NGSolve, FEniCS and FDM Solution Comparison

Öztürk S. B.

Middle East Technical University, Çankaya, Üniversiteler St., TR

E-mail: berkay.ozturk_01@metu.edu.tr

08.01.2023

Abstract. We present a benchmark project to solve a simple PDE problem, comparing NGSolve (Netgen) and FEniCS that use Finite Element Method(FEM), and a Python script that uses Finite Difference Method (FDM), with both iterative and matrix approach. We compared the results with the ease of implementation, computation time, and relative error with the known exact solution. We found that NGSolve is considerably faster than FEniCS, and the error during the computations was slightly lower in NGSolve when compared to FEniCS.

1. Introduction:

NGSolve and FEniCS are high performance FEM solvers that are used widely to analyze problems in fluid dynamics, electromagnetics and solid mechanics in a python environment. NGSolve is mainly developed in Georg-August-University Gottingen and has a variety of tutorials, called i-tutorials, covering such topics as the Poisson equation, adaptive mesh operation, magnetostatics, and Navier-stokes equations. NGSolve, furthermore, has an active and powerful community platform. FEniCS, on the other hand, is more collaborative software and developed by several institutions. Over the finite element method, we analyze the PDE problem with the finite difference method in python environment, with an iterative approach and matrix approach.

2. The Description of the Problem and Weak Form

While solving partial differential equations in computational environments with finite element method, it is crucial to get the weak form from the strong form. During the FEM, the mesh configuration is created and elements are generated within a proper function space. In our case, the strong form of the partial differential problem is,

$$-\vec{\nabla}.(\alpha_{(x,y)}\nabla u_{(x,y)}) + u_{(x,y)} = f_{(x,y)} \tag{1}$$

Here, $u_{(x,y)}$ is the primary function of interest, $f_{(x,y)}$ is the source function, $\alpha_{(x,y)}$ is a non-uniform coefficient through the space of interest. The $\alpha_{(x,y)}$ function may also be thought as the heat conduction coefficient of material recognized from the heat

diffusion/conduction problem. Where the $\alpha_{(x,y)}$ is described in a tensor form in 2 dimensional space as,

$$\alpha_{(x,y)} = (\alpha_{1(x,y)}, \alpha_{2(x,y)}) \tag{2}$$

Hence, the relation (1) reduces to,

$$-\frac{\partial}{\partial x}(\alpha_{1(x,y)}\frac{\partial u}{\partial x}) - \frac{\partial}{\partial y}(\alpha_{2(x,y)}\frac{\partial u}{\partial y}) + u = f_{(x,y)} \tag{3}$$

Where the known analytical values are, $u_{(x,y)} = 1 - x^2 - y^2$, $f_{(x,y)} = 5(1 + x^2 + y^2)$, $\alpha_{1(x,y)} = 1 + x^2$, $\alpha_{2(x,y)} = 1 + y^2$.

2.1. Obtaining the Weak Form of the Problem

The Galerkin principle is based on multiplying the strong form with a test function and integrating over the domain Ω of the problem. Then, introducing the test function "v", multiplying the strong form, and integrating over the domain,

$$\int_\Omega [-\frac{\partial}{\partial x}(\alpha_{1(x,y)}\frac{\partial u}{\partial x}) - \frac{\partial}{\partial y}(\alpha_{2(x,y)}\frac{\partial u}{\partial y}) + u]v \; dxdy = \int_\Omega f_{(x,y)} v \; dxdy \tag{4}$$

Applying integration by parts on the first term LHS of equation 4,

$$\int_\Omega -\frac{\partial}{\partial x}(\alpha_{1(x,y)}\frac{\partial u}{\partial x})v \; dxdy = \int_\Omega \alpha_{1(x,y)}\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} \; dxdy - \int_{\partial\Omega} v\alpha_{1(x,y)}\frac{\partial v}{\partial x}dS \tag{5}$$

On the second term of LHS of equation 4,

$$\int_\Omega -\frac{\partial}{\partial y}(\alpha_{2(x,y)}\frac{\partial u}{\partial y})v \; dxdy = \int_\Omega \alpha_{2(x,y)}\frac{\partial u}{\partial y}\frac{\partial v}{\partial y} \; dxdy - \int_{\partial\Omega} v\alpha_{2(x,y)}\frac{\partial v}{\partial y}dS \tag{6}$$

Hence, relation 4 reduces to,

$$\int_\Omega [\alpha_{1(x,y)}\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \alpha_{2(x,y)}\frac{\partial u}{\partial y}\frac{\partial v}{\partial y} + uv] \; dxdy = \int_\Omega f_{(x,y)} v \; dxdy + \int_{\partial\Omega} (\alpha_{1(x,y)}\frac{\partial u}{\partial x} + \alpha_{2(x,y)}\frac{\partial u}{\partial y})v \; dS \tag{7}$$

With the final weak formulation in relation 7, we can construct our system by additionally imposing the Neumann and Dirichlet boundary conditions as described,

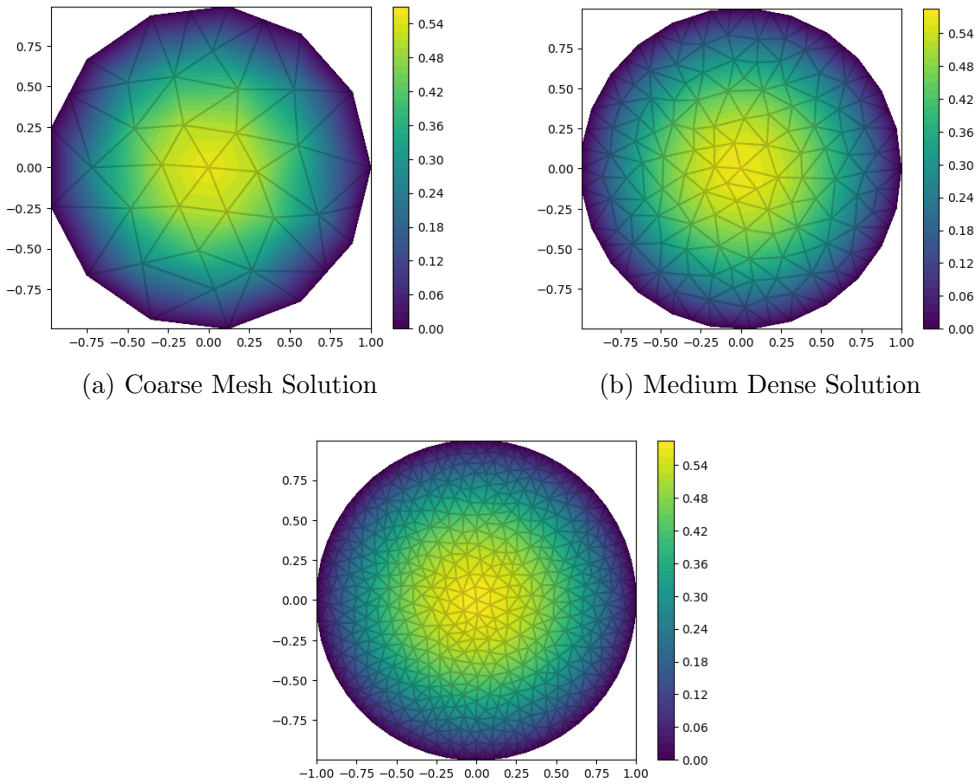
$$\Gamma_N : \alpha_1\frac{\partial u}{\partial x} + \alpha_2\frac{\partial u}{\partial y} = \hat{g} \tag{8}$$

$$\Gamma_D : u = \hat{u} = 0 \tag{9}$$

3. Solution in FEniCS

In this section, we present three solutions with varying mesh densities; coarse, medium, and dense. Furthermore, as we described in the introduction section, we compare the computational time and normalized errors by the built-in least square method for each case. The error calculation in FEniCS is done by taking the difference between the analytical solution and the numerical solution on the interpolated mesh points.

The simulation is done in Lagrange space family with degree 4, and the solutions for each mesh configuration are illustrated in figure 1.



(a) Coarse Mesh Solution

(b) Medium Dense Solution

(c) Dense Mesh Solution

Figure 1: : Illustration of the solution to the problem of weak form shown in the relation 7 by varying mesh densities, n=4, n=8, n=16, respectively.

3.1. Data Analysis

As seen from the figure, the expected maximum value was 1.0, however, the numerical solutions yield nearly half of it, with a value of about 0.54. Even though we predict the shape of the solution correctly, we suspect that the problem arose due to an error while setting the Dirichlet boundary condition, resulting in a pure Neumann boundary condition, which may be solved by imposing another boundary condition inside the

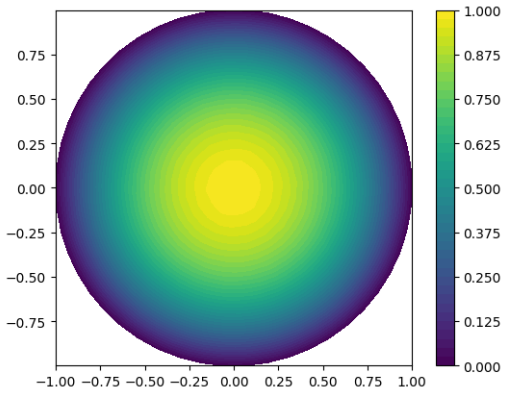


Figure 2: Illustration of the analytical solution of the problem.

domain. On the other hand, with the present values we have, the time required to run each simulation and the normal error are given in table 1, and the difference between the analytical solution and the FEM solution is illustrated in figure 3.

Table 1: Properties of the configurations in FEniCS.

Mesh Degree	Run Time [s]	Normal Error
4	1.157	0.211
8	1.203	0.188
16	1.253	0.182

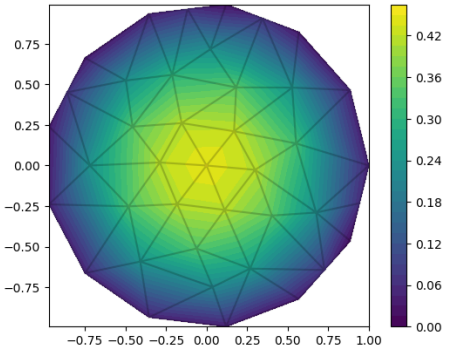
4. Solution in NGSolve

In this section, as in the FEniCS part, we present three solutions with varying mesh densities. The error analysis is done by calculating the least square method directly, without any built-in error calculation function, which is illustrated by the following equation.

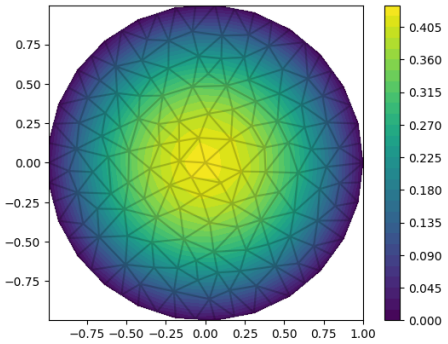
$$L2 = \sqrt{\sum_i (result_i - exact_i)^2} \tag{10}$$

Where the exact and numerical results are projected onto the mesh space.

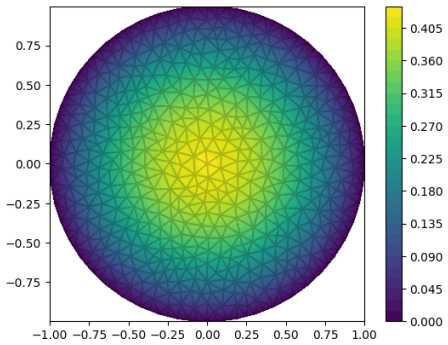
The simulation is done in Sobolev H1 space, a locally integrable set. Different from FEniCS, the mesh is generated by setting the maximum height of the elementary triangle elements, during the meshing operation. To apply coarse, medium, and dense mesh systems, we set the values of hmax as 1.0, 0.25, and 0.05. The results of the simulation are illustrated and may be seen in figure 4.



(a) Coarse Mesh Error



(b) Medium Dense Error



(c) Dense Mesh Error

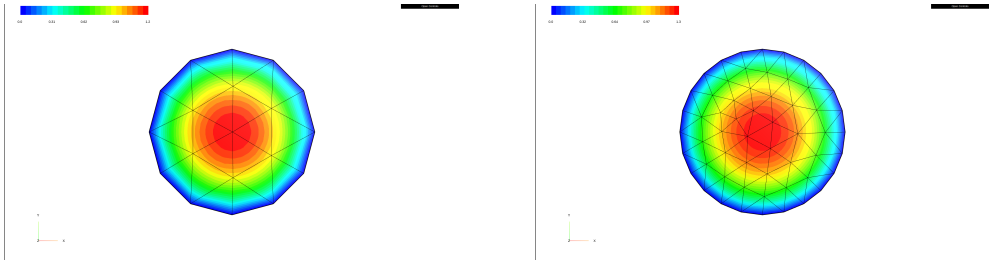
Figure 3: : Illustration of the difference between the analytical and the numerical results to the problem of the weak form shown in the relation 7 by varying mesh densities, n=4, n=8, n=16, respectively.

Table 2: Properties of the configurations in NGSolve.

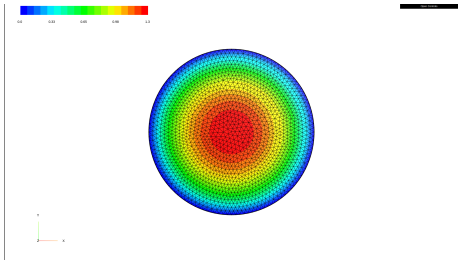
h_{max}	Run Time [s]	Normal Error
1.0	0.00548	0.0946
0.25	0.0207	0.152
0.05	0.611	0.173

4.1. Data Analysis

As was discussed in the error analysis in the FEniCS section, and seen in figure 2, the expected value of the primary variable was 1.0 at the center. However, the solutions in the NGSolve environment resulted in slightly higher values for each varying mesh case. The time required to run each simulation and the normal error in NGSolve are given in table 2, and the difference between the analytical solution and the FEM solution is illustrated in figure 5.

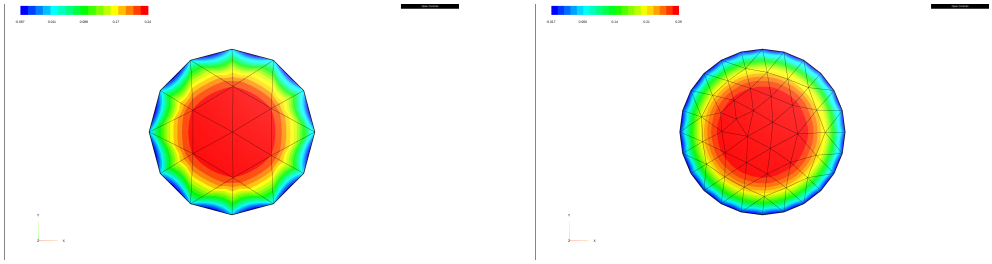


(a) Coarse Mesh Solution (b) Medium Dense Solution

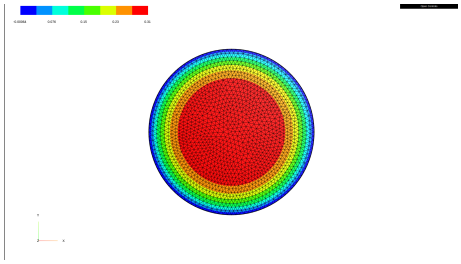


(c) Dense Mesh Solution

Figure 4: : Illustration of the solution to the problem of the weak form shown in the relation 7 by varying mesh densities, $h_{max}$of 1.0, 0.25, 0.05, respectively.



(a) Coarse Mesh Error (b) Medium Dense Error



(c) Dense Mesh Error

Figure 5: : Illustration of the difference between the analytical and the numerical results to the problem of the weak form shown in the relation 7 by varying h_{max} of 1.0, 0.25, 0.05, respectively.

5. Finite Difference Solution

The finite difference method uses the strong formulation of the partial differential problem. Taking the α value as 1, we solve the PDE described in equation 1. The

FDM gathers the numerical solution by approximating the derivatives in a discrete space, then the solution is gathered by iterating through the discrete space or using matrix elements to gather the solution for the primary variable.

Equation 1 in partial differential form with $\alpha = 1$ yields,

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + u = f_{(x,y)} \tag{11}$$

Hence, using the discretization of derivatives by central difference approximation,

$$\frac{\partial^2 u_{i,j}}{\partial x^2} = \left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} \right) \tag{12}$$

$$\frac{\partial^2 u_{i,j}}{\partial y^2} = \left(\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} \right) \tag{13}$$

Where i and j indices represent the elements of x and y axes, respectively. As we use a uniformly discretized space, setting $\Delta x = \Delta y = h$, and inserting the discrete derivatives into relation 11 we get,

$$-\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \right) - \left(\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} \right) + u_{i,j} = f_{(x_i,y_j)} \tag{14}$$

Leaving the primary term $u_{i,j}$ term alone, we get the final description for the FDM solution,

$$u_{i,j} = \frac{h^2 f_{(x_i,y_j)} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{4 + h^2} \tag{15}$$

5.1. Implementation of Iterative and Matrix Approaches

To solve the scheme described by a discretized space as in equation 15, we may use an iterative approach or setting matrix elements and solve the system numerically. In the iterative approach, one may choose a starting point, which we have chosen as the bottom left, and start iterating through the x-axis, then jumping to the next y-axis element and iterating through the x-axis again. After all elements are iterated and computed, it is crucial to restarting the procedure with the new values to get a converging value. The matrix approach, on the other hand, generates a sparse matrix and solves for the column vector of primary function values at grid points. The description of the matrix solution is,

$$(A_{N^2,N^2})\ (u_{N^2,1}) = (f_{N^2,1}) \tag{16}$$

where A is the sparse matrix for the coefficients of the discrete elements, u is the column matrix for the primary variable to be solved, and f is the column matrix of the discretized source values. More generally,

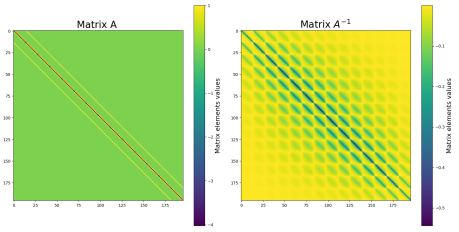
$$\begin{bmatrix} T & I & 0 & . & . & 0 \\ I & T & I & 0 & . & . \\ 0 & I & T & . & . & . \\ . & 0 & . & . & . & . \\ . & . & . & . & . & 0 \\ . & . & . & . & T & I \\ 0 & . & . & 0 & I & T \end{bmatrix} \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ . \\ . \\ . \\ u_{N,N} \end{bmatrix} = \begin{bmatrix} h^2 f_{1,1} \\ h^2 f_{2,1} \\ . \\ . \\ . \\ h^2 f_{N,N} \end{bmatrix} \tag{17}$$

where I is the identity matrix, and T is the sparse matrix with diagonal values of $-4 - h^2$, upper diagonal values, and lower diagonal values of 1 with matrix dimensions of N-N.

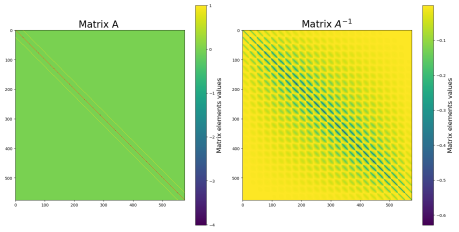
Hence, inverting the matrix A, and multiplying with the source matrix on the LHS, we acquire the solution for the u matrix as follows,

$$u = fA^{-1} \tag{18}$$

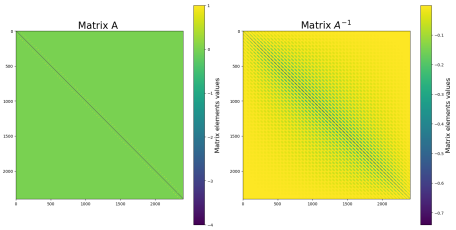
The sparse matrices and inverse of them generated in python for 15, 25, and 50 discrete points for each axis may be seen in the figure below.



(a) n=15



(b) n=25

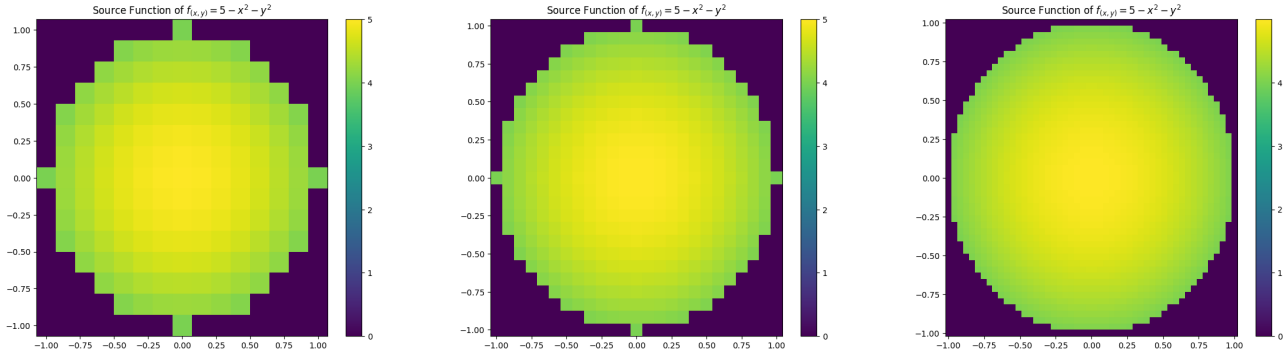


(c) n=50

Figure 6: Illustration of sparse matrices and inverse of them, for element values of 15, 25, and 50, respectively.

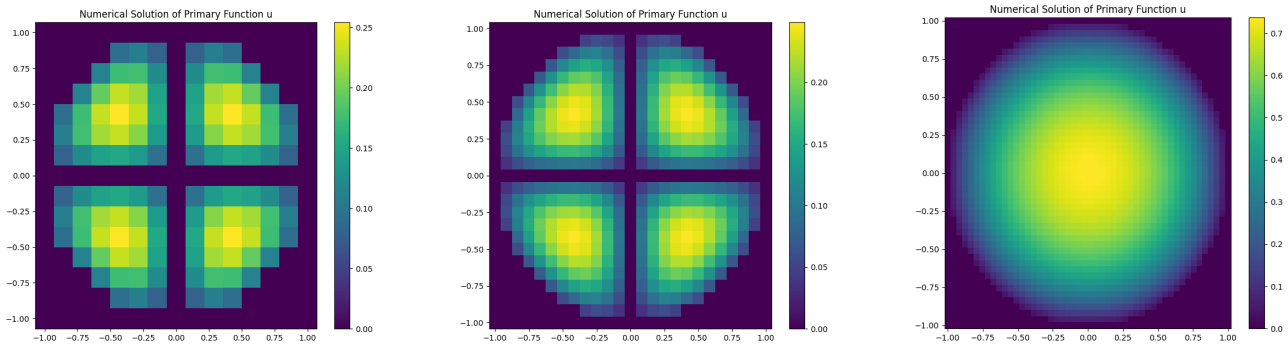
5.2. Results for Iteration

Using FDM, we solved equation 15 with 250 iterations for 15, 25, and 50 discretized elements. We illustrate the source function, numerical solution, and the absolute difference between the analytical and the numerical solution. The results may be seen in the figures below.



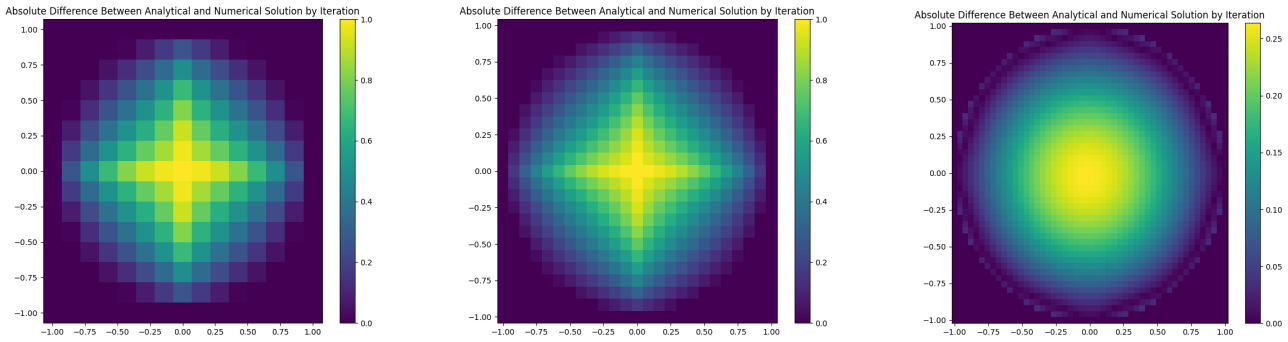
(a) n=15 (b) n=25 (c) n=50

Figure 7: Illustration of source function for element values of 15, 25, and 50, respectively.



(a) n=15 (b) n=25 (c) n=50

Figure 8: Illustration of numerical solution of u, for element values of 15, 25, and 50, respectively.

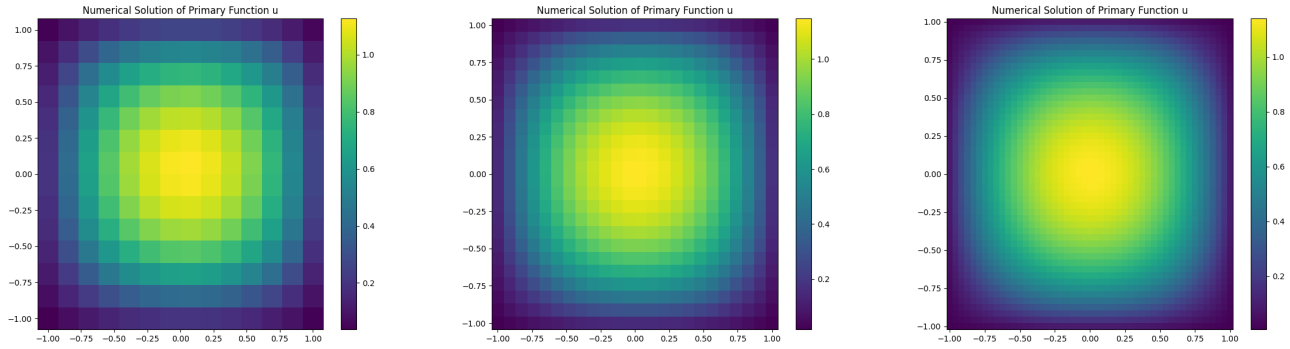


(a) n=15 (b) n=25 (c) n=50

Figure 9: Illustration of absolute error, for element values of 15, 25, and 50, respectively.

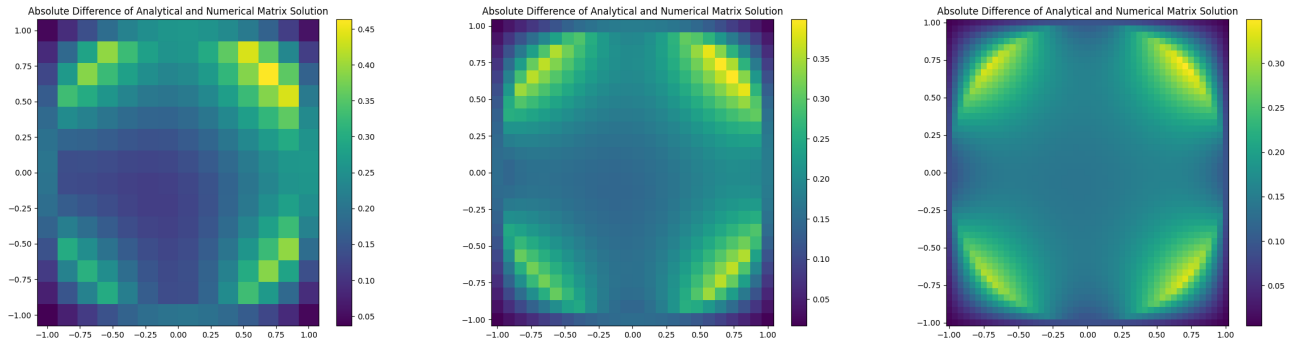
5.3. Results for Matrix Solution

With the use of the sparse matrices illustrated in figure 6, we acquired the primary value of u according to equation 18. The results may be seen in the figures below.



(a) n=15 (b) n=25 (c) n=50

Figure 10: Illustration of solution of primary value u by matrix solution, for element values of 15, 25, and 50, respectively.



(a) n=15 (b) n=25 (c) n=50

Figure 11: Illustration of absolute error of matrix solution, for element values of 15, 25, and 50, respectively.

5.4. Data Analysis

We applied χ^2 test of each acquired solution for both iterative approach and matrix solution. The results may be seen from table 3. Also note that there is an obvious error through the x and y axes for a low number of points, which is overcome with the increase of discrete points. On the other hand, there is an error trend on the edges of the matrix solution, which we believe is due to the numerical dispersion while calculating the sparse matrix and inverting it.

Table 3: χ^2 values for the iterative solution.

n	χ^2	dof	χ^2/dof
15	46.7	145	0.0946
25	137.1	437	0.152
50	47.1	1876	0.025

Table 4: χ^2 values for the matrix solution.

n	χ^2	dof	χ^2/dof
15	32.12	124	0.259
25	145.2	408	0.356
50	379.2	1789	0.212

6. Conclusion

We simulated the problem described in equation (1) by FEM and FDM. For FEM, we used FEniCS and NGSolve to solve the weak form acquired in equation (7). We found NGSolve is considerably faster and slightly more accurate when compared to FEniCs. We think why NGSolve was more successful is that NGSolve uses H1 Sobolev space rather than Lagrange space, which is found to be more accurate. We also saw that, for both libraries, normal error increases as we increase the mesh density. The reason, we believe, is due to the accumulation of computationally generated rounding errors. Another reason could be due to an error while setting the Dirichlet boundary condition, giving rise to a pure Neumann boundary condition. On the other hand, the implementation of the problem was easier in NGSolve as it directly recognizes the equations written in Python syntax, and FEniCS uses C++ syntax to recognize the generated equations. On the FDM side, the iterative approach for solving the problem described in equation (15) was more accurate than the matrix approach overall. Also, as seen in figure 11, we noticed that there was an error trend on the edges which may have arisen from the rounding off errors in sparse matrix calculation or an implementation error for the Dirichlet boundary condition.