# Solving Two-Dimensional Boundary Value Problems on Circular and Square Domains with FEM

Mudassar Amin

December 7, 2023

## 1 Introduction

The Finite Element Method (FEM) is a numerical technique for approximating solutions to Partial Differential Equations (PDEs) that describe continuous physical processes.FEM is particularly useful when analytical solutions are difficult or impossible to obtain. It simplifies complex problems by dividing them into smaller, manageable elements, each represented by simpler equations.

Originating from the need to solve intricate engineering challenges, FEM translates the principles of mechanics, thermodynamics, and other physical laws into a computational framework. It enables simulations of structural behaviors, thermal flows, and electromagnetic fields with remarkable accuracy.

## 2 Tools and Libraries used to develop the Application

- Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms[1]. The library excels in handling fixed-size matrices by avoiding dynamic memory allocation and using unrolled loops. Additionally, Eigen supports standard numeric types like complex numbers, integers, and floating-point numbers, and provides an interface for custom types[5].

- TTL is a generic triangulation library developed at SINTEF Applied Mathematics. It is generic in the sense that it does not rely on a specific data structure, allowing operation with custom application data structures while benefiting from a variety of generic algorithms in TTL that work directly on any data structure for triangulations. TTL is written in C++ and extensively uses function templates as a generic tool for the application programming interface (API)[4].

- Qt Creator is a cross-platform integrated development environment (IDE) that provides application developers with all the tools needed to create applications for multiple desktop and mobile device platforms[2].

## 3 One-Dimensional Boundary Value Problem Example

To demonstrate the differential approach to solving physical problems, we start with a one-dimensional boundary value problem using Poisson's equation as a typical example.[3]

### 3.1 Domain and Equation

- One-dimensional continuous domain $I = [0, 1]$ with length $L = 1$.
- Poisson's equation in one-dimensional space is given by:

$$-\frac{d}{dx}\left(a\frac{du}{dx}\right) = f \text{ in } I.$$

### 3.2 Analytical Solution

- Assuming constants $a = 1$ and source function $f = 1$, the equation simplifies to $-u'' = 1$.
- Integration leads to $u = -\frac{x^2}{2} - C1x - C2$.
- Applying Dirichlet boundary conditions $u(0) = 0.25$, $u(1) = 0$, we find $C1$ and $C2$.
- The exact solution is $u_{\text{exact}} = -0.5x^2 + 0.25x + 0.25$.

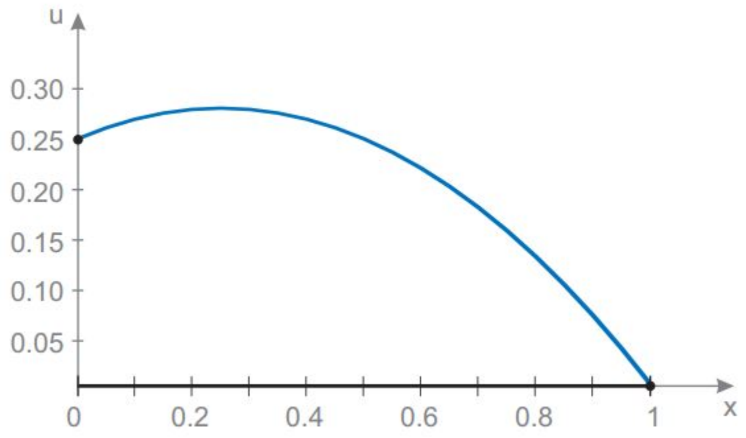

Figure 1: Exact Solution

### 3.3 Numerical Solution

- Finite Element Method (FEM) is employed as the numerical approach.
- The solution process involves discretizing the domain, formulating the problem variationally, and solving a system of linear equations.

- Boundary conditions are handled using Robin boundary conditions.
- The linear system $(A + R)\zeta = b + r$ is solved for coefficients $\zeta$.

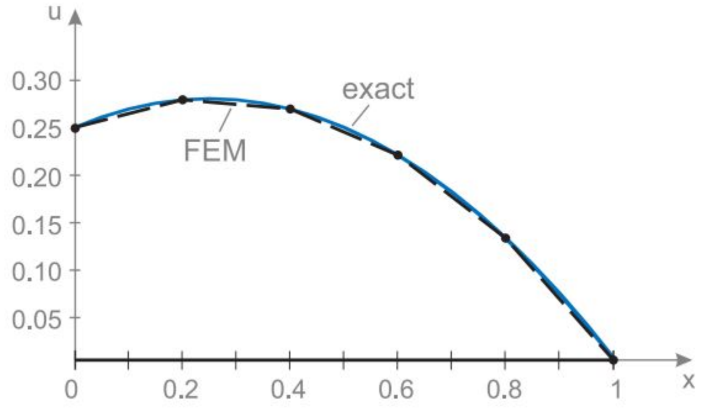

Figure 2: Comparison of fem solution and exact solution.

### 3.4 Error Analysis

- Error between the exact and computed solutions is measured using the Euclidean norm.
- Error analysis demonstrates how FEM converges with mesh refinement.

## 4 2-Dimensional Boundary Value Problem

In this case, we apply the finite element method to two-dimensional problems. The main goal is to create simple polynomial functions that we can easily use for calculations. These functions help us estimate the solution to our boundary value problem in a two-dimensional space.

### 4.1 Triangulation

A triangulation involves dividing a domain into triangles that are connected and do not overlap. These triangles are constructed using a set of points $P$ within the domain $\Omega$. The boundary of $\Omega$ forms a simple polygon, meaning it does not intersect itself. Typically, it is advantageous for $\Omega$ to be the convex hull of the points set, encompassing all the points.

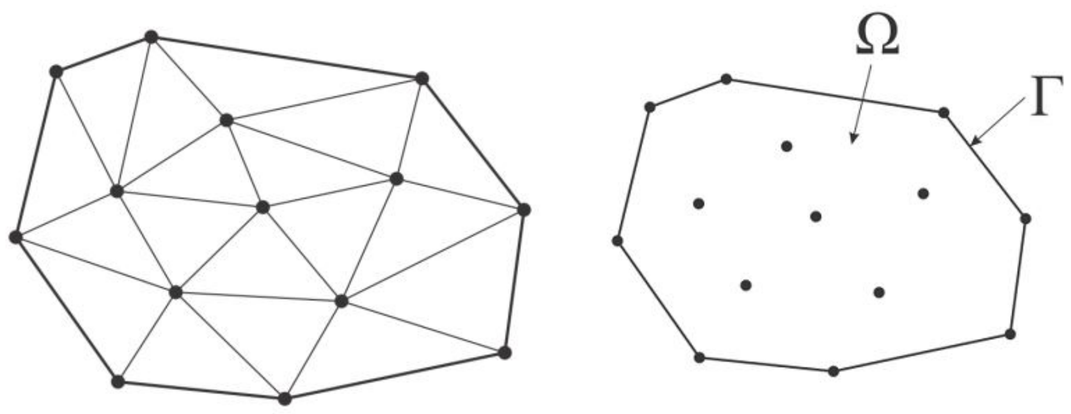

Figure 3: The convex hull of a set of points $P$ is defined as the smallest convex set that contains $P$. A set $S$ is considered convex if, for any two points within $S$, the line segment joining these points lies entirely within $S$.

Triangulations can vary, and they're often optimized to avoid thin or nearly flat triangles. A good triangulation typically has triangles that are almost equal in angles. When choosing from different triangulations of the same points, we prefer one with either the smallest largest angle or the largest smallest angle. These approaches are called the MinMax and MaxMin angle criteria, respectively. A Delaunay triangulation optimally uses the MinMax angle criterion and is based on the convex hull of a point set P. Its key feature is that no point in P lies inside the circumcircle of any triangle in the triangulation. The decision for triangulation in a quadrilateral depends on the sum of opposite interior angles, following the circumcircle test.[3]

In the Finite Element Method (FEM) for solving various Boundary Value Problems (BVPs), we need to find the following matrix and vector to compute the system of linear system:

- **Stiffness Matrix** $A$: The elements of the stiffness matrix are computed using the formula:
$$A_{ij} = \int_{\Omega} \nabla\phi_j \cdot \nabla\phi_i \, dx, \quad i, j = 1, \dots, N.$$
- **Mass Matrix** $M$: The entries of the mass matrix are given by:
$$M_{ij} = \int_{\Omega} \phi_j \phi_i \, dx, \quad i, j = 1, \dots, N.$$
- **Load Vector** $b$: The load vector is defined as:
$$b_i = \int_{\Omega} f \phi_i \, dx, \quad i = 1, \dots, N.$$
  where $f$ is the given source function.

- **Robin Matrix** $R$: The entries of the Robin matrix are given by:

$$R_{ij} = \int_{\Gamma} \kappa \phi_j \phi_i \, ds, \quad i, j = 1, \ldots, N.$$

  where $\kappa$ is the constant defining the type of boundary conditions.

- **Robin Vector** $r$: The elements of the Robin vector are computed as:

$$r_i = \int_{\Gamma} (\kappa g_D + g_N) \phi_i \, ds, \quad i = 1, \ldots, N.$$

  where $g_D$ and $g_N$ specify the inhomogeneous Dirichlet or Neumann boundary conditions, respectively.

## 5 Application Structure

The application is structured around a central class, FEMobject, which encapsulates the entire FEM process. Which include the following components:

### 5.1 FEMobject

This is the core class of the application, encapsulating the entire FEM process.

**Attributes:**

- Nodes, triangulation, matrices (A, M, R), vectors (b, r), problem type.

**Methods:**

- Mesh Generation: `CircleMesh`, `SquareMesh`. These methods generate meshes for circular and square domains, respectively. They discretize the domain into finite elements necessary for the FEM analysis, below in alogorihtm 1 & 2 we can see the pseudocodes, where n is the number of rings and m is the number of points while r is the radius.

- Material Properties and Boundary Conditions: `kappa`, `gN`, `gD`, `f`. These methods define material properties (kappa) and boundary conditions (gN for Neumann, gD for Dirichlet). The function f typically represents the source term or forcing function in the PDE.

- Geometric Calculations: `triarea`, `gradients`. Calculates the area of a triangle, a fundamental geometric operation in FEM for element integration. While the gradient method computes the gradient of shape functions within elements, crucial for assembling stiffness and mass matrices.

- Assembly of Matrices and Vectors: `stiffMat`, `massMat`, `loadVect`, `RobinMat`, `RobinVect`

- Solution: `solve`. This method solves the FEM problem by assembling all matrices and vectors and then applying a numerical solver.

- Output and Visualization: `visualization`

- Error Analysis: `uexact`, `getError`

- Degrees of Freedom: `getDoFs`

## 5.2 Main Function

In the main we created an instance of the class FEMobject named model then we can get access to the other methods to provide the required details for computation.

Implementation code: Mudassar Amin's FEM Repository.

**Algorithm 1** CircleMesh

```
1: function CircleMesh($n, m, r$)
2:     Initialize an empty list for nodes
3:     Create and add the central node at (0, 0)
4:     for each circle layer $k = 0$ to $n - 1$ do
5:         for each node $i = 0$ to $m \times (k + 1) - 1$ do
6:             $a \leftarrow 2 \times \pi \times i/(m \times (k + 1))$
7:             Calculate position $(x, y)$ using polar coordinates $(r \times (k + 1)/n, a)$
8:             Create a new node at $(x, y)$
9:             Add the new node to the nodes list
10:        end for
11:    end for
12:    Perform Delaunay triangulation on the nodes list
13: end function
```

**Algorithm 2** SquareMesh

```
1: function SquareMesh($n, d, Op$)
2:     Initialize an empty list for nodes
3:     for $j = 0$ to $n$ do
4:         for $i = 0$ to $n$ do
5:             $(x, y) \leftarrow (Op.x + i \times d/n, Op.y + j \times d/n)$
6:             Create a new node at $(x, y)$
7:             Add the new node to the nodes list
8:         end for
9:     end for
10:    Perform Delaunay triangulation on the nodes list
11: end function
```

# 6 Results

We solved the three following numerical problems with our applications with error analysis.

## 6.1 Laplace's Equation

Consider Laplace's equation in a domain $\Omega$ with the boundary $\Gamma$:

$$\begin{aligned} \Delta u &= 0, \text{ in } \Omega, \\ u &= \cos 4\phi, \text{ on } \Gamma \end{aligned}$$

**Exact Solution:**

$$u_{\text{exact}} = \rho^4 \cos 4\phi$$

**FEM Formulation:**
In FEM formulation, the equation can be represented as:

$$A + R\zeta = r$$

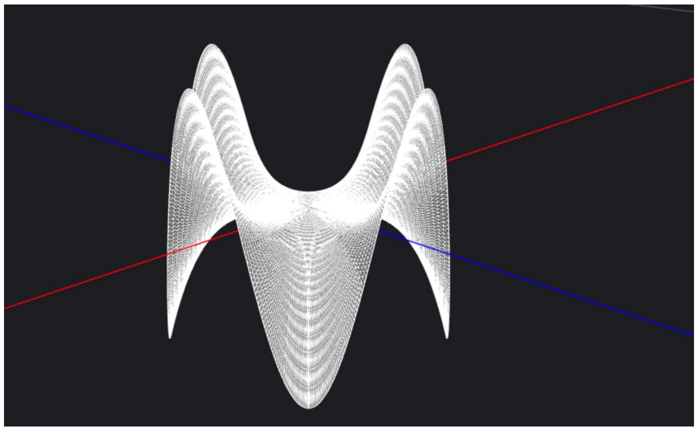

Figure 4: FEM result of Laplace's Equation using circulat mesh

## 6.2 Poisson's Equation

Consider Poisson's equation in a domain $\Omega$ with the boundary $\Gamma$:

$$-\Delta u = 1, \text{ in } \Omega,$$
$$u = \frac{y^2}{2}, \text{ on } \Gamma$$

The exact solution is given by:

$$u_{\text{exact}} = \frac{1 - x^2}{2}$$

**FEM Formulation:**
In FEM formulation, the equation can be represented as:

$$A + R\zeta = b + r$$

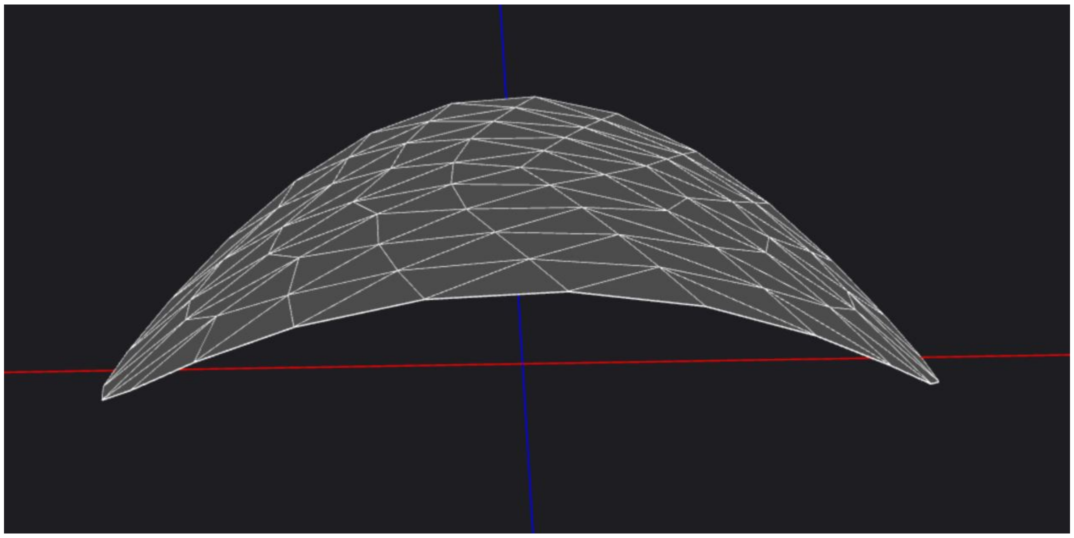

Figure 5: FEM result of Poisson's Equation using circular mesh

## 6.3 Helmholtz Equation

Consider the Helmholtz equation in a domain $\Omega$ with Dirichlet boundary $\Gamma_D$ and Neumann boundary $\Gamma_N$:

$$\begin{aligned} -\Delta u - \lambda u &= 0, \text{ in } \Omega, \\ u &= 0.25, \text{ on } \Gamma_D, \\ n \cdot \nabla u &= 0, \text{ on } \Gamma_N \end{aligned}$$

The exact solution is given by:

$$u_{\text{exact}} = (\cos(\lambda x) + \tan(\lambda)\sin(\lambda x)) * 0.25$$

**FEM Formulation:**
In FEM formulation, the equation can be represented as:

$$A + R - \lambda M\zeta = r$$

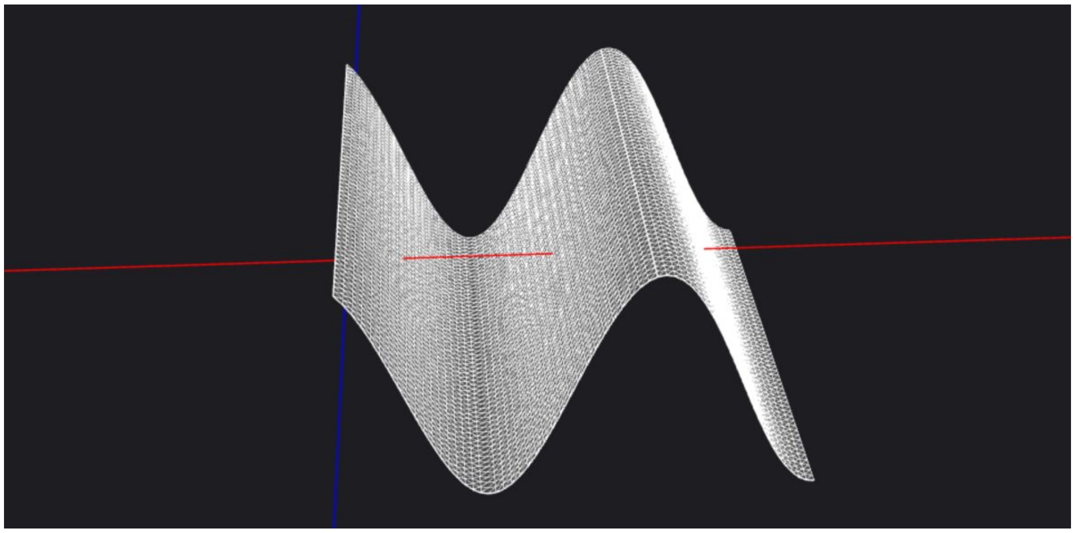

Figure 6: FEM result of Helmholtz's Equation where $\lambda$=81

| Rings | Nodes | DoFs | Laplace | Poisson |
|---|---|---|---|---|
| 5 | 5 | 76 | 0.031851 | 0.016443 |
| 7 | 8 | 225 | 0.010348 | 0.004128 |
| 12 | 10 | 781 | 0.0067635 | 0.0011044 |
| 14 | 12 | 1261 | 0.0050934 | 0.00070105 |
| 18 | 16 | 2737 | 0.0031583 | 0.00036521 |
| 20 | 18 | 3781 | 0.0025752 | 0.00028432 |
| 25 | 20 | 6501 | 0.001656 | 0.00017678 |
| 30 | 25 | 11626 | 0.0011577 | 0.00011767 |

Table 1: Data for various FEM problems

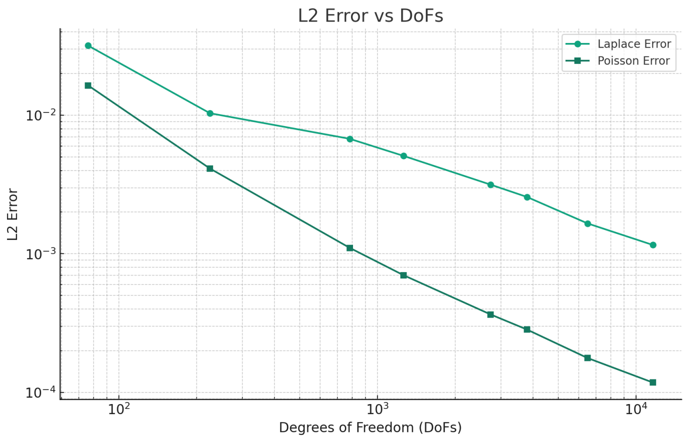

Figure 7: Error

**Increasing DoFs Means Finer Mesh:** As we can see from the tables the error is decreasing as the dof's increasing so, more degrees of Freedom (DoFs) typically imply a finer

| Helmholtz | DoFs | Nodes |
|---|---|---|
| 0.0099451 | 441 | 20 |
| 0.0068786 | 625 | 24 |
| 0.0050441 | 841 | 28 |
| 0.0010954 | 3721 | 60 |

Figure 8: Helmholtz Error Data

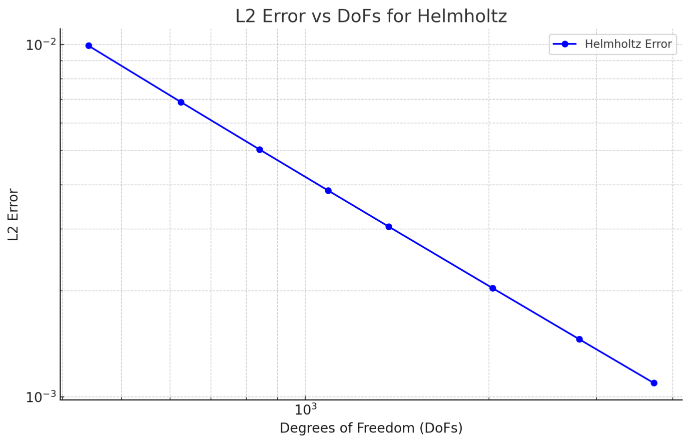

Figure 9: Error Analysis for Helmholtz Problem

mesh in Finite Element Method (FEM) analysis. A finer mesh can more accurately represent the geometry of the problem and capture the nuances of the solution field.

**Error Reduction with Finer Mesh:** As the mesh becomes finer, i.e., with more DoFs, the approximation of the solution becomes more accurate. This increased accuracy is due to the finer mesh's ability to better approximate the gradients and curvatures of the true solution, leading to a decrease in the L2 error.

**Convergence of the Numerical Method:** The decrease in error with increasing DoFs indicates that the numerical method is converging. In other words, as the mesh is refined, the numerical solution is getting closer to the actual (analytical) solution of the differential equation.

## References

[1] Eigen - a c++ template library for linear algebra. `https://eigen.tuxfamily.org/index.php?title=Main_Page`, 2023. Accessed: 2023-12-05.

[2] Qt creator documentation. `https://doc.qt.io/qtcreator/`, 2023. Accessed: 2023-12-05.

[3] Tatiana Kravetc. Fem theory and implementation. DTE-3612, Department of Computer Science and Computational Engineering, Faculty of Engineering Science and Technology, 2023. UiT - The Arctic University of Norway.

[4] SINTEF Applied Mathematics. Ttl: The triangulation template library, 2010. Accessed: 2023-12-05.

[5] Mikhail Sizov. Efficient c++ implementation of custom fem kernel with eigen, January 2019.