The Finite Difference Method for Elliptic Problems

Varun Shankar

February 19, 2016

1 Introduction

Previously, we saw the derivation of various methods from the MWR. Today, we will begin our study of the finite difference (FD) method. The best way to do so is by example. In this document, we will focus on 1D and 2D elliptic problems.

Specifically, we will focus on the Poisson equation with Dirichlet boundary conditions:

$$-\Delta u = f(\boldsymbol{x}), \boldsymbol{x} \in \Omega, \tag{1}$$
$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \boldsymbol{x} \in \partial\Omega. \tag{2}$$

We will also look at the Poisson equation with Neumann boundary conditions:

$$-\Delta u = f(\boldsymbol{x}), \boldsymbol{x} \in \Omega, \tag{3}$$
$$\frac{\partial u}{\partial n} = g(\boldsymbol{x}), \boldsymbol{x} \in \partial\Omega. \tag{4}$$

The Poisson equation describes many physical phenomena: distribution of charges under electrostatic forces, steady state for heat diffusion with a source, etc.

2 FD for the Poisson Equation with Dirchlet BCs

2.1 The 1D Poisson Equation

The 1D Poisson equation is a boundary-value ODE problem. However, it encapsulates many of the principles involved in solving the Poisson equation in 2D

(and higher), and so is worthy of our attention. The equation in 1D is:

$$\frac{d^2u}{dx^2} = f(x), x \in \Omega, \tag{5}$$
$$u(a) = \alpha, \tag{6}$$
$$u(b) = \beta. \tag{7}$$

We will take $[a, b] = [0, 1]$, the unit square.

To discretize the 1D Poisson equation with FD, we first need to set up a spatial grid. For this, we subdivide the interval $[0, 1]$ into $n + 1$subintervals, with $h = \frac{1}{n+1}$. We define $x_j = jh$, $U_j \approx u_j = u(x_j)$, $0 \leq j \leq n + 1$.

Let us take some point x_j in the interior of our grid, and ask ourselves how to approximate the Poisson equation at this point. We must now pick an FD scheme. For familiarity/convenience, let us select the second-order central difference approximation to the second derivative. Thus, we have

$$-\frac{U_{j+1} - 2U_j + U_{j-1}}{h^2} = f(x_j). \tag{8}$$

At the point x_0, we only have the equation $U_0 = \alpha 0$. Similarly, at the point x_{n+1}, we have $U_{n+1} = \beta$. Let us write out these equations for a few values of j:

$$U_0 = \alpha, \tag{9}$$
$$-U_2 + 2U_1 - U_0 = h^2 f(x_1), \tag{10}$$
$$-U_3 + 2U_2 - U_1 = h^2 f(x_2), \tag{11}$$
$$-U_4 + 2U_3 - U_2 = h^2 f(x_3), \tag{12}$$
$$\dots \tag{13}$$
$$-U_{n+1} + 2U_n - U_{n-1} = h^2 f(x_n), \tag{14}$$
$$U_{n+1} = \beta. \tag{15}$$

For ease of reading, we can carefully rewrite these equations:

$$U_0 + 0U_1 + 0U_2 + \ldots + 0U_{n+1} = \alpha, \tag{16}$$
$$-U_0 + 2U_1 - U_2 + 0U_3 + \ldots 0U_{n+1} = h^2 f(x_1), \tag{17}$$
$$0U_0 - U_1 + 2U_2 - U_3 + 0U_4 + \ldots + 0U_{n+1} = h^2 f(x_2), \tag{18}$$
$$\dots \tag{19}$$
$$0U_0 + 0U_1 + \ldots - U_{n-1} + 2U_n - U_{n+1} = h^2 f(x_n), \tag{20}$$
$$0U_0 + 0U_1 + \ldots + 0U_n + U_{n+1} = \beta. \tag{21}$$

We can now write this system of linear equations in matrix form:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & \ddots & \ddots & \ddots & 0 & \dots & 0 \\ & & & & \vdots & & & & \\ 0 & 0 & \dots & \dots & \dots & 0 & -1 & 2 & -1 \\ 0 & 0 & \dots & \dots & \dots & 0 & 0 & 0 & 1 \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ U_3 \\ \vdots \\ \vdots \\ U_n \\ U_{n+1} \end{bmatrix}}_{\tilde{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} \alpha \\ h^2 f(x_1) \\ h^2 f(x_2) \\ h^2 f(x_3) \\ \vdots \\ \vdots \\ h^2 f(x_n) \\ \beta \end{bmatrix}}_{\tilde{\mathbf{b}}} \tag{22}$$

Clearly, the first and last rows can be removed from the equation. Doing so removes the need to "solve" for U_0 and U_{n+1}. Further, we can use the known values of U_0 and U_{n+1} to alter the equations for U_1 and U_n:

$$2U_1 - U_2 = h^2 f(x_1) + \alpha, \tag{23}$$

$$2U_n - U_{n-1} = h^2 f(x_n) + \beta. \tag{24}$$

These transformations leave us the following linear system:

$$\underbrace{\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & \dots & \ddots & \ddots & \ddots & 0 & \dots \\ & & & \vdots & & & \\ 0 & \dots & \dots & \dots & 0 & -1 & 2 \end{bmatrix}}_{-L} \underbrace{\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ \vdots \\ U_n \end{bmatrix}}_{\boldsymbol{u}} = \underbrace{\begin{bmatrix} h^2 f(x_1) + \alpha \\ h^2 f(x_2) \\ h^2 f(x_3) \\ \vdots \\ \vdots \\ h^2 f(x_n) + \beta \end{bmatrix}}_{\boldsymbol{f}} \tag{25}$$

$-L$ is clearly symmetric. Let $\mathbf{v}$ be any vector (except the zero vector). Then, $\mathbf{v}^T(-L)\mathbf{v} = v_1^2 + \sum_{j=1}^{n}(v_j - v_{j-1})^2 + v_n^2$. This quantity is always positive, and is only zero if $\mathbf{v} = \mathbf{0}$. Thus, $-L$ is symmetric *and* positive-definite. L, of course, is symmetric negative-definite.

Consequently, $-L$ has positive real eigenvalues, and L has negative real eigenvalues. All FD discretizations of elliptic PDEs have this property.

$-L$ is tridiagonal! We can easily invert it in $O(n)$ operations.

2.2 The 2D Poisson Equation

We now look at our first real PDE: the 2D Poisson equation with Dirichlet BCs. As in the 1D case, we will discretize this with second-order centered finite

differences. Again, as in the 1D case, we will need a grid to do so. Since our domain is a 2D domain, we will need a 2D grid. Assume that the grid is uniform in both directions, with a spacing of h.

Let $x_i = ih$, $y_j = jh$, $i, j = 0, \ldots, n+1$. Let $U_{i,j} \approx u(x_i, y_j)$, and let $f_{i,j} = f(x_i, y_j)$. Discretizing the 2D Poisson equation, we have

$$-\frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2} - \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h^2} = f_{i,j}. \tag{26}$$

Looking at the (i, j) point, this formula shows that we use 5 points to approximate the Laplacian at that point. This is called a 5-point stencil (the same stencil in 3D would involve 7 points). Expanding out, we have

$$4U_{i,j} - U_{i-1,j} - Ui, j-1 - U_{i+1,j} - U_{i,j+1} = h^2 f_{i,j}. \tag{27}$$

If we want to rewrite this system of equations in matrix form, we will have to decide how to order the unknowns: based on the i index or the j index. If we wish to arrange based on rows (a lexicographical ordering), we will have to organize by increasing row index. This means that the unknowns appear in the order

$$\begin{bmatrix} U_{0,0} \\ U_{1,0} \\ U_{2,0} \\ U_{3,0} \\ \vdots \\ U_{n+1,0} \\ U_{0,1} \\ U_{1,1} \\ U_{2,1} \\ U_{3,1} \\ \vdots \\ U_{n+1,1} \\ U_{0,2} \\ U_{1,2} \\ \vdots \\ \vdots \\ U_{n+1,n+1} \end{bmatrix} \tag{28}$$

We need to carefully arrange the corresponding entries in the matrix also. For this, we need to write out some of the equations. Recall that anything with an index of 0 in either the i or j position is an unknown on boundary. Since the solution is known at all the boundary points (with Dirichlet BCs), we first have a set of trivial equations for the boundary values, then the equations for the discretized PDE. Let us use the shorthand of $g_{i,j} = g(x_i, y_j)$.

Let us write the equations for the first two rows of the grid out. This looks like:

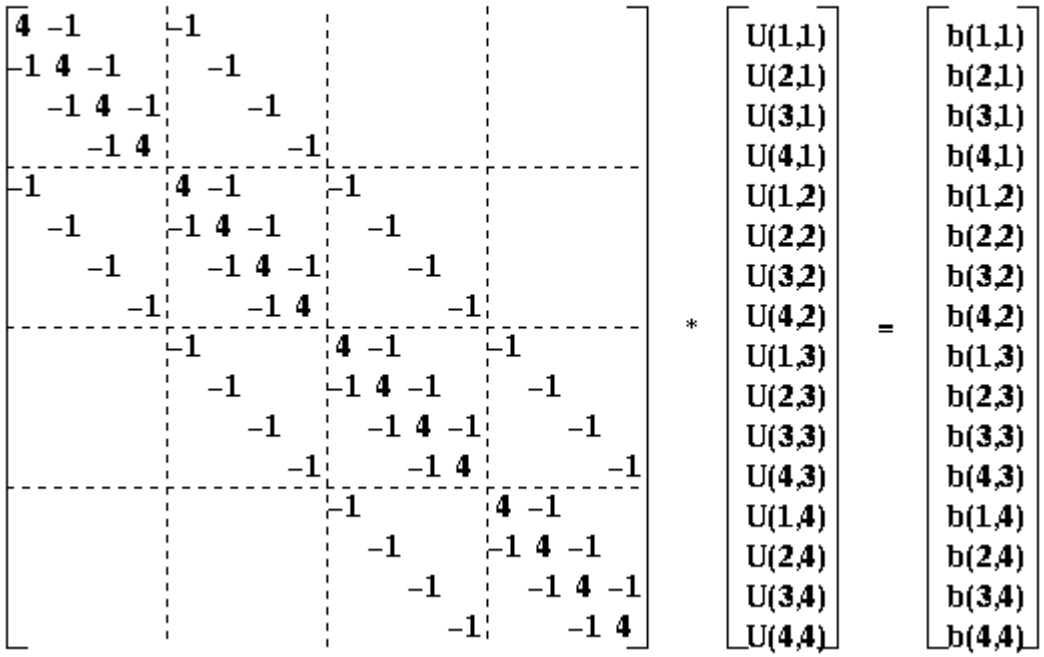
$$U_{0,0} = g_{0,0}, \tag{29}$$
$$U_{1,0} = g_{1,0}, \tag{30}$$
$$\vdots \tag{31}$$
$$U_{n+1,0} = g_{n+1,0}, \tag{32}$$
$$U_{0,1} = g_{0,1}, \tag{33}$$
$$4U_{1,1} - U_{1,0} - U_{0,1} - U_{2,1} - U_{1,2} = h^2 f_{1,1}, \tag{34}$$
$$4U_{2,1} - U_{2,0} - U_{1,1} - U_{3,1} - U_{2,2} = h^2 f_{2,1}, \tag{35}$$
$$\vdots \tag{36}$$
$$4U_{n,1} - U_{n,0} - U_{n-1,1} - U_{n+1,1} - U_{n,n+1} = h^2 f_{n,1}. \tag{37}$$

Note that in the equations on the first non-boundary row, a few terms on the left hand side are known in each equation. We can rearrange these equations to have known terms on one side, unknowns on another. This yields:

$$U_{0,0} = g_{0,0}, \tag{38}$$
$$U_{1,0} = g_{1,0}, \tag{39}$$
$$\vdots \tag{40}$$
$$U_{n+1,0} = g_{n+1,0}, \tag{41}$$
$$U_{0,1} = g_{0,1}, \tag{42}$$
$$4U_{1,1} - U_{2,1} - U_{1,2} = h^2 f_{1,1} + U_{1,0} + U_{0,1}, \tag{43}$$
$$4U_{2,1} - U_{1,1} - U_{3,1} - U_{2,2} = h^2 f_{2,1} + U_{2,0}, \tag{44}$$
$$\vdots \tag{45}$$
$$4U_{n,1} - U_{n-1,1} - U_{n+1,1} = h^2 f_{n,1} + U_{n,0} + U_{n,n+1}. \tag{46}$$

All the U values on the rhs can be replaced by g values since they are on the boundary. Applying this procedure on all four boundary adjacent rows, we can completely eliminate the $U_{i,0}$ or $U_{0,j}$ values from the system of equations, similar to what we did in the 1D case. We are now left with a system of equations only on the interior of the grid, with a carefully constructed right hand side. Fig. 2.2 shows this for $n = 4$. This system is very clearly $n^2 \times n^2$, rather than $(n+1)^2 \times (n+1)^2$. In 3D, the corresponding system would be $n^3 \times n^3$. Once again, this matrix is symmetric positive definite.

Discrete Poisson Problem on 4–by–4 Grid



3 FD for the Poisson equation with Neumann BCs

First, we note that by the divergence theorem, Neumann BCs are possible for the Poisson equation only if the rhs $f(\boldsymbol{x})$ satisfies a *compatibility condition*: $\int_\Omega f = -\int_{\partial\Omega} g$. Further, if you recall PDEs, the solution can only be determined up to an arbitrary constant, which means there are infinitely many solutions.

Now, the presence of a Neumann boundary condition means that we are not told what the solution is on the boundary, but rather the flux. This means the PDE must be solved up to and including the boundary! Unlike in the Dirichlet case, we cannot leave the boundaries out of our system of equations.

This leaves the question of how to discretize the Laplacian on the boundary. Clearly, for the boundary rows, we need to do *something* different. One option could be to use a one-sided difference at the boundary; however, this will ruin the symmetry of the matrix!

The most common approach to handle this is:

1. Introduce *ghost* points outside the boundary.
2. Use a second order centered approximation to the normal derivative across the boundary.
3. Rearrange this equation to get an expression for the ghost points in terms of the unknowns on the boundary.
4. Plug this back into the 5-point discretization of the PDE on the boundary; simplify the resulting formula.

For an example of how to do this, consider enforcing the Neumann condition on the left boundary ($i = 0$). Let the ghost points here have index $i = -1$. The outward normal derivative here is simply $-\frac{\partial}{\partial x}$. Then, at the $0, j$ grid point on the boundary, the Neumann boundary condition can be enforced as:

$$\frac{U_{-1,j} - U_{1,j}}{2h} = g_{0,j}, \tag{47}$$

$$\implies U_{-1,j} = U_{1,j} + 2hg_{0,j}. \tag{48}$$

At the boundary points, we also have the discretized PDE:

$$4U_{0,j} - U_{-1,j} - U_{1,j} - U_{0,j+1} - U_{0,j-1} = h^2 f_{0,j}. \tag{49}$$

We can substitute the expression for $U_{-1,j}$ in. This yields:

$$4U_{0,j} - U_{1,j} - 2hg_{0,j} - U_{1,j} - U_{0,j+1} - U_{0,j-1} = h^2 f_{0,j}. \tag{50}$$

We can move the $2hg_{0,j}$ term to the right hand side, and simplify the rest. This gives:

$$4U_{0,j} - 2U_{1,j} - U_{0,j+1} - U_{0,j-1} = h^2 f_{0,j} + 2hg_{0,j}. \tag{51}$$

Unfortunately, the presence of the -2 factor in front of $U_{1,j}$ ruins the symmetry of the matrix. If it were a -1, however, we would be good to go. Thus, it is common to divide this equation by 2. As long as we divide both sides, it should be fine.

$$2U_{0,j} - U_{1,j} - \frac{1}{2}U_{0,j+1} - \frac{1}{2}U_{0,j-1} = \frac{1}{2}h^2 f_{0,j} + hg_{0,j}. \tag{52}$$

We can do something similar for the other boundaries:

- Bottom boundary: normal derivative is $-\frac{\partial}{\partial y}$.
- Right boundary: normal derivative is $\frac{\partial}{\partial x}$.
- Top boundary: normal derivative is $\frac{\partial}{\partial y}$.

For the corners, one could either pick each corner to only be part of one boundary, or one could use an *average* of two FD approximations, one along the x-direction and the other along the y-direction.

The discrete negative Laplacian $-L$ with Neumann BCs encoded is *singular*. This means that, when solving the system, we can either expect infinitely-many solutions or no solution at all. If the rhs $f(\boldsymbol{x})$ does not satisfy the compatibility condition, there is no solution. If f satisfies the compatibility condition, we can usually recover a solution to this equation using any Krylov method.

On the other hand, if f does not satisfy the compatiblity condition, there is a way to solve the linear system anyway. The idea is to project out the part of the right hand side $\mathbf{b}$ that is not in $Range(-L)$. In this case, the null space of $(-L)^T$ has been found to be simply the vector of ones. We define a projection operator $P = I - \frac{1}{N}\boldsymbol{e}\boldsymbol{e}^T$, where $\boldsymbol{e} = [1, \ldots, 1]^T$ is the null vector. Then, we simply solve $-L\boldsymbol{x} = P\mathbf{b}$.

NOTE: This is a fancy way of saying that $\mathbf{b}$ should have a zero component in the null-space of A^T, for a general singular linear system $A\boldsymbol{x} = \mathbf{b}$.