Solving Boundary Value Problems Using Integral Equations

Zane Perry

7 May 2024

Abstract

Numerical methods for solving boundary value problems often rely on discretization techniques, introducing errors and potentially leading to ill-conditioned systems. However, an alternative approach involves using integral equations, particularly advantageous for linear, constant coefficient differential equations. By leveraging convolutions of the related Green's functions along the prescribed boundary the problem dimension is reduced and errors are localized, contingent upon the stability of the chosen solver method. This method allows for approximations of complex geometries and boundary conditions that are infeasible with traditional mesh grids. The derivation of Green's functions and fundamental solutions is explored, along with considerations for nonhomogeneous boundary conditions. Numerical evaluation involves quadrature methods and dense linear solvers. Extension to two-dimensional problems involves the use of fundamental solutions and boundary parameterization techniques. Numerical results demonstrate the high accuracy and efficiency of integral equation methods compared to traditional discretization techniques, particularly evident in irregular domains.

1 Introduction

Most methods within the numerical analysis of differential equation requires the discretization of space and/or time in order to create a mesh of approximation nodes that rely on each other in a chain or linear system. The major disadvantage to this strategy is that each step of discretization and derivative approximation creates more sources of error within the system and can lead to ill-conditioning and a relatively high magnitude of error depending on the stability of the method being used. By contrast, many linear, constant coefficient differential equations can be inverted by using the fundamental solutions or Green's functions related to the differential operator and boundary conditions of the specific problem. This in turn reduces the dimension of the problem being solved as well as reducing global error to localized errors dependent only on the stability of quadrature or iterative solver method being used. Additionally, because the differential equation is reduced to a single integral equation on the boundary, this allows for approximations of complicated geometries and boundary conditions that would not be possible using a simple mesh grid. While more complex in their formulation, and limited to the existence of the desired integral kernel, boundary value

integral equations lead to a high level of accuracy on complicated domains for many types of differential equations.

This report will begin with a derivation and understanding of Green's functions in a single dimension to create a foundation for the techniques being used. This will then be applied to the class of ODEs known as Sturm-Liouville problems to formulate a specific algorithm for approximating solutions to these problems. Various quadrature techniques will be explored with different examples to demonstrate the effectiveness of the method. The idea of Green's functions and their related integral equations will then be expanded to two-dimensional domains along with the challenges that result from adding another dimension to the problems, as well as the similarities between cases. This will be applied to the Laplace equation as an example to demonstrate how this technique can be useful on a variety of different boundary types using different quadrature methods.

2 Mathematical Formulation of 1D Problems

2.1 A Motivating Example

Consider the following Boundary Value Problem with homogeneous boundary conditions:

$$\begin{aligned} -\frac{d^2u}{dx^2} &= f(x) \\ u(0) = u(1) &= 0 \end{aligned}$$

We can solve for $u(x)$ directly by integrating twice (using "dummy" variables for integration purposes):

$$\begin{aligned} -\frac{du}{dx} &= \int_0^x f(y)dy + C_1 \\ -u(x) &= \int_0^x \int_0^y f(s)ds + C_1x + C_2 \end{aligned}$$

Note the following theorem for double integrals [7]:

$$\int_a^x \int_a^s f(y)dy = \int_a^x f(y)(x-y)dy$$

This, along with the prescribed boundary conditions, can help further reduce the expression and solve for the unknown constants:

$$
\begin{aligned}
-u(x) &= \int_0^x (x-s)f(s)ds + C_1 x + C_2 \\
&\quad \text{(Left Boundary Condition)} \\
-u(0) &= 0 \\
&= C_2 \\
&\quad \text{(Right Boundary Condition)} \\
-u(1) &= 0 \\
&= \int_0^1 (1-s)f(s)ds + C_1 \\
C_1 &= \int_0^1 (s-1)f(s)ds
\end{aligned}
$$

Inserting these constants into the solution and splitting the bounds of integration leads to

$$
\begin{aligned}
-u(x) &= \int_0^x (x-s)f(s)ds + x\int_0^1 (s-1)f(s)ds \\
u(x) &= \int_0^x (s-x)f(s)ds + \int_0^x x(1-s)f(s)ds + \int_x^1 x(1-s)f(s)ds \\
&= \int_0^x s(1-x)f(s)ds + \int_x^1 x(1-s)f(s)ds
\end{aligned}
$$

This solution is frequently combined into something called the Green's function[7], as follows:

$$
\begin{aligned}
u(x) &= \int_0^1 G(x,s)f(s)ds \\
G(x,s) &= \begin{cases} x(1-s) & x < s \\ s(1-x) & x > s \end{cases}
\end{aligned}
$$

In order to evaluate this solution numerically, it is split back into two integral form above and any desired quadrature method can be used to obtain a value for the desired x. It's important to note that this must be done for every desired value of x. Because this form of the solution is analytically exact, the accuracy and stability of the resulting approximation relies only on that of the quadrature method being used [4]. This is an important result because numerical methods involving integrals and quadratures tend to be more stable than those involved in the approximations of derivatives, and the specific quadrature chosen can be tailored to account for the specific condition of the desired problem.

2.2 Deriving Green's Functions

The formulation of Green's functions is motivated by the idea of inverting differential operators. Any linear differential equation can be written in the form $L[u] = f$ where L is a linear differential operator. In the example from before, the operator is given by $L[u] \equiv -\frac{d^2u}{dx^2}$. The goal is then to find a way to invert this operator; because the operator involves derivatives, the inverse will involve some form of integration. In fact, the inverse will be a convolution of the forcing term f with the Green's function associated with the given operator and boundary conditions [5].

$$u(x) = L^{-1}[f] = \int_a^b G(x,s)f(s)ds$$

The Green's function is also referred to as the kernel of the integral operator, and can be solved for in many different ways, including through variation of parameters (only for linear ordinary differential equations), eigenfunction expansions, and the Dirac delta distribution.

2.2.1 Understanding Green's Functions

In a physical sense, a Green's function is the response of a system to a single point charge. This can be mathematically represented by the solution to $L[u] = \delta(x - s)$, where $\delta(x - s)$ represents a single point charge that is translated to $x = s$, also noting that the operator L conforms to the prescribed homogeneous boundary conditions. In this way, the Green's function is determined by the operator itself as well as the given boundary conditions. Additionally, the Green's function only exists when the operator has an inverse, or equivalently if the only solution to $L[u] = 0$ is the trivial solution[7]. Using this definition, the Green's Function of any specific ODE with homogeneous boundary conditions can be derived by solving this equation, and is specific to that particular differential operator and boundary condition.

Green's functions for one-dimensional ODEs also satisfy the following properties[7]:

- Are symmetric in their arguments $G(x, s) = G(s, x)$
- Are continuous at $x = s$
- Have a jump discontinuity in the derivative at $x = s$

Essentially the Green's function associated with a boundary value problem is the fundamental behavior of that system at an isolated point. By convolving it with the forcing term of the differential equation, the effect of the forcing term on the system is incorporated into the solution everywhere within the desired interval. [5]

2.2.2 Application to Sturm-Liouville problems

Consider the general Sturm-Liouville boundary value problem with homogeneous Dirichlet boundary conditions

$$\frac{d}{dx}\left(p(x)\frac{du(x)}{dx}\right) + q(x)u(x) = f(x) \quad a < x < b$$

with $p(x) > 0$. The solution is found by superimposing the homogeneous solution with a particular solution $u(x) = u_h(x) + u_p(x)$. Because this is a second order differential equation, there will be 2 linearly independent solutions to the homogeneous form, $u_1(x)$ and $u_2(x)$ [3]. Assuming a particular solution of the form

$$u_p(x) = c_1(x)u_1(x) + c_2(x)u_2(x)$$

and using the method of variation of parameters to find a system of equations for the coefficient functions

$$\begin{aligned} c_1'(x)u_1(x) + c_2'(x)u_2(x) &= 0 \\ c_1'(x)u_1'(x) + c_2'(x)u_2'(x) &= \frac{f(x)}{p(x)} \end{aligned}$$

with solution

$$\begin{aligned} c_1'(x) &= -\frac{fu_2}{p(x)W(u_1,u_2)} \\ c_2'(x) &= -\frac{fu_1}{p(x)W(u_1,u_2)} \end{aligned}$$

where $W(u_1, u_2) = u_1u_2' - u_1'u_2$ is the Wronskian. An important and interesting property of Sturm-Liouville ODEs is that the Wronskian of a basis of the solution space to the homogeneous problem will never be 0 on the defined interval [3]. By construction $u_1(x)$ and $u_2(x)$ are linearly independent and the solution space of a second order ODE has a dimension of 2, so therefore they form a full basis of the solution space and the Wronskian will never be 0. Additionally, the product pW will always be a constant (proven in Appendix A). Integrating these gives the full solution

$$u(x) = u_2(x)\int_{x_1}^{x}\frac{f(s)u_1(s)}{pW}ds - u_2(x)\int_{x_0}^{x}\frac{f(s)u_2(s)}{pW}ds$$

If we choose x_0, x_1, u_1, and u_2 to account for the boundary conditions at a and b, the solution can be written in a compact form given by

$$\begin{aligned} u(x) &= \int_a^b G(x,s)f(s)ds \\ G(x,s) &= \begin{cases} \frac{u_1(s)u_2(x)}{pW}, & a \le x \le s \\ \frac{u_1(x)u_2(s)}{pW}, & s \le x \le b \end{cases} \end{aligned}$$

where $u_1(x)$ and $u_2(x)$ are solutions of the homogeneous problem satisfying $u_1(a) = 0, u_2(b) = 0$ and $u_1(b) \neq 0, u_2(a) \neq 0$. As before, when calculating a solution numerically, any quadrature method can be used for each desired value of x, with the integral split at x for both cases of the Green's function:

$$u(x) = \int_a^x \frac{u_1(s)u_2(x)}{pW} f(s)ds + \int_x^b \frac{u_1(x)u_2(s)}{pW} f(s)ds$$

Any 2nd order linear differential operator can be written in the form of a Sturm-Liouville problem (proven in Appendix A), making this a powerful method to approximate one-dimensional boundary value problems.

2.3 Nonhomogeneous Boundary Conditions

Like most other solutions to differential equations, the solution to any linear differential operator $L[u]$ with nonhomogeneous boundary conditions can be found by superimposing two separate solutions: one that satisfies the problem with homogeneous boundary conditions and the forcing term, and a particular solution that satisfies the actual boundary conditions with no forcing term:

$$u(x) = u_h(x) + u_p(x)$$

$$L[u_h] = 0 \quad u_h(a) = \alpha \;\; u_h(b) = \beta$$

$$L[u_p] = f(x) \quad u_p(a) = 0 \;\; u_p(b) = 0$$

The solution $u_p(x)$ is then found using the same method as above. Depending on the specific form of the operator L the solution $u_h(x)$ can often be found analytically, and frequently takes the form of a linear combination of the two solutions to the homogeneous linear operator.

3 Numerical Examples of Sturm-Liouville Problems

3.1 A Note on Quadrature Methods

Integrals can be approximated numerically using the following general rule[1]

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} f(x_i)w_i$$

Where the nodes x_i and weights w_j are based on the specific quadrature technique being used. The integrals in the following section were all approximated using two quadrature techniques: the trapezoidal method and the Gauss-Legendre method.

The trapezoidal technique consists of nodes $x_i = 2\pi i/N, i = 1, \ldots, N$ and weights $w_i = 2\pi/N$. The trapezoidal rule works well for functions that are periodic on the interval $[0, 2\pi)$ for smooth integrands. All examples were evaluated with $N = 100$ nodes.

The Gauss-Legendre quadrature uses nodes x_i defined as the zeros of the Nth degree Legendre polynomial and the corresponding optimal weight w_i. Gauss-Legendre polynomials are used to integrate on the interval $[-1, 1]$, and therefore a change of variables must be used to shift the interval to the desired endpoint. This formula is given by

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} f\left(\frac{b-a}{2}x_i + \frac{b+a}{2}\right) w_i$$

with nodes x_i and w_i defined as above. All examples were evaluated with $N = 10$ nodes with the resulting node values and corresponding weights derived from the scipy.special library function for Legendre polynomials.

3.1.1 A Simple homogeneous Example

Consider the boundary value problem given by

$$\begin{aligned} u''(x) &= x^2 \\ u(0) &= u(1) = 0 \end{aligned}$$

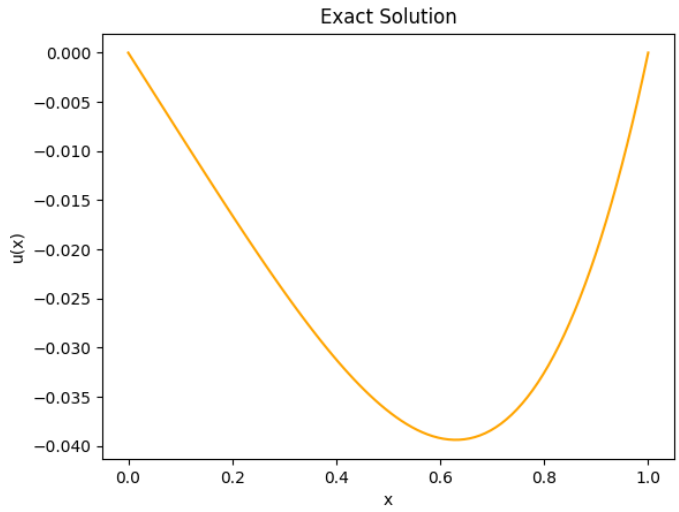
with exact solution given by $u(x) = \frac{1}{12}(x^4 - x)$. First we need to find the Green's function related to the operator $L[u] = u''(x)$ with the prescribed boundary conditions. It's easy to see that one solution to the homogeneous problem is $u_1(x) = x$ which satisfies the first boundary condition. The second solution can be given by $u_2(x) = x - 1$ which satisfies the second boundary condition. The Wronskian of these solutions is $W(x) = (x)(1) - (1)(x - 1) = 1$. Plugging all of these into the formula for Green's function above we get

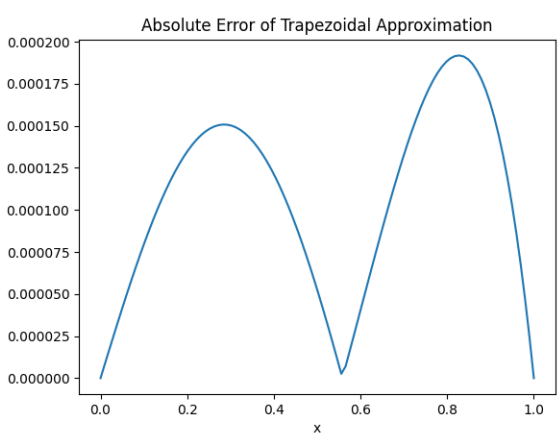
$$G(x, s) = \begin{cases} s(x-1) & 0 \leq x \leq s \\ x(s-1) & s \leq x \leq 1 \end{cases}$$

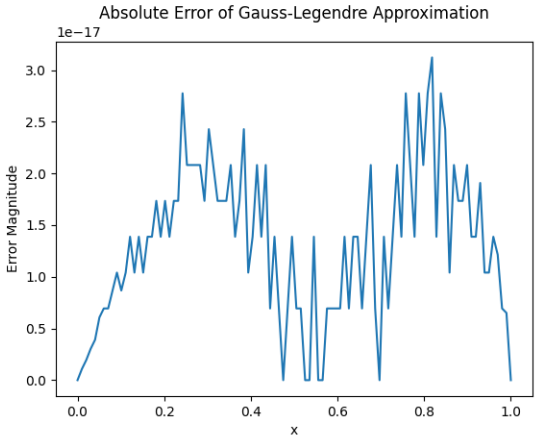
And therefore the full solution is given by

$$u(x) = \int_0^x s^3(x-1)ds + \int_x^1 s^2 x(s-1)ds$$

Using the quadrature methods described previously, the error of the approximations were as follows:







3.1.2 Nonhomogeneous Boundary Example

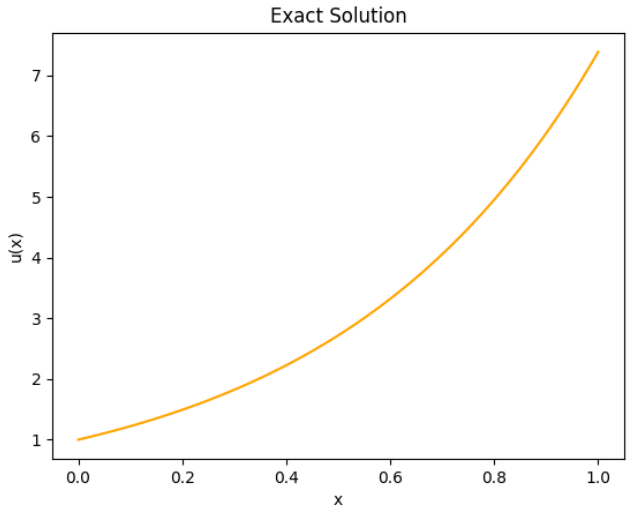
Consider the boundary value problem given by

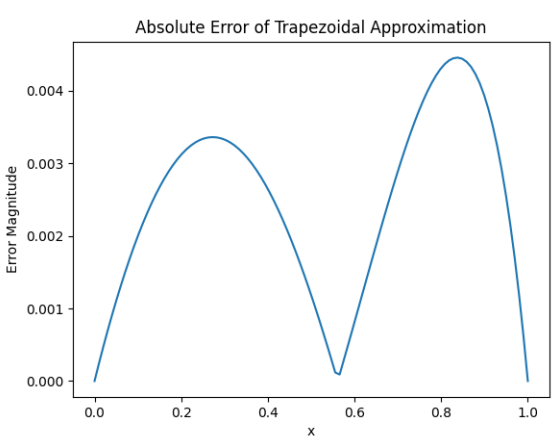
$$u''(x) = 4e^{2x}$$
$$u(0) = 1,\ u(1) = e^2$$

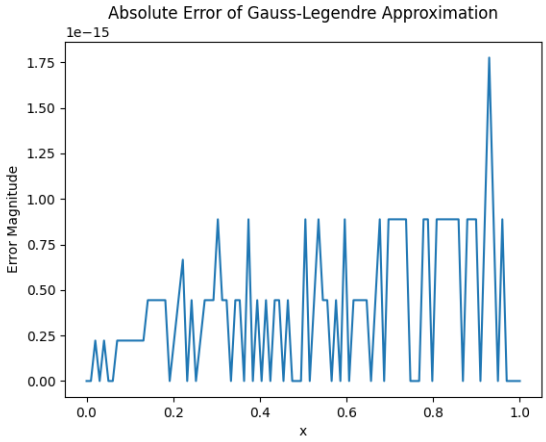
with exact solution given by $u(x) = e^{2x}$. The solution can be split into $u(x) = u_h(x) + u_p(x)$, where $u_h(x)$ satisfies the homogeneous problem with the prescribed boundary values, while $u_p(x)$ satisfies the forced problem with 0 boundaries. It's easy to verify that $u_h(x) = x(e^2 - 1) + 1$, and $u_p(x)$ is approximated using the same strategy as before. Because the differential operator and boundary conditions are the same as the previous problem, the Green's function is also the same. Therefore the full solution is given by

$$u(x) = x(e^2 - 1) + 1 + \int_0^x s(x-1)4e^{2s}ds + \int_x^1 x(s-1)4e^{2s}ds$$

Using the quadrature methods described previously, the error of the approximations were as follows:







3.1.3 A More Involved Problem

Consider the Sturm-Liouville Boundary Value Problem

$$\frac{d}{dx}\left(x^2\frac{du}{dx}\right) - 2u(x) = 4x^2$$
$$u(1) = 2$$
$$u(2) = -1$$

with exact solution $u(x) = x^2 - 3x + 4x^{-2}$. First we must find the solutions to the associated homogeneous problem. It can be verified that two linearly independent solutions take the form $u(x) = c_1 x$ and $u(x) = c_2 x^{-2}$. We must now use a linear combination of these to form solutions that are zero at the given boundaries in order to form Green's function. Take

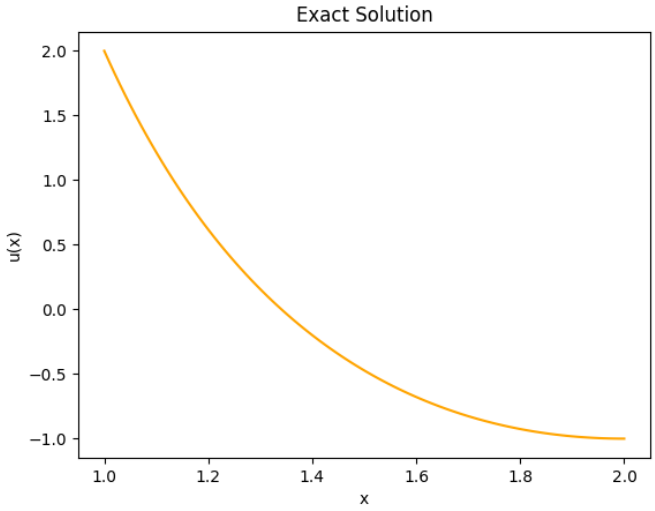
for example $u_1(x) = x - x^{-2}$ and $u_2(x) = x - 8x^{-2}$ which have a Wronskian of $W(x) = 21x^{-2}$. This leads to the Green's function

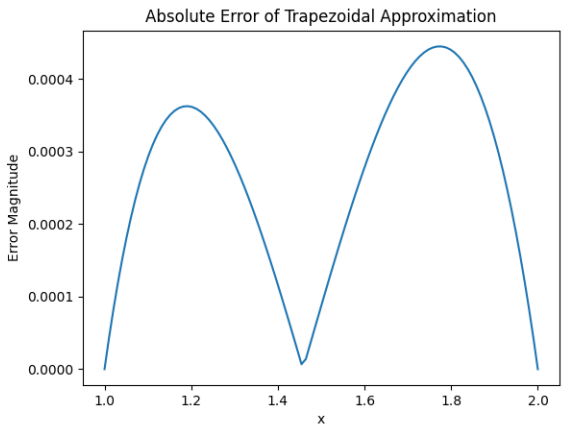
$$G(x,s) = \begin{cases} \frac{1}{21}(s - s^{-2})(x - 8x^{-2}), & 1 \leq x \leq s \\ \frac{1}{21}(x - x^{-2})(s - 8s^{-2}), & s \leq x \leq b \end{cases}$$

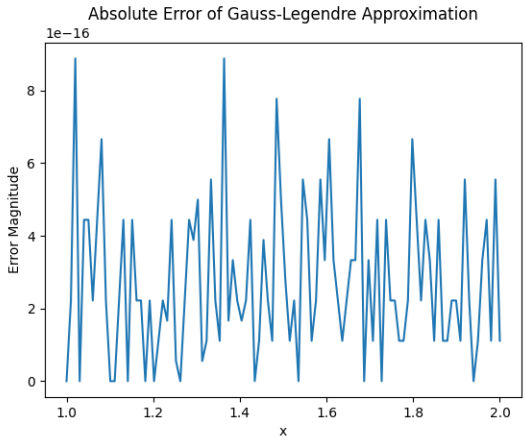
This Green's function is then convolved with the forcing function to approximate the particular solution with zero boundary conditions (as guaranteed by the property of the Green's function). In order to find the homogeneous solution, we need a linear combination of the two solutions from before that satisfy the boundary conditions given. In this case $u_h(x) = -\frac{6}{7}x + \frac{20}{7}x^{-2}$ meets these requirements. Therefore the full solution is given by

$$u(x) = -\frac{6}{7}x + \frac{20}{7}x^{-2} + \int_1^x \frac{1}{21}(s - s^{-2})(x - 8x^{-2})4s^2 ds + \int_x^2 \frac{1}{21}(x - x^{-2})(s - 8s^{-2})4s^2 ds$$

Using the quadrature methods described previously, the error of the approximations were as follows:







3.2 Discussion of Numerical Results

As can be seen in the example problems above, reformulating a boundary value problem into an integral equation leads to extremely accurate results. This makes sense because analytically they are the exact same problem, and the accuracy of these approximations relies only on the method of quadrature being used. Additionally, it is clear that there is no relationship between the x value being approximated and the error of the approximation. This is because the value at a specific x is not dependent on any other values of x unlike when using discretization methods such as time-stepping or finite element and difference methods. Instead, each value is simply a quadrature that only depends on the two boundary values given, greatly reducing the dimension of the problem.

Despite the high level of accuracy given by this method, it is limited by the need for the Green's function associated with the operator and the boundary conditions. In some cases, a Green's function may not exist or may be a significant challenge to solve for, meaning that this method would not be feasible. In general, this method requires a lot of work to be done before numerically integrating the approximation, any step of which may lead to error or not be possible for the given problem.

4 Mathematical Formulation of 2D Problems

4.1 The Fundamental Solution

When extending the use of integral equations to differential equations of multiple dimensions, similar concepts apply that allow us to reduce a problem to an equation just on the boundary of the domain. However, because Green's functions account for conditions on the boundary and the boundary conditions are often much more complicated in higher dimensions, the Green's function is not as easily found and instead we employ the use of the fundamental solution associated with the differential operator, which does not take the boundary conditions into account; the fundamental solution can equivalently be thought of as the Green's function on an infinite unbounded domain[5]. In this case, solutions to the boundary value problem are reformulated into a convolution with the fundamental solution and some unknown density function, to be solved for by implementing the boundary conditions. The fundamental solution is found by solving $L[\phi] = \delta(x - y)$ just as before with Green's functions, but this time without considering the boundary conditions. The analytic solution is then given by [2]

$$u(x) = \int_\Gamma \phi(x, y)\sigma(y)dl(y)$$

where $\sigma(y)$ is the unknown density function. In order to solve for $\sigma(y)$, the equation is forced to satisfy the prescribed boundary conditions

$$g(x) = \int_\Gamma \phi(x, y)\sigma(y)dl(y) \quad \text{where } u(x) = g(x),\ x \in \Gamma$$

Thus the solution has been reduced in dimension to an equation that lies purely on the boundary of the desired domain. The problem can then be discretized in a single dimension

in order to numerically solve for approximate solutions. This method can sometimes be even further improved by representing the solution as a double layer potential, which represents a dipole instead of a single point charge[9]

$$u(x) = \int_\Gamma \partial_{n_y}\phi(x,y)\sigma(y)dl(y)$$

where n_y is the outward normal at y on the boundary. The benefit to this method is that after discretizing and forming the linear system $A\sigma = h$, the matrix A, while dense, is a very well conditioned system and leads to fast convergence by iterative solvers [2], owing to the fact that the double layer is well defined at the center of the dipole as opposed to the infinite singularity of the single layer potential. This causes the eigenvalues of the resulting system to coalesce around 1/2 instead of diverging and creating an ill-conditioned system [9].

4.2 A Laplacian Example

Take the well known Laplacian boundary value problem

$$\begin{aligned} -\Delta u(x) &= 0 \quad x \in \Omega \\ u(x) &= g(x) \quad x \in \partial\Omega = \Gamma \end{aligned}$$

The fundamental solution for the laplacian operator is given by $\phi(x,y) = -\frac{1}{2\pi}\log|x-y|$ (derived in Appendix A). Writing the solution in the double layer potential form $d(x,y) = \partial_{n_y}\phi(x,y)$ gives [8]

$$u(x) = \int_\Gamma d(x,y)\sigma(y)dl(y) = \int_\Gamma \frac{n(y)\cdot|x-y|}{2\pi|x-y|^2}\sigma(y)dl(y)$$

Because the double-layer potential is discontinuous at the boundary, evaluating the limit as x approaches the boundary and enforcing the boundary conditions results in [2]

$$g(x) = \frac{1}{2}\sigma(x) + \int_\Gamma \frac{n(y)\cdot|x-y|}{2\pi|x-y|^2}\sigma(y)dl(y)$$

Finally, by discretizing and applying an appropriate quadrature technique, we are left with a linear system where row i is given by

$$g(x_i) = \frac{1}{2}\sigma(x_i) + \sum_{j=1}^{N} d(x_i,x_j)\sigma(x_j)w_j$$

$$\left(\frac{1}{2}I + D\right)\sigma = g$$

Solving this system leads to an approximation of $\sigma(x)$ that can then be used to calculate values for the approximation of $u(x)$. As the density function was calculated using a quadrature method, it is often most convenient to use the same quadrature rule to approximate $u(x)$

as the values of $\sigma(x)$ have already been calculated for the corresponding nodes. Otherwise a more complex interpolation technique is required to estimate values of $\sigma(x)$ that were not solved for in the preceding system.

4.2.1 Parameterization of the Boundary

In order to calculate the boundary integral using numerical techniques, the boundary shape and the integrand must be parameterized by a single variable using a function $\mathbf{G} : I \to \mathbb{R}^2$ where I is a defined interval, most often $[0, 2\pi)$. In doing so, the boundary can be written in the form $\Gamma = \{\mathbf{G}(t) : t \in I\}$ and the components broken down into $\mathbf{G}(t) = (G_1(t), G_2(t))$. This allows for all 2 dimensional points in the above formulas to be replaced by a single variable t, and also allows for easy computation of a normal vector at the point t as $n(t) = (G_2'(t), -G_1'(t))$ normalized by the magnitude. Finally, this allows for an expression for the double layer potential at the same point, which normally would result in a singularity owing to the $|x - x|$ in the denominator. Using L'Hopital's rule twice we get that [10]

$$\lim_{t' \to t} d(\mathbf{G}(t), \mathbf{G}(t')) = \frac{G_2'(t)G_1''(t) - G_1'(t)G_2''(t)}{4\pi|G'(t)|^3}$$

We now have all of the tools necessary to numerically evaluate the Laplacian operator in two dimensions.

5 Numerical Results of Laplace's Equation

5.1 Another Note on Quadrature Methods

After parameterizing the curve of a boundary, the same quadrature methods as described previously can be used as the integrals are still only with respect to a single variable, now parameterized by t instead of the coordinate dimension. However, owing to the complexity of a multi-dimensional domain, a much higher concentration of nodes needs to be used to achieve a comparable level of accuracy. For the trapezoidal quadrature rule this is as simple as increasing the number of nodes N being used. However, for Gauss-Legendre polynomials increasing the number of nodes can be problematic because using a high degree of Legendre polynomial can lead to unpredictability in performance; many Legendre root-finding functions are only reliably tested up to degree 100. Instead, the integral can be split up into multiple "panels", each of which has its own Gaussian-Legendre quadrature performed on a smaller number of nodes that are shifted according to the formula presented earlier. In this way, the total number of nodes N is equal to the product of the number of panels and the number of nodes within each panel [1]. Each of the following examples were performed using Trapezoidal rule with $N = 1000$ nodes and the Gaussian-Legendre quadrature with 10 panels of 16 nodes ($N = 160$). Additionally, as the purpose of this section is to explore the performance of different boundary parameterizations instead of different differential equations, all examples have an exact solution given by

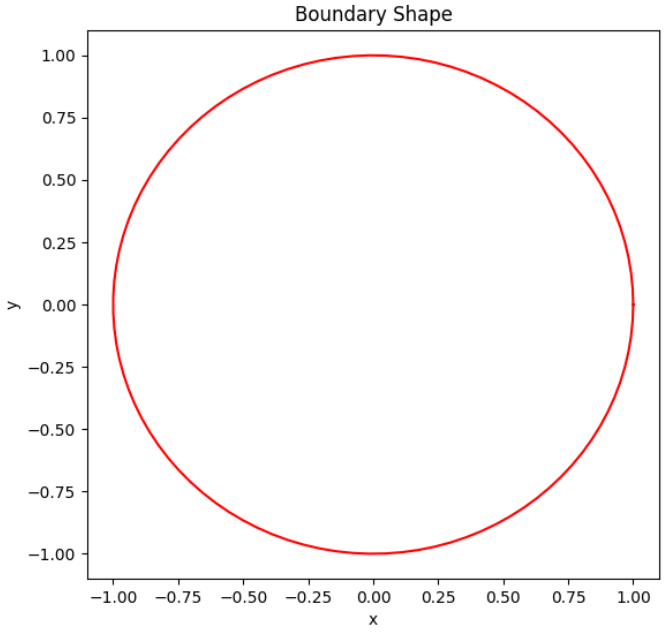
$$u(x) = -\frac{1}{2\pi} \log |x - (2, 3)|$$

5.1.1 A Simple Constant Boundary

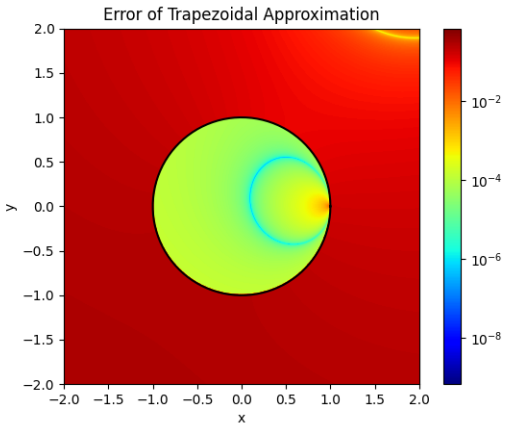
Consider the circular boundary given by the parameterization:

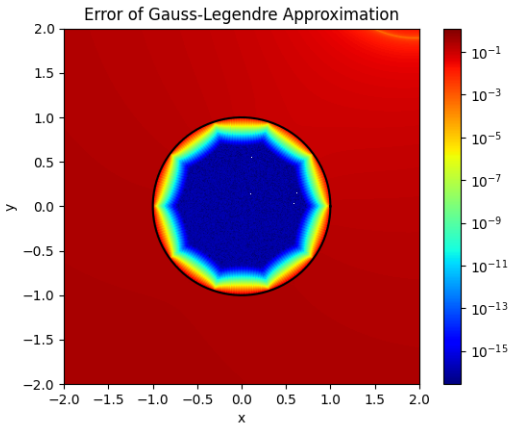
$$x(t) = \cos(t)$$
$$y(t) = \sin(t)$$

As visualized by



Applying the described technique to approximate the solution resulted in the following error plots



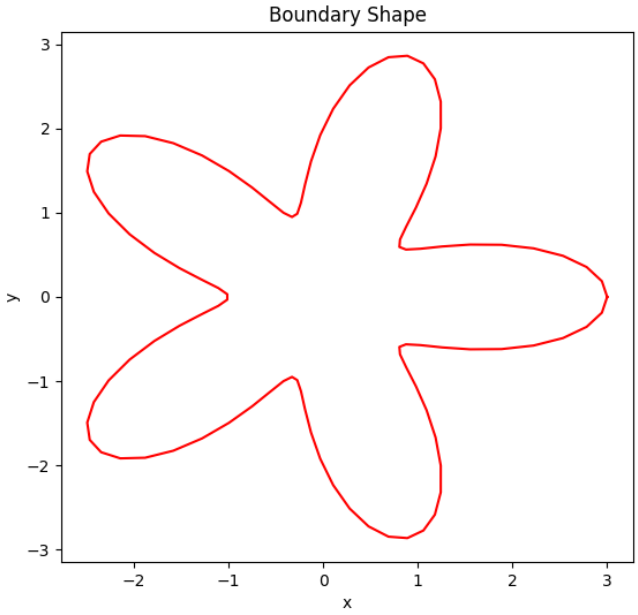


5.1.2 A More Interesting Boundary

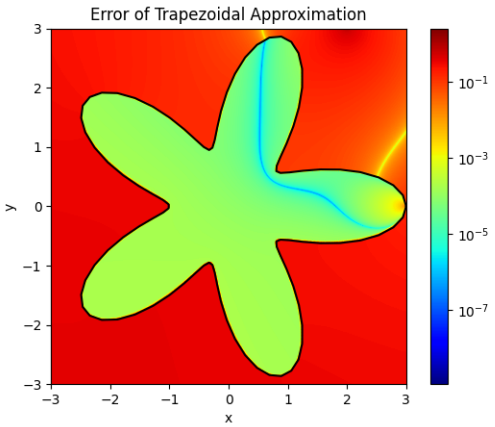
Consider the clover-shaped boundary given by the parameterization:

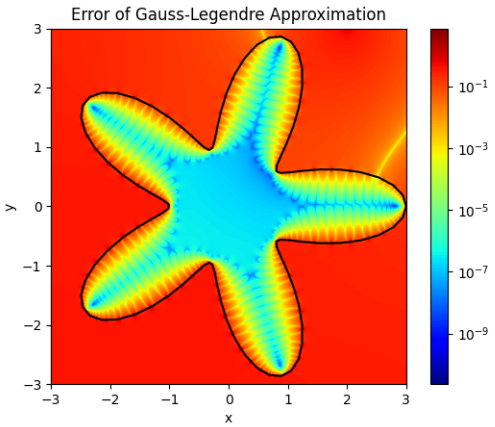
$$x(t) = (\cos(5t) + 2)\cos(t)$$
$$y(t) = (\cos(5t) + 2)\sin(t)$$

As visualized by



Applying the described technique to approximate the solution resulted in the following error plots



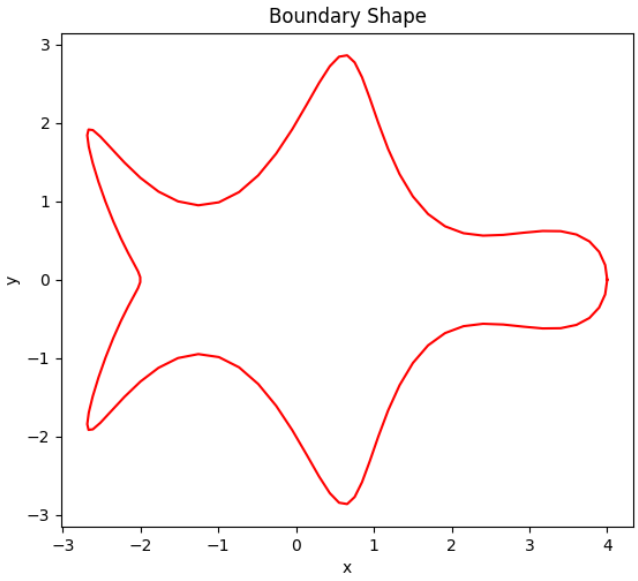


5.1.3 Loss of Radial Periodicity

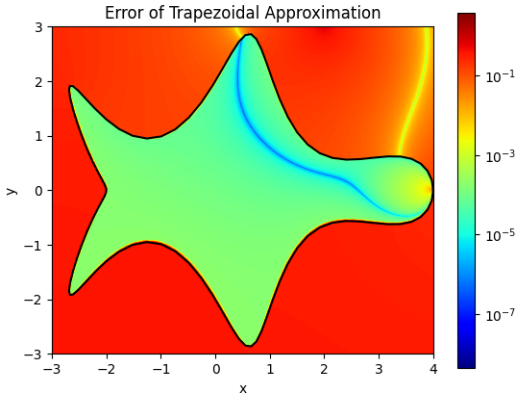
Consider the irregular symmetric boundary given by the parameterization:

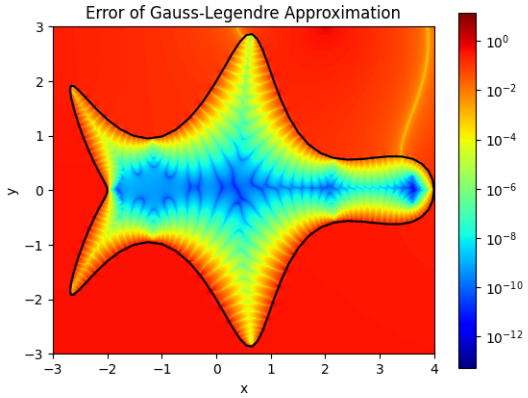
$$x(t) = (\cos(3t) + 3)\cos(t)$$
$$y(t) = (\cos(5t) + 2)\sin(t)$$

As visualized by



Applying the described technique to approximate the solution resulted in the following error plots



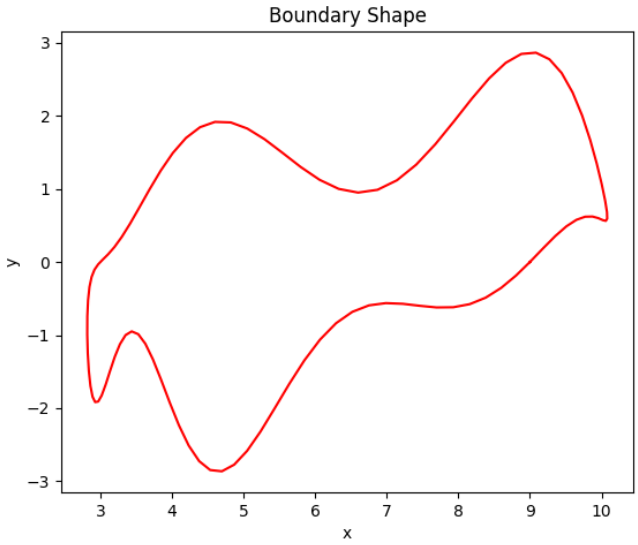


5.1.4 Loss of all Symmetry

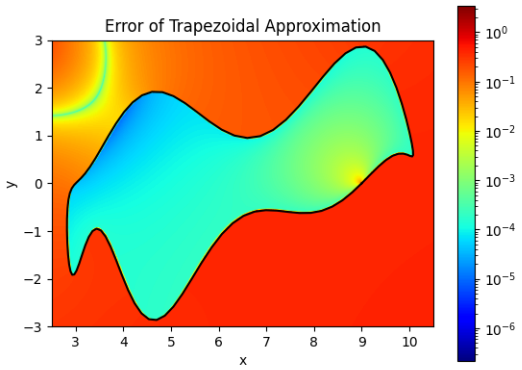
Consider the completely asymmetric boundary given by the parameterization:

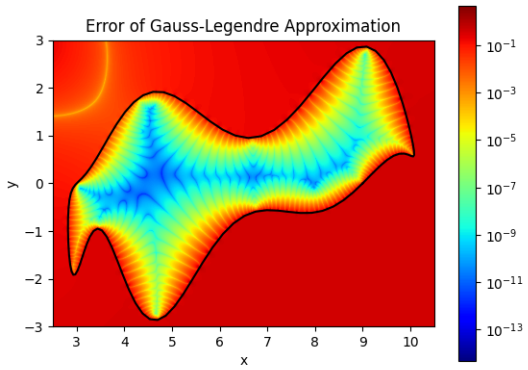
$$x(t) = (\cos(t) + 2)(\sin(t) + 3)$$
$$y(t) = (\cos(5t) + 2)\sin(t)$$

As visualized by



Applying the described technique to approximate the solution resulted in the following error plots





5.2 Discussion of Numerical Results

As seen above, using the technique of boundary integral equations leads to a very high level of accuracy when approximating solutions to partial differential boundary value problems inside the prescribed domain. It is important to note the decrease in accuracy as the evaluating point gets closer to the boundary of the domain, which is caused by the discontinuity of the normal derivative at these points owing to the dipole behavior. These errors can be minimized by increasing the number of nodes being used to evaluate a single point, as well as a variety of more complex techniques not presented in this report. However, the variety of boundary shapes presented above highlight the most important use of boundary integral equations as it relates to multi-dimensional problems. For all discretization techniques, such as finite element or finite difference methods, the above boundaries would require highly specialized and complex mesh grids, which can in turn lead to a high level of error very quickly. Integral equations on the other hand only require a parameterization of the boundary, reducing the dimension of the problem down to one and significantly simplifying the irregularity of uncommon domains. Additionally, the linear systems formed by boundary integral equations are often much smaller in size when compared to those created in other discretization methods, resulting in faster and more stable computational accuracy.

6 Conclusions and Further Extensions

The method of using boundary integral equations through the convolution of Green's functions and fundamental solutions results in extremely accurate approximations of differential equations in multiple dimensions. Together with the parameterization of boundary curves, this becomes a powerful technique for evaluating solutions defined on complicated domains that may be intractable or impossible with discretization techniques. However, boundary integral equations are limited to linear differential equations with constant coefficients in which a Green's function exists, otherwise this method cannot be used.

Beyond the limited introduction of this report, there are many further applications and developments that can be explored. The example of Laplace's equation for two dimensions was motivated by the fact that the resulting integral kernel was smooth and easily evaluated with standard quadrature techniques. More complex techniques and analysis is required to apply this method to other linear differential equations such as the Helmholtz equation or time-dependent PDEs. Additionally, the introduction of Neumann boundary conditions leads to a much more involved exploration of the single and double layer potentials associated with a problem in order to account for singularities and discontinuities. Finally, the numerical methods employed in this report can be improved even further by utilizing different quadratures, dense linear solvers, and interpolation methods applied to a particular problem.

References

[1] Alex Barnett. *Boundary integral equations for BVPs, and their high-order Nystrom quadratures: a tutorial*. June 2014.

[2] "Chapter 10: Integral equation formulations". In: *Fast Direct Solvers for Elliptic PDEs*, pp. 105–114. DOI: 10.1137/1.9781611976045.ch10. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611976045.ch10. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611976045.ch10.

[3] Jerry Farlow et al. *Differential equations & Linear Algebra*. 2nd ed. Pearson, 2018.

[4] Dominique Habault. "Chapter 6 - Boundary Integral Equation Methods - Numerical Techniques". In: *Acoustics*. Ed. by Paul Filippi et al. London: Academic Press, 1999, pp. 189–202. ISBN: 978-0-12-256190-0. DOI: https://doi.org/10.1016/B978-012256190-0/50007-5. URL: https://www.sciencedirect.com/science/article/pii/B9780122561900500075.

[5] Richard Haberman. *Applied partial differential equations with Fourier series and boundary value problems*. 5th ed. Pearson, 2019.

[6] Russell Herman. *4.2: Properties of sturm-liouville eigenvalue problems*. July 2022. URL: https://math.libretexts.org/Bookshelves/Differential_Equations/Introduction_to_Partial_Differential_Equations_(Herman)/04%5C%3A_Sturm-Liouville_Boundary_Value_Problems/4.02%5C%3A_Properties_of_Sturm-Liouville_Eigenvalue_Problems.

[7] John David Logan. *Applied mathematics*. 4th ed. Wiley-Interscience, 2006.

[8] Gunnar Martinsson. "CBMS Conference on Fast Direct Solvers". In: *Fast direct solvers for elliptic pdes*. Society for Industrial and Applied Mathematics, 2020.

[9] P G Martinsson. *A brief introduction to boundary integral equation techniques*. Mar. 2012.

[10] Per-Gunnar Martinsson. "Chapter 12: Discretization of integral equations". In: *Fast direct solvers for elliptic pdes*. Society for industrial and applied mathematics, 2020, pp. 121–135.

7 Appendix A

7.1 Proof of pW being constant

[5]

Consider the Sturm-Liouville linear operator given by

$$L[u] = \frac{d}{dx}\left(p(x)\frac{du}{dx}\right) + q(x)u(x)$$

Now take $u_1(x)$ and $u_2(x)$ to be two linearly independent solutions to the homogeneous problem $L[u] = 0$. The Wronskian of these solutions is given by $W = u_1u_2' - u_1'u_2$. These values can then be plugged into the following expression

$$(pW)' = (p(u_1u_2' - u_1'u_2))'$$

Using Lagrange's identity, which states for the Sturm-Liouville operator L that

$$uL[v] - vL[u] = [p(uv' - u'v)]'$$

We obtain the expression

$$(pW)' = u_1L[u_2] - u_2L[u_1]$$

And by construction, $L[u_1] = L[u_2] = 0$. Therefore $(pW)' = 0$. Because the derivative is equal to zero, the expression pW can only be a constant

7.2 Proof of Sturm-Liouville Form for 2nd Order ODEs

[6]

Consider the general form for a 2nd order linear ODE

$$a(x)y'' + b(x)y' + c(x)y = f(x)$$

If we were to multiply this equation by the term $\frac{\mu(x)}{a(x)}$ where $\mu(x)$ is some function to be determined, we obtain the equation

$$\mu(x)y'' + \mu(x)\frac{b(x)}{a(x)}y' + \mu(x)\frac{c(x)}{a(x)}y = \mu(x)\frac{f(x)}{a(x)}$$

If we add the constraint that $\mu'(x) = \mu(x)\frac{b(x)}{a(x)}$, then we can rewrite the equation into Sturm-Liouville form

$$\frac{d}{dx}\left(\mu(x)\frac{dy}{dx}\right) + \mu(x)\frac{c(x)}{a(x)}y = \mu(x)\frac{f(x)}{a(x)}$$

By now investigating the added constraint, we notice that it takes the form of a separable differential equation

$$\frac{d\mu}{dx} = \mu(x)\frac{b(x)}{a(x)}$$
$$\frac{1}{\mu(x)}d\mu = \frac{b(x)}{a(x)}dx$$
$$\log|\mu(x)| = \int \frac{b(x)}{a(x)}$$
$$\mu(x) = e^{\int \frac{b(x)}{a(x)}}$$

And with this integrating factor it is shown that any general 2nd order linear ODE can be written in the form of a Sturm-Liouville problem, noting that some may not have elementary representations based on the evaluation of the integrating factor.

7.3 Derivation of Fundamental Solution for Laplace's Equation

[5]

Consider the linear operator L defined by the Laplacian operator

$$L[u] = -\Delta u(x)$$

The fundamental solution is defined as the solution to the equation

$$L[\phi] = \delta(x - y)$$

Because the fundamental solution is the steady-state solution with a point charge at $x = y$ with no boundary considerations, there should be a solution that is radially symmetric about the point $x = y$. We can then define $r = |x - y|$ as the radial distance from the point charge, and convert the Laplacian operator to polar form. Because the solution is radially symmetric, any derivatives with respect to θ will be equal to 0 at all points.

$$-\Delta\phi(r) = -\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\phi}{\partial r}\right) - \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2} = -\frac{1}{r}\frac{d}{dr}\left(r\frac{d\phi}{dr}\right)$$

Because $\delta(x - y) = 0$ for any $r \neq 0$, we can write this as

$$-\frac{1}{r}\frac{d}{dr}\left(r\frac{d\phi}{dr}\right) = 0,\ (r \neq 0)$$
$$r\frac{d\phi}{dr} = c_1$$
$$\phi(r) = c_1 \log r + c_2$$

In order to find the unknown coefficients we can contain the singularity by integrating around a small circle of radius r and apply the divergence theorem. Note that the derivative over the singularity of the Dirac delta distribution is equal to 1 by definition (negated due to the integrand sign)

$$\int\int -\Delta\phi dA = -1$$

$$\int\int \nabla\cdot(\nabla\phi)dA = \oint \nabla\phi\cdot\hat{n}\,ds = -1$$

Because the normal vector to a circle always points outwards, and the fundamental solution only depends on r, the derivative of ϕ in the normal direction is just the derivative of ϕ with respect to r. The circumference of the circle is $2\pi r$ so we get the result

$$2\pi r\frac{d\phi}{dr} = -1$$

$$r\frac{d\phi}{dr} = -\frac{1}{2\pi}$$

Setting this equal to the equation for c_1 (after the first integration) gives us the value of that constant. c_2 is arbitrary and set to 0 for convenience. This gives us the final result for the fundamental solution

$$\phi(x,y) = -\frac{1}{2\pi}\log r = -\frac{1}{2\pi}\log|x-y|$$

8 Appendix B

8.1 Code used for 1D problems

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp

'''
This code approximates the solution to the following BVP

(p(x)u'(x))' + q(x)u(x) = f(x)
u(a) = alpha u(b) = beta
'''

def driver():

    a = 1
    b = 2
    n = 100

    f = lambda x: 4*x**2
    u1 = lambda x: x - x**-2
    u2 = lambda x: x-8*x**-2
    pW = 21
    uh = lambda x: -(6/7)*x + (20/7)*x**-2

    x = np.linspace(a,b,n)

    u = x**2 - 3*x + 4*x**-2

    uapp = evalApprox(x,f,u1,u2,pW,uh,a,b)


    plt.plot(x,u,label='Actual Solution')
    plt.title("Exact Solution")
    plt.xlabel("x")
    plt.ylabel("u(x)")
    plt.show()

    error = abs(uapp - u)

    plt.plot(x,error)
    #plt.title("Absolute Error of Trapezoidal Approximation")
    plt.title("Absolute Error of Gauss-Legendre Approximation")
```

```
    plt.xlabel("x")
    plt.ylabel("Error Magnitude")
    plt.show()


def evalApprox(x,f,u1,u2,pW,uh,a,b):

    uapp = np.zeros(x.size)

    for i in range(x.size):
        val = x[i]
        leftIntegrand = lambda s: (u1(s) * u2(val)) / pW * f(s)
        rightIntegrand = lambda s: (u1(val) * u2(s)) / pW * f(s)

        uapp[i] = quadrature(leftIntegrand,a,val) + quadrature(rightIntegrand,val,b) + u

    return uapp


def quadrature(f,a,b):

    N = 10

    #Used for Trapezoidal Rule
    nodes = np.linspace(a,b,N)
    weights = (b-a)/N * np.ones(N)

    #Used for Gauss-Legendre
    nodes, weights = sp.special.roots_legendre(N)
    nodes = (nodes * ((b-a) / 2)) + ((a+b) / 2)
    weights = weights * ((b-a)/2)

    return sum(f(nodes)*weights)

driver()
```

8.2 Code for 2D problems

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt


#Defines window of graph
left = 2.5
right = 10.5
bottom = -3
top = 3

#Defines resolution of Graph
dA = 500
dt = 100

#Code for parameterizations of boundary
'''
#CIRCLE DOMAIN
x = lambda t: np.cos(t)
y = lambda t: np.sin(t)

dx = lambda t: -np.sin(t)
dy = lambda t: np.cos(t)

ddx = lambda t: -np.cos(t)
ddy = lambda t: -np.sin(t)
#REQUIRES DIMENSIONS [-2,2] X [-2,2]
'''

'''
#CLOVER DOMAIN 2
x = lambda t: (np.cos(5*t) + 2) * np.cos(t)
y = lambda t: (np.cos(5*t) + 2) * np.sin(t)

dx = lambda t: -np.sin(t)*(np.cos(5*t)+2) - 5*np.sin(5*t)*np.cos(t)
dy = lambda t: np.cos(t)*(np.cos(5*t)+2) - 5*np.sin(5*t)*np.sin(t)

ddx = lambda t: 10*np.sin(t)*np.sin(5*t)-26*np.cos(5*t)*np.cos(t)-2*np.cos(t)
ddy = lambda t: -10*np.cos(t)*np.sin(5*t)-26*np.cos(5*t)*np.sin(t)-2*np.sin(t)
#REQUIRES DIMENSIONS [-3,3] X [-3,3]
'''

'''
```

```
#LITTLE GUY DOMAIN
x = lambda t: (np.cos(3*t) + 3) * np.cos(t)
y = lambda t: (np.cos(5*t) + 2) * np.sin(t)

dx = lambda t: -np.sin(t)*(np.cos(3*t)+3) - 3*np.sin(3*t)*np.cos(t)
dy = lambda t: np.cos(t)*(np.cos(5*t)+2) - 5*np.sin(5*t)*np.sin(t)

ddx = lambda t: 6*np.sin(t)*np.sin(3*t)-10*np.cos(3*t)*np.cos(t)-3*np.cos(t)
ddy = lambda t: -10*np.cos(t)*np.sin(5*t)-26*np.cos(5*t)*np.sin(t)-2*np.sin(t)
#REQUIRES DIMENSIONS [-3,4] X [-3,3]
'''



#BLOB DOMAIN
x = lambda t: (np.cos(t)+2)*(np.sin(t)+3)
y = lambda t: (np.cos(5*t) + 2) * np.sin(t)

dx = lambda t: np.cos(t)*(np.cos(t) + 2) - np.sin(t)*(np.sin(t)+3)
dy = lambda t: np.cos(t)*(np.cos(5*t)+2) - 5*np.sin(5*t)*np.sin(t)

ddx = lambda t: -2*np.sin(t)-(4*np.sin(t) + 3)*np.cos(t)
ddy = lambda t: -10*np.cos(t)*np.sin(5*t)-26*np.cos(5*t)*np.sin(t)-2*np.sin(t)
#REQUIRES DIMENSIONS [2.5,10.5] X [-3,3]


#Exact solution
f = lambda point: (-1/(2*np.pi))*np.log(np.linalg.norm(point - np.array([2,3])))



def driver():

    plotBoundary()
    plotDomainTrap()
    plotDomainGauss()




#Evaluates the solution at a specified point
def evalSolution(xi,sigma,nodes,weights,N):

    ny1 = dy(nodes)
    ny2 = -1*dx(nodes)
```

```
    speed = np.sqrt(ny1**2 + ny2**2)
    ny1 = ny1 / speed
    ny2 = ny2 / speed

    diff1 = xi[0] - x(nodes)
    diff2 = xi[1] - y(nodes)

    numerator = (ny1*diff1) + (ny2*diff2)
    denominator = 2*np.pi*(diff1**2 + diff2**2)

    a = numerator / denominator * speed * sigma * weights

    return sum(a)



#Sets up and solves the system for the density function
def solveDensity(f,nodes,weights,N):


    D = np.zeros([N,N])
    g = np.zeros(N)


    for i in range(N):
        s = nodes[i]
        ny1 = dy(nodes)
        ny2 = -1*dx(nodes)
        speed = np.sqrt(ny1**2 + ny2**2)
        ny1 = ny1 / speed
        ny2 = ny2 / speed

        diff1 = x(s) - x(nodes)
        diff2 = y(s) - y(nodes)

        numerator = (ny1*diff1) + (ny2*diff2)
        denominator = 2*np.pi*(diff1**2 + diff2**2)

        a = numerator / denominator * speed * weights

        D[i,:] = a
        D[i,i] = ((dy(s) * ddx(s)) - (dx(s) * ddy(s))) /
         (4 * np.pi * (np.linalg.norm([dx(s),dy(s)]) ** 3)) * speed[i] * weights[i]
        g[i] = f(np.array([x(nodes[i]),y(nodes[i])]))
```

```
    A = (-1/2 * np.eye(N)) + D

    sigma = np.linalg.solve(A,g)

    return sigma

#Returns the nodes and weights for Trapezoidal
def makeTrapNodes(N):

    nodes = np.linspace(0,2*np.pi,N)
    weights = 2*np.pi/N * np.ones(N)

    return nodes, weights

#Returns the nodes and weights for Gauss-Legendre
def makeGaussNodes(panels,num):

    nodes = np.array([])
    weights = np.array([])
    step = 2*np.pi / panels
    a = 0
    b = step

    for i in range(panels):

        localNodes, localWeights = sp.special.roots_legendre(num)

        localNodes = (localNodes * ((b-a) / 2)) + ((a+b) / 2)
        localWeights = localWeights * ((b-a)/2)

        nodes = np.append(nodes, localNodes)
        weights = np.append(weights, localWeights)

        a += step
        b += step

    return nodes, weights

#Plots the boundary specified
def plotBoundary():

    t = np.linspace(0,2*np.pi, dt)
    xt = np.zeros(dt)
```

```
    yt = np.zeros(dt)

    for i in range(dt):
        xt[i] = x(t[i])
        yt[i]  = y(t[i])


    plt.plot(xt,yt, color='red')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Boundary Shape')
    plt.show()



#Makes the error plot for a generalized quadrature rule
def makePlot(f,sigma,nodes, weights, N, method):

    t = np.linspace(0,2*np.pi, dt)
    xvals = np.linspace(left,right,dA)
    yvals = np.linspace(bottom,top,dA)

    errors = np.zeros([dA,dA])

    for i in range(dA):
        for j in range(dA):

            point = np.array([xvals[i],yvals[j]])
            uex = f(point)
            uapp = evalSolution(point, sigma, nodes, weights, N)
            err = abs(uex - uapp)

            errors[i][j] = err if not err == 0 else np.finfo(np.float64).eps

    xt = np.zeros(dt)
    yt = np.zeros(dt)

    for i in range(dt):
        xt[i] = x(t[i])
        yt[i]  = y(t[i])


    plt.imshow(np.transpose(errors), norm='log',
     extent=(left,right,bottom,top),origin='lower',cmap='jet')
    plt.colorbar()
```

```
    plt.plot(xt,yt, color='black')
    plt.xlabel('x')
    plt.ylabel('y')
    if method == 'Trapezoidal':
        plt.title('Error of Trapezoidal Approximation')
    if method == 'Gauss-Legendre':
        plt.title('Error of Gauss-Legendre Approximation')
    plt.show()



#Makes the error plot for Trapezoidal
def plotDomainTrap():
    N = 1000
    nodes,weights = makeTrapNodes(N)

    sigma = solveDensity(f,nodes,weights,N)

    makePlot(f,sigma,nodes, weights, N, 'Trapezoidal')


#Makes the error plot for Gauss-Legendre
def plotDomainGauss():

    panels = 10
    num = 16
    N = panels * num
    nodes, weights = makeGaussNodes(panels,num)


    sigma = solveDensity(f,nodes,weights,N)

    makePlot(f,sigma,nodes, weights, N, 'Gauss-Legendre')


driver()
```