Finite Elements Method of Poisson's Equation in a 2D Rectangular Domain

Aidan Olson, Max Rubin, and Griffin Hunt

December 14th, 2023

Background

General Overview

The Poisson equation, named after the French mathematician Siméon Denis Poisson, is a second-order partial differential equation (PDE) that is quite popular and commonly used in physics and engineering, as it describes the distribution of a scalar field in the presence of a source term.

Analytical Decomposition of Equation Components

The Poisson equation, given in three-dimensional Cartesian coordinates (x, y, z), is:

$$\nabla^2 u(x, y) = f \tag{1}$$

The u is the solution, f is the source term, and ∇^2 (often written as Δ) is the Laplacian operator and can be expressed as the sum of second spatial derivatives: $\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_n^2}$.
Poisson's equation is closely related to Laplace's equation, $\Delta u = 0$. Laplace's equation, however, has no source term. The source term makes the Poisson equation applicable to situations where there are external influences affecting the scalar field. This is why the Poisson is often referred to as the general Laplacian equation. The Laplacian is also a second-order PDE that commonly appears in physics and engineering. It was named after Pierre-Simon Laplace. The Laplacian describes a scalar function u that has no sources or sinks within the region of interest. It simply states that the sum of an expression's second partial derivatives is 0. As we'll see soon, the solutions to Laplace's equation are harmonic functions.

Applications

These equations are used in multiple fields. Firstly, the they are used in electrostatics. The Poisson equation is used to model the distribution of electric charge or magnetic charge when there are charges present, and even allows for the derivation of the electric potential in a given charge distribution in a given region of space. The Laplacian, on the other hand, can only describe the electric potential in regions where there are no charges. The equations are used to solve steady-state heat conduction problems. The Laplacian can model the distribution of temperature in a conductive medium when temperature is not changing with time as well as when there are no heat-generating

elements within the material. The Poisson is used when there are localized heat sources. Lastly, the equations are used in fluid dynamics to solve potential flow and used in image processing for edge detection.

1 Solving the Problem

1.1 Objective

In this section, we will solve the following 2D Laplacian equation:

$$\nabla^2 u = 0 \tag{2}$$

With the following boundary conditions: $u(x,1) = u(x,0) = u(0,y) = 0,\ \ u(1,y) = g_R(y) = 4y(1-y),\ \ x \in (0,1),\ \ \ y \in (0,1)$.

Recall, this is a version of Poisson's equation whose solution has a known expression. Notice, the boundary conditions are not Dirichlet. Dirichlet boundary conditions imply that $u(x,y) = 0$ on the boundary of the domain. Solving the Laplacian with Dirichlet boundary conditions simply returns a null solution, so we want to avoid that. We will first solve it using separation of variables and then using the finite elements method (FEM). After acquiring these solutions, we will compare the solution from FEM with the solution from separation of variables using an error convergence test of a fixed point. Finally, we will test the limits of our FEM and showcase its possibilities for the 2D Poisson of any source term, square domain, and boundary conditions.

1.2 Separation of Variable

To get an expression for the solution to Laplace's equation inside the unit square domain with the given boundary conditions, we use the method of separation of variables. Assume a solution of the form $u(x,y) = X(x)Y(y)$. Substituting this into Laplace's equation, dividing both sides by $X(x)Y(y)$, equating both quotients to λ^2, and rearranging a bit gives us two second order linear ODEs:

$$\begin{cases} X''(x) - \lambda^2 X(x) &= 0 \\ Y''(y) + \lambda^2 Y(y) &= 0 \end{cases} \tag{3}$$

The solutions to these ODEs are:

$$\begin{cases} X(x) &= A\cosh(\lambda x) + B\sinh(\lambda x) \\ Y(y) &= C\cos(\lambda y) + D\sin(\lambda y) \end{cases} \tag{4}$$

Applying the boundary conditions $u(x,0) = Y(0) = 0$ and $u(x,1) = Y(1) = 0$, we find that $C = 0$ and $\sin(\lambda) = 0$. Therefore, $\lambda = n\pi$ for $n = 1, 2, 3, \ldots, N$. The solution for $Y_n(y)$ becomes:

$$Y_n(y) = D_n \sin(n\pi y) \tag{5}$$

For $X(x)$, we apply the boundary condition $u(0,y) = X(0) = 0$ to deduce that $A = 0$. This is the format of our solution:

$$u(x,y) = \sum_{n=1}^{N} B_n \sinh(n\pi x)\sin(n\pi y) \tag{6}$$

We can also enforce the boundary condition $u(1, y) = X(1)Y(y) = g_R(y)$ to determine B_n:

$$g_R = \sum_{n=1}^{N} B_n \sinh(n\pi) \sin(n\pi y) \tag{7}$$

This is a Fourier sine series. The coefficients B_n are given by:

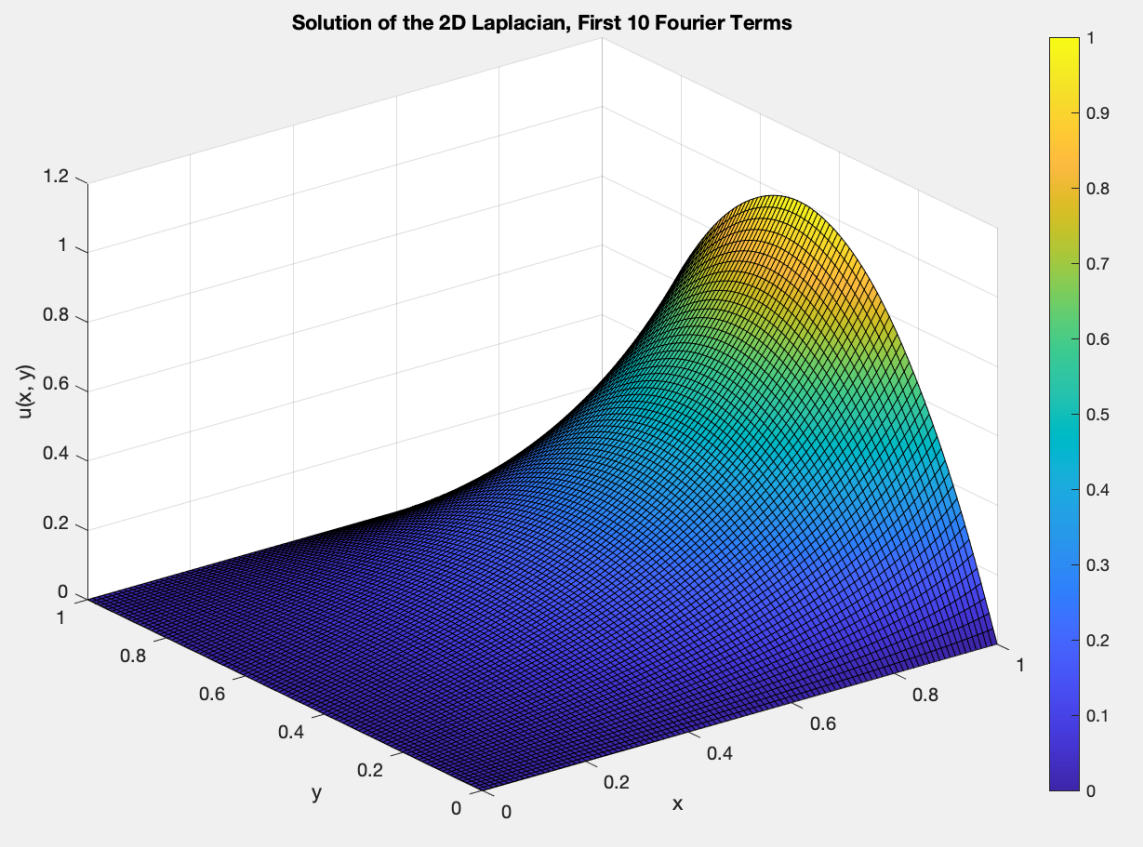
$$B_n = \frac{2}{\sinh(n\pi)} \int_0^1 g_R(y) \sin(n\pi y)\, dy \tag{8}$$

This can be computed for all n from 1 to N as $N \to \infty$ to increase accuracy. This infinite series represents the solution to Laplace's equation in the unit square with an unknown right boundary condition. Let $g_R(y)$ be defined as outlined in our problem:

$$g_R(y) = 4y(1 - y) \tag{9}$$

We can solve for the first N terms in the summation for u. Since we know the coefficients B_n approach 0 as $n \to \infty$, the first 10 terms will suffice. Using some MATLAB code, we can graph these solutions on our unit square domain, as given by the problem.

Figure 1: Graph of $u(x, y)$ using Separation of Variables



To summarize our current standing, we have a viable, known solution to the Poisson equation with specific parameters. Now we can take this homogeneous solution on the unit square with Dirichlet and quadratic boundary conditions and try to mimic it using the Finite Elements Method.

2 Finite Elements Methods

The finite elements method is a technique invented in the 1950's used to solve PDEs in engineering and physics. The overall idea behind the finite elements method is to divide a complex domain into much smaller and simpler subdomains. This creates a mesh with individual components called elements. Using these elements, it is possible to then approximate the behavior within each element using simple functions. In order to do this, we must deduce a weak form of our PDE, express this in terms of our individual elements, form a global stiffness matrix and load vector, and solve our system for basis function coefficients.

2.1 Theory and Process

To solve a PDE using the finite elements method, one follows certain steps. Many of these steps require wise decision making foresight to best fit the equation at hand.

2.1.1 Define and Analyze the Problem

Specify the problem, including the domain, the PDE itself, and the boundary conditions. In our case, we want to apply FEM to Poisson's equation, so we allow room for f to be defined, and the boundaries and domain constraints to change within our code. For our testing purposes, we will use the same homogeneous Laplacian defined above. Here is a reminder:

$$\begin{aligned} \Delta u(x,y) &= f(x,y) = 0 \\ u(0,y) &= u(x,0) = u(x,1) = 0 \\ u(1,y) &= g_R(y) = 4y(1-y) \\ x &\in (0,1), \quad y \in (0,1) \end{aligned}$$

These parameters imply that we are solving on a unit square domain with partially Dirichlet boundary conditions just like before in our separation of variables section. For the sake of flexibility in later applications, we will treat our function like we don't know f, Ω, or $\partial\Omega$. That way, we can change these parameters to develop solutions to various types of Poisson equations.

2.1.2 Create the Mesh and Choose the Element Type and Basis Functions

Divide the domain into smaller, non-overlapping elements. In 2D, these elements are typically triangles. The mesh consists of nodes and elements. Basis functions are used to approximate the solution within each element. We will use linear basis functions and triangular elements.

2.1.3 Formulate the Weak Form

The weak form of a PDE transforms the PDE into a more digestible expression with lower degree differentials. In the case of Poisson's equation, we go from a second degree PDE to a first degree PDE which is easier to work with in numerical models like FEM. Additionally, we can break down the weak form to an elemental scale, and take advantage of the divergence of the nodal gradients (seen in the next step). To acquire this form, we multiply the PDE by a test function v (essentially the same as a basis function) and integrate over the domain.

$$\begin{aligned} \int_\Omega \Delta u \cdot v \, d\Omega &= \int_\Omega f \cdot v \, d\Omega \\ \int_\Omega \nabla u \cdot \nabla v \, d\Omega &= \int_\Omega f \cdot v \, d\Omega \quad \text{(LHS Using Green's theorem)} \end{aligned}$$

2.1.4 Compute Local Stiffness Matrices and Local Load Vectors

We convert the weak form to a elemental basis, and compute these local stiffness matrices and load vectors for each element. This step involves breaking down the weak solution from our large domain, Ω, into its triangular elements. We then integrate the basis functions and their gradients (which come from the weak solution) over each element. These are then used to compute their local

stiffness matrix and load vector. The element stiffness matrices, K_e, are derived from the LHS of the weak form:

$$K_{ij}^e = \int_{\Omega^e} \nabla N_i \cdot \nabla N_j \, dA$$

Each elemental stiffness matrix, K^e, will look like this:

$$K^e = \begin{pmatrix} K_{1,1}^e & K_{1,2}^e & K_{1,3}^e \\ K_{2,1}^e & K_{2,2}^e & K_{2,3}^e \\ K_{3,1}^e & K_{3,2}^e & K_{3,3}^e \end{pmatrix} \tag{10}$$

Where N_i and N_j are the basis functions associated with the nodes of the element. In total, we have 9 K_{ij} per element since our triangles have 3 nodes and thus, 3 i and 3 j per element gives us a 3×3 matrix. The element load vectors, F_e, come from the RHS of the weak form:

$$F_i^e = \int_{\Omega^e} f \cdot N_i \, dA$$

Each elemental load vector, F^e, will look like this:

$$F^e = \begin{pmatrix} F_{1,1}^e \\ F_{2,1}^e \\ F_{3,1}^e \end{pmatrix} \tag{11}$$

In total, we have 3 F_i per element since our triangles have 3 nodes. For each element, calculate the gradients of the associated shape functions and integrate each K_{ij} and F_i over the area, A, defined by the element in question. This is done with numerical integration. Since we have linear basis functions and elements, this can be achieved with a simple trapezoidal rule.

2.1.5 Global Stiffness Matrix and Load Vector Assembly

Using the local matrices and vectors for each element, we combine these into a global stiffness matrix and load vectors that will be used to solve for our constants. If there are N nodes in the mesh, the global stiffness matrix, K, is size $N \times N$. The K_{ij} entry represents the interaction between the ith and jth nodes. Often, these entries are 0 which makes it easier to solve the system at the end.

The interaction between the local matrices, K_e, and the global stiffness matrix, K, can be seen as follows: K_e describes the interaction between 3 specific nodes while K explains the interaction between all nodes, including all possible combinations of K_e and more (nodes that are too far from each other). For instance, if an element connects global nodes 2, 5, and 7, the entries of its local stiffness matrix K_e will contribute to the $(2,2)$, $(2,5)$, $(2,7)$, $(5,2)$, $(5,5)$, $(5,7)$, $(7,2)$, $(7,5)$, and $(7,7)$ positions in the global matrix K. Since there is overlap between elemental nodes, some entries are the sums of multiple K^e. Notice that places like the $(1,42)$ position (depending on your mesh generation) will be 0 because those nodes do not interact. Here is what we have so far.

$$K = \begin{bmatrix} K_{11} & K_{12} & 0 & \dots & 0 \\ K_{21} & K_{22} & K_{23} & \ddots & \vdots \\ 0 & K_{32} & K_{33} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & K_{N-1N} \\ 0 & \dots & 0 & K_{NN-1} & K_{NN} \end{bmatrix} \tag{12}$$

2.1.6 Apply Boundary Conditions

The application of our boundary conditions can be seen in the modification of edge positions in the global stiffness matrix and load vector. This involves setting certain values in the solution vector or altering rows and columns in the stiffness matrix according to your boundary conditions. When we apply the boundary conditions, we can eliminate some unknown constants for the boundary nodes and simplify our system a bit more to something like this:

$$K_{BC} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & K_{22} & K_{23} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & K_{N-1N-2} & K_{N-1N-1} & 0 \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

2.1.7 Solve the System of Equations

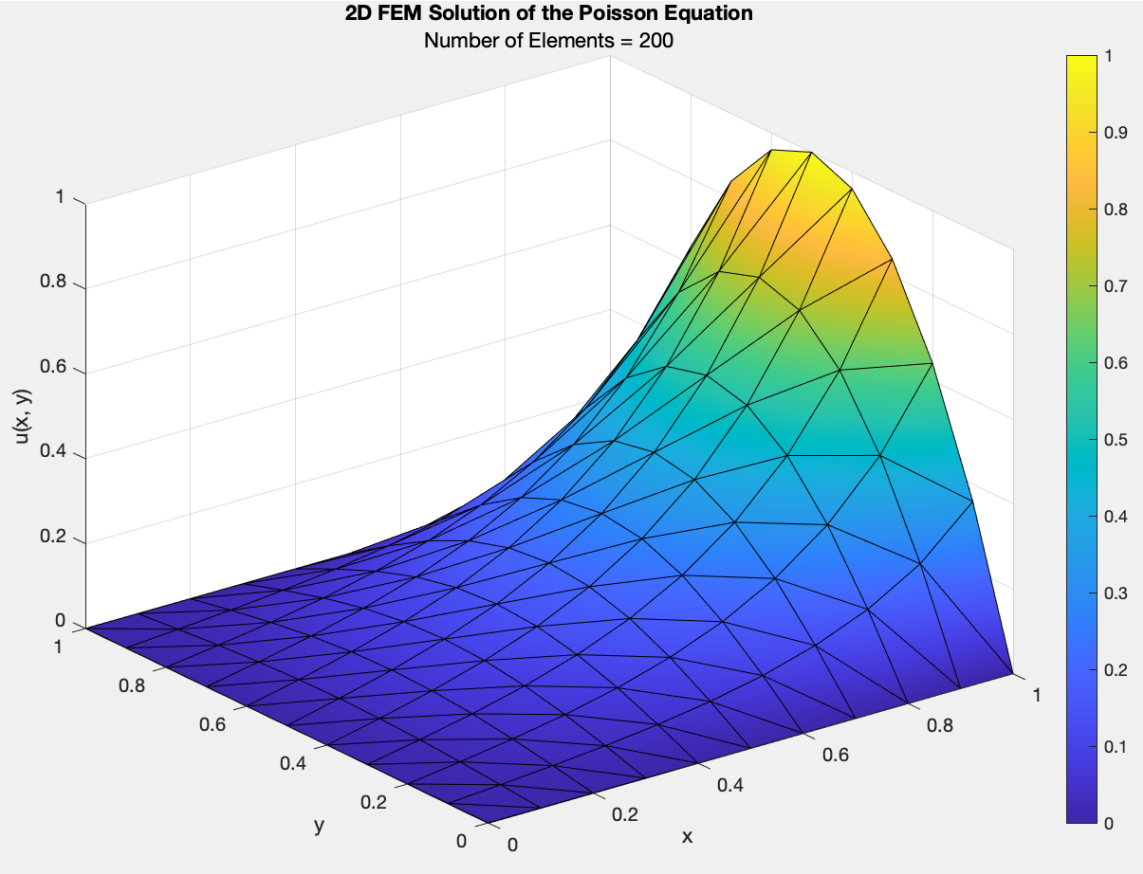
Now we have a linear system with a global stiffness matrix, K, a global load vector, F, and a vector of unknown constants, U:

$$K\vec{U} = \vec{F} \tag{14}$$

Since K and F are populated with stress coefficients and data from our source function, we can use the backslash in MATLAB to solve the linear system and find coefficients associated to each nodal. These constants stretch the basis functions to the appropriate location and help create a smooth surface. Our code intakes a source function, domain, 4 boundary conditions, and the dimensions of our x and y subdivisions. We can finally visualize $u(x, y)$ in the x, y, z space.

2.2 FEM Solution

Figure 2: Graph of $u(x, y)$ for 100 Elements[2]



Notice, this solution mimics that of our separation of variables graph only with larger elements. If we increase the number of elements, it should increase in accuracy and become nearly indistinguishable to the actual solution.

Now that we have a working solution to the 2D homogeneous Laplacian on a square domain with 3 Dirichlet boundaries and a quadratic right boundary, we continue with an analysis.

3 FEM Analysis and Extrapolation

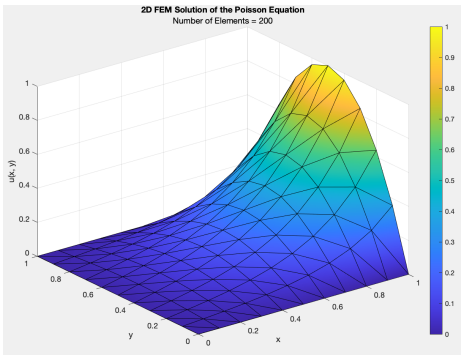
Having developed a working FEM process, we can analyze how efficiently it works. In this section, we intend to conduct an elementary accuracy/error convergence analysis and expand upon the potentials of FEM.

3.1 Accuracy Analysis

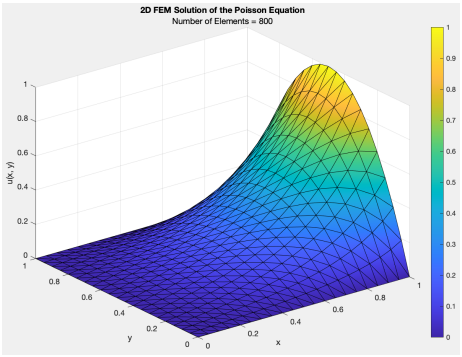
To do this, we can compare the solution we get with FEM to that from the separation of variables by including the calculation of a random point in both methods. This step is continued in the next section. By selecting the same point, in this case $u_{test} = u(4.12, 5.63)$, we can analyze how the increase in elements impacts the accuracy or percent error of the solution.

The following 4 plots show the solution to the familiar 2D unit square semi Dirichlet Laplacian equation as they vary in elements. Since we double the subdivision of our unit distances in each dimension, the amount of elements increases by a factor of 4. The underlying table shows the value of u_{test} extrapolated from the solutions of increasing granularity.

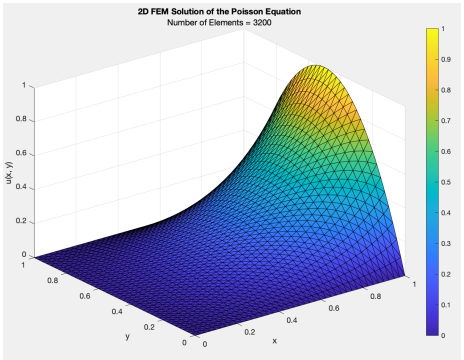
Figure 3: FEM solutions to the homogeneous Laplacian increasing in element granularity



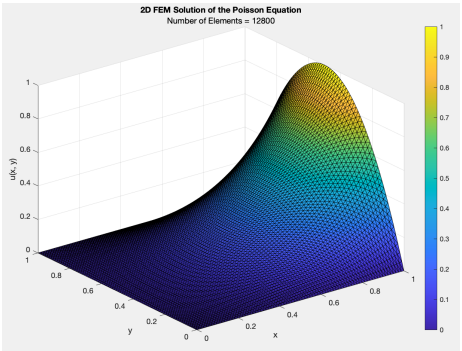
(a) 200 Elements^2



(b) 800 Elements^2



(c) 3200 Elements^2



(d) 12800 Elements^2

Table 1: Example of a 6x5 Table with Various Data Types

Graph	Num Elems	Actual Laplace	FEM Laplace	% Error	Error Factor
3 a	200	0.14771	0.14898	0.8600	N/A
3 b	800	0.14771	0.14826	0.3724	2.3
3 c	3200	0.14771	0.14789	0.1219	3.1
3 d	12800	0.14771	0.14775	0.027	4.5

Num Elems = The number of square elements for each FEM approximation
Actual Laplace = The value of u_{test} on the actual solution graph obtained from separation of variables
FEM Laplace = The value of the same test point on our approximated graph of FEM.
% Error = $\frac{(\text{FEM Laplace} - \text{Actual Laplace})}{\text{Actual Laplace}} \times 100\%$
Error Factor = The error compared to that of the last graph was x times smaller.

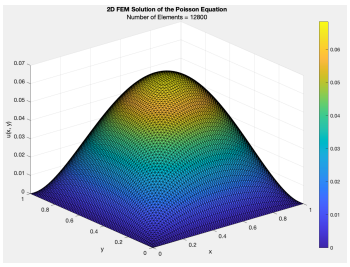
This provides us with a few takeaways. Using FEM with just 200 elements gives us a % error of only 0.86! For so few elements in this model, this is pretty accurate. Then, when we quadruple the elements to 800, the accuracy doubles. This is not remarkable since it only reflects $O(h^{\frac{1}{2}})$ accuracy convergence which is to say that a quadrupling of granularity only impacts accuracy two-fold. What is peculiar is that the next quadrupling of our elemental basis triples the accuracy. Furthermore, the last quadrupling as seen in graph 3 (d), more than fourths the error from graph 3 (c) which reflects $O(h)$ convergence. These steps are demonstrating an increase in convergence as we get more granular.

Through this step, we have verified that our application of FEM to the 2D Laplacian on a square domain is accurate. Now, we can this extends to all Poisson equations and use our model to draft some more complicated solutions.

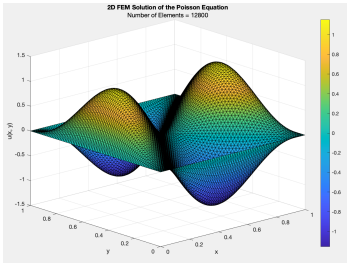
3.2 Experimenting with the 2D Non-homogeneous Poisson FEM

See the next page to observe a few example outputs of our FEM 2D Poisson code for the following domains (Ω), boundary conditions ($\partial\Omega$), and source functions (f).

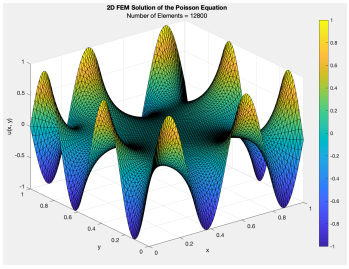
Figure 4: Three example FEM solutions with various parameters to the Poisson equation



(a) Gaussian



(b) Saddle



(c) Sine BC's

$$f_a = e^{-((x-\frac{1}{2})^2+(y-\frac{1}{2})^2)} \qquad f_b = 800 * ((x-\frac{1}{2})^2 + (y-\frac{1}{2})^2) \qquad f_c = 0$$

$$\Omega_a = (0,1) \text{ x } (0,1) \qquad \Omega_b = (0,1) \text{ x } (0,1) \qquad \Omega_c = (0,1) \text{ x } (0,1)$$

$$\partial\Omega_a = Dirichlet \qquad \partial\Omega_b = Dirichlet \qquad \partial\Omega_c = Sinusoidal$$

Look at how easily we can compute the solution to any 2D Poisson problem on a square domain. We can effortlessly model solutions to the Poisson with Gaussian and saddle like source functions, strange sinusoidal boundary conditions, and any domain we want. Domain variation was not experimented because it is dependent on the source function and boundary conditions. There's no need to solve with separation of variable if this solution is so fast and reliable.

Discussion and Further Research

Using FEM it is possible to both solve the homogeneous Poisson equation and then inhomogenous equation with a source term. Moreover, as the number of square elements quadruples, the accuracy doubles, then triples, then quadruples, showing that FEM converges at an increasing rate. It's possible that the point selected gets closer and closer to a newly generated node as the elements keep quadrupling in amount. If this is the case then the accuracy ratios should also increase at all points which would indicate that the whole solution increases in accuracy in a similar pattern. This hypothesis is certainly worth exploring in future research.

Beyond investigating the hypothesis surrounding our error convergence, experimenting with other accuracy metrics may be beneficial. We only examined one point, on 4 different mesh generations, providing 3 ratios of convergence. While it is trivial that FEM converges to the real solution as we further divide the mesh, it's worth examining how much and to what degree it improves. We know it's somewhere in the $O(h^{\frac{1}{2}})$ to $O(h)$ range.

Lastly, we could investigate the validity of FEM with another set of parameters (Ω, $\partial\Omega$, and f). Perhaps comparing a solution from FEM to that of another known problem will uncover hidden dynamics that did not appear in our selected unit square Laplacian with one boundary condition.

In conclusion, we have established a working method for approximating the solution to Poisson's equation on a 2D, bounded, unit square domain.

References

1. Binegar, B. (n.d.). Finite Element Method. Oklahoma State University. Retrieved from https://math.okstate.edu/people/binegar/4263/4263-l21.pdf

2. Chen, L. (n.d.). Programming of Finite Element Methods in MATLAB. Retrieved from https://lyc102.github.io/ifem/fem/Poisson/

3. Burkardt, J. (n.d.). FEM2D_POISSON_RECTANGLE, a MATLAB program which solves the 2D Poisson equation on a rectangle, using the finite element method. Retrieved from https://people.math.sc.edu/Burkardt/m_src/fem2d_poisson_rectangle/fem2d_poisson_rectangle.html

4. Roth, J. (2020). The Finite Element Method - Numerical Analysis. YouTube. Retrieved from https://www.youtube.com/watch?v=P4lBRuY7pC4