Algorithms to Approximate Solutions of Poisson's Equation in Three Dimensions

Ray Dambrose

University of Mary Washington, rsdambrose@gmail.com

Follow this and additional works at: https://scholar.rose-hulman.edu/rhumj

Part of the Numerical Analysis and Computation Commons, Numerical Analysis and Scientific Computing Commons, Partial Differential Equations Commons, and the Special Functions Commons

Recommended Citation

Available at: https://scholar.rose-hulman.edu/rhumj/vol20/iss1/2
Algorithms to Approximate Solutions of Poisson's Equation in Three Dimensions

Cover Page Footnote
I'd like to thank Dr. Jangwoon Lee for giving me the opportunity and guidance to complete this research, Dr. Larry Lehman and Dr. Debra Hydorn for their time and edits as I was writing this paper, and to the University of Mary Washington's Summer Science Institute where my research began.
Algorithms to Approximate Solutions of Poisson’s Equation in Three Dimensions

By Ray Dambrose

Abstract. The focus of this research is to develop numerical algorithms to approximate solutions of Poisson's equation in three dimensional rectangular prism domains. Numerical analysis of partial differential equations is vital to understanding and modeling these complex problems. Poisson's equation can be approximated with a finite difference approximation. A system of equations can be formed that gives solutions at internal points of the domain. A computer program was developed to solve this system with inputs such as boundary conditions and a nonhomogenous source function. Approximate solutions are compared with exact solutions to prove their accuracy. The program was tested with an increasing number of subintervals to ensure that the approximations get closer to the actual solution.

1 Introduction

Mathematics can be used to simplify and understand complex phenomena that occur in our everyday world. One such equation, Poisson's equation, is a steady state, time-independent variation of Laplace's equation \[ \nabla^2 \phi = 0 \] that models energy distribution in equilibrium systems. Solutions of this equation model heat distribution through a region, concentration of particles after diffusion, electrostatic potential, and Newtonian gravity potential \[ \phi \]. Although actual solutions to Poisson's equation are known, these solutions are complicated and difficult to calculate. Numerical methods become necessary to efficiently model solutions of this partial differential equation.

In this paper, we will approximate Poisson's equation with a finite difference approximation in various rectangular prism domains. This approximation will be used to form a system of linear equations that gives solutions at internal points of our domain. First, we will identify patterns in this system for a small number of subintervals. With these patterns, we will define algorithms to construct the matrices and vectors of this equation with a general number of subintervals along the x, y, and z axes.

Mathematics Subject Classification. 65N06, 65Y99

Keywords. Poisson's equation, partial differential equations, numerical analysis, matlab, finite difference methods
In this research, we develop a new method to approximate solutions of Poisson's equation in three dimensions that has not been previously explored in other literature. A Matlab program was developed to approximate solutions using the algorithms mentioned above to construct and solve approximate solutions in a given region. We will demonstrate the accuracy of this numerical method by comparing approximate solutions with exact solutions and show that approximate solutions get closer to the exact solution as the number of subintervals increase. With this testing, we prove that this method can accurately model solutions of Poisson's equation for real world situations where exact solutions are not known.

2 Finite Difference Approximation of Poisson's Equation

Poisson's equation in three dimensions is defined as

\[ \nabla^2 u = Q(x, y, z) \]

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = Q(x, y, z). \]  

(1)

Here, \( Q(x, y, z) \) is the nonhomogenous source function of our region, \( R \). Boundary conditions, \( \alpha \), of our region must also be given, that is, the solution \( u \) at the edges of our domain. Let our domain be a rectangular prism of length \( L \), height \( H \), and width \( W \). The boundaries of this region are \( x = 0, x = L \), the length of our surface, \( y = 0, y = H \), the height of our surface, \( z = 0 \), and \( z = W \), the width of our surface. The boundary conditions are given as six functions of two variables:

\[ u(x, 0, z) = f_1(x, z) \quad u(0, y, z) = g_1(y, z) \quad u(x, y, 0) = h_1(x, y) \]
\[ u(x, H, z) = f_2(x, z) \quad u(L, y, z) = g_2(y, z) \quad u(x, y, W) = h_2(x, y) \]

We can approximate the second derivatives, \( \nabla^2 u \), at any point in our region, \((x_i, y_j, z_k)\), with a finite difference approximation [3] as

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \approx \frac{u(x_i + \Delta x, y_j, z_k) + u(x_i - \Delta x, y_j, z_k) - 2u(x_i, y_j, z_k)}{(\Delta x)^2} \]
\[ + \frac{u(x_i, y_j + \Delta y, z_k) + u(x_i, y_j - \Delta y, z_k) - 2u(x_i, y_j, z_k)}{(\Delta y)^2} \]
\[ + \frac{u(x_i, y_j, z_k + \Delta z) + u(x_i, y_j, z_k - \Delta z) - 2u(x_i, y_j, z_k)}{(\Delta z)^2}. \]

(2)

Let \( \Delta x = \Delta y = \Delta z \). Equation (2) can be simplified and applied to Poisson's equation (1)
as
\[
\begin{align*}
&u(x_i + \Delta x, y_l, z_k) + u(x_i - \Delta x, y_l, z_k) \\
&+ u(x_i, y_l + \Delta y, z_k) + u(x_i, y_l - \Delta y, z_k) \\
&+ u(x_i, y_l, z_k + \Delta z) + u(x_i, y_l, z_k - \Delta z) \\
&- 6u(x_i, y_l, z_k) \\
= & (\Delta x)^2 \cdot Q(x, y, z).
\end{align*}
\]

Let us apply a new notation to equation (3). Let \( x_i + \Delta x = x_{i+1}, \ y_l + \Delta y = y_{l+1}, \) and \( z_k + \Delta z = z_{k+1} \). Then, let every \( u(x_i, y_l, z_k) = u_{i,j,k} \). We can simplify equation (3) as
\[
\begin{align*}
u_{i+1,l,k} + u_{i-1,l,k} + u_{i,l+1,k} + u_{i,l-1,k} + u_{i,l,k+1} + u_{i,l,k-1} - 6u_{i,j,k} \\
= & (\Delta x)^2 \cdot Q(x_i, y_l, z_k).
\end{align*}
\]

\section{Cube Domains}

First, consider a cube domain where \( H = L = W \). If \( \Delta x = \Delta y = \Delta z \), then we can separate our region into \( n = \frac{L}{\Delta x} = \frac{H}{\Delta y} = \frac{W}{\Delta z} \) subintervals along the \( x \), \( y \), and \( z \) axes.

The goal is to approximate all solutions, \( u_{i,j,k} \) where \( 0 \leq i \leq n, \ 0 \leq j \leq n, \) and \( 0 \leq k \leq n. \) As we have seen from equation (4), any point \( u_{i,j,k} \) in the region is related to the six points surrounding it, as depicted in Figure 1.

![Figure 1: Point in 3D Region](image)

Note that many of the values in this region are already defined. From the boundary conditions, it is known that \( u_{0,l,k} = g_1(y_l, z_k), \ u_{n,l,k} = g_2(y_l, z_k), \ u_{i,0,k} = f_1(x_i, z_k), \ u_{i,n,k} = f_2(x_i, z_k), \ u_{i,l,0} = h_1(x_i, y_l), \) and \( u_{i,l,n} = h_2(x_i, y_l). \) The remaining \((n-1)^3\) points will be approximated by building a linear system of equation (4). We will create a system of \((n-1)^3\) equations, one for each solution at an internal point of our cube by iterating through all possible values of \( i, \ l, \) and \( k, \) where \( 0 < i < n, \ 0 < l < n, \) and \( 0 < k < n. \) For example, if working with \( n = 4 \) subintervals, the system of equations would look something like the following:
These values may not be the same, so we state that

For convenience, we will denote each of the \((n-1)^3\) unknown values as \(v_r\) so that

For example, where \(n = 4\), consider how system (5) changes where \(k = 1 \) and \( l = 1 \).

Note that some points qualify for multiple boundary condition functions. In instances where two or more boundary conditions overlap, the point will be taken as the average of the relevant boundary conditions, assuring that the solutions over the rectangular prism domain are continuous. For instance, \(u_{000}\) is along three boundaries, \(x = 0, y = 0,\) and \(z = 0\) and is given as three boundary conditions, \(f_1(x_0, z_0), g_1(y_0, z_0)\) and \(h_1(x_0, y_0)\). These values may not be the same, so \(u_{000}\) is taken as the average of all three boundary conditions.

\[
u_{000} = \frac{f_1(x_0, z_0) + g_1(y_0, z_0) + h_1(x_0, y_0)}{3}
\]
After plugging boundary conditions and internal points into system (5) as we saw in system (6), we can simplify these equations into a single equation of matrices and vectors,

\[ A \vec{v} + \vec{b} = (\Delta x)^2 \cdot \vec{q}. \] (7)

We want to solve for \( \vec{v} \), the vector of approximate solutions at each internal point of our domain. \( A \) is the coefficient matrix of these solutions, \( \vec{b} \) is the vector of boundary conditions at these points, and \( \vec{q} \) is the vector of source functions. The following sections with be devoted to constructing algorithms to construct \( A \), \( \vec{q} \), and \( \vec{b} \) for any number of subintervals.

### 3.1 A Matrix Pattern

To identify the patterns of the \( A \) matrix for any number of subintervals \( n \), first consider \( A \) for the \( n = 4 \) example in system (5). Instead of thinking of \( A \) as a 27 \( \times \) 27 matrix, note that \( A \) is a 3 \( \times \) 3 matrix of 3 \( \times \) 3 block matrices holding 3 \( \times \) 3 submatrices. On the main diagonal of \( A \), we see a notable block matrix \( A' \). On the main diagonal of \( A' \) is a matrix \( A'' \) with -6 along the main diagonal and 1 on either side. On either side of the main diagonal in \( A' \) is the 3 \( \times \) 3 identity matrix, \( I' \).

\[
A'' = \begin{bmatrix}
-6 & 1 & 0 \\
1 & -6 & 1 \\
0 & 1 & -6
\end{bmatrix}
\Rightarrow
A' = \begin{bmatrix}
A'' & I' & 0 \\
I' & A'' & I' \\
0 & I' & A''
\end{bmatrix}
\]

In \( A, A' \) is on the main diagonal. On either side of \( A' \), is a block matrix \( I \) that has 3 \( \times \) 3 identity matrices along main diagonal. In other words, \( A \) can be simplified as

\[
A = \begin{bmatrix}
A' & I & 0 \\
I & A' & I \\
0 & I & A'
\end{bmatrix}.
\]

Using the patterns seen above, it is possible to define an algorithm for constructing \( A \) for any number of subintervals \( n \):

1. \( A \) is a \((n - 1) \times (n - 1)\) block matrix of \((n - 1) \times (n - 1)\) block matrices. These block submatrices hold \((n - 1) \times (n - 1)\) submatrices.

2. Define a notable \((n - 1) \times (n - 1)\) matrix, \( A'' \), with -6 on the main diagonal and 1 on either side.

\[
A'' = \begin{bmatrix}
-6 & 1 & 0 & \cdots & 0 \\
1 & -6 & 1 & \cdots & 0 \\
0 & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & 1 & -6 & 1 \\
0 & \cdots & 0 & 1 & -6
\end{bmatrix}
\]
3. Define $A'$ as a $(n-1) \times (n-1)$ block matrix comprised of $(n-1) \times (n-1)$ submatrices. $A''$ appears on the main diagonal of $A'$. On either side of $A''$ is the $(n-1) \times (n-1)$ identity matrix, $I'$. Here, 0 represents the $(n-1) \times (n-1)$ zero matrix.

$$A' = \begin{bmatrix}
A'' & I' & 0 & \cdots & 0 \\
I' & A'' & I' & \cdots & 0 \\
0 & \cdots & I' & A'' & I' \\
0 & \cdots & 0 & I' & A'' \\
\end{bmatrix}$$

4. Define $I$ to be a $(n-1) \times (n-1)$ block matrix with $I'$ along the main diagonal, and all other entries the $(n-1) \times (n-1)$ zero matrix.

$$I = \begin{bmatrix}
I' & 0 & 0 & \cdots & 0 \\
0 & I' & 0 & \cdots & 0 \\
0 & \cdots & I' & 0 & \cdots & 0 \\
0 & \cdots & 0 & I' & \cdots & 0 \\
0 & \cdots & 0 & 0 & I' \\
\end{bmatrix}$$

5. Let $A$ be a $(n-1) \times (n-1)$ block matrix with $A'$ across the diagonal. $I$ lies on either side of the main diagonal. Here, 0 represents an $(n-1) \times (n-1)$ block matrix comprised of $(n-1) \times (n-1)$ zero matrices.

$$A = \begin{bmatrix}
A' & I & 0 & \cdots & 0 \\
I & A' & I & \cdots & 0 \\
0 & \cdots & I' & A' & I \\
0 & \cdots & 0 & I & A' \\
\end{bmatrix}$$

### 3.2 Boundary Condition Vector Pattern

First, note how we can iterate through the source function vector and boundary condition vector. For any $v_r = u_{i,l,k} = u(x_i, y_l, z_k)$, we can find $i$, $l$, and $k$ as:

$$i = (r - 1) \mod (n - 1) + 1,$$

$$l = \left\lfloor \frac{r - 1}{n - 1} \right\rfloor \mod (n - 1) + 1,$$

$$k = \left\lfloor \frac{r - 1}{(n - 1) \cdot (n - 1)} \right\rfloor + 1.$$

Note that for $w \in \mathbb{R}$, the floor of $w$, denoted $\lfloor w \rfloor$, is the largest integer that is less than or equal to $w$. 

Rose-Hulman Undergrad. Math. J. | Volume 20, Issue 1, 2019
Consider how points \( u_{i,l,k} \) are mapped to \( v_r \). The first few points are denoted as: \( v_1 = u_{111}, v_2 = u_{211}, \ldots, v_{n-1} = u_{(n-1),1,1} \). Then, at \( v_n \), \( l \) increments and \( i \) returns to 1, so \( v_n = u_{121} \). This pattern continues until we reach \( v_{(n-1)^2} = u_{(n-1),(n-1),1} \). Then, \( k \) will increment, while \( i \) and \( l \) return to 1 so that \( v_{(n-1)^2+1} = u_{112} \). Equations (8) capture how \( i, l, \) and \( k \) increment and help build the source vector, \( \tilde{q} \), and boundary condition vector, \( \tilde{b} \).

For a domain separated into \( n \) subintervals, the source function and boundary function vectors have length of \( (n-1)^3 \). The \( r \)th entry of both vectors can be mapped back to a corresponding \( u_{i,l,k} \) for some point in our domain using equations (8). For example, in a cube domain with \( n = 4 \) subintervals, the eighth entry \( v_8 = u_{i,l,k} \), where \( i = 2, l = 3, \) and \( k = 1 \).

\[
\begin{align*}
  i &= (8 - 1) \left( \text{mod} \ (4 - 1) \right) + 1 = 2 \\
  l &= \left\lfloor \frac{8 - 1}{4 - 1} \right\rfloor \left( \text{mod} \ (4 - 1) \right) + 1 = 3 \\
  k &= \left\lfloor \frac{8 - 1}{(4 - 1)^2} \right\rfloor + 1 = 1
\end{align*}
\]

Thus, \( v_8 = u_{231} \). Then, the eighth entry of our source function vector holds \( Q(x_2, y_3, z_1) \).

Note the boundary conditions we must consider where \( v_8 = u_{231} \). Note that \( u(x_2, y_3, z_1 - \Delta z) = u(x_2, y_3, z_0) = h_1(x_2, y_3). \) \( u_{231} \) lies directly above the \( z = 0 \) boundary, so it relies on the boundary condition \( h_1(x_2, y_3). \) Similarly, \( u(x_2, y_3 + \Delta y, z_1) = u(x_2, y_4, z_1) = f_2(x_2, z_1). \) Thus, the eighth entry in the boundary condition vector holds \( f_2(x_2, z_1) + h_1(x_2, y_3). \) The pattern of the boundary condition vector depends on the location of solutions \( v_r = u_{i,l,k} \) in the region and their relative position to the six boundaries. Points \( u_{i,l,k} \) are related to boundary conditions where \( i, k, \) or \( l \) are 1 or \( n - 1 \), as these points are one increment away from a boundary.

Let us consider more generally how the boundary condition functions appear in \( \tilde{b} \). Note that \( \tilde{b} \) is a block vector of block vectors. \( \tilde{b} \) is a vector of length \( (n-1) \) of block vectors of length \( (n-1) \) with entries of vectors of length \( (n-1) \). \( h_1(x, y) \) appears in the first block vector, or the first \( (n-1)^2 \) points. Similarly, \( h_2(x, y) \) appears in the last block vector or the last \( (n-1)^2 \) points. Within each block vector, the first subvector contains all instances of \( f_1(x, z) \) while the last subvector contains all instances of \( f_2(x, z). \) In each subvector, \( g_1(y, z) \) appears in the first entry, while \( g_2(y, z) \) appears in the last entry.

Consider how we can construct the boundary condition vector for any number of subintervals \( n \):

1. \( \tilde{b} \) is a block vector of length \( (n-1) \). Entries in this vector are also block vectors of length \( (n-1) \) that contain vectors of length \( (n-1) \).

2. \( h_1(x_i, y_j) \) appears in the first block vector in \( \tilde{b} \). This occurs when \( v_r = u_{i11} \) or where \( k = \left\lfloor \frac{r - 1}{(n-1)^2} \right\rfloor + 1 = 1 \). Note that inside our cube, these points are directly above the
boundary where \( z = 0 \). \( h_2(x_i, y_l) \) appears in the last block vector. This occurs when \( v_r = u_{i1l(n-1)} \), or when \( k = \left\lfloor \frac{r-1}{(n-1)^2} \right\rfloor + 1 = n - 1 \). These points lie at the top of our cube, where \( z = W \).

3. Now consider the subvectors inside each block vector in \( \vec{b} \). In each block vector, \( f_1(x, z) \) appears in the first subvector. This occurs when \( v_r = u_{i1k} \) or where \( l = \left\lfloor \frac{r-1}{n-1} \right\rfloor \mod (n-1) + 1 = 1 \). These points lie directly next to the \( y = 0 \) boundary. Similarly, \( f_2(x, z) \) appears in the last subvector within each block vector. This occurs when \( v = u_{i(n-1)k} \), or where \( l = \left\lfloor \frac{r-1}{n-1} \right\rfloor \mod (n-1) + 1 = n - 1 \). These points lie beside the \( y = H \) boundary.

4. Similarly, consider the subvectors inside the block vectors of \( \vec{b} \). In each subvector, \( g_1(y, z) \) appears in the first entry. This occurs for all solutions \( v_r = u_{1lk} \) or when \( i = (r - 1) \mod (n-1) + 1 = 1 \). These points lie directly next to the \( x = 0 \) boundary condition. Similarly, \( g_2(y, z) \) appears in the last entry of each subvector. Thus occurs when \( v_r = u_{n1lk} \) or when \( i = (r - 1) \mod (n-1) + 1 = n - 1 \). In our cube, these points lie directly next to the \( x = L \) boundary.
The general boundary condition vector is depicted below.

\[
\begin{pmatrix}
    f_1(x_1, z_1) + g_1(y_1, z_1) + h_1(x_1, y_1) \\
    f_1(x_2, z_1) + h_1(x_2, y_1) \\
    \vdots \\
    f_1(x_{n-1}, z_1) + g_2(y_1, z_1) + h_1(x_{n-1}, y_1) \\
    \vdots \\
    g_1(y_1, z_1) + h_1(x_1, y_1) \\
    \vdots \\
    g_2(y_1, z_1) + h_1(x_{n-1}, y_1) \\
    \vdots \\
    f_2(x_1, z_1) + g_1(y_{n-1}, z_1) + h_1(x_1, y_{n-1}) \\
    f_2(x_2, z_1) + h_1(x_2, y_{n-1}) \\
    \vdots \\
    f_2(x_{n-1}, z_1) + g_2(y_{n-1}, z_1) + h_1(x_{n-1}, y_{n-1}) \\
    \vdots \\
    f_1(x_1, z_{n-1}) + g_1(y_1, z_{n-1}) + h_2(x_1, y_1) \\
    f_1(x_2, z_{n-1}) + h_2(x_2, y_1) \\
    \vdots \\
    f_1(x_{n-1}, z_{n-1}) + g_2(y_1, z_{n-1}) + h_2(x_{n-1}, y_1) \\
    \vdots \\
    g_1(y_1, z_{n-1}) + h_2(x_1, y_1) \\
    \vdots \\
    g_2(y_1, z_{n-1}) + h_2(x_{n-1}, y_1) \\
    \vdots \\
    f_2(x_1, z_{n-1}) + g_1(y_{n-1}, z_{n-1}) + h_2(x_1, y_{n-1}) \\
    f_2(x_2, z_{n-1}) + h_2(x_2, y_{n-1}) \\
    \vdots \\
    f_2(x_{n-1}, z_{n-1}) + g_2(y_{n-1}, z_{n-1}) + h_2(x_{n-1}, y_{n-1}) \\
    \vdots \\
    \end{pmatrix}
\]

\[\vec{b} = \begin{pmatrix} f_2(x_1, y_1) + g_1(y_{n-1}, y_1) + h_2(x_1, y_{n-1}) \\
    f_2(x_2, y_1) + h_2(x_2, y_{n-1}) \\
    \vdots \\
    f_2(x_{n-1}, y_1) + g_2(y_{n-1}, y_1) + h_2(x_{n-1}, y_{n-1}) \end{pmatrix} \]

4 Rectangular Prism Domains

Now, let us consider a more general case, where \( L = H = W \) is not necessarily true. In order to keep \( \Delta x = \Delta y = \Delta z \), we will have a different number of subintervals on the \( x, y, \) and \( z \) axes denoted as \( n_x = \frac{L}{\Delta x}, n_y = \frac{H}{\Delta y}, \) and \( n_z = \frac{W}{\Delta z} \). Let us consider two cases, alternating the values for \( n_x, n_y, \) and \( n_z \) to compare the shape of \( A, \vec{b}, \) and \( \vec{q} \).

1. Let \( L = 3, H = 4, W = 5, \) and \( \Delta x = \Delta y = \Delta z = 1. \) Then, \( n_x = \frac{L}{\Delta x} = 3, \) so the length of
our rectangular prism is separated into 3 subintervals. Similarly, \( n_y = 4 \) and \( n_z = 5 \). Here we have \( (n_x - 1)(n_y - 1)(n_z - 1) = 24 \) internal points of our domain.

2. Now, consider a very similar case, where \( L = 5 \), \( H = 3 \), and \( W = 4 \), and \( \Delta x = \Delta y = \Delta z = 1 \). Now, \( n_x = 5 \), \( n_y = 3 \), and \( n_z = 4 \).

We will identify how different subintervals along the \( x \), \( y \), and \( z \) axes affect the pattern in the matrix and boundary condition vector.

### 4.1 A Matrix Pattern

**Matrix 1.:** \( n_x = 3 \), \( n_y = 4 \), and \( n_z = 5 \)

Note that \( A_{3,4,5} \) is a \( 4 \times 4 = (n_z - 1) \times (n_z - 1) \) block matrix of \( 3 \times 3 = (n_y - 1) \times (n_y - 1) \) matrices with \( 2 \times 2 = (n_x - 1) \times (n_x - 1) \) submatrices as entries. Using a similar definition for \( A' \) and \( I \) as in the cube domain case, we find that \( A_{3,4,5} \) can be simplified as

\[
A_{3,4,5} = \begin{bmatrix} A' & I & 0 & 0 \\ I & A' & I & 0 \\ 0 & I & A' & I \\ 0 & 0 & I & A' \end{bmatrix}.
\]

**Matrix 2.:** \( n_x = 5 \), \( n_y = 3 \), and \( n_z = 4 \)

Note that \( A_{5,3,4} \) is a \( 3 \times 3 = (n_z - 1) \times (n_z - 1) \) block matrix of \( 2 \times 2 = (n_y - 1) \times (n_y - 1) \) matrices with \( 4 \times 4 = (n_x - 1) \times (n_x - 1) \) submatrices as entries. \( A_{5,3,4} \) can be simplified as

\[
A_{5,3,4} = \begin{bmatrix} A' & I & 0 \\ I & A' & I \\ 0 & I & A' \end{bmatrix}.
\]

Let us identify the pattern as \( n_x \), \( n_y \), and \( n_z \) change to form a general algorithm for constructing \( A \).

1. \( A \) is a \( (n_z - 1) \times (n_z - 1) \) block matrix of \( (n_y - 1) \times (n_y - 1) \) matrices that have entries of \( (n_x - 1) \times (n_x - 1) \) matrices.

2. Define a notable \( (n_x - 1) \times (n_x - 1) \) submatrix, \( A'' \), with -6 on the main diagonal and 1 on either side.

\[
A'' = \begin{bmatrix} -6 & 1 & 0 & \cdots & 0 \\ 1 & -6 & 1 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 1 & -6 & 1 \\ 0 & \cdots & 0 & 1 & -6 \end{bmatrix}
\]
3. Now define $A'$ as a $(ny - 1) \times (ny - 1)$ block matrix comprised of $(nx - 1) \times (nx - 1) \times (nx - 1)$ submatrices. $A''$ appears on the main diagonal of $A'$. On either side of $A''$ is the $(nx - 1) \times (nx - 1)$ identity matrix, $I'$. Here, 0 represents the $(nx - 1) \times (nx - 1)$ zero matrix.

$$A' = \begin{bmatrix} A'' & I' & 0 & \cdots & 0 \\ I' & A'' & I' & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & I' & A'' & I' \\ 0 & \cdots & 0 & I' & A'' \end{bmatrix}$$

4. Define $I$ to be a $(ny - 1) \times (ny - 1)$ block matrix with $I'$ along the main diagonal, and all other entries the $(nx - 1) \times (nx - 1)$ zero matrix.

$$I = \begin{bmatrix} I' & 0 & 0 & \cdots & 0 \\ 0 & I' & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & I' & 0 \\ 0 & \cdots & 0 & 0 & I' \end{bmatrix}$$

5. Let $A$ be a $(nz - 1) \times (nz - 1)$ block matrix with $A'$ across the diagonal. $I$ lies on either side of the main diagonal. Here, 0 represents an $(ny - 1) \times (ny - 1)$ block matrix of $(nx - 1) \times (nx - 1)$ zero matrices.

$$A = \begin{bmatrix} A' & I & 0 & \cdots & 0 \\ I & A' & I' & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & I & A' & I \\ 0 & \cdots & 0 & I & A' \end{bmatrix}$$

4.2 Boundary Condition Vector Pattern

First, we must consider how we iterate through a rectangular prism. For every $v_r = u_{iilk} = u(x_i, y_l, z_k)$, we can find $i$, $l$, and $k$ as:

$$i = (r - 1) \pmod{(nx - 1)} + 1,$$
$$l = \left\lfloor \frac{r - 1}{nx - 1} \right\rfloor \pmod{(ny - 1)} + 1,$$
$$k = \left\lfloor \frac{r - 1}{(ny - 1)(nx - 1)} \right\rfloor + 1.$$
Given the two cases above, consider the corresponding boundary condition vectors:

\[
\begin{align*}
 b_1 &= \begin{cases}
 f_1(x_1, z_1) + g_1(y_1, z_1) + h_1(x_1, y_1) \\
 f_1(x_2, z_1) + g_2(y_1, z_1) + h_1(x_2, y_1) \\
 g_1(y_2, z_1) + h_1(x_1, y_2) \\
 g_2(y_2, z_1) + h_1(x_2, y_2) \\
 f_2(x_1, z_1) + g_1(y_3, z_1) + h_1(x_1, y_3) \\
 f_2(x_2, z_1) + g_2(y_3, z_1) + h_1(x_2, y_3) \\
 f_1(x_1, z_2) + g_1(y_1, z_2) \\
 f_1(x_2, z_2) + g_2(y_1, z_2) \\
 g_1(y_2, z_2) \\
 g_2(y_2, z_2) \\
 f_2(x_1, z_2) + g_1(y_3, z_2) \\
 f_2(x_2, z_2) + g_2(y_3, z_2) \\
 f_1(x_1, z_3) + g_1(y_1, z_3) \\
 f_1(x_2, z_3) + g_2(y_1, z_3) \\
 g_1(y_2, z_3) \\
 g_2(y_2, z_3) \\
 f_2(x_1, z_3) + g_1(y_3, z_3) \\
 f_2(x_2, z_3) + g_2(y_3, z_3) \\
 f_1(x_1, z_4) + g_1(y_1, z_4) + h_1(x_1, y_1) \\
 f_1(x_2, z_4) + g_2(y_1, z_4) + h_1(x_2, y_1) \\
 g_1(y_2, z_4) + h_2(x_1, y_2) \\
 g_2(y_2, z_4) + h_2(x_2, y_2) \\
 f_2(x_1, z_4) + g_1(y_3, z_4) + h_2(x_1, y_3) \\
 f_2(x_2, z_4) + g_2(y_3, z_4) + h_2(x_2, y_3)
\end{cases}
\]

\[
\begin{cases}
 f_1(x_1, z_1) + g_1(y_1, z_1) + h_1(x_1, y_1) \\
 f_1(x_2, z_1) + h_1(x_2, y_1) \\
 f_1(x_3, z_1) + h_1(x_3, y_1) \\
 f_1(x_4, z_1) + g_2(y_1, z_1) + h_1(x_4, y_1) \\
 f_2(x_1, z_1) + g_1(y_2, z_1) + h_1(x_1, y_2) \\
 f_2(x_2, z_1) + h_1(x_2, y_2) \\
 f_2(x_3, z_1) + h_1(x_3, y_2) \\
 f_2(x_4, z_1) + g_2(y_2, z_1) + h_1(x_4, y_2)
\end{cases}
\]

\[
\begin{align*}
 b_2 &= \begin{cases}
 f_1(x_1, z_2) + g_1(y_1, z_2) \\
 f_1(x_2, z_2) + g_1(y_1, z_2) \\
 f_1(x_3, z_2) \\
 f_1(x_4, z_2) + g_2(y_1, z_2) \\
 f_2(x_1, z_2) + g_1(y_2, z_2) \\
 f_2(x_2, z_2) \\
 f_2(x_3, z_2) \\
 f_2(x_4, z_2) + g_2(y_2, z_2) \\
 f_1(x_1, z_3) + g_1(y_1, z_3) + h_2(x_1, y_1) \\
 f_1(x_2, z_3) + h_2(x_2, y_1) \\
 f_1(x_3, z_3) + h_2(x_3, y_1) \\
 f_1(x_4, z_3) + g_2(y_1, z_3) + h_2(x_4, y_1) \\
 f_2(x_1, z_3) + g_1(y_2, z_3) + h_2(x_1, y_2) \\
 f_2(x_2, z_3) + h_2(x_2, y_2) \\
 f_2(x_3, z_3) + h_2(x_3, y_2) \\
 f_2(x_4, z_3) + g_2(y_2, z_3) + h_2(x_4, y_2)
\end{cases}
\]

Let us compare the pattern in \( \vec{b}_1 \) and \( \vec{b}_2 \). Note that \( \vec{b}_1 \) is a block vector of length \( 4 = (n_z - 1) \) that contains block vectors of length \( 3 = (n_y - 1) \). This block subvector holds subvectors of length \( 2 = (n_x - 1) \). On the other hand \( \vec{b}_2 \) is a block vector of length \( 3 = (n_z - 1) \) that contains block subvectors of length \( 2 = (n_y - 1) \). These block subvectors hold subvectors of length \( 4 = (n_x - 1) \). Within both vectors, \( \vec{b}_1 \) and \( \vec{b}_2 \), we see the pattern from our cube domain examples. \( h_1(x, y) \) appears in the first block vector, or the first \((n_x - 1)(n_y - 1)\) points, while \( h_2(x, y) \) appears in the last subvector, or the last \((n_x - 1)(n_y - 1)\) points. In each of these block vectors, \( f_1(x, z) \) appears in the first subvector, while \( f_2(x, z) \) appears in the last subvector. \( g_1(y, z) \) appears in the first entry of each subvector, while \( g_2(y, z) \) appears in the last entry.

The algorithm for constructing \( \vec{b} \) for any rectangular prism domain:

1. The boundary condition vector \( \vec{b} \) has length \((n_x - 1)(n_y - 1)(n_z - 1)\). This vector can be broken into \((n_z - 1)\) block vectors of length \((n_y - 1)\) that hold subvectors of length \((n_x - 1)\).
2. $h_1(x_i, y_j)$ appears in the first block vector in $\vec{b}$. This occurs when $v_r = u_{i1}$ or where $k = \left\lfloor \frac{r-1}{(n_y-1)(n_x-1)} \right\rfloor + 1 = 1$. Note that inside our cube, these points are directly above the $z = 0$ boundary. $h_2(x_i, y_j)$ appears in the last block vector. This occurs when $v_r = u_{i(n-1)}$, or when $k = \left\lfloor \frac{r-1}{(n_y-1)(n_x-1)} \right\rfloor + 1 = n_z - 1$. These points lie at the edge of our cube where $z = W$.

3. Now consider the subvectors inside each block vector in $\vec{b}$. In each block vector, $f_1(x, z)$ appears in the first subvector. This occurs when $v_r = u_{1k}$ or where $l = \left\lfloor \frac{r-1}{n_x-1} \right\rfloor \mod (n_y - 1) + 1 = 1$. These points lie directly next to the boundary condition where $y = 0$. Similarly, $f_2(x, z)$ appears in the last subvector within each block vector. This occurs when $v_r = u_{i(n_y-1)k}$, or when $l = \left\lfloor \frac{r-1}{n_x-1} \right\rfloor \mod (n_y - 1) + 1 = n_y - 1$. These points lie beside the $y = H$ boundary.

4. Similarly, consider the subvectors inside the block vectors of $\vec{b}$. In each subvector, $g_1(y, z)$ appears in the first entry. This occurs for all solutions $v_r = u_{1k}$ or when $i = (r-1) \mod (n_x - 1) + 1 = 1$. These points lie directly next to the $x = 0$ boundary. Similarly, $g_2(y, z)$ appears in the last entry of each subvector. This occurs when $v_r = u_{i(n_y-1)k}$, or when $i = (r-1) \mod (n_x - 1) + 1 = n_x - 1$. In our cube, these points lie directly next to the $x = L$ boundary.

5 3D Program

Using the algorithms found above, a program in Matlab was created to approximate solutions of Poisson’s equation in three dimensions. In order to run the program, one would need the following input:

- L, the length of our region along the x-axis.
- H, the height of our region along the y-axis.
- W, the width of our region along the z-axis.
- $\Delta x = \Delta y = \Delta z$, the size of the subintervals. This will dictate the number of subintervals along each axis. In order for the subintervals on each axes to have consistent sizes, $\Delta x$ should evenly divide L, H, and W.
- Q(x, y, z), source function through our region.
- $f_1(x, z)$, $f_2(x, z)$, $g_1(y, z)$, $g_2(y, z)$, $h_1(x, y)$, and $h_2(x, y)$, the boundary conditions of our region.

The program uses the patterns defined above to construct $A$, $\vec{b}$, and $\vec{q}$. Then, it solves for the approximate solutions in $\vec{v}$. To test the results of the program, one can compare the graph of the approximate solution with the exact solution at different $z$ values.
5.1 Examples

Example 5.1. Let our region be a $4 \times 4 \times 4$ cube so $H = L = W = 4$. Let $\Delta x = \Delta y = \Delta x = 0.25$, so that the $x$, $y$, and $z$ axes are separated into $n = 16$ subintervals, respectively. We will approximate the function

$$u(x, y, z) = 4 \cos(x^2 + z^2) + \sin(yz)$$

We can find the source function by finding the second derivatives of $u(x, y, z)$,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = Q(x, y, z)$$

$$Q(x, y, z) = -16 \sin(x^2 + z^2) - 16x^2 \cos(x^2 + z^2) - 16z^2 \cos(x^2 + z^2)$$

$$-y^2 \sin(yz) - z^2 \sin(yz)$$

We can find the six boundary conditions by solving $u(x, y, z)$ where $x = 0$, $x = L = 4$, $y = 0$, $y = H = 4$, $z = 0$, and $z = W = 4$.

$$u(x, 0, z) = f_1(x, z) = 4 \cos(x^2 + z^2)$$

$$u(x, 4, z) = f_2(x, z) = 4 \cos(x^2 + z^2) + \sin(4z)$$

$$u(0, y, z) = g_1(y, z) = 4 \cos(z^2) + \sin(yz)$$

$$u(4, y, z) = g_2(y, z) = 4 \cos(16 + z^2) + \sin(yz)$$

$$u(x, y, 0) = h_1(x, y) = 4 \cos(x^2)$$

$$u(x, y, 4) = h_2(x, y) = 4 \cos(x^2 + 16) + \sin(4y)$$

To test the accuracy of the results, compare the approximate and exact solutions at different values of $z$. 

(a) Approximation at $z = 0.25$ 
(b) Actual Solution at $z = 0.25$
Example 5.2. Let us approximate a function \( u(x, y, z) \) in a \( 3 \times 3 \times 3 \) cube, so that \( 0 \leq x \leq 3 \), \( 0 \leq y \leq 3 \), and \( 0 \leq z \leq 3 \).

\[
u(x, y, z) = \cos(x^2) + \sin(y) \cos(z) + \sin(y^2)\]

We can find the source function and boundary conditions as:

\[
Q(x, y, z) = 2\cos(y^2) - 2\sin(x^2) - 2\cos(z)\sin(y) - 4x^2 \cos(x^2) - 4y^2 \sin(y^2)
\]
First, compare the numerical approximation to the exact solution. Note the animations given in the supplementary files. The program approximated $u(x, y, z)$ with $n = 24$ subintervals. The animation depicts the approximate function, $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, by iterating through the various widths, the 24 $z$ values. We can compare the approximation to the animation of the exact solution.

To show that these approximate solutions get closer to the actual solutions as the number of subintervals increase, consider approximations with subintervals $n = 4, 8, 16, 24$ at a stationary $z = 1.5$.  

\begin{align*}
  f_1(x, z) &= \cos(x^2) \\
  f_2(x, z) &= \cos(x^2) + \sin(3) \cos(z) + \sin(9) \\
  g_1(y, z) &= 1 + \sin(y) \cos(z) + \sin(y^2) \\
  g_2(y, z) &= \cos(9) + \sin(y) \cos(z) + \sin(y^2) \\
  h_1(x, y) &= \cos(x^2) + \sin(y) + \sin(y^2) \\
  h_2(x, y) &= \cos(x^2) + \sin(y) \cos(3) + \sin(y^2)
\end{align*}
This example also shows that using too few subintervals can have inaccurate and unpredictable approximations, especially if the function has multiple direction changes. For instance, where \( n = 4 \), the graph has a drastic, unexpected minimum. Ideally, the program could be run with an incredibly large number of subintervals, but solving these linear systems becomes computationally expensive. In this example, where \( n = 24 \), the corresponding system has \( (n - 1)^3 = 12,167 \) equations. Alternately, other numerical methods could be used to approximate the linear system, so that a larger number of subintervals is possible.

Let us now approximate this function in a rectangular prism domain where \( L = 2 \), \( H = 5 \), and \( W = 3.5 \). We can choose \( \Delta x = \Delta y = \Delta z = 0.25 \), so that \( n_x = 8 \), \( n_y = 20 \), and \( n_z = 14 \).

The boundary conditions with this new region are given as

\[
\begin{align*}
  f_1(x, z) &= \cos(x^2) \\
  g_1(y, z) &= 1 + \sin(y) \cos(z) + \sin(y^2) \\
  h_1(x, y) &= \cos(x^2) + \sin(y) + \sin(y^2)
\end{align*}
\]

\[
\begin{align*}
  f_2(x, z) &= \cos(x^2) + \sin(5) \cos(z) + \sin(25) \\
  g_2(y, z) &= \cos(4) + \sin(y) \cos(z) + \sin(y^2) \\
  h_2(x, y) &= \cos(x^2) + \sin(y) \cos(3.5) + \sin(y^2).
\end{align*}
\]

Now, we can compare the numerical approximation with the exact solution at different values of \( z \).
Algorithms to Approximate Solutions of Poisson’s Equation

6 Conclusion

Although actual solutions to Poisson’s equation are difficult to find, we have shown that numerical methods can be used to accurately approximate these mathematical models. We expanded Poisson’s equation with a finite difference approximation and created a system of equations to approximate solutions at internal points of our region. We found patterns in the A matrix, boundary condition vector, and source vector by solving for solutions with a small number of subintervals. Based on these patterns, we were able to write an algorithm to construct the matrices and vectors with a much larger number of subintervals along the x, y, and z axes. With these algorithms, a program was written in Matlab to approximate solutions of Poisson’s equation in three dimensions. Through many examples, we have shown that the program can accurately approximate the function given the necessary source and boundary condition functions and that these approximations get closer to the exact solution as the number of subintervals increase.

References


Ray Dambrose  
University of Mary Washington  
rsdambrose@gmail.com