Chapter 5

Finite difference methods for 2D elliptic PDE

There are many important applications of elliptic PDE, and we now give some examples of linear and then nonlinear equations.

- The 2-D Laplace equation,

\[ u_{xx} + u_{yy} = 0. \]  

(5.1)

The solution \( u \) is sometimes called a potential function, because a conservative vector field \( \mathbf{v} \) (i.e., such that \( \nabla \times \mathbf{v} = 0 \)) is given by \( \mathbf{v} = \nabla u \) (or alternatively \( \mathbf{v} = -\nabla u \)), and we often also have the incompressible constraint \( \nabla \cdot \mathbf{v} = 0 \) such that \( \nabla^2 u = 0 \).

(The gradient operator in 2-D is \( \nabla = [\frac{\partial}{\partial x}, \frac{\partial}{\partial y}]^T \).)

- The 2-D Poisson equation,

\[ u_{xx} + u_{yy} = f. \]  

(5.2)

- The modified Helmholtz equation,

\[ u_{xx} + u_{yy} - \lambda^2 u = f. \]  

(5.3)

Many incompressible flow solvers are based on solving one or several Poisson or Helmholtz equations, e.g., the projection method for solving incompressible Navier-Stokes equations for flow problems \([?, ?]\) at low or modest Reynolds number, or the stream-vorticity formulation method for large Reynolds number \([?, ?]\). In particular, there are some fast Poisson solvers available for regular domains, e.g., in Fishpack \([?]\).

- The Helmholtz equation,

\[ u_{xx} + u_{yy} + \lambda^2 u = f. \]  

(5.4)
The Helmholtz equation arises in scattering problems, when $\lambda$ is the wave number, and the corresponding problem may not have a solution if $\lambda^2$ is an eigenvalue. Further, the problem is hard to solve if $\lambda$ is big.

- General self-adjoint elliptic PDE,

$$\nabla \cdot (a(x,y)\nabla u(x,y)) + \lambda(x,y)u = f(x,y) \quad (5.5)$$

or

$$(au_x)_x + (au_y)_y + \lambda(x,y)u = f(x,y), \quad (5.6)$$

where we should assume that $a(x,y)$ does not change sign in the solution domain, e.g., $a(x,y) \geq a_0 > 0$, where $a_0$ is a constant.

- General elliptic PDE (diffusion and advection equations),

$$a(x,y)u_{xx} + 2b(x,y)u_{xy} + c(x,y)u_{yy}$$

$$+ d(x,y)u_x + e(x,y)u_y + g(x,y)u(x,y) = f(x,y), \quad (x,y) \in \Omega,$$

if $b^2 - ac < 0$ for all $(x,y) \in \Omega$. This equation can be re-written as

$$\nabla \cdot (a(x,y)\nabla u(x,y)) + w(x,y) \cdot \nabla u + c(x,y)u = f(x,y) \quad (5.7)$$

after a transformation, where $w(x,y)$ is a vector.

- Diffusion and reaction equation,

$$\nabla \cdot (a(x,y)\nabla u(x,y)) = f(u). \quad (5.8)$$

Here $\nabla \cdot (a(x,y)\nabla u(x,y))$ is the diffusion term, the typically nonlinear $f(u)$ is the reaction term, and if $a(x,y) \equiv 1$ the PDE is called a nonlinear Poisson equation.

- $p$-Laplacian equation,

$$\nabla \cdot (|Du|^{p-2}Du) = 0, \quad p \geq 2. \quad (5.9)$$

- Minimal surface equation,

$$\nabla \cdot \left( \frac{Du}{\sqrt{1 + |Du|^2}} \right) = 0. \quad (5.10)$$

We note that an elliptic PDE ($P(\frac{\partial}{\partial x}, \frac{\partial}{\partial y})u = 0$) can be regarded as the steady state solution of a corresponding parabolic PDE ($u_t = P(\frac{\partial}{\partial x}, \frac{\partial}{\partial y})u$). Furthermore, if a linear PDE is defined on a rectangle domain then a finite difference approximation (in each dimension) can be used for both the equation and the boundary conditions, but the most difficult part is to solve the resulting linear system of algebraic equations efficiently.
5.1 Boundary and compatibility conditions

Let us consider a 2-D second-order elliptic PDE on a domain $\Omega$, with boundary $\partial \Omega$ and $\mathbf{n}$ the unit normal directed according to the “right side rule”, cf., Figure 5.1. Some common boundary conditions are as follows.

Figure 5.1: A diagram of a two dimensional domain $\Omega$, its boundary $\partial \Omega$ and its unit normal direction.

- Dirichlet boundary condition: the solution is known on the boundary,

$$u(x, y)|_{\partial \Omega} = u_0(x, y).$$

- Neumann or flux boundary condition: the normal derivative is given along the boundary,

$$\frac{\partial u}{\partial \mathbf{n}} \equiv \mathbf{n} \cdot \nabla u = u_n = u_x n_x + u_y n_y = g(x, y),$$

where $\mathbf{n} = (n_x, n_y)$ is the unit normal.

- Mixed boundary condition:

$$\left( \alpha(x, y)u(x, y) + \beta(x, y) \frac{\partial u}{\partial \mathbf{n}} \right)|_{\partial \Omega} = \gamma(x, y)$$

is given along the boundary $\partial \Omega$.

- In some cases, a boundary condition is periodic, e.g., for $\Omega = [a, b] \times [c, d]$, $u(a, y) = u(b, y)$ periodic in the $x$-direction and $u(x, c) = u(x, d)$ periodic in the $y$-direction.

There can be different boundary conditions on different parts of the boundary, e.g., for a channel flow in a domain $[a, b] \times [c, d]$, the flux boundary condition may apply at $x = a$ and $x = b$, and a no-slip boundary condition $\mathbf{u} = 0$ at the boundaries $y = c$ and $y = d$. 
For a Poisson equation with a purely Neumann boundary condition, there is no solution unless a compatibility condition is satisfied. Consider the following problem:

\[ \Delta u = f(x, y), \quad (x, y) \in \Omega, \quad \frac{\partial u}{\partial n} \mid \partial \Omega = g(x, y). \]

On integrating over the domain \( \Omega \)

\[ \iint_{\Omega} \Delta u \, dx \, dy = \iint_{\Omega} f(x, y) \, dx \, dy, \]

and applying the Green’s theorem gives

\[ \iint_{\Omega} \Delta u \, dx \, dy = \oint_{\partial \Omega} \frac{\partial u}{\partial n} \, ds, \]

so we have the compatibility condition

\[ \iint_{\Omega} \Delta u \, dx \, dy = \oint_{\partial \Omega} g \, ds = \iint_{\Omega} f(x, y) \, dx \, dy \tag{5.1} \]

for the solution to exist. If the compatibility condition is satisfied and \( \partial \Omega \) is smooth, then the solution does exist but it is not unique. Indeed, \( u(x, y) + C \) is a solution for arbitrary constant \( C \) if \( u(x, y) \) is a solution, but we can specify the solution at a particular point (e.g. \( u(x_0, y_0) = 0 \)) to render it well defined.

5.2 The central finite difference method for Poisson equations

Let us now consider the following problem, involving a Poisson equation and Dirichlet BC:

\[ u_{xx} + u_{yy} = f(x, y), \quad (x, y) \in \Omega = (a, b) \times (c, d), \tag{5.1} \]
\[ u(x, y) \mid \partial \Omega = u_0(x, y). \tag{5.2} \]

If \( f \in L^2(\Omega) \), then the solution \( u(x, y) \in C^2(\Omega) \) exists and it is unique. An analytic solution is often unavailable, and the FD procedure is explained below.

- Step 1: Generate a grid. For example, a uniform Cartesian grid can be generated:

\[ x_i = a + ih_x, \quad i = 0, 1, 2, \ldots, m, \quad h_x = \frac{b - a}{m}, \tag{5.3} \]
\[ y_j = c + jh_y, \quad j = 0, 1, 2, \ldots, n, \quad h_y = \frac{d - c}{n}. \tag{5.4} \]

In seeking an approximate solution \( U_{ij} \) at the grid points \((x_i, y_j)\) where \( u(x, y) \) is unknown, there are \((m - 1)(n - 1)\) unknowns.
FD methods for linear parabolic PDE

• Step 2: Represent the partial derivatives with FD formulas involving the function values at the grid points. For example, if we adopt the three-point central FD formula for second-order partial derivatives in the $x$- and $y$-directions respectively, then

$$u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j) \left( \frac{h_x}{2} \right)^2 + u(x_i, y_{j-1}) - 2u(x_i, y_j) + u(x_i, y_{j+1}) \left( \frac{h_y}{2} \right)^2 = f_{ij} + T_{ij}, \quad i = 1, \ldots, m - 1, \quad j = 1, \ldots, n - 1, \quad (5.5)$$

where $f_{ij} = f(x_i, y_j)$. The local truncation error satisfies

$$T_{ij} \sim \left( \frac{h_x}{12} \right)^2 \frac{\partial^4 u}{\partial x^4} (x_i, y_j) + \left( \frac{h_y}{12} \right)^2 \frac{\partial^4 u}{\partial y^4} (x_i, y_j) + O(h^4), \quad (5.6)$$

where

$$h = \max \{ h_x, h_y \}. \quad (5.7)$$

and the finite difference discretization is consistent if

$$\lim_{h \to 0} \| T \| = 0, \quad (5.8)$$

so this FD discretization is consistent and second-order accurate.

We ignore the error term in Eq. (5.5) and replace the exact solution values $u(x_i, y_j)$ at the grid points with the approximate solution values $U_{ij}$ obtained from solving the linear system of algebraic equations, i.e.,

$$\frac{U_{i-1,j} + U_{i+1,j}}{(h_x)^2} + \frac{U_{i,j-1} + U_{i,j+1}}{(h_y)^2} - \left( \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} \right) U_{ij} = f_{ij}, \quad (5.9)$$

$$i = 1, 2, \cdots, m - 1, \quad j = 1, 2, \cdots, n - 1.$$

The FD equation at the grid point $(x_i, y_j)$ involves five grid points in a five-point stencil, $(x_{i-1}, y_j), (x_{i+1}, y_j), (x_i, y_{j-1}), (x_i, y_{j+1})$, and $(x_i, y_j)$. The grid points in the FD stencil are sometimes labeled east, north, west, south, and the center in the literature. The center $(x_i, y_j)$ is called the master grid point, where the FD equation is used to approximate the partial differential equation.

• Solve the linear system of algebraic equations (5.9), to get the approximate values for the solution at all of the grid points.

• Error analysis, implementation, visualization etc.

5.2.1 The matrix-vector form of the FD equations

In solving the algebraic system of equations by a direct method such as Gaussian elimination or some sparse matrix technique, knowledge of the matrix-vector structure is important.
although less so for an iterative solver such as the Jacobi, Gauss-Seidel or SOR(\(\omega\)) methods. In the matrix-vector form \(AU = F\), the unknown \(U\) is a 1-D array. From 2-D Poisson equations the unknowns \(U_{ij}\) are a 2-D array, but we can order it to get a 1-D array. We may also need to re-order the FD equations, and it is a common practice to use the same ordering for the equations as for the unknown array. There are two commonly used orderings, namely, the natural ordering, a natural choice for sequential computing; and red-black ordering, considered to be a good choice for parallel computing.

\[
\begin{array}{ccc}
7 & 8 & 9 \\
4 & 5 & 6 \\
1 & 2 & 3
\end{array}
\quad
\begin{array}{ccc}
4 & 9 & 5 \\
7 & 3 & 8 \\
1 & 6 & 2
\end{array}
\]

Figure 5.2: The natural ordering (left) and the red-black ordering (right).

**The natural row ordering**

In the natural row ordering, we order the unknowns and equations row by row. Thus the \(k\)-th FD equation corresponding to \((i, j)\) has the following relation:

\[
k = i + (m - 1)(j - 1), \quad i = 1, 2, \cdots, m - 1, \quad j = 1, 2, \cdots, n - 1.
\]  

(5.10)

see the left diagram in Figure 5.2.

Referring to Figure 5.2, suppose that \(h_x = h_y = h\), \(m = n = 4\). Then there are nine equations and nine unknowns, so the coefficient matrix is 9 by 9. To write down the matrix-vector form, use a 1-D array \(x\) to express the unknown \(U_{ij}\) according to the ordering, we should have

\[
\begin{align*}
x_1 &= U_{11}, & x_2 &= U_{21}, & x_3 &= U_{31}, & x_4 &= U_{12}, & x_5 &= U_{22}, \\
x_6 &= U_{32}, & x_7 &= U_{13}, & x_8 &= U_{23}, & x_9 &= U_{33}.
\end{align*}
\]  

(5.11)

Now if the algebraic equations are ordered in the same way as the unknowns, the nine
equations from the standard central FD scheme using the five-point stencil are

\begin{align*}
\text{Eqn.}\ 1 : & \quad \frac{1}{h^2} (-4x_1 + x_2 + x_4) = f_{11} - \frac{u_{01} + u_{10}}{h^2} \\
\text{Eqn.}\ 2 : & \quad \frac{1}{h^2} (x_1 - 4x_2 + x_3 + x_5) = f_{21} - \frac{u_{20}}{h^2} \\
\text{Eqn.}\ 3 : & \quad \frac{1}{h^2} (x_2 - 4x_3 + x_6) = f_{31} - \frac{u_{30} + u_{41}}{h^2} \\
\text{Eqn.}\ 4 : & \quad \frac{1}{h^2} (x_1 - 4x_4 + x_5 + x_7) = f_{12} - \frac{u_{02}}{h^2} \\
\text{Eqn.}\ 5 : & \quad \frac{1}{h^2} (x_2 + x_4 - 4x_5 + x_6 + x_8) = f_{22} \\
\text{Eqn.}\ 6 : & \quad \frac{1}{h^2} (x_3 + x_5 - 4x_6 + x_9) = f_{32} - \frac{u_{03} + u_{14}}{h^2} \\
\text{Eqn.}\ 7 : & \quad \frac{1}{h^2} (x_4 - 4x_7 + x_8) = f_{13} - \frac{u_{03} + u_{14}}{h^2} \\
\text{Eqn.}\ 8 : & \quad \frac{1}{h^2} (x_5 + x_7 - 4x_8 + x_9) = f_{23} - \frac{u_{02}}{h^2} \\
\text{Eqn.}\ 9 : & \quad \frac{1}{h^2} (x_6 + x_8 - 4x_9) = f_{33} - \frac{u_{34} + u_{43}}{h^2}.
\end{align*}

The corresponding coefficient matrix is \textit{block tridiagonal},

\[
A = \frac{1}{h^2} \begin{bmatrix}
B & I & 0 \\
I & B & I \\
0 & I & B
\end{bmatrix},
\tag{5.12}
\]

where \(I\) is the \(3 \times 3\) identity matrix and

\[
B = \begin{bmatrix}
-4 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & -4
\end{bmatrix}.
\]

In general, for an \(n + 1\) by \(n + 1\) grid we obtain

\[
A = \frac{1}{h^2} \begin{bmatrix}
B & I & & & \\
I & B & I & & \\
& I & B & I & \\
& & & \ddots & \ddots \\
& & & & I & B
\end{bmatrix}_{n^2 \times n^2}, \quad B = \begin{bmatrix}
-4 & 1 & & & \\
1 & -4 & 1 & & \\
& 1 & -4 & 1 & \\
& & \ddots & \ddots & \ddots \\
& & & 1 & -4
\end{bmatrix}_{n^2 \times n^2}.
\]

Since \(-A\) is symmetric positive definite and weakly diagonally dominant, the coefficient matrix \(A\) is a non-singular, and hence the solution of the system of the FD equations is unique.
The matrix-vector form is useful to understand the structure of the linear system of algebraic equations, and as mentioned it is required when a direct method (such as Gaussian elimination or a sparse matrix technique) is used to solve the system. However, it can sometimes be more convenient to use a two-parameter system, especially when an iterative method is preferred but also as more intuitive and to visualize the data. The eigenvalues and eigenvectors of $A$ can also be indexed by two parameters $p$ and $k$, corresponding to wave numbers in the $x$ and $y$ directions. The $(p,k)$-th eigenvector $u^{p,k}$ has $n^2$ components,

$$u_{ij}^{p,k} = \sin(p\pi ih) \sin(k\pi jh), \quad i, j = 1, 2, \cdots, n$$  \hspace{1cm} (5.13)

for $p, k = 1, 2, \cdots, n$; and the corresponding $(p,k)$-th eigenvalue is

$$\lambda^{p,k} = \frac{2}{h^2} \left( \cos(p\pi h) - 1 \right) + \cos(k\pi h) - 1.$$  \hspace{1cm} (5.14)

The least dominant (smallest magnitude) eigenvalue is

$$\lambda^{1,1} = -2\pi^2 + O(h^2),$$  \hspace{1cm} (5.15)

obtained from the Taylor expansion of (5.14) in terms of $h \sim 1/n$; and the dominant (largest magnitude) eigenvalue is

$$\lambda^{\text{int}(n/2),\text{int}(n/2)} \sim -\frac{8}{h^2}.$$  \hspace{1cm} (5.16)

It is notable that the dominant and least dominant eigenvalues are twice the magnitude of those in 1-D representation, so we have the following estimates:

$$\|A\|_2 \sim \max |\lambda^{p,k}| = \frac{8}{h^2}, \quad \|A^{-1}\|_2 = \frac{1}{\min |\lambda^{p,k}|} \sim \frac{1}{2\pi^2},$$  \hspace{1cm} (5.17)

$$\text{cond}_2(A) = \|A\|_2\|A^{-1}\|_2 \sim \frac{4}{\pi^2 h^2} = O(n^2).$$

Note that the condition number is about the same as that in the 1-D case; and since it can be large, double precision is recommended to reduce the effect of round-off errors.

### 5.3 Finite difference methods for general second-order elliptic PDE

If the domain of the interest is a rectangle $[a, b] \times [c, d]$ and there is no mixed derivative term $u_{xy}$ in the PDE, then the PDE can be discretized dimension by dimension. Consider the following example:

$$\nabla \cdot (p(x, y) \nabla u) - q(x, y) u = f(x, y), \quad \text{or} \quad (pu_x)_x + (pu_y)_y - qu = f,$$

with a Dirichlet boundary condition at $x = b$, $y = c$, and $y = d$ but a Neumann boundary condition $u_x = g(y)$ at $x = a$. 

For simplicity, let us adopt a uniform Cartesian grid again

\[ x_i = a + ih_x, \quad i = 0, 1, \ldots, m, \quad h_x = \frac{b - a}{m}, \]
\[ y_j = c + jh_y, \quad j = 0, 1, \ldots, n, \quad h_y = \frac{d - c}{n}. \]

If we discretize the PDE dimension by dimension, at a typical grid point \((x_i, y_j)\) the FD equation is

\[
\frac{p_{i+\frac{1}{2}, j} U_{i+1,j} - (p_{i+\frac{1}{2}, j} + p_{i-\frac{1}{2}, j}) U_{ij} + p_{i-\frac{1}{2}, j} U_{i-1,j}}{(h_x)^2} + \frac{p_{i,j+\frac{1}{2}} U_{i,j+1} - (p_{i,j+\frac{1}{2}} + p_{i,j-\frac{1}{2}}) U_{ij} + p_{i,j-\frac{1}{2}} U_{i,j-1}}{(h_y)^2} - q_{ij} U_{ij} = f_{ij}
\]

(5.1) for \(i = 1, 2, \ldots, m - 1\) and \(j = 1, 2, \ldots, n - 1\), where \(p_{\pm\frac{1}{2}, j} = p(x_i \pm h_x/2, y_j)\) and so on.

For the indices \(i = 0, j = 1, 2, \ldots, n - 1\), we can use the ghost point method to deal with the Neumann boundary condition. Using the central FD scheme for the flux boundary condition

\[ \frac{U_{1,j} - U_{-1,j}}{2h_x} = g(y_j), \quad \text{or} \quad U_{-1,j} = U_{1,j} - 2h_x g(y_j), \quad j = 1, 2, \ldots, n - 1, \]

on substituting into the FD equation at \((0, j)\), we obtain

\[
\frac{(p_{-\frac{1}{2}, j} + p_{\frac{1}{2}, j}) U_{1,j} - (p_{\frac{1}{2}, j} + p_{-\frac{1}{2}, j}) U_{0,j}}{(h_x)^2} + \frac{p_{0,j+\frac{1}{2}} U_{0,j+1} - (p_{0,j+\frac{1}{2}} + p_{0,j-\frac{1}{2}}) U_{0,j} + p_{0,j-\frac{1}{2}} U_{0,j-1}}{(h_y)^2} - q_{0,j} U_{0,j} = f_{0,j} + \frac{2 p_{-\frac{1}{2}, j} g(y_j)}{h_x}.
\]

(5.2)

For a general second-order elliptic PDE with no mixed derivative term \(u_{xy}\), i.e.,

\[ \nabla \cdot (p(x,y)\nabla u) + w \cdot \nabla u - q(x,y)u = f(x,y), \]

the central FD scheme when \(|w| \ll 1/h\) can be used, but an upwinding scheme may be preferred to deal with the advection term \(w \cdot \nabla u\).

**A finite difference formula for approximating the mixed derivative \(u_{xy}\)**

If there is a mixed derivative term \(u_{xy}\), we cannot proceed dimension by dimension but a centered FD scheme (3.5) for \(u_{xy}\) can be used, i.e.,

\[
u_{xy}(x_i, y_j) \approx \frac{u(x_{i-1}, y_{j-1}) + u(x_{i+1}, y_{j+1}) - u(x_{i+1}, y_{j-1}) - u(x_{i-1}, y_{j+1})}{4h_x h_y}. \]

(5.3)
From a Taylor expansion at \((x, y)\), this FD formula can be shown to be consistent and the discretization is second-order accurate, and the consequent central FD formula for a second-order linear PDE involves nine grid points. The resulting linear system of algebraic equations for PDE is more difficult to solve, because it is no longer symmetric nor diagonally dominant. Furthermore, there is no known upwinding scheme to deal with the PDE with mixed derivatives.

### 5.4 Solving the resulting linear system of algebraic equations

The linear systems of algebraic equations resulting from FD discretizations for two or higher dimensional problems are often very large, e.g., the linear system from an \(n \times n\) grid for an elliptic PDE has \(O(n^2)\) equations, so the coefficient matrix is \(O(n^2 \times n^2)\). Even for \(n = 100\), a modest number, the \(O(10^4 \times 10^4)\) matrix cannot be stored in most modern computers if the desirable double precision is used. However, the matrix from a self-adjoint elliptic PDE is sparse since the non-zero entries are about \(5n^2\), so an iterative method or sparse matrix technique may be used. For an elliptic PDE defined on a rectangle domain or a disk, frequently used methods are listed below.

- Fast Poisson solvers such as the fast Fourier transform (FFT) or cyclic reduction \([?]\). Usually the implementation is not so easy, and the use of existing software packages is recommended, e.g., Fishpack, written in Fortran and free on the Netlib.
- Multigrid solvers, either structured multigrid, e.g., MGD9V \([?]\) that uses a nine-point stencil, or AMGs (algebraic multi-grid solvers).
- Sparse matrix techniques.
- Simple iterative methods such as Jacobi, Gauss-Seidel, SOR(\(\omega\)). They are easy to implement, but often slow to converge.
- Other iterative methods such as the conjugate gradient (CG) or pre-conditioned conjugate gradient (PCG), generalized minimized residual (GMRES), conjugate gradient (CG) or biconjugate gradient (BICG) method for non-symmetric system of equations.

We refer the reader to \([?]\) for more information and references.

An important advantage of an iterative method is that zero entries play no role in the matrix-vector multiplications involved and there is no need to manipulate the matrix and vector forms, as the algebraic equations in the system are used directly. Given a linear system of equation \(Ax = b\) where \(A\) is non-singular \((\text{det}(A) \neq 0)\). If \(A\) can be written as \(A = M - N\) where \(M\) is an invertible matrix, then \((M - N)x = b\) or \(Mx = Nx + b\) or \(x = M^{-1}Nx + M^{-1}b\). We may iterate starting from an initial guess \(x^0\), via

\[
x^{k+1} = M^{-1}Nx^k + M^{-1}b, \quad k = 0, 1, 2, \ldots \tag{5.1}
\]
and the iteration converges or diverges depending the spectral radius of \( \rho(M^{-1}N) = \max |\lambda_i(M^{-1}N)| \). (Incidentally, if \( T = M^{-1}N \) is a constant matrix, the iterative method is called stationary.)

### 5.4.1 The Jacobi iterative method (solving the diagonals)

Solving for \( x_1 \) from the first equation in the algebraic system, \( x_2 \) from the second, and so forth, we have

\[
\begin{align*}
  x_1 &= \frac{1}{a_{11}} \left( b_1 - a_{12}x_2 - a_{13}x_3 \cdots - a_{1n}x_n \right) \\
  x_2 &= \frac{1}{a_{22}} \left( b_2 - a_{21}x_1 - a_{23}x_3 \cdots - a_{2n}x_n \right) \\
  \vdots &= \vdots \vdots \vdots \\
  x_i &= \frac{1}{a_{ii}} \left( b_i - a_{i1}x_1 - a_{i2}x_2 \cdots - a_{i,i-1}x_{i-1} - a_{i,i+1}x_{i+1} - \cdots - a_{in}x_n \right) \\
  \vdots &= \vdots \vdots \vdots \\
  x_n &= \frac{1}{a_{nn}} \left( b_n - a_{n1}x_1 - a_{n2}x_2 \cdots - a_{n,n-1}x_{n-1} \right).
\end{align*}
\]

Given some initial guess \( x^0 \), the corresponding Jacobi iterative method is

\[
\begin{align*}
  x_1^{k+1} &= \frac{1}{a_{11}} \left( b_1 - a_{12}x_2^k - a_{13}x_3^k \cdots - a_{1n}x_n^k \right) \\
  x_2^{k+1} &= \frac{1}{a_{22}} \left( b_2 - a_{21}x_1^k - a_{23}x_3^k \cdots - a_{2n}x_n^k \right) \\
  \vdots &= \vdots \vdots \vdots \\
  x_i^{k+1} &= \frac{1}{a_{ii}} \left( b_i - a_{i1}x_1^k - a_{i2}x_2^k \cdots - a_{in}x_n^k \right) \\
  \vdots &= \vdots \vdots \vdots \\
  x_n^{k+1} &= \frac{1}{a_{nn}} \left( b_n - a_{n1}x_1^k - a_{n2}x_2^k \cdots - a_{n,n-1}x_{n-1}^k \right).
\end{align*}
\]

It can be written compactly as

\[
x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1,j \neq i}^n a_{ij}x_j^k \right), \quad i = 1, 2, \ldots, n, \quad (5.2)
\]

which is the basis for easy programming. Thus for the FD equations

\[
\frac{U_{i+1} - 2U_i + U_{i+1}}{h^2} = f_i
\]
with Dirichlet boundary conditions $U_0 = ua$ and $U_n = ub$, we have

\[
U_1^{k+1} = \frac{ua + U_2^k}{2} - \frac{h^2 f_1}{2},
\]
\[
U_i^{k+1} = \frac{U_{i-1}^k + U_{i+1}^k}{2} - \frac{h^2 f_i}{2}, \quad i = 2, 3, \ldots n - 1
\]
\[
U_{n-1}^{k+1} = \frac{U_n^k + ub}{2} - \frac{h^2 f_{n-1}}{2} ;
\]

and for a two dimensional Poisson equation,

\[
U_{i,j}^{k+1} = \frac{U_{i-1,j}^k + U_{i+1,j}^k + U_{i,j-1}^k + U_{i,j+1}^k}{4} - \frac{h^2 f_{i,j}}{4}, \quad i, j = 1, 2, \ldots, n - 1.
\]

### 5.4.2 The Gauss-Seidel iterative method (solving the diagonals and using the most updated information)

In the Jacobi iterative method, all components $x^{k+1}$ are updated based on $x^k$, whereas in the Gauss-Seidel iterative method the *most updated* information is used as follows:

\[
x_1^{k+1} = \frac{1}{a_{11}} \left( b_1 - a_{12} x_2^k - a_{13} x_3^k - \cdots - a_{1n} x_n^k \right)
\]
\[
x_2^{k+1} = \frac{1}{a_{22}} \left( b_2 - a_{21} x_1^k - a_{23} x_3^k - \cdots - a_{2n} x_n^k \right)
\]
\[
\vdots \quad \vdots \quad \vdots 
\]
\[
x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - a_{i1} x_1^{k+1} - a_{i2} x_2^{k+1} - \cdots - a_{i,i-1} x_{i-1}^{k+1} - a_{i,i+1} x_{i+1}^{k+1} - \cdots - a_{in} x_n^k \right)
\]
\[
\vdots \quad \vdots \quad \vdots
\]
\[
x_n^{k+1} = \frac{1}{a_{nn}} \left( b_n - a_{n1} x_1^{k+1} - a_{n2} x_2^{k+1} - \cdots - a_{n,n-1} x_{n-1}^{k+1} \right),
\]

or in a compact form

\[
x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij} x_j^k \right), \quad i = 1, 2, \ldots, n. \tag{5.3}
\]

Below is a pseudo-code, where the Gauss-Seidel iterative method is used to solve the FD equations for the Poisson equation, assuming $u_0$ is an initial guess:

```plaintext
% Give u0(i,j) and a tolerance tol, say 1e-6.

err = 1000; k = 0; u = u0;
while err > tol
    for i=1:n
        for j=1:n
```
\[ u(i,j) = \left( (u(i-1,j)+u(i+1,j)+u(i,j-1)+u(i,j+1)) - h^2 f(i,j) \right) / 4; \]

end

end

err = max(max(abs(u-u0)));

u0 = u; k = k + 1; % Next iteration if err > tol
end

Note that this pseudo-code has a framework generally suitable for iterative methods.

5.4.3 The successive over-relaxation (SOR(\(\omega\))) method: an extrapolation technique

Suppose \(x_{GS}^{k+1}\) denotes the update from \(x^k\) in the Gauss-Seidel method. Intuitively one may anticipate the update

\[ x^{k+1} = (1-\omega)x^k + \omega x_{GS}^{k+1}, \tag{5.4} \]

a linear combination of \(x^k\) and \(x_{GS}^{k+1}\), may give a better approximation for a suitable choice of the relaxation parameter \(\omega\). If the parameter \(\omega < 1\), the combination above is an interpolation, and if \(\omega > 1\) it is an extrapolation or over-relaxation. For \(\omega = 1\), we recover the Gauss-Seidel method. For elliptic problems, we usually choose \(1 \leq \omega < 2\). In component form, the SOR(\(\omega\)) method can be represented as

\[
\begin{align*}
x_i^{k+1} = (1-\omega)x_i^k + \omega \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k \right),
\end{align*}
\tag{5.5}
\]

for \(i = 1, 2, \cdots, n\). Therefore, only one line in the pseudo-code of the Gauss-Seidel method above need be changed, namely,

\[
\begin{align*}
  u(i,j) &= (1-\text{omega})*u0(i,j) + \text{omega}*(u(i-1,j) + u(i+1,j) \\
  &
  + u(i,j-1) + u(i,j+1) - h^2 f(i,j)) / 4
\end{align*}
\]

The convergence of the SOR(\(\omega\)) method depends on the choice of \(\omega\). For the linear system of algebraic equations obtained from the standard five-point stencil applied to a Poisson equation with \(h = h_x = h_y = 1/n\), it can be shown that the optimal \(\omega\) is

\[
\omega_{opt} = \frac{2}{1 + \sin(\pi/n)} \sim \frac{2}{1 + \pi/n}, \tag{5.6}
\]

which approaches two as \(n\) approaches infinity. Although the optimal \(\omega\) is unknown for general elliptic PDE, we can use the optimal \(\omega\) for the Poisson equation as a trial value, and in fact larger rather than smaller \(\omega\) values are recommended. If \(\omega\) is so large that the iterative method diverges, this is soon evident because the solution will “blow-up”. Incidentally, the optimal choice of \(\omega\) is independent of the right-hand side.
5.4.4 Convergence of stationary iterative methods

For a stationary iterative method, the following theorem provides a necessary and sufficient condition for convergence.

Theorem 5.1 Given a stationary iteration
\[ \mathbf{x}^{k+1} = T\mathbf{x}^k + c, \] (5.7)
where \( T \) is a constant matrix and \( c \) is a constant vector, the vector sequence \( \{\mathbf{x}^k\} \) converges for arbitrary \( \mathbf{x}^0 \) if and only if \( \rho(T) < 1 \) where \( \rho(T) \) is the spectral radius of \( T \) defined as
\[ \rho(T) = \max |\lambda_i(T)|, \] (5.8)
i.e., the largest magnitude of all the eigenvalues of \( T \).

Another commonly used sufficient condition to check the convergence of a stationary iterative method is given in the following theorem.

Theorem 5.2 If there is a matrix norm \( \|\cdot\| \) such that \( \|T\| < 1 \), then the stationary iterative method converges for arbitrary initial guess \( \mathbf{x}^0 \).

We often check whether \( \|T\|_p < 1 \) for \( p = 1, 2, \infty \), and if there is just one norm such that \( \|T\| < 1 \), then the iterative method is convergent. However, if \( \|T\| \geq 1 \) there is no conclusion about the convergence.

Let us now briefly discuss the convergence of the Jacobi, Gauss-Seidel, and SOR(\( \omega \)) methods. Given a linear system \( A\mathbf{x} = \mathbf{b} \), let \( D \) denote the diagonal matrix formed from the diagonal elements of \( A \), \( -L \) the lower triangular part of \( A \), and \( -U \) the upper triangular part of \( A \). The iteration matrices for the three basic iteration methods are thus

- Jacobi method: \( T = D^{-1}(L + U) \), \( c = D^{-1}\mathbf{b} \).
- Gauss-Seidel method: \( T = (D - L)^{-1}U \), \( c = (D - L)^{-1}\mathbf{b} \).
- SOR(\( \omega \)) method: \( T = (I - \omega L)^{-1}((1 - \omega)I + \omega U) \), \( c = \omega(I - \omega L)^{-1}D^{-1}\mathbf{b} \).

Theorem 5.3 If \( A \) is strictly row diagonally dominant, i.e.,
\[ |a_{ii}| > \sum_{j=1,j\neq n}^n |a_{ij}|, \] (5.9)
then both the Jacobi and Gauss-Seidel iterative methods converge. The conclusion is also true when (1) \( A \) is weakly row diagonally dominant
\[ |a_{ii}| \geq \sum_{j=1,j\neq n}^n |a_{ij}|; \] (5.10)
(2): the inequality holds for at least one row; (3) \( A \) is irreducible.
We refer the reader to [?] for the definition of irreducibility. From this theorem, both the Jacobi and Gauss-Seidel iterative methods converge when they are applied to the linear system of algebraic equations obtained from the standard central FD method for Poisson equations.

5.5 A finite difference method for Poisson equations in polar coordinates

If the domain of interest is a circle, an annulus, or a fan etc., (cf., Figure 5.3 for some illustrations), it is natural to use plane polar coordinates

\[ x = r \cos \theta, \quad y = r \sin \theta. \] (5.1)

The Poisson equation in these coordinates is

\[ \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial u}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 u}{\partial \theta^2} = f(r, \theta), \text{ conservative form} \]

or

\[ \frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} + \frac{1}{r^2} \frac{\partial^2 u}{\partial \theta^2} = f(r, \theta). \]

Figure 5.3: Diagrams of domains and boundary conditions that may be better solved in polar coordinates.
For $0 < R_1 \leq r \leq R_2$ and $\theta_l \leq \theta \leq \theta_r$, where the origin is not in the domain of interest, on a uniform grid in plane polar coordinates

$$r_i = R_1 + i\Delta r, \quad i = 0, 1, \cdots, m, \quad \Delta r = \frac{R_2 - R_1}{m},$$

$$\theta_j = \theta_l + j\Delta \theta, \quad j = 0, 1, \cdots, N, \quad \Delta \theta = \frac{\theta_r - \theta_l}{N},$$

the discretized finite difference equation (in conservative form) is

$$\begin{align*}
&\frac{1}{r_i} \left( r_{i-\frac{1}{2}} U_{i-1,j} - (r_{i-\frac{1}{2}} + r_{i+\frac{1}{2}}) U_{ij} + r_{i+\frac{1}{2}} U_{i+1,j} \right) \\
&\quad + \frac{1}{r_i^2} U_{ij} - 2U_{ij} + U_{i,j+1} = f(r_i, \theta_j),
\end{align*}
$$

where again $U_{ij}$ is an approximation to the solution $u(r_i, \theta_j)$.

### 5.5.1 Treating polar boundary conditions

If the origin is within the domain and $0 \leq \theta < 2\pi$, we have a periodic boundary condition in the $\theta$ direction (i.e., $u(r, \theta) = u(r, \theta + 2\pi)$), but in the radial ($r$) direction the origin $R_1 = 0$ needs special attention. There are different ways of dealing with a singularity at the origin, but some methods lead to an undesirable structure in the coefficient matrix from the FD equations. One approach discussed is to use a staggered grid:

$$r_i = (i - \frac{1}{2})\Delta r, \quad \Delta r = \frac{R_2 - \frac{1}{2}}{m - \frac{1}{2}}, \quad i = 1, 2, \cdots, m,$$

where $r_1 = \Delta r/2$ and $r_m = R_2$. Except for $i = 1$ (i.e., at $i = 2, \cdots, m - 1$), the conservative form of the FD discretization can be used. At $i = 1$, the following non-conservative form is used to deal with the pole singularity at $r = 0$:

$$\frac{U_{0j} - 2U_{1j} + U_{2j}}{(\Delta r)^2} + \frac{1}{r_1^2} U_{2j} - U_{0j} + \frac{1}{r_1^2} U_{1,j-1} - 2U_{1j} + U_{1,j+1} = f(r_1, \theta_j).$$

Note that $r_0 = -\Delta r/2$ and $r_1 = \Delta r/2$. The coefficient of $U_{0j}$ in the above FD equation, the approximation at the ghost point $r_0$, is zero! The above FD equation simplifies to

$$\frac{-2U_{1j} + U_{2j}}{(\Delta r)^2} + \frac{1}{r_1^2} U_{2j} + \frac{1}{r_1^2} U_{1,j-1} - 2U_{1j} + U_{1,j+1} = f(r_1, \theta_j),$$

and we still have a diagonally dominant system of linear algebraic equations.

### 5.5.2 Using the FFT to solve Poisson equations in polar coordinates

When the solution $u(r, \theta)$ is periodic in $\theta$, we can approximate $u(r, \theta)$ by the truncated Fourier series

$$u(r, \theta) = \sum_{n=-N/2}^{N/2-1} u_n(r) e^{in\theta},$$

where again $u_n(r)$ is an approximation to the solution $u(r, \theta)$.
where \( i = \sqrt{-1} \) and \( u_n(r) \) is the complex Fourier coefficient given by

\[
u_n(r) = \frac{1}{N} \sum_{k=0}^{N-1} u(r, \theta) e^{-ink\theta}.
\] (5.5)

Substituting (5.4) into the Poisson equation gives

\[
\frac{1}{r} \frac{\partial}{\partial r} \left( \frac{1}{r} \frac{\partial u_n}{\partial r} \right) - \frac{n^2}{r^2} u_n = f_n(r), \quad n = -N/2, \ldots, N/2 - 1,
\] (5.6)

where \( f_n(r) \) is the \( n \)-th coefficient of the Fourier series of \( f(r, \theta) \) defined in (5.5). For each \( n \), we can discretize the above ODE in the \( r \) direction using the staggered grid, to get a tridiagonal system of equations that can be solved easily.

With a Dirichlet boundary condition \( u(r_{max}, \theta) = u_{BC}(\theta) \) at \( r = r_{max} \), the Fourier transform

\[
u_n^{BC}(r_{max}) = \frac{1}{N} \sum_{k=0}^{N-1} u^{BC}(\theta) e^{-ink\theta}
\] (5.7)

can be invoked to find \( u_n^{BC}(r_{max}) \), the corresponding boundary condition for the ODE. After the Fourier coefficient \( u_n \) is obtained, the inverse Fourier transform (5.4) can be applied to get an approximate solution to the problem.
Index

autonomous, 31
backward Euler’s method, 24
consistency, 23
convergency, 24
Crank-Nicholson method, 25

equilibrium, 31
Euler’s method, 21
explicit finite difference method, 21

initial condition, 15
local truncation error, 23
Lotka-Volterra model, 18

Matlab ODE-Suite, 29
Newton cooling model, 15
pendulum model, 17
Runge-Kutta Method, 26
stability condition, 23
steady state solution, 31