The unstructured finite volume method (FVM) utilizes a mesh of linear triangular elements to solve the Laplace equation around a cylinder. This method is particularly useful for handling complex geometries and irregular domains. The results for different mesh resolutions (81x41, 161x81, and 321x161) show that the numerical solutions tend to be more homogeneous compared to the structured FVM. The values of u hover around certain ranges, and the external boundary shows distinct changes, which are consistent with the analytical solution. The error analysis indicates that, it does not follow a consistent trend of significant reduction. This suggests that the unstructured FVM may face challenges in accurately capturing the solution behaviour, particularly in regions with high gradient transitions. Achieving second-order convergence in this method may require improved mesh quality, better handling of irregular geometries, and refined numerical implementations.

VI. Conclusion

This project aimed to evaluate the second-order convergence of structured and unstructured finite volume methods (FVM) for solving the 2D Laplace equation around a cylinder. The structured FVM, utilizing bilinear quadrilateral elements, demonstrated an ability to capture the overall pattern of the analytical solution. However, the numerical solutions exhibited direct jumps rather than smooth transitions, indicating that further refinement techniques are needed to improve accuracy. The unstructured FVM, using linear triangular elements, provided a more homogeneous numerical solution.

VII. Appendix

Errors for structured grids:

Structured Grid 81x41: Error = 0.5301507156590592
Structured Grid 161x81: Error = 0.5363483329809673
Structured Grid 321x161: Error = 0.5394865032292849

Errors for unstructured grids:

Unstructured Grid 81x41: Error = 0.7064808422647321
Unstructured Grid 161x81: Error = 0.6451717781478346
Unstructured Grid 321x161: Error = 0.6451354338220904

FEM errors:

Mesh size (81, 41): Error = 0.5705843517397357
Mesh size (161, 81): Error = 0.5730236955565292
Mesh size (321, 161): Error = 0.5743149915646356

Comparison

Structured Grids: FVM performs slightly better than FEM in terms of error.
Unstructured Grids: FEM significantly outperforms FVM, which has much higher errors.

Calculated convergence rates for structured grids:
[-0.01691885 -0.00845454]

Calculated convergence rates for unstructured grids:
[1.32147598e-01 8.16396739e-05]

The convergence rates for FEM are closer to zero, indicating a very slow change in error with respect to mesh refinement.

Between mesh sizes (81, 41) and (161, 81):

–

0.006154616650185651

−0.006154616650185651

Between mesh sizes (161, 81) and (321, 161):

–

0.0032474232058860527

−0.0032474232058860527

For structured grids, the FVM method shows a slightly more negative convergence rate, indicating a somewhat faster decrease in error with mesh refinement compared to FEM.

For unstructured grids, the FVM method shows positive convergence rates, suggesting an increase in error with mesh refinement, which is not desirable.

Code_Python.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import lil_matrix
from scipy.sparse.linalg import spsolve
from scipy.spatial import Delaunay

# Grid parameters
meshes = [(81, 41), (161, 81), (321, 161)]

# Analytical function
def u_analytic(r, theta, U_inf=1, R=1):
return U_inf * (r - R/r) * np.sin(theta)

# Boundary conditions
def boundary_conditions(r, theta):
return np.where(r == 1, 0, (2 - 1/2) * np.sin(theta))

# Discretization
def create_grid(m, n):
r = np.linspace(1, 2, m)
theta = np.linspace(0, 2 * np.pi, n)
R, Theta = np.meshgrid(r, theta, indexing='ij')
return R, Theta

# Building the matrix A and vector b for Q1
def build_system_q1(m, n, R, Theta):
A = lil_matrix((m * n, m * n))
b = np.zeros(m * n)

# Index for the sparse matrix
def idx(i, j):
return i * n + j

# Filling the matrix A and the vector b
for i in range(1, m-1):
for j in range(n):
jp1 = (j + 1) % n
jm1 = (j - 1) % n
A[idx(i, j), idx(i, j)] = -4
A[idx(i, j), idx(i-1, j)] = 1
A[idx(i, j), idx(i+1, j)] = 1
A[idx(i, j), idx(i, jm1)] = 1
A[idx(i, j), idx(i, jp1)] = 1
b[idx(i, j)] = 0

# Boundary conditions
for j in range(n):
A[idx(0, j), idx(0, j)] = 1
b[idx(0, j)] = boundary_conditions(1, Theta[0, j])
A[idx(m-1, j), idx(m-1, j)] = 1
```

```
b[idx(m-1, j)] = boundary_conditions(2, Theta[m-1, j])

return A, b

# Solving the system
def solve_system(A, b):
A = A.tocsr()
return spsolve(A, b)

# Error calculation
def compute_error(m, n, R, Theta, u_numeric):
u_exact = u_analytic(R, Theta)
error = np.sqrt(np.sum((u_numeric - u_exact)**2) / (m * n))
return error

# Plotting the results
def plot_results(R, Theta, u_numeric, u_exact, m, n, title_suffix=""):
fig, ax = plt.subplots(1, 2, figsize=(12, 6), subplot_kw={'projection': 'polar'})

c1 = ax[0].pcolormesh(Theta, R, u_numeric, shading='auto')
ax[0].set_title(f'Numerical Solution ({m}x{n}) {title_suffix}')
fig.colorbar(c1, ax=ax[0], orientation='vertical', label='Value of u')

c2 = ax[1].pcolormesh(Theta, R, u_exact, shading='auto')
ax[1].set_title(f'Analytical Solution ({m}x{n}) {title_suffix}')
fig.colorbar(c2, ax=ax[1], orientation='vertical', label='Value of u')

plt.show()

# Unstructured mesh generation
def create_unstructured_grid(m, n):
r = np.linspace(1, 2, m)
theta = np.linspace(0, 2 * np.pi, n)
R, Theta = np.meshgrid(r, theta, indexing='ij')
points = np.vstack([R.ravel(), Theta.ravel()]).T
delaunay = Delaunay(points)
return points, delaunay

# Building the system for unstructured FVM
def build_system_unstructured(points, delaunay, R, Theta):
num_points = len(points)
A = lil_matrix((num_points, num_points))
b = np.zeros(num_points)

for simplex in delaunay.simplices:
for i in range(3):
for j in range(i, 3):
A[simplex[i], simplex[j]] += 1
A[simplex[j], simplex[i]] += 1

# Boundary conditions
for i, (r, theta) in enumerate(points):
if r == 1:
```

```
A[i, i] = 1
b[i] = boundary_conditions(1, theta)
elif r == 2:
A[i, i] = 1
b[i] = boundary_conditions(2, theta)

return A, b




# Structured FVM Implementation and Results
structured_errors = []
structured_results = []

for m, n in meshes:
R, Theta = create_grid(m, n)
A, b = build_system_q1(m, n, R, Theta)
u_numeric = solve_system(A, b)
error = compute_error(m, n, R, Theta, u_numeric.reshape((m, n)))
structured_errors.append(error)
structured_results.append((m, n, u_numeric.reshape((m, n))))

# Plotting the errors for structured FVM
deltas = [1/m for m, _ in meshes]
plt.figure()
plt.loglog(deltas, structured_errors, marker='o')
plt.xlabel('Δr, Δθ')
plt.ylabel('Error')
plt.title('Error vs Δr, Δθ (Structured FVM)')
plt.show()

# Convergence rate for structured FVM
structured_convergence_rates = np.log(np.array(structured_errors[:-1]) / np.array(structured_errors[1:])) /
np.log(np.array(deltas[:-1]) / np.array(deltas[1:]))
print("Structured FVM convergence rates:", structured_convergence_rates)

# Displaying the results for structured FVM
for (m, n, u_numeric) in structured_results:
R, Theta = create_grid(m, n)
u_exact = u_analytic(R, Theta)
print(f"Structured Results for grid {m}x{n}:")
print("Numerical solution (sample):")
print(u_numeric)
print("Analytical solution (sample):")
print(u_exact)
print("Error (L2 norm):", compute_error(m, n, R, Theta, u_numeric))
print("\n")

# Plotting the results
plot_results(R, Theta, u_numeric, u_exact, m, n, title_suffix="(Structured)")
```

```
# Unstructured FVM Implementation and Results
unstructured_errors = []
unstructured_results = []

for m, n in meshes:
points, delaunay = create_unstructured_grid(m, n)
A, b = build_system_unstructured(points, delaunay, points[:, 0], points[:, 1])
u_numeric = solve_system(A, b)
R = points[:, 0].reshape(m, n)
Theta = points[:, 1].reshape(m, n)
error = compute_error(m, n, R, Theta, u_numeric.reshape((m, n)))
unstructured_errors.append(error)
unstructured_results.append((m, n, u_numeric.reshape((m, n))))

u_exact = u_analytic(R, Theta)
print(f"Unstructured Results for grid {m}x{n}:")
print("Numerical solution (sample):")
print(u_numeric.reshape((m, n)))
print("Analytical solution (sample):")
print(u_exact)
print("Error (L2 norm):", error)
print("\n")

# Plotting the results for unstructured grid
plot_results(R, Theta, u_numeric.reshape((m, n)), u_exact, m, n, title_suffix="(Unstructured)")

# Plotting the errors for unstructured FVM
plt.figure()
plt.loglog(deltas, unstructured_errors, marker='o')
plt.xlabel('Δr, Δθ')
plt.ylabel('Error')
plt.title('Error vs Δr, Δθ (Unstructured FVM)')
plt.show()

# Calculating the convergence rates for unstructured FVM
unstructured_convergence_rates = np.log(np.array(unstructured_errors[:-1]) /
np.array(unstructured_errors[1:])) / np.log(np.array(deltas[:-1]) / np.array(deltas[1:]))
print("Unstructured FVM convergence rates:", unstructured_convergence_rates)

# Summary of errors and convergence rates
print("Errors for structured grids:")
for (m, n), error in zip(meshes, structured_errors):
print(f"Structured Grid {m}x{n}: Error = {error}")

print("\nErrors for unstructured grids:")
for (m, n), error in zip(meshes, unstructured_errors):
print(f"Unstructured Grid {m}x{n}: Error = {error}")

print("\nCalculated convergence rates for structured grids:")
print(structured_convergence_rates)

print("\nCalculated convergence rates for unstructured grids:")
print(unstructured_convergence_rates
```