# Project 1 - TMA 4220 - 2022

Bjørnar Ø. Kaarevik Isak Ytrøy Gjermund Oscar Lyckander

October 1, 2022

## 1 Introduction

To start our foray into numerical solution of problems using the finite element method, we need a library of functioning routines for posing questions and calculating answers.

In this project, we implement routines for solving the Poisson problem on the unit disk with an arbitrary function on the righthand side. Treatment of both homogeneous Dirichlet boundary conditions on the entire boundary and Neumann boundary conditions on the upper half of the boundary are included.

### Testing the quadratures

Both the 1D and 2D quadratures were tested against known values and found to be precise to several decimal places.

The 1D quadrature was successfully adapted to perform line integrals later on in the project. The 2D quadrature was only used to integrate the linear form, because the bilinear form had a simple structure. It could be expressed as an integral over constants and thus reduced to an area times the constants.

## 2 The Poisson Problem

One famous partial differential equations is the Poisson equation, which is a heterogeneous variant of Laplace's equation. Here we will study the following variant:

$$\begin{cases} \nabla^2 u = -f & , \quad (x,y) \in \Omega \\ u(x,y) = 0 & , \quad (x,y) \in \partial\Omega \end{cases} \tag{1}$$

In this case the load function $f$ is given as

$$f(x,y) = -8\pi \cos(2\pi(x^2+y^2)) + 16\pi^2(x^2+y^2)\sin(2\pi(x^2+y^2)).$$

Our domain is the unit disk

$$\Omega = \{(x,y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$$

### 2.1 Finding a solution

Finding an exact solution can be a hairy process. Fortunately, a solution has been suggested:

$$u(x,y) = \sin(2\pi(x^2+y^2)) \tag{2}$$

We need now only verify that it satisfies equation 1. Applying the $\nabla$ -operator twice, and checking the boundary value should be sufficient. By the definition of $\nabla$, the left hand side of the boundary value problem in equation 1 is

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

The fact that $x$ and $y$ appear in identical manners in the function $u(x,y)$, makes this process simple. We need only find the second derivative for one variable and replace it for the other wherever the chain rule was applied, so:

$$\frac{\partial u}{\partial x} = 4\pi x \cos(2\pi(x^2+y^2))$$

$$\frac{\partial^2 u}{\partial x^2} = 4\pi \cos(2\pi(x^2+y^2)) - 8\pi^2 x^2 \sin(2\pi(x^2+y^2))$$

Next, we insert $y$ instead and add up our second derivatives, yielding:

$$\begin{aligned}\nabla^2 u &= 4\pi \cos(2\pi(x^2+y^2)) - 8\pi^2 x^2 \sin(2\pi(x^2+y^2)) + 4\pi \cos(2\pi(x^2+y^2)) - 8\pi^2 y^2 \sin(2\pi(x^2+y^2)) \\ &= 8\pi \cos(2\pi(x^2+y^2)) - 16\pi^2 (x^2+y^2) \sin(2\pi(x^2+y^2))\end{aligned}$$

This equals $-f$! And as $x^2 + y^2 = 1$ on our boundary we get

$$u(x,y)\big|_{\partial\Omega} = \sin(2\pi) = 0$$

Thus, our suggested function solves the boundary value problem in equation 1.

## 2.2 The weak form

The PDE, in its current form, requires the solution to be a twice differentiable function. This sounds reasonable, but a smarter solution would be to utilize weak derivatives, i.e. expressing the PDE in terms of integration by parts. The consequence is that we no longer look for a solution in the space $C^2$, but instead in a Sobolev space. A Sobolev space is still a function space, and in this case the relevant Sobolev space is $H^1$. This is the space of functions which have square integrable derivatives, in addition to being square integrable themselves. Thus, the inner product becomes

$$\langle f, g \rangle_{H^1} = \langle f, g \rangle_{L^2} + \langle f', g' \rangle_{L^2}$$

For a single function, the induced norm is

$$\|f\|^2_{H^1} = \|f\|^2_{L^2} + \|f'\|^2_{L^2}$$

This norm is hard to visualize, but its purpose is to require a well-behaved function with finite "energy".

With this in mind, we can find a new expression for our PDE. First we will need to generalize integration by parts in higher dimensions, this is done with Green's first identity:

$$\int_\Omega v \nabla^2 u \, d\Omega = \oint_{\partial\Omega} v(\nabla u \cdot \mathbf{n}) \, ds - \int_\Omega \nabla v \cdot \nabla u \, d\Omega \tag{3}$$

Here $\mathbf{n}$ is the typically outward facing unit vector on the boundary. To find the weak formulation we multiply the PDE with the test function $v$, then integrate:

$$\begin{aligned} v\nabla^2 u &= -vf \\ \int_\Omega v \nabla^2 u \, d\Omega &= -\int_\Omega vf \, d\Omega \end{aligned}$$

Next we apply Green's first identity 3 on the left hand side:

$$\oint_{\partial\Omega} v(\nabla u \cdot \mathbf{n}) \, ds - \int_\Omega \nabla v \cdot \nabla u \, d\Omega = -\int_\Omega vf \, d\Omega$$

Due to the homogenous Dirichlet conditions on the boundary, this simplifies to:

$$\int_\Omega \nabla v \cdot \nabla u \, d\Omega = \int_\Omega vf \, d\Omega \tag{4}$$

This is our weak form of the Poisson problem in equation 1, which also gives the linear form $l(v)$, and the bilinear form $a(u, v)$ as:

$$\begin{aligned} a(u,v) &= \iint_\Omega \nabla v \cdot \nabla u \, dxdy \\ l(v) &= \iint_\Omega vf \, dxdy \end{aligned}$$

## 2.3 Galerkin Projection

To solve the boundary value problem we will apply Galerkin's method, which is based on looking for the projection of $u(x, y)$ in a subspace of $H^1$. Practically, this means that the subspace $X_h \subset H^1$ is spanned by basis functions $\varphi_i \in \mathbb{P}^1$, which have local support on compact triangular elements. The triangular elements are simply triangles in $\mathbb{R}^2$ defined by their corner nodes, where we also evaluate $u(x, y)$. With this we can express the projection $u_h$ and the test function $v$ as:

$$u_h = \sum_{i=1}^{n} u_h(x_i, y_i)\varphi_i(x, y)$$
$$v = \sum_{i=1}^{n} v(x_i, y_i)\varphi_i(x, y)$$

Recalling the variational form we can insert the above and get

$$a(u_h, v) = a(\sum_{i=1}^{n} u_h(x_i, y_i)\varphi_i, \sum_{i=1}^{n} v(x_i, y_i)\varphi_i),$$

and thus by bilinearity

$$a(u_h, v) = \sum_{i=1}^{n}\sum_{j=1}^{n} v_i \; a(\varphi_i, \varphi_j) \; u_{hj}.$$

This can be written more compactly using vector notation:

$$a(u_h, v) = \mathbf{v}^T \mathbb{A} \mathbf{u}_h$$

Where $\mathbb{A}$ is the so-called stiffness matrix, $\mathbf{v}^T$ represents the test function and $\mathbf{u}_h$ will hold our numerical approximation of $u(x, y)$. Next, we look at the linear form

$$l\left(\sum_{i=1}^{n} v_i\varphi_i\right) = \sum_{i=1}^{n} v_i \; l(\varphi_i),$$

which again can be simplified with vector notation

$$l(v) = \mathbf{v}^T \mathbb{F}_h$$

The vector $\mathbb{F}_h$ will be the corresponding numerical load to the discrete points in $\mathbf{u}_h$ which we are solving for. Finally, we can observe that the vector equation

$$\mathbf{v}^T \mathbb{A} \mathbf{u}_h = \mathbf{v}^T \mathbb{F}_h$$

has the same solution as

$$\mathbb{A}\mathbf{u}_h = \mathbb{F}_h \tag{5}$$

Thus we have shown that the problem of finding a $u_h \in X_h \subset H^1$ such that $a(u_h, v) = l(v)$ is solved by the matrix equation 5.

## 2.5 Stiffness Matrix

Before treatment of the boundary conditions, the stiffness matrix is singular because the boundary nodes mistakenly are described to depend on the internal nodes. The stiffness matrix at this point is built from elemental contributions which also contribute to determine the nodal values at the boundary. This results in a linear dependency in the columns of the stiffness matrix, since some nodal values may be described in terms of the others.

To rectify this issue, we need to treat the boundary conditions by imposing them onto the system by means of some algorithm.

## 2.7 Boundary Conditions

### 2.7.1 Modifying the Equations

One algorithm to impose homogeneous Dirichlet conditions is:

- Identifying boundary nodes by usage of the boundary node array from the `GetDisc`-function
- Zeroing out the row and column in the matrix corresponding to the node
- Setting a 1 on the diagonal element of that row
- Zeroing the element in the load vector corresponding to the node

This embeds the equation $u\big|_{\partial\Omega} = 0$ in the system and ensures that the equations controlling the nodal values at the boundary are decoupled from the interior of the system.

### 2.7.2 The Submatrix Approach

Since these equations are trivial to solve, we may remove them from the system matrix which is given to the solver and reinsert them after the solver has finished. Since the `GetDisc` function constructs the system such that all the boundary nodes are at the end of the list of nodes, they also end up at the end of the vector of nodal values. Thus, we may crop the system matrix by slicing it in python, such that:

For the entire system matrix $A$, the nodal value vector $u$ and the load vector $F$, we extract a submatrix $B$, a subvector $\tilde{u}$ and a load subvector $\tilde{F}$:

- $n$: number of total nodes
- $b$: number of boundary nodes
- $s = n - b$, $s$: number of interior nodes
- $B = A[0:s, 0:s]$, $\tilde{u} = u[0:s]$, $\tilde{F} = F[0:s]$

Solving $B\tilde{u} = \tilde{F}$ means only solving for the interior nodes. The boundary nodes are added back into the system by appending them to the $\tilde{u}$-vector after solving the system. Since we use homogeneous Dirichlet conditions, their values are zero.

Both approaches were used in the project. The submatrix approach was effective for the homogeneous Dirichlet approach, while the modification approach was better suited when handling Neumann conditions since the equations to determine nodal values had to be preserved for the Neumann nodes.

## 2.8 Verification

Finally, it is time to solve our boundary value problem. Having built the stiffness matrix and load vector as described, the system of equations is solved using conjugate gradient descent. For 2000 nodes the final result is shown as contour plot below.

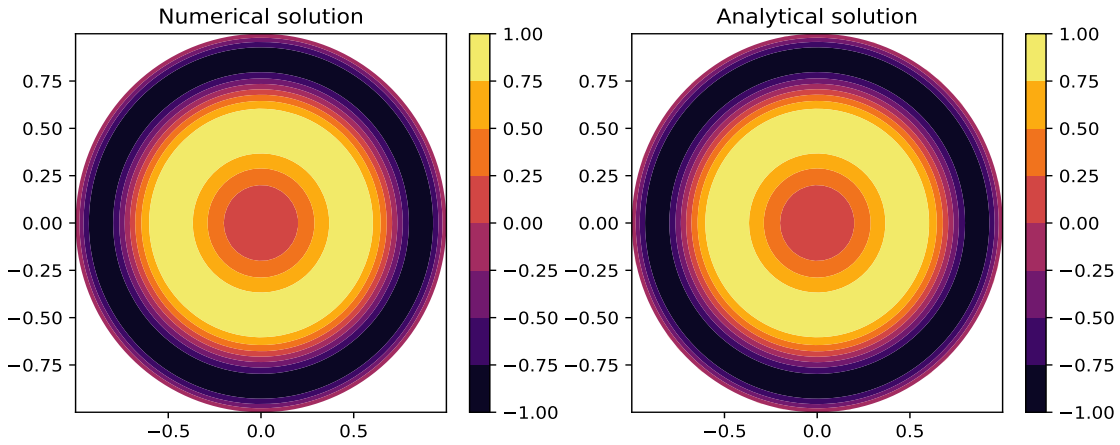


Figure 1: Left: The numerical solution
Right: The analytic solution

The initial impression is that our solution is good, however we must find out exactly how good the solution is. Since we have the analytical solution, we can make use of the residual. The residual is the difference between the

analytical and numerical solution. By storing the residual in vector form, $\mathbf{r}$, we can utilize the stiffness matrix $\mathbb{A}$ to form the so-called energy norm:

$$||\mathbf{r}||_{\mathbb{A}} = \sqrt{\mathbf{r}^T \mathbb{A} \mathbf{r}}$$

Thus, by increasing the number of nodes, we can plot the change in error as the number of elements increase.

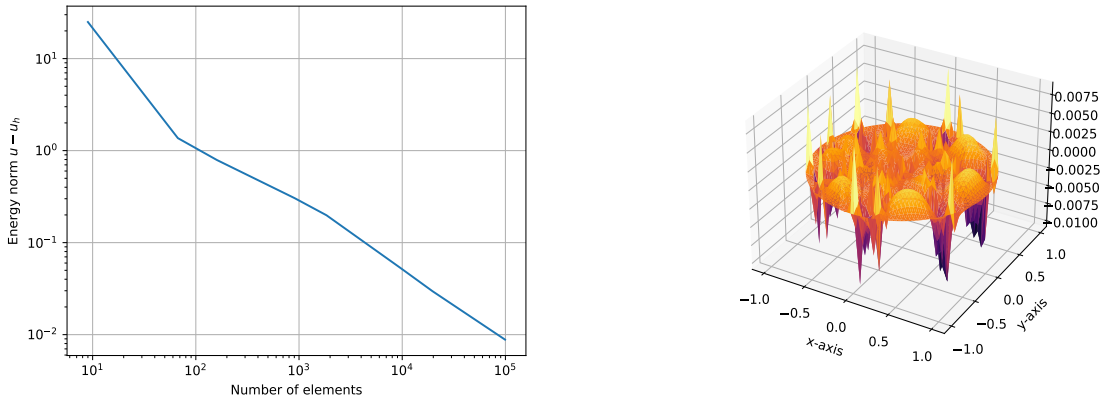


Figure 2: Left: Observed error as number of elements increase
Right: The residual error plot with 2000 nodes

As can be seen in on the left in figure 2, the error decreases roughly linearly with the order of magnitude of elements. Through linear regression the order of convergence was found to be $p \approx 0.96$. On the right figure the residual for 2000 nodes is provided, note that the boundary error is 0. Later we will see that the boundary error need not be zero if we do not have Dirichlet boundary conditions.

# 3 Neumann and homogeneous Dirichlet boundary conditions

We are going to look at the following boundary value problem:

$$\begin{cases} \nabla^2 u(x,y) = -f(x,y) & , \quad (x,y) \in \Omega \\ u(x,y) = 0 & , \quad (x,y) \in \partial\Omega_D \\ \frac{\partial u(x,y)}{\partial n} = g(x,y) & , \quad (x,y) \in \partial\Omega_N \end{cases}$$

Integration with a test function gives the start of the variational formulation as:

$$\nabla^2 u = -f \quad \Rightarrow \quad \int_\Omega v \nabla^2 u \, d\Omega = \int_\Omega -fv \, d\Omega \tag{6}$$

The prescribed flux at the boundary is:

$$g(x,y) = 4\pi\sqrt{x^2+y^2}\cos\left(2\pi(x^2+y^2)\right)$$

## 3.1 Boundary Conditions

The directional derivative $\partial u/\partial n$ can be computed as:

$$\frac{\partial u}{\partial n} = n \cdot \nabla u$$

Using the analytical solution from equation 2, we get that the gradient of $u$ is:

$$\begin{aligned} \nabla u &= \nabla\left(\sin\left(2\pi(x^2+y^2)\right)\right) \\ &= \nabla\left(x^2+y^2\right) \cdot 2\pi\cos\left(2\pi(x^2+y^2)\right) \\ &= 4\pi\cos\left(2\pi(x^2+y^2)\right) \cdot \begin{pmatrix} x \\ y \end{pmatrix} \end{aligned}$$

Since the boundary of the domain is a circle, the normal vector is simply:

$$n = \frac{1}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix}$$

The scalar product between the normal vector and the gradient becomes:

$$\begin{aligned} n \cdot \nabla u &= 4\pi \cos\left(2\pi(x^2 + y^2)\right) \cdot \frac{1}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} x \\ y \end{pmatrix} \\ &= 4\pi \cos\left(2\pi(x^2 + y^2)\right) \cdot \sqrt{x^2 + y^2} \end{aligned}$$

Since this is the same as the function $g$, the analytical solution respects our proposed flux at the boundary.

## 3.2 Variational Formulation

Applying Green's first identity 3 to equation 6 gives us

$$\begin{aligned} -\int_\Omega v \nabla^2 u \, d\Omega &= \int_\Omega \nabla u \cdot \nabla v \, d\Omega - \int_{\partial\Omega} \frac{\partial u}{\partial n} v \, dl = \int_\Omega f v \, d\Omega \\ \Rightarrow \int_\Omega \nabla u \cdot \nabla v \, d\Omega - \int_{\partial\Omega_D} \frac{\partial u}{\partial n} v \, dl - \int_{\partial\Omega_N} \frac{\partial u}{\partial n} v \, dl &= \int_\Omega f v \, d\Omega \end{aligned}$$

We require that $v|_{\partial\Omega_D} = 0, \forall v \in X$ and insert the Neumann condition $\frac{\partial u}{\partial n}|_{\partial\Omega_N} = g$:

$$\int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega + \int_{\partial\Omega_N} g v \, dl.$$

From this we obtain the bilinear and linear forms, $a(u, v)$ and $l(v)$:

$$\begin{aligned} a(u, v) &= \iint_\Omega \nabla u \cdot \nabla v \, d\Omega \\ l(v) &= \iint_\Omega f v \, d\Omega + \int_{\partial\Omega_N} g v \, dl \end{aligned}$$

The difference between this form and the form in problem 2 is the inclusion of the boundary flux term on the linear form. The update to the algorithms should simply be to add a line integration over the respective boundary edges to the load vector, while the construction of the stiffness matrix is largely unchanged.

## 3.3 Line integration

Since our Neumann boundary requires a line integral in the linear form we must implement this numerically. Fortunately, this is somewhat simple with our triangular elements. By parameterizing the straight line between two points on the boundary such that any point between them is given by a value $t \in [-1, 1]$, we can utilize one-dimensional Gaussian quadrature. So for $P_1, P_2 \in \partial\Omega$ we can write

$$P(t) = \frac{1 - t}{2} P_1 + \frac{1 + t}{2} P_2.$$

Thus, we can model 2D coordinates with the 1D reference interval used in the quadrature, allowing us to approximate a line integral along the boundary. By modifying our original one-dimensional integrator such that it handles line integrals we tested it over the function $e^x$ from $(1, 0)$ to $(2, 0)$, which yielded $e^2 - e$ accurate to seven digits.

## 3.4 Verification

By modifying the previous code such that it keeps track of which nodes are on the boundary and thus adds their line integral to the linear form, we have managed to solve the boundary value problem once again. In this case it is also interesting to have a look at the numerical load given by $l(v)$.

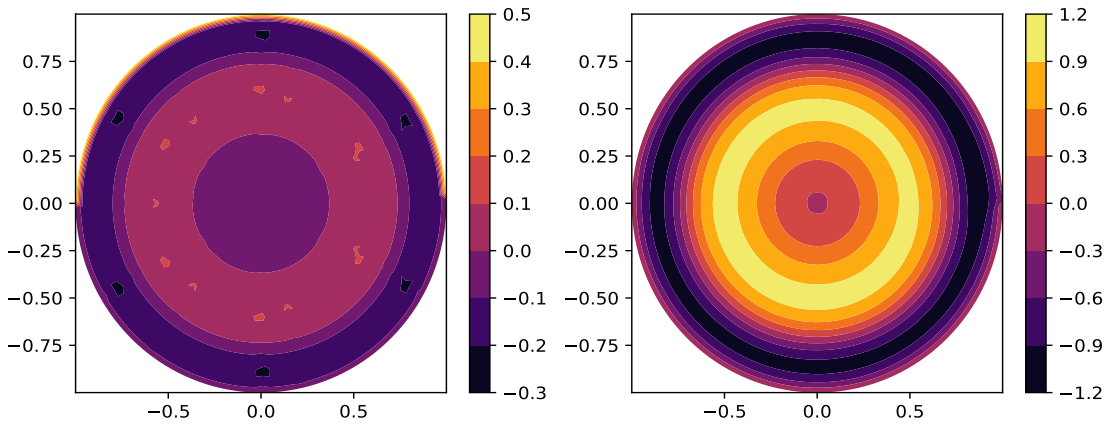

Figure 3: Left: The numerical load
Right: The numerical solution

From figure 3 we clearly see the difference in boundary conditions on the numerical load when we change from $y < 0$ to $y > 0$, which indicates that things have worked out correctly. The numerical solutions looks good at first, but if we take a closer look around the point $(1, 0)$, we notice a little "kink" in the solution. This turns out to be a rather large error in the solution as we transition between the boundary conditions. Error analysis shows just how large this "kink" is.

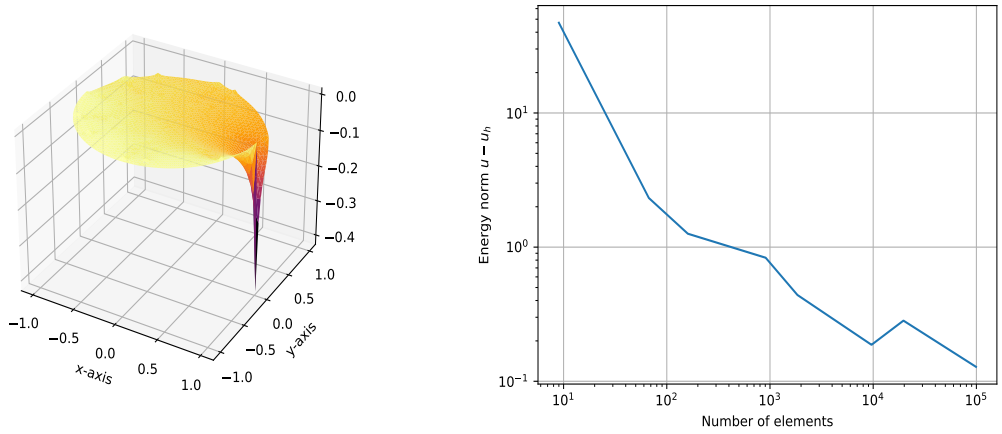


Figure 4: Left: The mixed boundary residual for 2000 nodes
Right: Change in error as number of elements increase

Our model struggles with the transition between boundary conditions. This should not be a surprise as the grid we use is quite naïve.

One way to mitigate the error around the transition is to increase the number of nodes around the point. In addition, the nodes might have been placed more appropriately. Furthermore, the interior of our residual seems to have changed, in the Dirichlet case there is clearly some structured behaviour in our residual, but this does not seem to be the case for mixed boundary conditions as we observe a much smoother interior. A direct comparison of residual contours show this change:

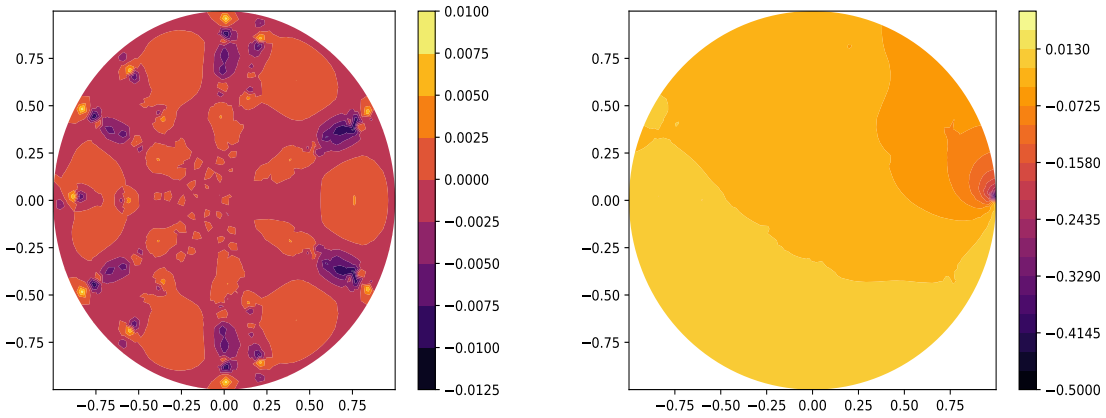


Figure 5: Left: The Dirichlet contour
Right: The mixed case contour

In total, these problems reverberate in the convergence analysis as the norm of our residual does not decrease significantly. The final result of this is an order of convergence of $p \approx 0.86$.

To summarize the comparison of solutions, both Dirichlet conditions and Neumann conditions seem to have solved the system well for the interior. The error on the interior in the Dirichlet case is generally low, but somewhat jagged. For the Neumann case, the interior error is both low and smooth. For the boundary, however, the error in the Dirichlet case is zero, while in the Neumann case, there is a serious error in the point $(1, 0)$. The solution using the Neumann boundary conditions seem to be preferable, especially if the serious point error can be rectified.