Programming a Linear Algebra Solution to the Poisson Equation

Elizabeth Drueke

February 9, 2016

Abstract

Poisson's equation comes into play frequently in physical situations. In every field of physics, from mechanics to electromagnetism, we are forced to model physical systems subject to boundary conditions with this versatile equation. In this report, we analyze not the applications of the Poisson equation, but how to apply it, specifically subject to the Dirichlet boundary conditions, using a C++ computer program.

1 Introduction

It is vital in physics to develop ways to deal with large quantities of data and to use that data to make close approximations of physical conditions. In particular, we wish to develop computer programs which can both automate this process and complete it in a timely manner. To this end, we investigate a linear algebra solution to the Poisson equation, given by

$$-u''(x) = f(x), \tag{1.1}$$

subject to the Dirichlet boundary conditions,

$$u(0) = u(1) = 0. \tag{1.2}$$

2 Theory

The mathematics behind the solution to the Poisson equation presented here is mathematically rich in approximations. From Eq. 1.1, we can see that we are required to compute the second derivative of $u(x)$. To do this, we note that we can always approximate the first derivative as

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \tag{2.1}$$

for $h << 1$ because the derivative is the slope of the line tangent to f at that point. This then implies that

$$
\begin{aligned}
f''(x) &\approx \frac{f'(x+h) - f'(x-h)}{2h} \\
&\approx \frac{\frac{f(x+h+h)-f(x+h-h)}{2h} - \frac{f(x-h+h)-f(x-h-h)}{2h}}{2h} \\
&\approx \frac{\frac{f(x+2h)-f(x)}{2h} - \frac{f(x)-f(x-2h)}{2h}}{2h} \\
&\approx \frac{f(x+2h) - 2f(x) + f(x-2h)}{(2h)^2}.
\end{aligned} \tag{2.2}
$$

Letting $2h \to h$, we then have

$$
f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{2.3}
$$

Eq. 2.3 is what we will use as our approximation to the second derivative throughout.

Now, we have an expression which lends itself to the creation of vectors. Letting

$$
h = \frac{1}{n+1}
$$

for some $n \in \mathbb{N}$, we see that we can treat this as a step partition of the interval $[0, 1]$, on which the Dirichlet conditions are valid. Thus, we define each x_i in the partition as

$$
x_i = ih, i = 0, \ldots, n+1.
$$

From these x_i we can create a vector $\mathbf{x}$ [1]

$$
\mathbf{x} = \left(0, \frac{1}{n+1}, \frac{2}{n+1}, \ldots, \frac{n}{n+1}, 1\right)^T \tag{2.4}
$$

Then, let $\mathbf{b}$ be a vector of the function $f(x)$ evaluated at the points in $\mathbf{x}$. That is,

$$
\begin{aligned}
\mathbf{b} &= \left(f(0), f\left(\frac{1}{n+1}\right), f\left(\frac{2}{n+1}\right), \ldots, f\left(\frac{n}{n+1}\right), f(1)\right)^T \\
&= \left(0, f\left(\frac{1}{n+1}\right), f\left(\frac{2}{n+1}\right), \ldots, f\left(\frac{n}{n+1}\right), 0\right)^T.
\end{aligned} \tag{2.5}
$$

Now, we see that Eq. 2.3 lends itself nicely to the introduction of a linear algebra problem. In particular, we can define an $n \times n$ matrix A such that

$$
A = \frac{-1}{h^2} \begin{pmatrix}
2 & -1 & 0 & \cdots & \cdots & 0 \\
-1 & 2 & -1 & 0 & \cdots & 0 \\
0 & -1 & 2 & -1 & 0 & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & -1 & 2
\end{pmatrix} \tag{2.6}
$$

[1] Note that throughout we indicate vectors as bold, lowercase letters (eg. $\mathbf{x}$), and matrices as uppercase letters (eg. A).

Then, we have that the Poisson Equation given by Eq. 1.1 can be approximated as

$$A\mathbf{v} = \mathbf{b} \tag{2.7}$$

for some $\mathbf{v}$ which represents $f''(x)$ at various values of x. And so we have a system of n equations in n unknowns.

In general, there are two main ways in which we might solve Eq. 2.7, known as Gaussian elimination and LU Decomposition. We will discuss these in general in Section 2.1 and Section 2.2, respectively, before discussing how we dealt with our particular matrix A given by Eq. 2.6 in Section 3.

2.1 Gaussian Elimination

Gaussian elimination is the method of solving linear systems typically taught in a linear algebra class. In particular, this method involves adding multiples of rows to other rows in order to eliminate (or set to zero) off-diagonal elements. In most situations, it is necessary to employ both a forward and backward Gaussian elimination method in order to solve a full system. In order to demonstrate this method, we will provide an example here. The algorithm used will be more explicitly discussed in Section 3.

Suppose we have a matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{12} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{2.8}$$

and we wish to reduce it to row echelon form. We might do this using Gaussian elimination. To begin, we would use forward elimination to set the a_{i1} components to zero for $i \neq 1$. Letting R_i denote the i^{th} row, we notice that letting

$$\begin{aligned} R_2 &= R_2 - \frac{a_{21}}{a_{11}} R_1 \\ R_3 &= R_3 - \frac{a_{31}}{a_{11}} R_1 \end{aligned} \tag{2.9}$$

ought to do the trick. That is, we have

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} & a_{12} & a_{12} \\ 0 & a_{22} - \frac{a_{21}a_{12}}{a_{11}} & a_{23} - \frac{a_{21}a_{13}}{a_{11}} \\ 0 & a_{32} - \frac{a_{31}a_{12}}{a_{11}} & a_{33} - \frac{a_{31}a_{13}}{a_{11}} \end{pmatrix}. \tag{2.10}$$

Continuing in this manner, we will eventually have a matrix of the form

$$A\prime = \begin{pmatrix} a_{11}\prime & a_{12}\prime & a_{13}\prime \\ 0 & a_{22}\prime & a_{23}\prime \\ 0 & 0 & a_{33}\prime \end{pmatrix}. \tag{2.11}$$

At this point, we may begin the backward Gaussian elimination. In the forward process, we were able to set all of the $a_{ij} = 0$ for $i > j$. In the backward process, we wish to do the same for the a_{ij} with $i < j$. In the end, we should have a pure diagonal matrix.

To begin the backward process, we note that we can set the a_{i3} elements to zero by similar calculations as those performed in Eq. 2.9. In particular, we can let

$$\begin{aligned} R_2 &= R_2 - \frac{a_{23}}{a_{33}} R_3 \\ R_1 &= R_1 - \frac{a_{13}}{a_{33}} R_3 \end{aligned} \tag{2.12}$$

Doing this, we see

$$\begin{pmatrix} a_{11}\prime & a_{12}\prime & a_{13}\prime \\ 0 & a_{22}\prime & a_{23}\prime \\ 0 & 0 & a_{33}\prime \end{pmatrix} \rightarrow \begin{pmatrix} a_{11}\prime & a_{12}\prime & 0 \\ 0 & a_{22}\prime & 0 \\ 0 & 0 & a_{33}\prime \end{pmatrix}. \tag{2.13}$$

Continuing in this manner, we can see that we will eventually come across a pure diagonal matrix. From here, having performed the row operations on the solution vector $\mathbf{b}$ as well as the matrix A, we can find our solution $\mathbf{x}$ by noting that

$$x_i = \frac{\widetilde{b_i}}{\widetilde{a_{ii}}}, \tag{2.14}$$

where the $\sim$ indicates that this is the component of the Gaussian eliminated matrix/vector. This procedure is easily generalized to an $n \times n$ matrix.

One downside to this method of solving the system of linear equations is that whatever row operations are performed on A in Eq. 2.7 must also be performed on $\mathbf{b}$. This means that the process must be repeated every time the solution vector is changed. This restriction can be a serious time constraint on any program written to implement Gaussian elimination in order to solve systems of linear equations. The time limitations of such an algorithm are discussed in Section 3.

2.2 LU Decomposition

The second major linear system solving method is what is known as LU decomposition. In this procedure, we decompose our A matrix into a lower-triangular L and an upper-triangular U of the form

$$\begin{array}{ccccc} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} & = & \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} & \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \\ A & = & L & U \end{array} \tag{2.15}$$

In contrast with the Gaussian elimination method, this method does not need to be repeated for every choice of solution vector $\mathbf{b}$. Instead, once L and U have been computed, they can be used to determine the solution $\mathbf{x}$ for any solution vector $\mathbf{b}$.

As with the Gaussian elimination method, we present an example as an illustration of how the LU decomposition method works. Assume we have some

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}. \tag{2.16}$$

Based on the multiplication as shown in Eq. 2.15, we can see that we can directly solve for the l_{ij} and u_{ij}. In particular, in order, we see[2]

$$\begin{aligned}
a_{11} =& \mathbf{u_{11}} && \Rightarrow \\
a_{21} =& \mathbf{l_{21}} u_{11} && \Rightarrow \\
a_{31} =& \mathbf{l_{31}} u_{11} && \Rightarrow \\
a_{41} =& \mathbf{l_{41}} u_{11} && \Rightarrow \\
a_{12} =& \mathbf{u_{12}} && \Rightarrow \\
a_{22} =& \mathbf{l_{21}} u_{12} + u_{22} && \Rightarrow \\
a_{32} =& l_{31} u_{12} + \mathbf{l_{32}} u_{22} && \Rightarrow \\
a_{42} =& l_{41} u_{12} + \mathbf{l_{42}} u_{22} && \Rightarrow \\
a_{13} =& \mathbf{u_{13}} && \Rightarrow \\
a_{23} =& l_{21} u_{13} + \mathbf{u_{23}} && \Rightarrow \\
a_{33} =& l_{31} u_{13} + l_{32} u_{23} + \mathbf{u_{33}} && \Rightarrow \\
a_{43} =& l_{42} u_{23} + \mathbf{l_{43}} u_{33} + l_{41} u_{13} && \Rightarrow \\
a_{14} =& \mathbf{u_{14}} && \Rightarrow \\
a_{24} =& l_{21} u_{14} + \mathbf{u_{24}} && \Rightarrow \\
a_{34} =& l_{31} u_{14} + l_{32} u_{24} + \mathbf{u_{34}} && \Rightarrow \\
a_{44} =& l_{41} u_{14} + l_{42} u_{24} + l_{43} u_{34} + \mathbf{u_{44}}.
\end{aligned} \tag{2.17}$$

This result generalizes **more**.

3 The Algorithm

4 Results and Benchmarks

5 Conclusions

Something about how the class isn't complete actually.

[2]Here, the boldface text indicates the unknown variable in each equation.