# Chapter 3

# Finite Difference Methods

## 3.1 Introduction

The finite difference methods have dominated computational science since its inception and were *the* method of choice in the 1960s and 1970s. While other methods, such as the finite element method and boundary element have enjoyed recent popularity, finite difference methods are still utilized for a wide array of computational engineering and science problems.

Briefly, the finite difference method can be characterized as follows. **Finite Difference Method**

- Utilizes uniformly spaced grids.
- At each node, each derivative is approximated by an algebraic expression which references the adjacent nodes.
- A system of algebraic equations is obtained by evaluating the previous step for each node.
- The system is solved for the dependent variable.

## 3.2 Taylor Series

Before proceeding to discuss the mathematical formalities of the finite difference method, it would probably be a good idea to review some facts concerning Taylor's series. From elementary calculus, we remember that the Taylor series formula is used to expand a function, $u(x)$ as a power series,

$$u(x) = u(a) + \frac{(x-a)}{1!}u'(a) + \frac{(x-a)^2}{2!}u''(a) + \ldots = \sum_{n=0}^{\infty} \frac{(x-a)^n}{n!}u^{(n)}(a) \qquad (3.1)$$

For the numerical treatment of field equations, we often represent the Taylor series in somewhat different forms. Usually, we replace $x$ and $a$ in the previous formula by $x + h$ and $x$ respectively.

Here, $h$ is denoted as an increment in $x$ and is assumed positive. The Taylor series now becomes

$$u(x+h) = u(x) + \frac{h}{1!}u'(x) + \frac{(h)^2}{2!}u''(x) + \ldots = \sum_{n=0}^{\infty} \frac{(h)^n}{n!}u^{(n)}(x) \tag{3.2}$$

For the treatment of many problems, it is convenient to take only the first two terms of the right hand side of the previous equation,

$$u(x+h) = u(x) + hu'(x) + O(h^2) \tag{3.3}$$

where the expression $O(h^2)$, represents the error of the approximation is proportional to $h^2$. From this relationship, we can easily define what is known as the first order, forward difference approximant to $\frac{du}{dx}$,

$$\frac{du}{dx} = \frac{u(x+h) - u(x)}{h} + O(h) \tag{3.4}$$

Likewise, we can define the first order, backward difference approximant to $\frac{du}{dx}$ (just substitute $-h$ for $h$ in the expansion),

$$\frac{du}{dx} = \frac{u(x) - u(x-h)}{h} + O(h) \tag{3.5}$$

If instead, we take terms up to and including $h^3$ on the right hand side of the two previous equations, then subtract them, we can obtain,

$$\frac{du}{dx} = \frac{u(x+h) - u(x-h)}{2h} + O(h^3). \tag{3.6}$$

This is the second order, central difference approximation to $\frac{du}{dx}$. Taking up to forth order yields,

$$u(x+h) + u(x-h) = 2u(x) + h^2u''(x) + O(h^4) \tag{3.7}$$

the terms $h$ and $h^3$ having canceled. Now we can obtain the second order, central difference approximation to $\frac{d^2u}{dx^2}$

$$\frac{d^2u}{dx^2} = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} + O(h^2). \tag{3.8}$$

Now that were warming up, let's see what excitement awaits us by considering the Taylor series for a function of two independent variables, $u(x,t)$,

$$u(x+h,t) = u(x,t) + \frac{h}{1!}\frac{\partial u(x,t)}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u(x,t)}{\partial x^2} + \ldots \tag{3.9}$$

likewise,

$$u(x-h,t) = u(x,t) - \frac{h}{1!}\frac{\partial u(x,t)}{\partial x} + \frac{h^2}{2!}\frac{\partial^2 u(x,t)}{\partial x^2} + \ldots \tag{3.10}$$

thus, we can generate first and second (and higher) order approximations for partial derivatives, for example:

$$\frac{\partial u}{\partial x} = \frac{u(x+h,t) - u(x,t)}{h} + O(h) \tag{3.11}$$

$$\frac{\partial u}{\partial x} = \frac{u(x,t) - u(x-h,t)}{h} + O(h) \tag{3.12}$$

$$\frac{\partial u}{\partial x} = \frac{u(x+h,t) - u(x-h,t)}{2h} + O(h^2) \tag{3.13}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h} + O(h^2). \tag{3.14}$$

It is easy to show that there are equivalent expressions for time,

$$\frac{\partial u}{\partial t} = \frac{u(x,t+l) - u(x,t)}{l} + O(l) \tag{3.15}$$

as well as for other spacial variables and mixed partial derivatives,

$$\begin{aligned} \frac{\partial^2 u}{\partial x \partial y} &= \frac{u(x+h,y+k) - u(x+h,y-k) - u(x-h,y+k) + u(x-h,y-k)}{4hk} + \\ & O(\frac{(h+k)^4}{hk}) \end{aligned} \tag{3.16}$$

Alright, now that we know everything about how to derive difference formulations from Taylors series, let's talk about how to use such results in approximating a field problem.

## 3.3 Finite difference formulation for Elliptic Equations

As a simple illustration, let's consider Laplace's equation for a planar region, $\Omega$ with Dirichlet boundary data on the boundary of the region, $\partial\Omega$. To make this even simpler still, consider the region to be bounded by the lines $x = 0$, $x = 1$, $y = 0$, and $y = 1$. We now discretize the bounded region with a square mesh as show in Figure 3.3. We note that we can use non-square elements, but for now, squares will suffice. We might ask ourselves the pertinant question, "how many squares do we use to discretize the solution domain?" While we intuitively know that our solution will probably get better as the discretization level increases, how fast it will get better and how much it will cost are other questions to think about. We impose a mesh of grid points and define $h$ to be the distance between mesh points. The **interior** grid points are given by $(x_i, y_j) = (ih, jh)$, where $i, j = 1, \ldots, N$. Now we look up above for the Taylors series expansion which represents second order approximations for $x$ and $y$ and substitute them in:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{3.17}$$

$$\nabla^2 u = \frac{u(x_{i+1},y_j) - 2u(x_i,y_j) + u(x_{i-1},y_j)}{(\Delta x)^2} + \frac{u(x_i,y_{j+1}) - 2u(x_i,y_j) + u(x_i,y_{j-1})}{(\Delta y)^2} = 0. \tag{3.18}$$

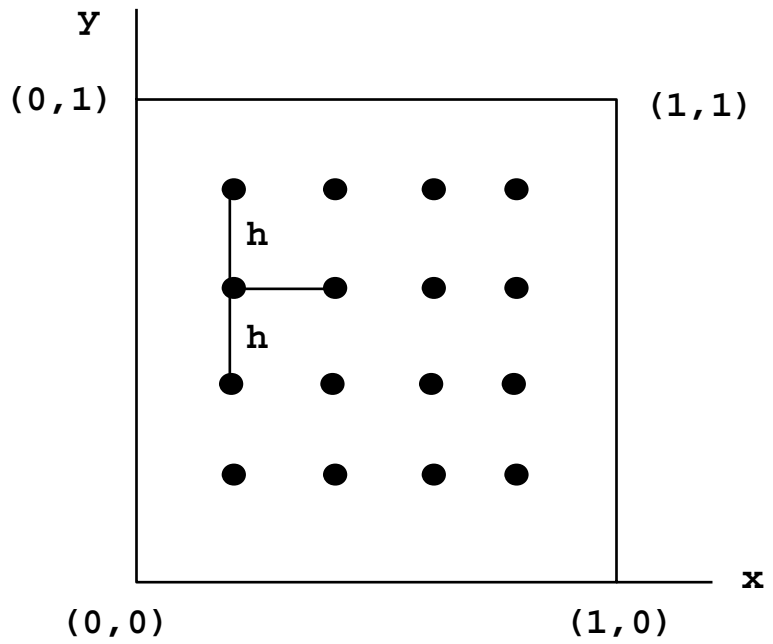


Figure 3.1: Discretization for Laplace's Equation.

As we mentioned before, we chose a uniform grid such that $\Delta x = \Delta y = h$. This yields the simplification,

$$\nabla^2 u_{i,j} = \frac{1}{h^2}[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}] = 0 \tag{3.19}$$

Which is the finite difference approximation of Laplace's equation. If we had carried a higher order term through the calculations, we would have found out that the local truncation error is

$$\frac{1}{12}h^2(\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4})_{i,j} \tag{3.20}$$

which is usually written as $O(h^2)$, such that the error is proportional to the space constant squared. Note that five points are involved in the approximation of the Laplacian (points to the left, right, above and below of a central point, $(x_i, y_j)$) such that this is called the *five point* formula.

We will derive similar relationships for parabolic and hyperbolic problems. As such, it is often convient to use a stencil notation when discussing difference equations for the second order Laplacian on a Cartesian grid (or one that is logically rectangular). Below, we define the stencil for 2D problems, however, we will create 3D stencils later in this chapter.

First, let's create a general stencil notation and then look at different finite difference stencils.

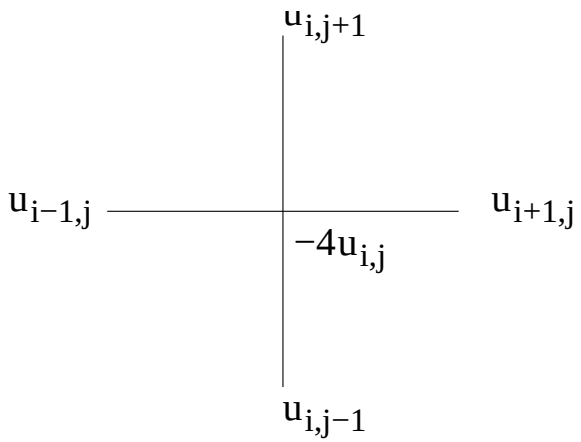


Figure 3.2: Five point finite difference stencil for the Laplacian operator.

A general stencil can be represented as

$$[s_{\alpha,\beta}]_h = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & s_{-1,1} & s_{0,1} & s_{1,1} & \cdots \\ \cdots & s_{-1,0} & s_{0,0} & s_{1,0} & \cdots \\ \cdots & s_{-1,-1} & s_{0,-1} & s_{1,-1} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}_h \tag{3.21}$$

This defines a discrete set of grid operations

$$[s_{\alpha,\beta}]_h u_h(x,y) = \sum_{(\alpha,\beta)} s_{\alpha,\beta} u_h(x+\alpha h_x, y+\beta h_y) \tag{3.22}$$

We will mainly consider five-point or nine-point stencils (or their 3D equivalent). The five-point stencil is of the form:

$$[s_{\alpha,\beta}]_h = \begin{bmatrix} & s_{0,1} & \\ s_{-1,0} & s_{0,0} & s_{1,0} \\ & s_{0,-1} & \end{bmatrix}_h \tag{3.23}$$

and the nine-point stencil of the form:

$$[s_{\alpha,\beta}]_h = \begin{bmatrix} s_{-1,1} & s_{0,1} & s_{1,1} \\ s_{-1,0} & s_{0,0} & s_{1,0} \\ s_{-1,-1} & s_{0,-1} & s_{1,-1} \end{bmatrix}_h \tag{3.24}$$

When using stencil notation, we usually must restrict the stencil application to the domain $\Omega_h$ excluding the boundaries. Near the boundaries, the stencils often have to be modifed due to boundary conditions.

Using Figure 3.3 and the notation above, we can now rewrite equation (3.19) in stencil form

$$[s_{i,j}]_h = \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}_h u_h(x,y) \tag{3.25}$$

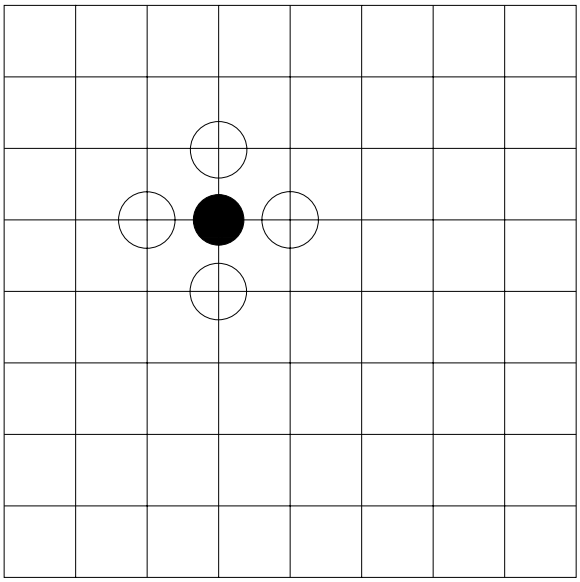

Figure 3.3: Five point finite difference stencil within a 2D domain.

One often finds the following notation to denote the discrete Laplacian,

$$\nabla_h^2 = \frac{1}{h^2} \left[ \begin{array}{ccc} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{array} \right]_h u_h(x,y) \tag{3.26}$$

The above stencil works for areas within the solution domain that are not close to a boundary, such as in Figure 3.3

However, for nodes near boundaries, we would need to modify the stencil, for example, for the situation in Figure 3.3, we would use

$$\nabla_h^2 = \frac{1}{h^2} \left[ \begin{array}{ccc} & 1 & \\ 0 & -4 & 1 \\ & 1 & \end{array} \right]_h u_h(x,y) \tag{3.27}$$

and for the situation in Figure 3.3 we would use

$$\nabla_h^2 = \frac{1}{h^2} \left[ \begin{array}{ccc} & 0 & \\ 0 & -4 & 1 \\ & 1 & \end{array} \right]_h u_h(x,y) \tag{3.28}$$

If we have Dirichlet boundary conditions, then we know the values of the mesh points (nodes) on the region of the boundary, we are left applying the approximation to $N^2$ internal mesh points. This yields a linear system of equations of $N^2$ equations of the function $u_{i,j}$. We usually write the system as:

$$A\mathbf{u} = \mathbf{b} \tag{3.29}$$

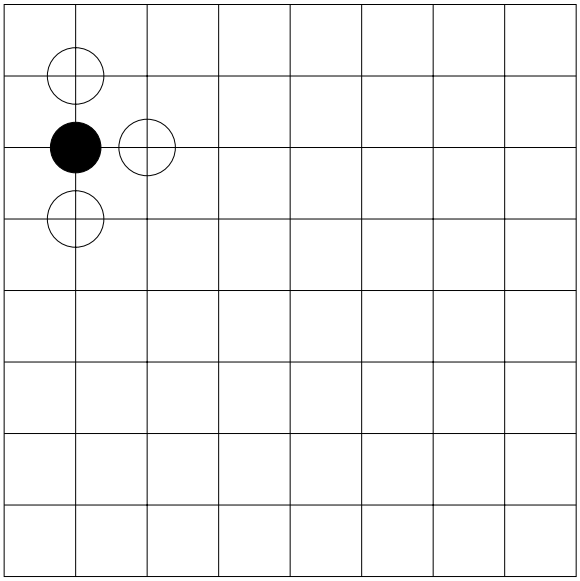

Figure 3.4: Five point finite difference stencil within a 2D domain near a boundary.

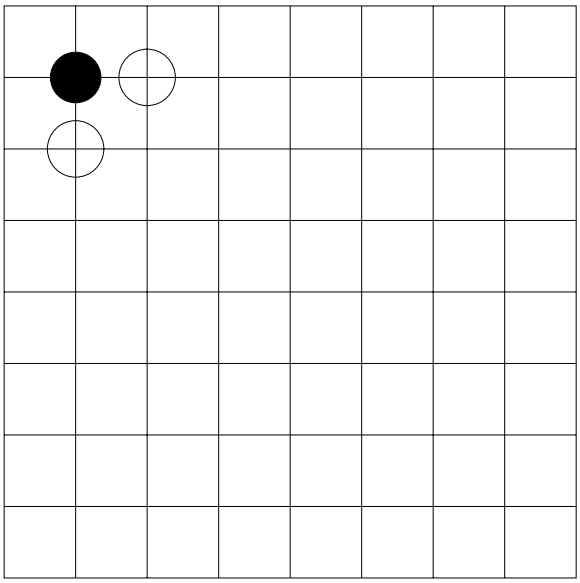

Figure 3.5: Five point finite difference stencil within a 2D domain near a boundary.

where $A$ is an $N^2 \times N^2$ matrix of block tridiagonal form:

$$A = \begin{pmatrix} T_N & I_N & & \\ I_N & T_N & \ddots & \\ & \ddots & \ddots & I_N \\ & & I_N & T_N \end{pmatrix} \tag{3.30}$$

Here, $I_N$ is the $N \times N$ identity matrix and $T_N$ is the $N \times N$ tridiagonal matrix,

$$T_N = \begin{pmatrix} -4 & 1 & & \\ 1 & -4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -4 \end{pmatrix} \tag{3.31}$$

The vector $\mathbf{u}$ consists of the unknown values at the nodes and is of dimension $N^2 \times 1$, while $\mathbf{b}$ is the $N^2 \times 1$ vector consisting of boundary data (with $4N - 4$ non-zero elements).

Thus the solution to our problem can be found by solving for the vector $u$,

$$\mathbf{u} = A^{-1}\mathbf{b} \tag{3.32}$$

since $A$ is nonsingular, we will obtain a unique solution.

### 3.3.1 Laplace's Equation – A Numerical Example

To illustrate the method, consider the steady-state heat conduction problem defined over our unit square. We want to find the finite difference approximation to Laplace's equation subject to the Dirichlet boundary conditions,

$$u(x, 0) = u_0 = 300; \qquad u(1, y) = u(x, 1) = u(0, y) = 0 \tag{3.33}$$

If we discretize our square into a uniform mesh with $h = 0.25$ as shown in Figure 3.3.1. The nodes are numbered from $i, j = 0, 1, \ldots, N + 1 = 4$, where nodes $i, j = 1, 2, 3$ represent the interior nodes which we are solving for. We can apply the finite difference memthod using equation 3.19. Writing out a few of the nodes, we obtain for $u_{1,1}$

$$u_{2,1} + u_{0,1} + u_{1,2} + u_{1,0} - 4u_{1,1} = 0 \tag{3.34}$$

and for $u_{1,2}$

$$u_{2,2} + u_{0,2} + u_{1,3} + u_{1,1} - 4u_{1,2} = 0 \tag{3.35}$$

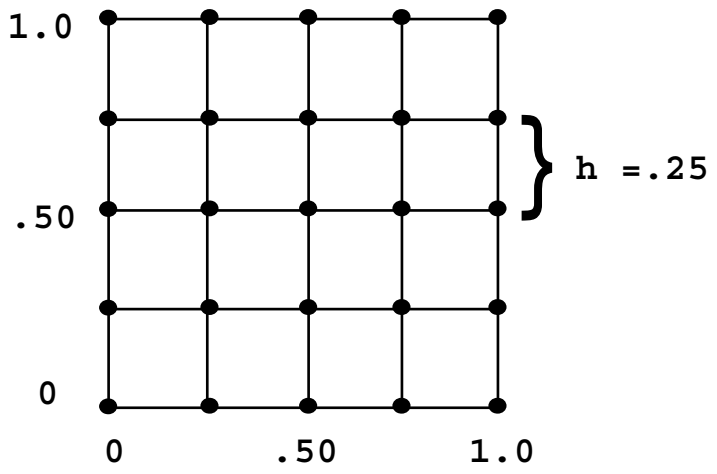


Figure 3.6: Mesh used for the numerical solution of the steady-state heat equation.

We continue the process for all nine interior nodes numbered from the left corner, proceeding along the x-axis, $u_{0,0}, u_{1,0}, u_{2,0}, u_{3,0}, u_{0,1}, u_{1,1}, u_{2,1}, u_{3,1} \ldots u_{3,3}$ and obtain the following linear system of equations (notice that we have multiplied through equations 3.34 and 3.35 by a negative 1):

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{pmatrix} = \begin{pmatrix} 300 \\ 300 \\ 300 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.36}$$

Upon solving the linear system, we obtain a vector $\mathbf{u}$ of the temperatures. It is interesting to look at the distributions of temperature as a function of the number of elements (and thus step-size, $h$). The solution of the system shown in figure 3.3.1 and equation 3.36, we obtain the temperature distribution in 3.3.1. You'll note that the solution does not look very smooth. This is because we a fair amount of error due to inadequately discretizing the solution domain. As we increase the number of elements (and thus decrease the step-size), we are able to reduce the discretization area. Figures 3.3.1, 3.3.1, and 3.3.1 show the iso-tempurature contours for succesive double of the number of elements along the x and y axes. As you can see, the solutions become smoother as we increase the number of elements (although there is still a problem near the bottom corners). Note that we have significantly increased the computational costs to obtain a more accurate solution. We started with a $9 \times 9$ system of equations, and ended up using a $30 \times 30$ system of equations.

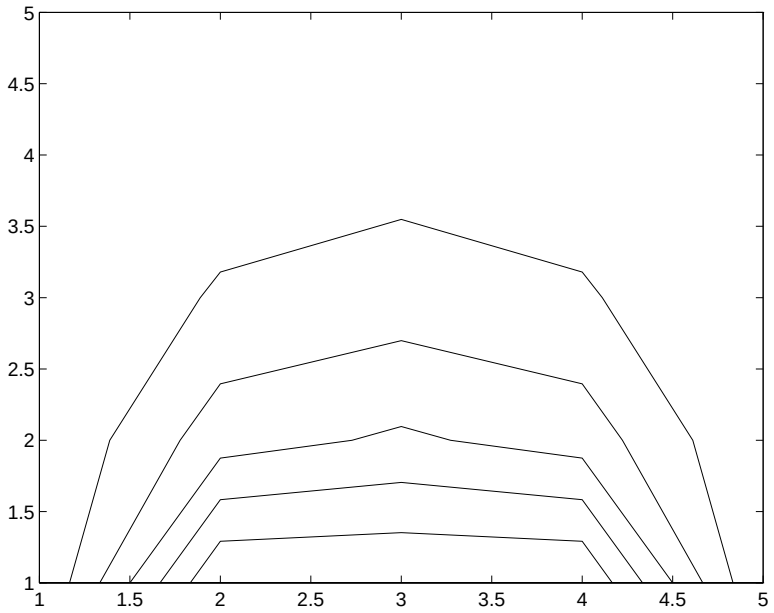


Figure 3.7: Temperature distribution, shown as iso-temperature contours for the solution of Laplace's equation from example 3.3.1. Here we have divided up the solution domain into 8 panels with $h = 0.25$.

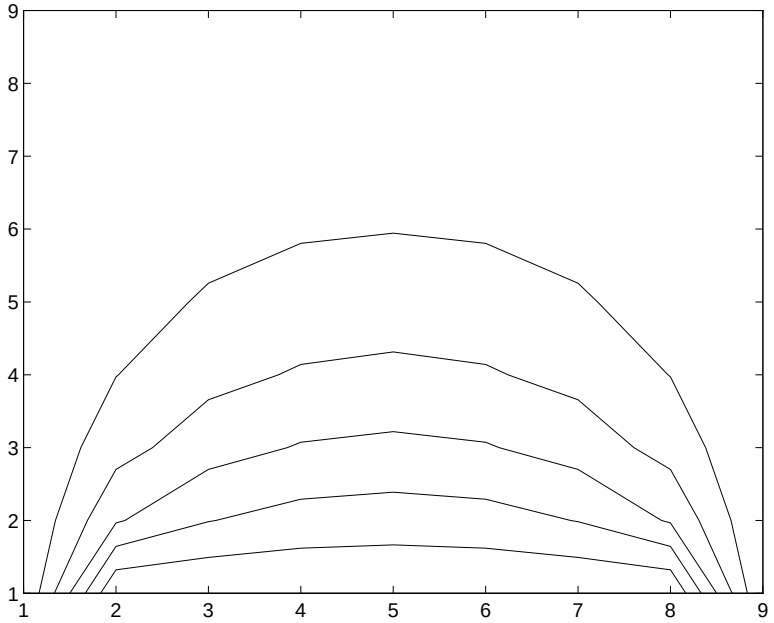


Figure 3.8: Iso-temperature contours for the solution of Laplace's equation from example 3.3.1. Here the solution domain is divided into 64 panels with $h = 0.125$.

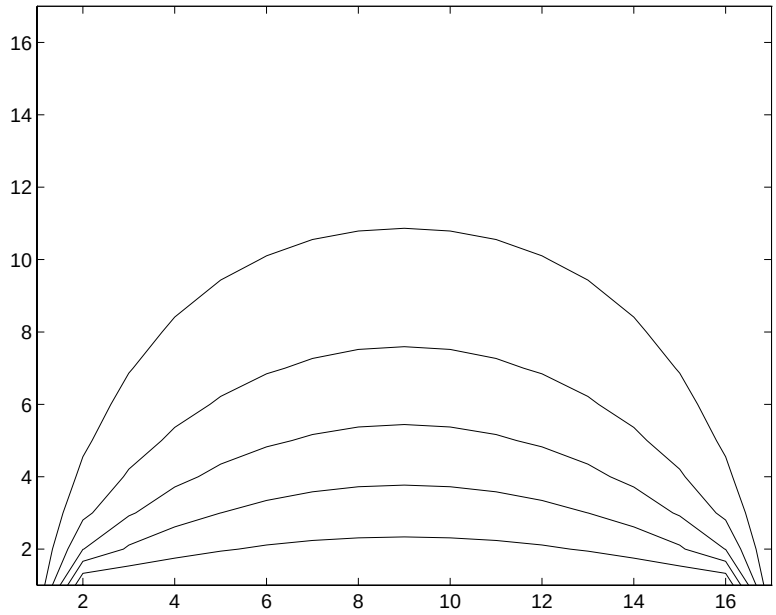


Figure 3.9: Iso-temperature contours for the solution of Laplace's equation from example 3.3.1. Here the solution domain is divided into 256 panels with $h = 0.0625$.

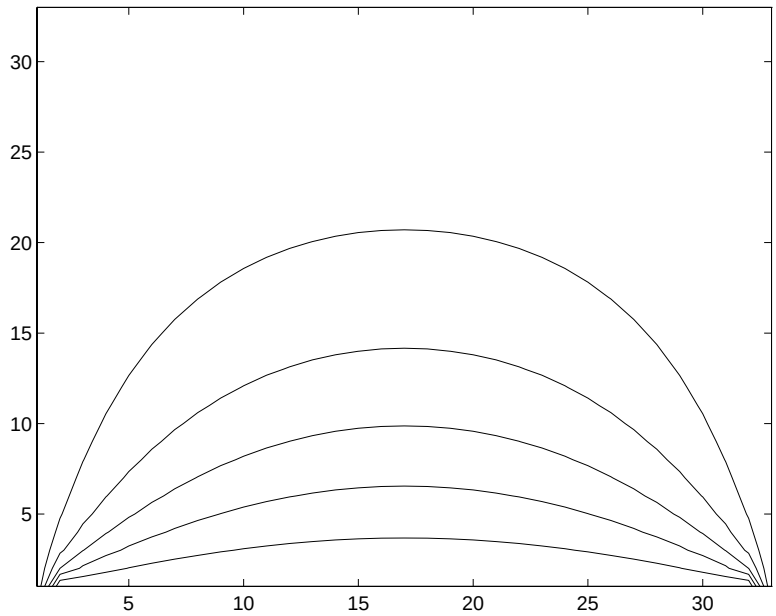


Figure 3.10: Iso-temperature contours for the solution of Laplace's equation from example 3.3.1. Here the solution domain is divided into 1024 panels with $h = 0.03125$.

Since the solution of a system of equations is $O(N^3$, we increased the number of operations by a factor of 37 and our storage costs by a factor of 100 (assuming double precision floats).

We have increased the solution accuracy by uniformly dividing the number of elements into smaller paritions. However, as you might imagine, this isn't necessarily the most computationally efficient way to decrease the discretization error. It would seemingly make sense to only increase the number of elements in the regions with the largest amount of error. But how do we do that if the finite different methodology utilizes uniform partitions? One answer is to use overlapping grids. We will address this method later on in the section on Adaptive Methods.

I note that this problem can be solved analytically using a Fourier series [26]:

$$u(x,y) = \frac{4u_0}{\pi}\sum_{n=0}^{\infty}\frac{1}{2n+1}\sin((2n+1)\pi x)sinh[(1-y)(2n+1)\pi]cosech(2n+1). \tag{3.37}$$

thus we can compare our finite difference solution with the solution generated by the Fourier series [26]:

| Node | Fourier Solution | Finite Difference Solution | Absolute Error |
|---|---|---|---|
| (1,1) | 129.608 | 128.571 | -1.0 |
| (2,1) | 162.159 | 158.036 | -4.1 |
| (1,2) | 54.608 | 56.250 | 1.6 |
| (2,2) | 75.000 | 75.000 | 0.0 |
| (1,3) | 20.391 | 21.429 | 1.0 |
| (2,3) | 28.624 | 29.464 | 0.8 |

Table 3.1: Comparison of F.D. and Fourier Solutions

### 3.3.2 Poisson's Equation

Before going on, let's take a minute to extend our method to including solving Poisson's equation. The Poisson equation has a right hand side which does not equal zero,

$$\nabla^2 u = f(x,y) \tag{3.38}$$

Since we have already discretized the left hand side of the equation in 3.19, we simply need to add the right hand side,

$$\nabla^2 u_{i,j} = \frac{1}{h^2}[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}] = f(x_i, y_j) \tag{3.39}$$

Such that our solution can be written as,

$$-u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} + 4u_{i,j} = -h^2 f(x_i, y_j), \tag{3.40}$$

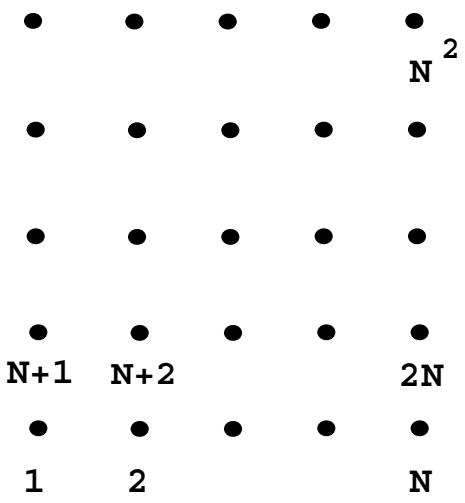


Figure 3.11: Natural ordering for the interior grid points.

where we have multiplied through by minus one. As an illustration of the kind of system this produces, we order the points in *natural* or *row wise* ordering and look at the system for $N = 2$.

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} u_{11} \\ u_{21} \\ u_{12} \\ u_{22} \end{pmatrix} = -h^2 \begin{pmatrix} f_{11} \\ f_{21} \\ f_{12} \\ f_{22} \end{pmatrix} + \begin{pmatrix} u_{01} + u_{10} \\ u_{20} + u_{31} \\ u_{02} + u_{13} \\ u_{32} + u_{23} \end{pmatrix} \tag{3.41}$$

Where, for the Dirichlet problem, the boundary data is given as:

$$\begin{aligned} u_{0,j} = g(0, y_j), \quad u_{N+1,j} = g(1, y_j), \quad j = 0, 1, \ldots N+1 \\ u_{i,0} = g(x_i, 0), \quad u_{i,N+1} = g(x_i, 1), \quad j = 0, 1, \ldots N+1 \end{aligned} \tag{3.42}$$

We can then generalize the Poisson problem with Dirichlet boundary conditions as the $N^2 \times N^2$ linear system,

$$\begin{pmatrix} T_N & -I_N & & \\ -I_N & \ddots & \ddots & \\ & \ddots & & -I_N \\ & & -I_N & T_N \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_N \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 - h^2 \mathbf{f}_1 \\ \mathbf{b}_2 - h^2 \mathbf{f}_2 \\ \vdots \\ \mathbf{b}_N - h^2 \mathbf{f}_N \end{pmatrix}. \tag{3.43}$$

where

$$\mathbf{u}_i = (u_{1i}, \ldots, u_{Ni})^T, \quad \mathbf{f}_i = (f_{1i}, \ldots, f_{Ni})^T, \quad i = 1, \ldots, N \tag{3.44}$$

$$\mathbf{b}_1 = (u_{01} + u_{10}, u_{20}, \ldots, u_{N-1,0}, u_{N,0} + u_{N+1,1})^T \tag{3.45}$$

$$\mathbf{b}_i = (u_{0i}, 0, \ldots, 0, u_{N+1,1})^T, \quad i = 2, \ldots, N-1 \tag{3.46}$$

$$\mathbf{b}_N = (u_{0,N} + u_{1,N+1}, u_{2,N+1}, \ldots, u_{N-1,N}, u_{N,N+1} + u_{N+1,N})^T \tag{3.47}$$

### 3.3.3 Poisson's Equation - A Numerical Example

To illustrate the finite difference method as applied to Poisson's equation, consider same steady-state heat conduction problem defined over our unit square as in our last example. This time we find the finite difference approximation to Poisson's equation subject to the Dirichlet boundary conditions,

$$u(x,0) = u(1,y) = u(x,1) = u(0,y) = 0 \tag{3.48}$$

with the additional source term $f(x,y) = 1$ at the center of the square.

We again discretize our square into a uniform mesh with $h = 0.25$ as shown in Figure 3.3.1. This time, we can the finite difference memthod using equation 3.40.

Upon solving the resulting linear system, we obtain a vector $\mathbf{u}$ of the temperatures. Again we look at the distributions of temperature as a function of the number of elements (and thus step-size, $h$).

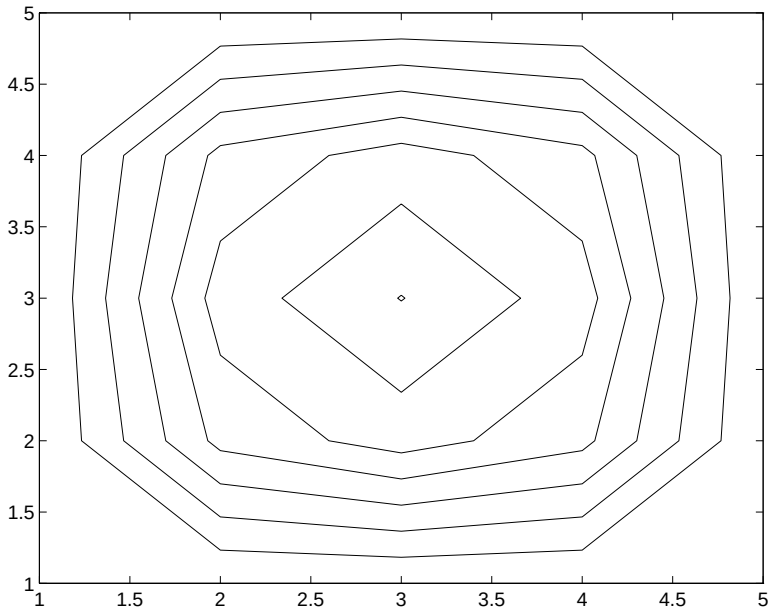


Figure 3.12: Temperature distribution, shown as iso-temperature contours for the solution of Poisson's equation. Here we have divided up the solution domain into 8 panels with $h = 0.25$.

Again we notice that as the number of elements increase (and the step-size decreases), the solution becomes smoother - and our computational costs increase.

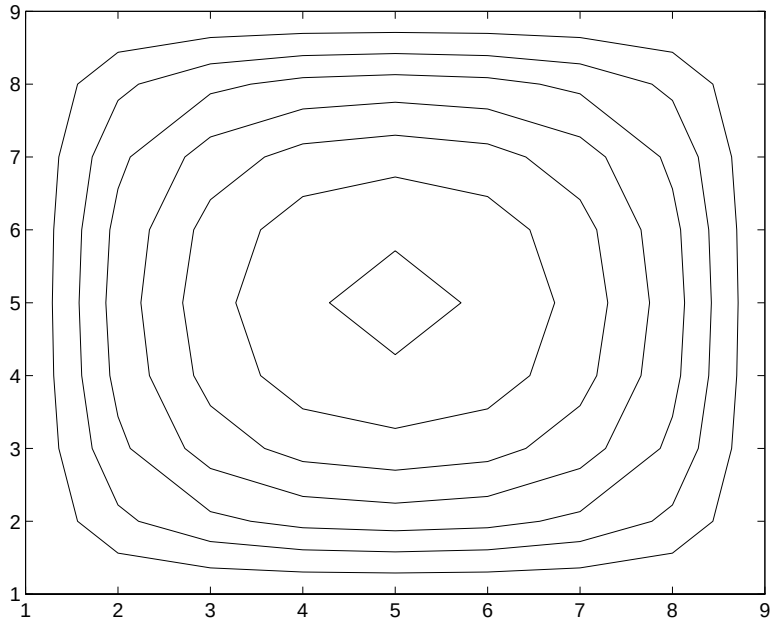


Figure 3.13: Temperature distribution, shown as iso-temperature contours for the solution of Poisson's equation. Here we have divided up the solution domain into 16 panels with $h = 0.125$.

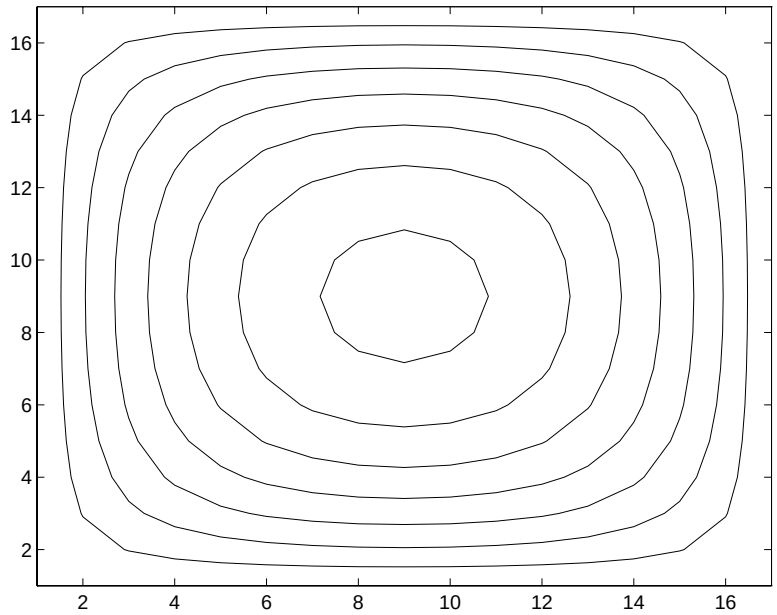


Figure 3.14: Temperature distribution, shown as iso-temperature contours for the solution of Poisson's equation. Here we have divided up the solution domain into 32 panels with $h = 0.0625$.

### 3.3.4 Iterative Finite Difference Methods for Elliptic Equations

As you will recall from CS 5210, iterative methods are powerful techniques for approximating the solution to a large-scale linear system, $\mathbf{Ax} = \mathbf{b}$. Some of the simpler iterative methods have

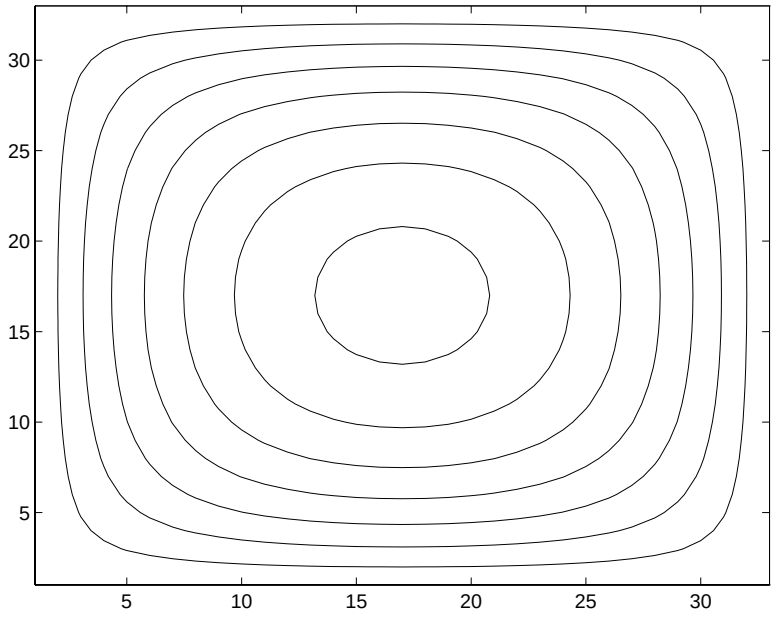


Figure 3.15: Temperature distribution, shown as iso-temperature contours for the solution of Poisson's equation. Here we have divided up the solution domain into 64 panels with $h = 0.03125$.

straightforward extensions to iterative methods for solving finite difference approximations for elliptic equations.

Recall the Gauss-Seidel method. We first split the $A$ matrix into two pieces, $A = M - N$. For the Jacobi method, $M = D$ and $N = -(L+U)$, where $L$ and $U$ are the lower and upper triangular parts of the matrix $A$ respectively. For the Gauss-Seidel method, the splitting is $M = D + L$ and $N = -U$. This leads to the Gauss-Seidel technique for solving $Ax = b$:

$$x^{k+1} = D^{-1}(b - Lx^{k+1} - Ux^k). \tag{3.49}$$

We can cast a five-point finite difference approximation of Laplace's equation as an iterative method:

$$u_{i,j}^{k+1} = \frac{1}{4}(u_{i-1,j}^{k+1} + u_{i,j-1}^{k+1} + u_{i+1,j}^{k} + u_{i,j+1}^{k}) \tag{3.50}$$

which can easily be extended to a Gauss-Seidel method for Poisson's equation:

$$u_{i,j}^{k+1} = \frac{1}{4}(u_{i-1,j}^{k+1} + u_{i,j-1}^{k+1} + u_{i+1,j}^{k} + u_{i,j+1}^{k} + F_{i,j}) \tag{3.51}$$

Similarly, one can write a Successive Over Relaxation (SOR) point iteratiion method for Poisson's equation:

$$\bar{u}_{i,j}^{k+1} = \frac{1}{4}(u_{i-1,j}^{k+1} + u_{i,j-1}^{k+1} + u_{i+1,j}^{k} + u_{i,j+1}^{k} + F_{i,j}) \tag{3.52}$$

where

$$u_{i,j}^{k+1} = \omega \bar{u}_{1,j}^{k+1} + (1-\omega)u_{i,j}^{k} \quad (0 < \omega < 2). \tag{3.53}$$

Some properties of these iterative methods for finite difference approximations of elliptic equations include

- The Dirichlet boundary condition can be applied by setting $u_{i,j}^k = g_{i,j}$ at all of the boundary nodes for $k = 0, 1, 2, \ldots$
- The initial estimate, $u_{i,j}^0$ can be chosen arbitrarily.
- The value for the right hand side is determined by $F_{i,j} = -h^2 f_{i,j}$.
- The methods as stated above are to be applied to a natural node ordering as shown in Figure 3.11.
- The $\omega$ in the SOR method is the relaxation parameter. If $\omega = 1$ then the SOR method reduces to the Gauss-Seidel method.

### 3.3.5 Example: Algorithm for Finite Difference Approximation for Poisson's Equation

Here we provide a simple algorithm to solve the two-dimensional Poisson equation with Dirichlet boundary conditions for a rectangular (or square) region using an iterative Gauss-Seidel or SOR method.

**Algorithm FD 1: Iter_Poisson(imax,jmax,$\omega$,kmax,tol,rmax)**
**Function**: Solves a two-dimensional Poisson equation using GS or SOR.
**Output**: $u_{i,j}$, the approximations of $u(x, y)$
**Inputs**: imax and jmax - the number x and y intervals, $\omega$, kmax - the number of iterations, tol - tolerance, rmax - max residual, $x_1, x_2$ and $y_1, y_2$ - the $x$ and $y$ intervals, Dirichlet boundary conditions $g_1, g_2, g_3, g_4$ on the lower, right, upper, and left edges of the sqaure respectively, $f(x, y)$ - the right hand side

**Begin**

Define parameters $imax = (x_2 - x_1)/h, jmax = (x_2 - x_1)/h$
Set boundary conditions
For i $\leftarrow$ 1 to $imax - 1$ Do
For j $\leftarrow$ 1 to $jmax - 1$ Do
$x = x_1 + ih$
$y = y_1 + jh$
$u_{i,j} = 0, u_{i,0} = g_1(x), u_{imax,j} = g_2(y), u_{i,jmax} = g_3(x) u_{0,j} = g_4(y)$
End
Calculate the k-th iterate

For k $\leftarrow$ 1 to $kmax$ Do
For i $\leftarrow$ 1 to $imax - 1$ Do
For j $\leftarrow$ 1 to $jmax - 1$ Do
$x = x_1 + ih$
$y = y_1 + jh$
$u_{temp} = u_{i,j}$
$u_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1} - h^2 f_{x,y})$
$u_{i,j} = \omega u_{i,j} + (1 - \omega)u_{temp}$
End
End
Calculate Residual
If rmax $<$ tol then converged, otherwise, continue the loop as long as k $<$ kmax
End
Output Solution
**End**

### 3.3.6 Neumann Boundary Conditions

Now let's consider another type of boundary conditiont that occurs frequently in engineering field problems, the Neumann problem. The Neumann boundary condition, which is often denoted as the *natural* boundary condition (as opposed to the Dirichlet boundary condition which is often termed the *essential* boundary condition) is concerned with the *flux* (i.e. the normal of the gradient) at the boundary. The so-called, Neumann problem (when the Neumann boundary condition is applied over the entire boundary) requires the solution of $u(x, y)$ over $\Omega \subset \partial\Omega$ subject to,

$$\nabla u \cdot \mathbf{n} = \frac{\partial u}{\partial n} = f(x, y) \quad \text{on } \partial\Omega \tag{3.54}$$

For many physical situations, $f(x, y)$ is equal to zero. This occurs when there is a so-called, no-flux boundary condition, such that the solution domain is somehow isolated and the field is not allowed to flow outside the domain. A simple example is in electrostatics when the solution domain is surrounded by a non-conducting media (like air) such that the current (flux) cannot flow outside the conducting solution domain. Suppose that the right edge of the square domain has a no flux boundary condition. We would then write equation (3.54) as

$$\nabla u \cdot \mathbf{n} = \frac{\partial u(x_n, y_j)}{\partial x} = u_x(x_n, y_j) = 0 \tag{3.55}$$

Then the Laplacian difference equation operator for the point $(x_n, y_j)$ is

$$u_{n+1,j} + u_{n-1,j} + u_{n,j+1} + u_{n,j-1} - 4u_{n,j} = 0 \tag{3.56}$$

The value for $u_{n+1}$ is unknown because it lies outside of the domain $\Omega$. What we can do is to use either a first order difference operator or a centered difference operator to approximate this quantity.

$$\frac{u_{n+1,j} - u_{n-1,j}}{2h} \approx u_x(x_n, y_j) = 0 \tag{3.57}$$

This yields an approximation that $u_{n+1,j} \approx u_{n-1,j}$, which has an accuracy of $O(h^2)$. Substituting equation (3.57) into equation (3.56) yields

$$2u_{n-1,j}u_{n,j+1} + u_{n,j-1} - 4u_{n,j} = 0 \tag{3.58}$$

The stencil for this Neumann condition is show in Figure 3.3.6 in the bottom right corner. The stencils for the other three edges of a square domain are shown in Figure 3.3.6.

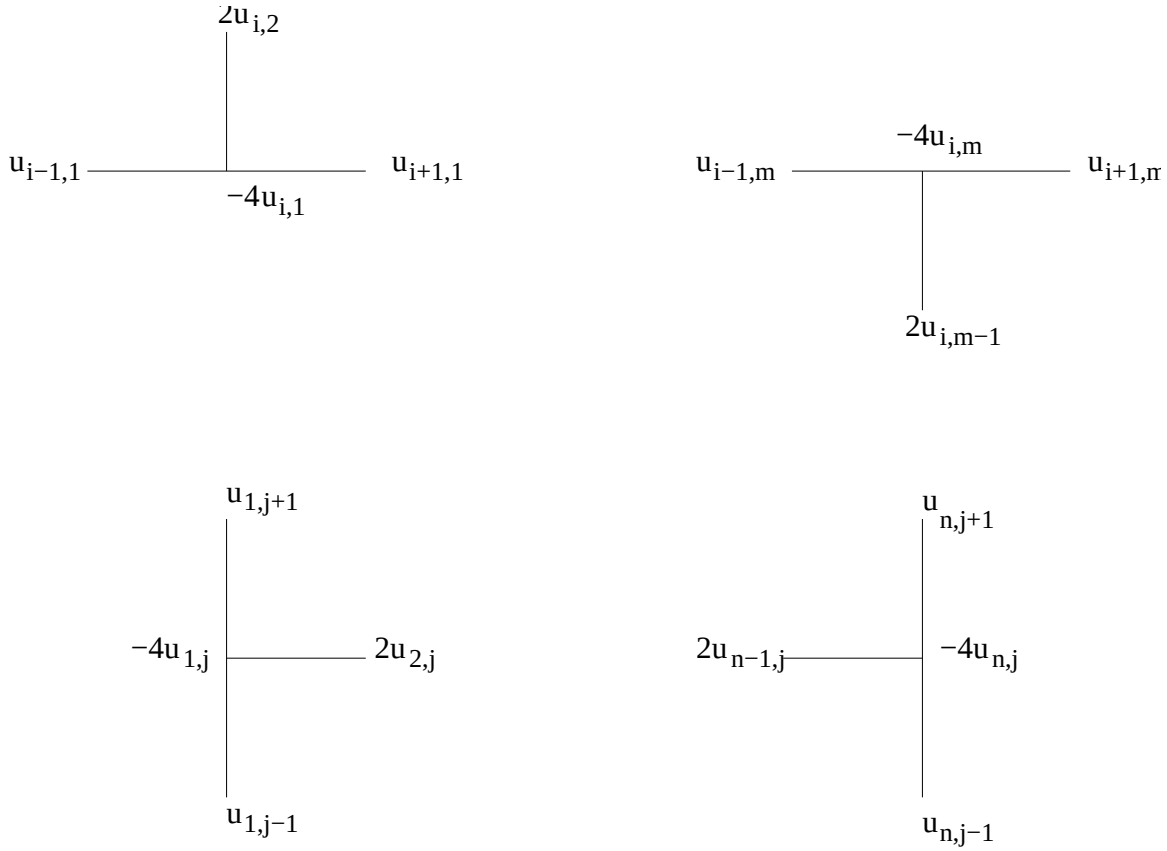


Figure 3.16: Stencils for Neumann no flux boundary conditions applied to a square domain.

While a zero Neumann condition is often the case, it is not always the case, therefore let's look at the general case. To illustrate how to incorporate non-zero Neumann boundary conditions into our finite difference formulation, let's consider our example of solving a heat conduction problem on the unit square [27].

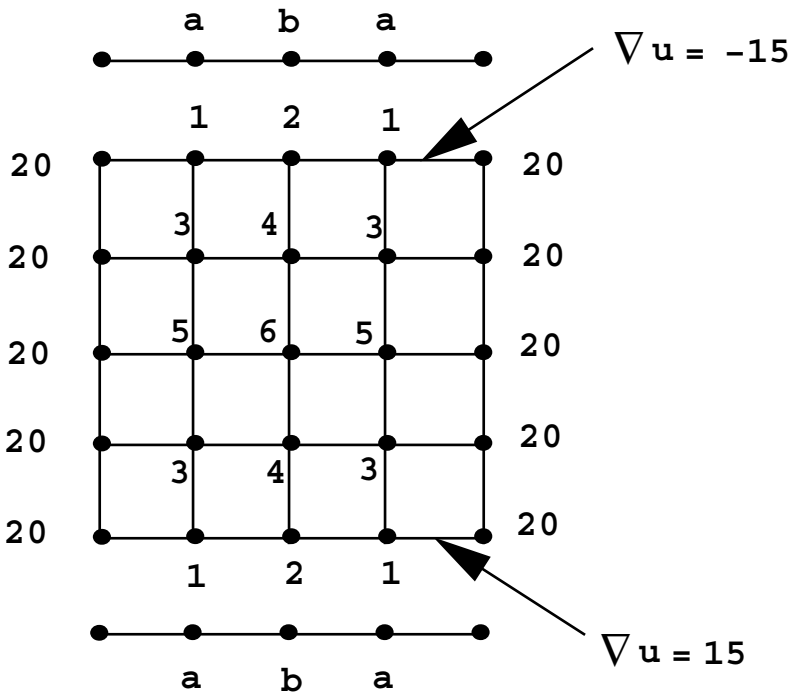


Figure 3.17: Set-up for the the steady-state heat equation with Neuman boundary conditions.

### 3.3.7 Poisson's Equation with Neumann Boundary Data - A Numerical Example

Consider our unit square as a steel plate which is uniformly generating heat at each point a t a rate of $Q = 5cal/cm^3 \cdot sec$. The thermal conductivity of the plate is $k = 0.16$cal/sec $\cdot$ cm$^2$ $\cdot$ C/cm. For this problem, Poisson's equation holds

$$\nabla^2 u = -\frac{Q}{k} \tag{3.59}$$

Assume that the top and bottom faces are perfectly insulated so that no heat is lost, but the upper and lower edges lose heat according to

$$\frac{\partial u}{\partial y} = -15 \text{ C/cm} \tag{3.60}$$

in the outward direction as shown in Figure 3.3.7. Furthermore, the right and left edges are held at a constant temparture $u = 20$C. Our goal is to find the steady-state tempartures within the area of the plate. In this problem, the upper and lower edge temperatures are also unknown, which increases the number of unknowns we need to solve for. Because of the symmetry, we can limit ourselves to a system of $6$ values. We now apply the five-point difference formula to the system. We notice that at the upper and lower edges, we cannot form the five-point combination as there are not enough points outside the boundary. We get around this by extending our mesh to include

an additional row (upper and lower) of exterior points. This gives us the following set of equations:

$$
\begin{aligned}
(20 + u_a + u_2 + u_3 - 4u_1) &= -\frac{(.25)^2 5.0}{0.16}, \\
(u_1 + u_b + u_1 + u_4 - 4u_2) &= -\frac{(.25)^2 5.0}{0.16}, \\
(20 + u_1 + u_4 + u_5 - 4u_3) &= -\frac{(.25)^2 5.0}{0.16}, \\
(u_3 + u_2 + u_3 + u_6 - 4u_4) &= -\frac{(.25)^2 5.0}{0.16}, \\
(20 + u_3 + u_6 + u_3 - 4u_5) &= -\frac{(.25)^2 5.0}{0.16}, \\
(u_5 + u_4 + u_5 + u_4 - 4u_6) &= -\frac{(.25)^2 5.0}{0.16}, \qquad (3.61)
\end{aligned}
$$

(3.62)

Now we apply the Neuman boundary condition utilizing the difference formula for the first partial derivative. Note that we make sure to choose the order of the points so that it reflects that the outward normal is negative, since heat is flowing outwardly.

$$
\begin{aligned}
-(\frac{\partial u}{\partial y})_1 &= \frac{u_3 - u_a}{2(.25)} = 15, \quad u_a = u_3 - 7.5, \\
-(\frac{\partial u}{\partial y})_2 &= \frac{u_4 - u_b}{2(.25)} = 15, \quad u_b = u_4 - 7.5,
\end{aligned}
\qquad (3.63)
$$

We now eliminate the exterior rows of points we added and then solve our system of equations.

Now let's generalize things a bit, the Neumann conditions for Laplace's equation on the unit square can be summarized as,

$$-\frac{\partial u}{\partial x} = f(0, y) \quad \frac{\partial u}{\partial x} = f(1, y) \qquad (3.64)$$

$$-\frac{\partial u}{\partial y} = f(x, 0) \quad \frac{\partial u}{\partial y} = f(x, 1) \qquad (3.65)$$

along the $x$ and $y$ axis respectively. The minus signs denote that the gradients always are pointing outward, away from the boundary. We now apply a finite grid of points to the square (as we did before). Then we apply the five point formula to all of the $(N + 2)^2$ grid points within the domain $\Omega$. When we apply our five point formula to the points on $\partial\Omega$ we note that it involes the set of points: $(ih, -h), (ih, 1 + h), (-h, jh), (1 + h, jh)$, where $i, j = 0, 1, \ldots N + 1$, which lie outside the boundary. We use the derivative boundary conditions to "eliminate" these points outside of the boundary. Recalling the Taylor series approximation for first order partial derivatives,

$$\frac{\partial u}{\partial x} = \frac{[u(x + h, y) - u(x - h, y)]}{2h} + O(h^2) \qquad (3.66)$$

$$\frac{\partial u}{\partial y} = \frac{[u(x, y+h) - u(x, y-h)]}{2h} + O(h^2) \tag{3.67}$$

we use these as replacements for the exterior points in the five point formula,

$$u_{-1,j} = u_{1,j} + 2hf(0, jh) + O(h^3) \tag{3.68}$$

$$u_{N+2,j} = u_{N,j} + 2hf(1, jh) + O(h^3) \tag{3.69}$$

$$u_{i,-1} = u_{i,1} + 2hf(ih, 0) + O(h^3) \tag{3.70}$$

$$u_{i,N+2} = u_{i,N} + 2hf(ih, 1) + O(h^3) \tag{3.71}$$

where $i, j = 0, 1, \ldots, N+1$. These may be used in the five point formula along $x = 0, x = 1, y = 0, y = 1$ respectively.

The $(N+2)^2$ equations which arise from the Neumann problem then can be written as,

$$A\mathbf{U} = 2h\mathbf{f} \tag{3.72}$$

where the matrix $A$ now has the form,

$$A = \begin{pmatrix} T_N & 2I_N & & & & & \\ I_N & T_N & I_N & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & & & & \\ & & & I_N & T_N & I_N \\ & & & & 2I_N & T_N \end{pmatrix} \tag{3.73}$$

Here, $I$ is the identity matrix of order $N+2$ and now $T_N$ is the tridiagonal matrix,

$$T_N = \begin{pmatrix} -4 & 2 & & & & & \\ 1 & -4 & 1 & & & & \\ & 1 & \ddots & \ddots & & & \\ & & \ddots & & & & \\ & & & & & & \\ & & & & 1 & -4 & 1 \\ & & & & & 2 & -4 \end{pmatrix} \tag{3.74}$$

thus the solution to the general Neumann problem is,

$$\mathbf{u} = 2hA^{-1}\mathbf{f}. \tag{3.75}$$

One can prove that the matrix $A$ is singular, and thus, does not have a unique solution. Usually the field problems that we encounter in engineering and science have mixed boundary conditions, i.e. portions of the surface have Neumann boundary conditions, while other portions have Dirichlet boundary conditions, thus rendering a nonsingular $A$ matrix (and unique solution).

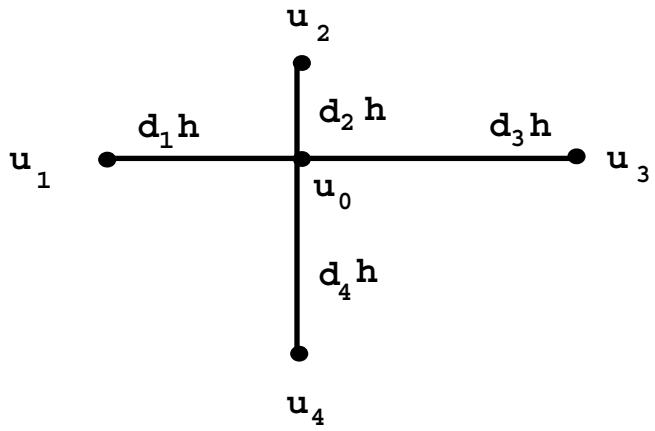


Figure 3.18: Irregular spacing for the five-point finite difference formulation.

### 3.3.8 Irregular Regions

So far we've treated solution domains in which the boundary coincided with nodes from our mesh. For many problems, we will have irregular domains and thus must re-think our approach. One approach to this problem is to continually subdivide our mesh into smaller and smaller grid points such that the uniform mesh can approximate the irregular boundary. While this will usually work, this adds a tremendous amount of additional computation since we must usually add a large number of additional degrees of freedom. Another approach is to redefine our difference formulas to take into account the irregular region. Still other approaches use domain decomposition approaches with meshes of different levels of refinement, interpolation methods, and/or combinations of all the above. Let's explore the option of modifying the difference formulas to account for the irregular boundary.

Consider the case of using the five-point formula but for unequal lengths between nodes as shown in Figure 3.3.8. Along the line from $u_1$ to $u_0$ to $u_3$, we have the following approxmations for the first derivatives:

$$(\frac{\partial u}{\partial x})_{1-0} = \frac{u_0 - u_1}{d_1 h} \tag{3.76}$$

$$(\frac{\partial u}{\partial x})_{0-3} = \frac{u_3 - u_0}{d_3 h} \tag{3.77}$$

Since

$$(\frac{\partial^2 u}{\partial x^2}) = \frac{\partial}{\partial x}(\frac{\partial u}{\partial x}) \tag{3.78}$$

we have

$$(\frac{\partial^2 u}{\partial x^2}) = \frac{(u_3 - u_0)/d_3 h - (u_0 - u_1)/d_1 h}{(d_1 + d_3)h/2}. \tag{3.79}$$

A similar expression exists for $y$. Combining and rearranging, we obtain

$$\nabla^2 u = \frac{2}{h^2}[\frac{u_1}{d_1(d_1 + d_3)} + \frac{u_2}{d_2(d_2 + d_4)} + \frac{u_3}{d_3(d_1 + d_3)} + \frac{u_4}{d_4(d_2 + d_4)} - (\frac{1}{d_1 d_3} + \frac{1}{d_2 d_4})u_0] \tag{3.80}$$

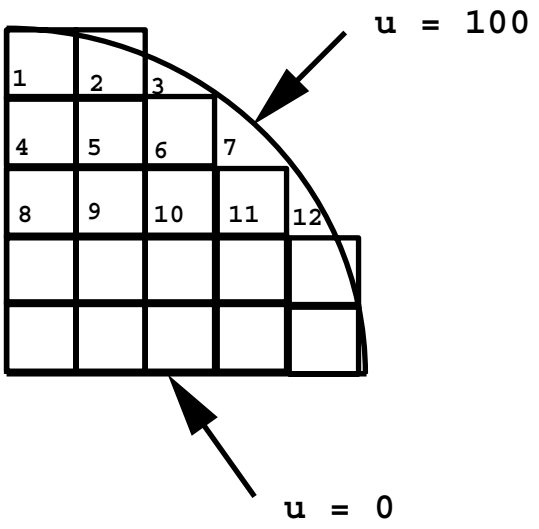


Figure 3.19: Example of a curved boundary.

We then use this new operator for points adjacent to the boundary points when the boundary points do not coincide with the mesh, instead of our standard five-point rule. If there are Neuman boundary conditions on the curved part of the boundary, then we must go back and re-derive the approximation for the Neuman condition (this gets rather invovled) [26].

### 3.3.9 Numerical Example: Curved Boundary

Consider the following example of solving the steady-state heat flow equation on a semicircular plate of radius 1. The temperature at the base is 0 and the temperature on the circumference is 100. The analytic solution is

$$u(r,\theta) = \frac{(4)(100)}{\pi} \sum_{n=1}^{\infty} \frac{1}{2n-1} (\frac{r}{(1)})^{2n-1} \sin(2n-1)\theta \tag{3.81}$$

We now impose a uniform mesh on the semicircular region as shown in Figure 3.3.9. Notice that the mesh points, $u_2, u_3, u_{12}, \ldots$ don't coincide with the boundary. What we do is use the geometry to figure out what the values for $d_2, d_3, \ldots$. It turns out that the short part for $u_2 = .8990h$ and for $u_3 = .5826h$. We continue to apply this method for the rest of the points using our new non-uniform operator. For this particular problem, the finite difference solution is in very good agreement with the analytic solution. As you can imagine, this method would not be very efficient for domains with many different curved regions. A more reasonible way to approach the problem would be to simply use the mesh point which is closest to the point on the boundary - in effect, perturbing the actual region. Obviously one introduces errors by using this method, however, if the distance between the mesh point and boundary is not too large, then the result can be satisfactory.

Now, if you are working on circular or cylindrical boundaries, then you can use a finite difference approximation for Laplace's or Poisson's equation in polar or cylindrical coordinates. For example, consider the two-dimensional Laplace's equation in polar coordinates:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = 0 \tag{3.82}$$

One defines a grid in polar coordinates $(r, \theta) = (i\Delta r, j\Delta\theta)$. Represent the solution $u$ as $u(r_i, \theta_j)$. Then one can use a centered difference representation of equation (3.82). This yields a polar coordinate finite difference version of Laplace's equation.

$$(1 - \frac{1}{2i})u_{i-1,j} + \frac{1}{(i\Delta\theta)^2}u_{i,j-1} - 2[1 + \frac{1}{(i\Delta\theta)^2}]u_{i,j} + (1 + \frac{1}{2i})u_{i+1,j} + \frac{1}{(i\Delta\theta)^2}u_{i,j+1} = 0. \tag{3.83}$$

### 3.3.10 Higher Order Approximation

Thus far, we have discussed the five-point stencil. By including nearest neighbors on a 2D stencil, we can decrease the truncation error and increase the accuracy (at a computational cost, of course). This yields the so-called nine-point finite difference approximation for the Laplacian operator:

$$\nabla^2 u_{i,j} = \frac{1}{6h^2}[u_{i+1,j-1} + u_{i-1,j-1} + u_{i-1,j+1} + 4u_{i+1,j} + 4u_{i-1,j} + 4u_{i,j+1} + 4u_{i,j-1} - 20u_{i,j}] \tag{3.84}$$

Or in stencil format:

$$\nabla^2 u_{i,j} = \frac{1}{6h^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}_h \tag{3.85}$$

The truncation error for the nine-point difference formula is on the order of $O(h^4)$ for Poisson's equation and $O(h^6)$ for Laplace's equation

### 3.3.11 The Laplacian Operator in Three Dimensions

We can extend the finite difference formulation to three-dimensions by just adding the approximation for the z-axis

$$\nabla^2 u_{i,j,k} = \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{h^2} + \frac{u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}}{h^2} + \frac{u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}}{h^2} \tag{3.86}$$

or

$$\nabla^2 u_{i,j,k} = \frac{1}{h^2}[(u_{i+1,j,k} + u_{i-1,j,k}) + (u_{i,j+1,k} + u_{i,j-1,k}) + (u_{i,j,k+1} + u_{i,j,k-1}) - 6u_{i,j,k}] \tag{3.87}$$

Now the value is the arithmetic average of its six nearest neighboring values instead of four in the two-dimensional case.