UNIVERSITY OF
WESTERN MACEDONIA

Solving Elliptic Partial Differential Equations with Dirichlet Boundary Conditions

Mid-studies dissertation

Stancu Gabriela Raluca

Department of Electrical and Computer Engineering

Faculty of Engineering

Laboratory of Applied and Computational Electromagnetics

Advisor: Professor Theodoros Zygiridis

2023

Student signature

..... .

Abstract

The efficient solution of sparse linear systems from PDE discretisations is central to scientific computing. This study evaluates iterative methods for the discrete five-point Laplacian on a unit square with Dirichlet boundary conditions, benchmarking Jacobi, Gauss-Seidel, SOR, Conjugate Gradient (CG), and PCG (Preconditioned Conjugate Gradient) with Jacobi (diagonal scaling), SSOR, and IC(0) preconditioners. Performance is assessed using iteration counts to a prescribed tolerance, wall-clock runtimes, and residual histories across grid sizes. The results are interpreted using spectral diagnostics, such as condition numbers and eigenvalue clustering. The results show that CG outperforms Jacobi and GS, while SOR narrows the gap, but remains slower to converge. Among preconditioners, PCG-SSOR yields the lowest runtimes and iteration counts across all system dimensions, PCG-IC(0) displays intermediate improvements, while PCG-Jacobi is indistinguishable from CG in terms of number of iterations and almost runtimes.

Keywords: Partial Differential Equations (PDEs), Spectral Graph Theory, Numerical Linear Algebra, Computational Mathematics, Computational Engineering

Solving Elliptic Partial Differential Equations with Dirichlet Boundary Conditions

Stancu Gabriela Raluca
ece01906@uowm.gr

2023

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Large scale scientific and engineering models most of the times reduce to solving sparse linear systems that are derived from PDE discretisations. Even in the case of the 2D Laplace equation, a modest $N \times N$ grid produces $n = N^2$ unknowns. Direct solvers quickly become memory and time bounded because of fill-in, while basic stationary schemes converge too slowly on fine meshes. Krylov subspace methods, such as the Conjugate Gradient (CG) are attractive for symmetric positive definite (SPD) problems, but their efficiency is determined by the spectrum of the system matrix. In the case of the discrete Laplacian we have $\kappa(A) = \Theta(N^2)$.
Preconditioning is the key to improving the efficiency of such methods. We attempt to transform the system so that its eigenvalues are better clustered and the condition number is smaller, making CG convergence faster. In this project, a clean and simple testbed is used: the 2D Laplace equation on a unit square with Dirichlet boundary conditions discritised by the five-point stencil. Classic stationary methods (Jacobi, Gauss-Seidel, SOR) are compared with CG and PCG using three easy-to-implement preconditioners: Jacobi (diagonal scaling), SSOR, and IC(0).

1.2 Structure overview

Chapter 2 - Theoretical background. Briefly described second order Partial Differential Equations, boundary and initial conditions, and specialises to the two-dimensional Laplace equation and it finite-difference discretisation, and provides the basic matrix algebra needed later.

Chapter 3 - Graph theoretic interpretation. Relates grid discretisations to graph Laplacians, reviews structural properties such as sparsity, symmetry, kernels, and the Kronecker form, and spectral properties used in the spectral analysis part.

Chapter 4 - Solvers for linear systems. Describes and analyses the main solvers used in this assignment: Jacobi, Gauss-Seidel, SOR, Krylov subspace methods (CG), preconditioning, and the PCG algorithm.

Chapter 5 - Experimental evaluation. In this chapter, the test setup (exact solution, grids, stopping rule) is explained, and all solvers are compared across grid sizes, reporting residual histories, iteration counts, and runtimes. Also, PCG is tested with Jacobi (Diagonal), SSOR, and IC(0) preconditioners, and a simple spectral view is used to explain why some methods converge faster than others.

Chapter 6 - Conclusions and recommendations for future work. Summarises the main findings, notes limitations, and outlines next steps.

Chapter 2

Theoretical Background

2.1 Second-Order Partial Differential Equations

Partial differential equations (PDEs) arise naturally in the mathematical modelling of many physical and engineering systems. A PDE is an equation involving unknown multivariable functions and their partial derivatives and governs how quantities such as temperature, displacement, electric potential, or fluid velocity vary over space and time [1].

Second-order PDEs are commonly classified into three types - elliptic, parabolic, and hyperbolic - according to the structure of their second derivative terms. Consider the general linear second-order PDE in a domain $\Omega \subset \mathbb{R}^2$:

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G. \tag{2.1}$$

The coefficients A, B, C, D, E, F, G are functions given in Ω that are assumed to be sufficiently smooth, and at the point of interest, (x_0, y_0), we require that $(A, B, C) \neq (0, 0, 0)$ [1, 2].
The principal part of 2.1, $Au_{xx} + Bu_{xy} + Cu_{yy}$, plays a central role in determining the qualitative behaviour of the solutions. After isolating the principal part of equation 2.1, we obtain the following equivalent form.

$$Au_{xx} + Bu_{xy} + Cu_{yy} = H(x, y, u, u_x, u_y), \tag{2.2}$$

with $H(x, y, u, u_x, u_y) = G(x, y) - D(x, y)u_x - E(x, y)u_y - F(x, y)u$.
The role of A, B, C mirrors the general quadratic form of conic sections in analytic geometry,

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0, \tag{2.3}$$

whose discriminant, $B^2 - 4AC$, determines the type: elliptic, parabolic, and hyperbolic [1, 2]. By analogy, the classification of second order PDEs at (x_0, y_0) is determined by the discriminant

$$\Delta(x, y) = B(x, y)^2 - 4A(x, y)C(x, y).$$

Definition 2.1. *A partial differential equation of the form given in 2.1, given that* $(A, B, C) \neq (0, 0, 0)$ *at a point* $(x_0, y_0) \in \mathbb{R}^2$ *as of:*

- *elliptic type, if* $B(x_0, y_0)^2 - 4A(x_0, y_0)C(x_0, y_0) < 0$*,*
- *parabolic type, if* $B(x_0, y_0)^2 - 4A(x_0, y_0)C(x_0, y_0) = 0$*,*
- *hyperbolic type, if* $B(x_0, y_0)^2 - 4A(x_0, y_0)C(x_0, y_0) > 0$*.*

Equivalently, for the principal symbol matrix

$$\mathcal{A}(x, y) = \begin{bmatrix} A(x, y) & \frac{1}{2}B(x, y) \\ \frac{1}{2}B(x, y) & C(x, y) \end{bmatrix},$$

a PDE is of elliptic type if $\mathcal{A}$ is definite, parabolic if $\mathcal{A}$ is semidefinite, and hyperbolic if $\mathcal{A}$ is indefinite [2, 3].

Although the classification corresponds to canonical forms of conic sections, the analogy is algebraic rather than geometric; it stems from the possibility of transforming a PDE into a canonical form through a chain of transformations and does not lead to any conclusions regarding the geometrical properties of the solutions.

2.1.1 Boundary and Initial Conditions

Second-order partial differential equations can describe several physical phenomena such as heat conduction, wave propagation, and fluid dynamics. To obtain a unique solution, additional constraints must be imposed to ensure that the problem is well-posed in the sense of Hadamard [2]; that is, a solution exists, it is unique and depends continuously on the given data, meaning that small fluctuations in the initial conditions cause little or no significant changes to the solution. These constraints take the form of boundary and initial conditions.

Boundary conditions (BCs) describe how the solution behaves on the boundary, $\partial\Omega$, of the spatial domain, $\Omega \subset \mathbb{R}^n$, of the PDE. The most common types are **Dirichlet**, **Neumann**, and **Robin** conditions.

Dirichlet boundary conditions, specify the value of the solution on the boundary; that is, $u(x) = g(x)$ for $x \in \partial\Omega$. These conditions are commonly used to model fixed quantities, such as temperature or displacement, along the boundary of the domain.

Neumann boundary conditions prescribe the normal derivative of the solution on the boundary; that is, $\frac{\partial u}{\partial n} = h(x)$ for $x \in \partial\Omega$, where $\frac{\partial u}{\partial n}$ denotes the directional derivative in the outward normal direction with respect to the boundary of the domain. Neumann conditions often represent the flux of a quantity across the boundary, such as heat or force.

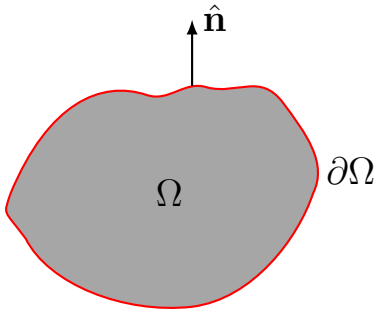


Figure 2.1: A schematic representation of a domain Ω, its boundary $\partial\Omega$, and the outward normal vector $\hat{\mathbf{n}}$.

Robin boundary conditions, also called mixed or third-type conditions, combine both Dirichlet and Neumann aspects in the form $a(x)u(x) + b(x)\frac{\partial u}{\partial n} = r(x)$, for $x \in \partial\Omega$. These types of conditions are typically encountered in problems that involve convective heat transfer or interaction with an external environment [4, 5].

Initial Conditions (ICs) capture how the system is initially set into motion and determine the subsequent evolution of the solution, and are essential when dealing with time-dependent partial differential equations. Such conditions specify the state of the system at the starting time, typically $t = 0$, and are essential to ensure that the problem is well-posed. For example, in the case of the heat equation which describes the diffusion of the temperature over time, we must pre-define the initial temperature distribution throughout the domain as $u(x, t = 0) = f(x)$. On the other hand, hyperbolic equations, such as the wave equation, require initial conditions for both the initial displacement and velocity, given as $u(x, t = 0) = f(x)$ and $\frac{\partial u}{\partial t}(x, t = 0) = g(x)$, respectively.

From a computational standpoint, both the domain and the data associated with the initial and boundary conditions can only be approximated rather than exactly represented in the discrete model. Additional errors arise from the finite precision of arithmetic oper-

ations during computation. Nevertheless, if the problem is well-posed, then it is possible to obtain an accurate approximation of the exact solution, provided that the data associated with the problem are also approximated with sufficient accuracy. In the case of an ill-posed problem, this cannot be taken for granted.

2.1.2 Elliptic Equations

Elliptic Partial Differential Equations are central to the mathematical representation of steady-state or equilibrium phenomena, that is, systems that are not evolving in time but have instead reached a stable state, where spatial variations exist, but no further temporal dynamics are present. Examples of such systems include the distribution of temperature in a stationary object, the electrostatic potential in a charge-free region, and pressure fields in incompressible fluids at rest.

A fundamental elliptic PDE is the **Poisson equation**. In a domain $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega$, the Poisson equation is defined as

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y), \tag{2.4}$$

where $f(x, y)$ is a known function often interpreted as a source term. In this equation, we assume that $f(x, y)$ describes the input to the problem. The Poisson equation models a variety of physical phenomena, such as electric or gravitational potentials influenced by a distributed source.
When the source term, $f(x, y)$, vanishes, the Poisson equation reduces to the **Laplace equation**:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \tag{2.5}$$

Or, equivalently,

$$\Delta u = 0.$$

The Laplace equation arises naturally in modelling phenomena where sources or sinks are absent, such as in the equilibrium distribution temperature in a homogeneous medium with fixed temperatures along the boundaries. The solutions of the Laplace equation, known as harmonic functions, satisfy important properties such as the maximum principle and uniqueness under appropriate boundary conditions [6, 2].

An essential property of elliptic PDEs, particularly the Laplace and Poisson equations, is that when discretised using finite differences of finite element methods, they result in

linear systems with certain algebraic properties. The resulting matrices are usually large, sparse, symmetric, and often diagonally dominant. These structural properties allow the application of efficient iterative numerical solvers tailored for such systems.

2.1.3 Parabolic Equations

Parabolic partial differential equations describe physical systems in which a quantity diffuses (or smooths) over time. They model time-dependent phenomena that tend to converge to a steady state as time progresses. A representative example is the **heat equation**, which governs how heat flows along or through a given medium [1, 4].
On a one-dimensional rod, $0 < x < L$, the standard model is

$$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}, \tag{2.6}$$

where $u(x, y)$ is the temperature or any other diffusing quantity, at position x and time t, and α^2 is the thermal diffusivity coefficient of the medium, which represents how quickly the heat spreads through the medium.

To uniquely determine the solution, the initial and boundary conditions must be specified, that is, the initial distribution of temperature, $f(x, y)$ at $t = 0$, along the rod, and Dirichlet and/or Neumann conditions to fix the temperature at the ends of the rod, $x = 0$ and $x = L$, and define the flow rate out of the rod, respectively.
The formulation of a problem with non-homogenous Dirichlet boundary conditions is given below.

$$\begin{cases} u_t = \sigma u_{xx}, \quad 0 < x < L, \quad t > 0 \\ u(0, t) = U_1, \quad t > 0 \\ u(L, t) = U_2, \quad t > 0 \\ u(x, 0) = f(x), \quad 0 \leq x \leq L \end{cases}$$

If the ends are insulated (no heat flux), one imposes the following Neumann boundary conditions: $u_x(0, t) = 0$ and $u_x(L, t) = 0$ for $t > 0$.

Long-time behaviour. The long-term behaviour of the solution of the heat equation reflects the physical expectation of the phenomenon, that is, the temperature distribution tends to converge to a steady state as $t \to \infty$, which is typically expressed by a linear or

constant profile depending on the boundary conditions. This limiting state satisfies an elliptic equation, usually the Laplace or Poisson equation, thereby establishing a strong link between parabolic and elliptic PDEs.

Given that in a steady state phase the solution is not time dependent, the solution, $u_\infty(x)$, satisfies the condition $u''_\infty(x) = 0$, therefore $u_\infty(x, y) = Ax + B$. Applying the initial Dirichlet conditions, we see that the steady-state solution is defined as follows.

$$\lim_{t\to\infty} u(x,t) = u_\infty(x) = \frac{T_2 - T_1}{L}x + T_1$$

In the case of insulated ends (pure Neumann conditions), the total heat, $\int_0^L u(x,t)\,dx$, is conserved, and $u(\cdot, t)$ converges to the constant equal to the spatial average of the initial data:

$$u_\infty(x) \;=\; \frac{1}{L}\int_0^L f(s)\,ds.$$

In general, in higher dimensions the heat equation, $u_t = \alpha^2\Delta u$, relaxes toward a harmonic steady state, $\Delta u_\infty = 0$, compatible with the boundary conditions.

Discretisation and stability. In numerical simulations, parabolic PDEs are often treated using finite-difference methods for the spatial domain, combined with explicit or implicit time-stepping schemes. Unlike the Laplace equation, which is purely spatial, solving parabolic PDEs requires advancing the solution in discrete time steps where stability becomes a central concern.

On a uniform grid with spacing h and time step Δt, the explicit (FTCS) update

$$u_i^{n+1} \;=\; u_i^n \;+\; \lambda\big(u_{i+1}^n - 2u_i^n + u_{i-1}^n\big), \qquad \lambda \coloneqq \frac{\alpha^2\,\Delta t}{h^2},$$

is stable given that $\lambda \le \frac{1}{2}$ in a one-dimensional setting [7, 8].

2.1.4 Hyperbolic Equations

Hyperbolic partial differential equations describe systems in which information or disturbances propagate as waves in space and time. In contrast to parabolic equations, which model diffusive behaviour that smooths over time, hyperbolic PDEs capture the dynamics of oscillations, vibrations, and signal transmission phenomena, where speed and direction of propagation are essential.
A well-known example of a hyperbolic PDE is the one-dimensional **wave equation**,

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \tag{2.7}$$

where $u(x,t)$ is the displacement at position x from the equilibrium position and time t, and c is the constant wave speed. This equation models the transverse motion of a taut and flexible string.

To obtain a unique solution to a hyperbolic PDE like the wave equation, initial conditions must be provided for both the displacement and velocity and boundary conditions for the endpoints of the vibrating string.
A well-posed problem for the one-dimensional wave equation is given below.

$$\begin{cases} u_{tt} = c^2 u_{xx}, & 0 < x < L, \quad t > 0 \\ u(0,t) = u(L,t) = 0, & \qquad\quad t \geq 0 \\ u(x,0) = f(x), & 0 \leq x \leq L \\ u_t(x,0) = g(x), & 0 \leq x \leq L \end{cases}$$

Characteristics and finite propagation speed. Hyperbolic PDEs arise naturally in many physical problems beyond the vibration of strings. They model the motion of elastic membranes, longitudinal waves in rods, electromagnetic waves in free space, and even electrical signals propagating through transmission lines. A prominent feature of such equations is that their solutions typically involve **characteristics**, $x \pm ct = \text{const}$, - curves along which information travels - and often require a different numerical approach from their elliptic and parabolic counterparts. On the infinite line the d'Alembert equation gives the explicit solution

$$u(x,t) \;=\; \tfrac{1}{2}\big[f(x-ct) + f(x+ct)\big] \;+\; \frac{1}{2c}\int_{x-ct}^{x+ct} g(s)\,ds, \tag{2.8}$$

which displays **finite speed of propagation**; that is, the value at (x,t) depends only on the initial data in the interval $[x-ct,\, x+ct]$ [1, 2].

Energy conservation. For fixed ends, the mechanical energy,

$$E(t) \;=\; \tfrac{1}{2}\int_0^L \Big(u_t^2(x,t) + c^2\, u_x^2(x,t)\Big)\,dx \tag{2.9}$$

is conserved: $E(t) \equiv E(0)$ [1]. This contrasts with the heat equation, where the energy decays monotonically.

Computational remarks and CFL stability. From a computational point of view, solving hyperbolic equations requires time-stepping methods that respect the Courant-Friedrichs-Lewy (CFL) condition. This ensures numerical stability by linking the time step with the spatial resolution and wave speed. Second order centred finite differences in space combined with a leapfrog/central time discretisation scheme yield

$$u_i^{n+1} \;=\; 2u_i^n - u_i^{n-1} \;+\; \lambda^2\big(u_{i+1}^n - 2u_i^n + u_{i-1}^n\big), \qquad \lambda := \frac{c\,\Delta t}{h},$$

which is stable under the Courant–Friedrichs–Lewy (CFL) condition $\lambda \leq 1$ in a one-dimensional setting [7, 9]. To start the two-step scheme, Taylor expansion with initial velocity, g, is used:

$$u_i^1 \;=\; u_i^0 + \Delta t\, g_i \;+\; \tfrac{1}{2}\lambda^2\big(u_{i+1}^0 - 2u_i^0 + u_{i-1}^0\big).$$

2.2 The two-dimensional Laplace equation

The two-dimensional Laplace equation arises naturally when stationary physical systems and equilibrium phenomena are modelled. Examples include electrostatics, when determining the potential, u, in a charge-free space, heat conduction without heat sources, and fluid dynamics.

As mentioned above, the two-dimensional Laplace equation is expressed as follows.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{2.10}$$

Alternatively, we can write the Laplace equation using the Laplacian operator:

$$\Delta u = 0 \tag{2.11}$$

This equation is a canonical example of a second-order linear elliptic PDE. Its solutions, **called harmonic functions**, possess several analytical properties: they are infinitely differentiable within the domain and satisfy the **maximum principle**, which states that any maximum or minimum must occur on the boundary, assuming that the domain is bounded and not in the interior region, except if the solution is constant.

In the case of a rectangular or square domain, the Laplace equation with Dirichlet boundary conditions serves as a prototypical boundary value problem. It can model, for exam-

ple, the steady-state distribution of temperature in a metal plate when the temperature along the boundary is kept fixed.

The main focus of this assignment is the numerical study and solution of the two-dimensional Laplace equation on a unit square $(0,1) \times (0,1)$, subject to Dirichlet boundary conditions on all four sides, as described by the following boundary value problem.

$$\begin{cases} u_{xx} + u_{yy} = 0, & (x,y) \in (0,1) \times (0,1) \\ u(0,y) = g_1(y), & 0 \leq x \leq 1 \\ u(1,y) = g_2(y), & 0 \leq x \leq 1 \\ u(x,0) = f_1(x), & 0 \leq y \leq 1 \\ u(x,1) = f_2(x), & 0 \leq y \leq 1 \end{cases}$$

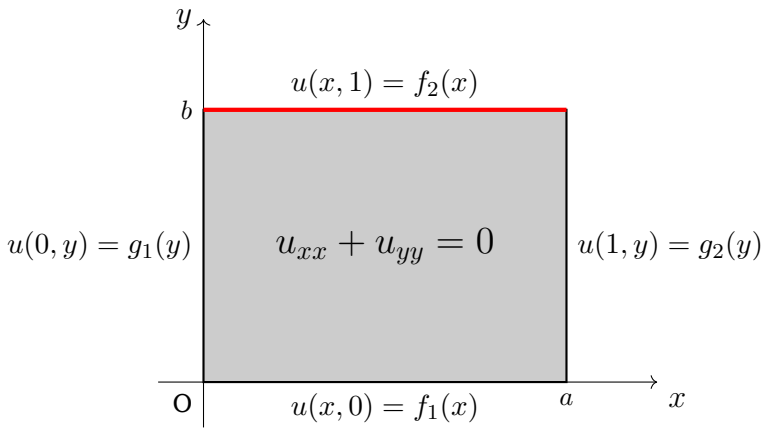


Figure 2.2: Boundary value problem for the Laplace equation on a rectangular domain.

Discretising the Laplace equation. To approximate the numerical solution of this boundary value problem, a two-dimensional adaptation of the finite-difference method for linear boundary value problems will be used. This method will transform the continuous Laplace equation into a sparse linear system, which can then be solved by applying several iterative methods. The discretisation procedure is described in detail below [8, 7].

The first step is to define a Cartesian grid containing the points at which we aim to approximate the solution. The continuous second-order partial derivatives will be replaced with finite-difference expressions at every grid point. The goal is to derive a system of linear equations that approximate the original PDE at each point inside the grid. Assuming that the problem domain is a unit square, $\Omega = \{(x,y) \mid x \in (0,1), \ \ y \in (0,1)\}$, the

step size across both axes is defined as $h = \frac{1}{N+1}$, resulting in N interior points per axis and $(N+2)^2$ grid points in total, including the boundary.
The grid points are defined as:

$$x_i = i \cdot h \quad \text{and} \quad y_j = j \cdot h$$

for each $i, j = 0, 1, 2, \ldots, N+1$. Indices $i = 0$ and $i = N+1$ correspond to the left and right boundaries, respectively, while labels $j = 0$ and $j = N+1$ correspond to the top and bottom, respectively.

If the domain Ω, is defined as $\Omega = \{(x, y) \mid x \in (a, b), \quad y \in (c, d)\}$ and is not a unit square, the step sizes across the axes and the grid points are defined as follows.

$$h_x = \frac{b-a}{N+1} \quad \text{and} \quad h_y = \frac{d-c}{N+1}$$

and

$$x_i = a + i \cdot h_x \quad \text{and} \quad y_j = c + j \cdot h_y$$

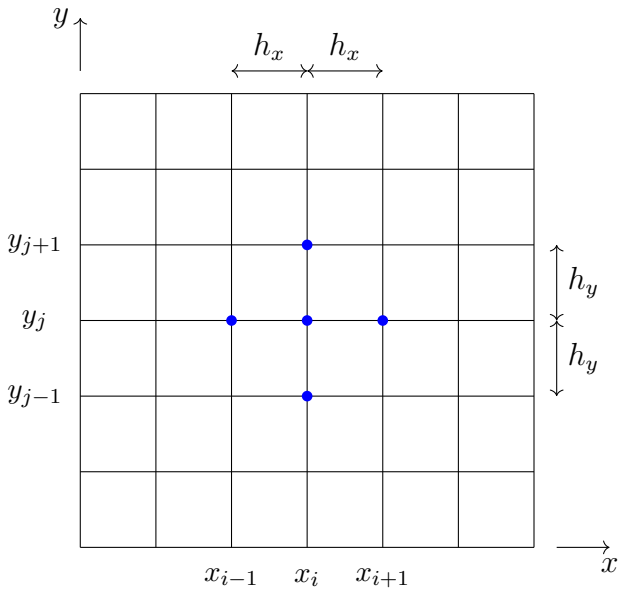


Figure 2.3: Uniform Cartesian grid with central stencil for (x_i, y_j) and grid spacings h_x, h_y.

The interior values (mesh points) of the unknown function, $u(x_i, y_j) \approx u_{i,j}$, are treated as unknowns, whereas the boundary values are determined by the Dirichlet conditions. For each mesh point, (x_i, y_j), for $i, j = 1, 2, \ldots, N-1$, we can use the Taylor series to generate the centred difference formula.

The Taylor series of a function $f(x)$ is given by the following expression.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n$$

If we expand $f(x)$ around x_0 we get the following.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^n(x_0)}{n!}(x-x_0)^n = f(x_0)+f'(x_0)(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\frac{f^{(3)}(x_0)}{3!}(x-x_0)^3+\dots$$

Let $x = x_0 + h \implies h = x - x_0$.
Therefore,

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f^{(3)}(x_0) + \dots \tag{2.12}$$

We perform the same process for $x = x_0 - h \implies h = -x + x_0$ accordingly, and we get:

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2!}f''(x_0) - \frac{h^3}{3!}f^{(3)}(x_0) + \frac{h^4}{4!}f^{(4)}(x - x_0)^4 + \dots \tag{2.13}$$

Adding 2.12 and 2.13 we get the following.

$$f(x + h) + f(x - h) = 2f(x_0) + h^2 f''(x_0) + \frac{h^4}{12}f^{(4)}(x_0) + \dots \tag{2.14}$$

We solve for $f''(x_0)$ and obtain the expression of the second derivative with respect to x at $x = x_0$:

$$f''(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} + O(h^2) \tag{2.15}$$

Recall that in a two-dimensional setting, the Laplace equation is defined as follows.

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{2.16}$$

Using the one-dimensional finite difference formula in both directions, we get the follow-

ing.

$$\frac{\partial^2}{\partial x^2}u(x_i, y_j) = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} \tag{2.17}$$

and

$$\frac{\partial^2}{\partial y^2}u(x_i, y_j) = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} \tag{2.18}$$

The Laplacian is formed by adding 2.17 and 2.18.

$$\Delta u_{i,j} \approx \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = 0 \tag{2.19}$$

The full discrete Laplace equation becomes:

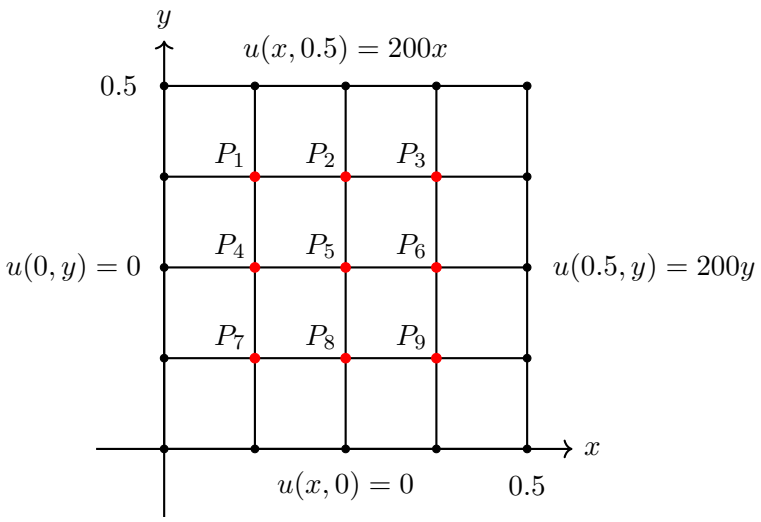
$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0 \tag{2.20}$$

Dirichlet values are imposed at boundary nodes:

$$u(x_0, y_j) = g_1(x_0, y_j) \quad \text{and} \quad u(x_N, y_j) = g_2(x_N, y_j) \quad \text{for each } j = 0, 1, \ldots N \tag{2.21}$$

$$u(x_i, y_0) = f_1(x_i, y_0) \quad \text{and} \quad u(x_i, y_N) = f_2(x_i, y_n) \quad \text{for each } i = 0, 1, \ldots N-1 \tag{2.22}$$

Example 2.1. *Consider the uniform grid shown in the figure below.*



The internal nodes, labeled P_1 through P_9, correspomd to the unknown values w_1 through

w_9. *The boundary values are imposed via Dirichlet boundary conditions:*

$$\begin{aligned} u(x,0) &= 0 \\ u(0,y) &= 0 \\ u(x,0.5) &= 200x \\ u(0.5,y) &= 200y \end{aligned}$$

For each internal node, P_i, we apply the 5-point stencil approximation of the Laplace Equation and we get the following system.

$$\begin{aligned}
P_1 : \quad & 4u_1 - u_2 - u_4 = u_{0,3} + u_{1,4} \\
P_2 : \quad & 4u_2 - u_3 - u_1 - u_5 = u_{2,4} \\
P_3 : \quad & 4u_3 - u_2 - u_6 = u_{4,3} + u_{3,4} \\
P_4 : \quad & 4u_4 - u_5 - u_1 - u_7 = u_{0,2} \\
P_5 : \quad & 4u_5 - u_6 - u_4 - u_2 - u_8 = 0 \\
P_6 : \quad & 4u_6 - u_5 - u_3 - u_9 = u_{4,2} \\
P_7 : \quad & 4u_7 - u_8 - u_4 = u_{0,1} + u_{1,0} \\
P_8 : \quad & 4u_8 - u_9 - u_7 - u_5 = u_{2,0} \\
P_9 : \quad & 4u_9 - u_8 - u_6 = u_{3,0} + u_{4,1}
\end{aligned}$$

The linear system associated with this problem has the following form.

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} 25 \\ 50 \\ 150 \\ 0 \\ 0 \\ 50 \\ 0 \\ 0 \\ 25 \end{bmatrix}$$

Determining the values of the internal points deduces the problem to solving a linear system of the form

$$Au = b$$

2.3 Matrix Algebra

Matrix algebra provides a compact language for expressing the discrete systems that arise from PDE discretisations into a compact and manipulable form. When the continuous domain is discretised, using finite-differences methods, for instance, the resulting equations for each grid point can be assembled into a system of linear equations. This system is written as $Au = b$, where A is the matrix that encodes the structure of the differential equation and the geometry of the domain, u is the vector of unknowns, and b contains the predefined contributions of the boundary conditions. Understanding the algebraic properties of this system is essential for both constructing it and determining the numerical methods to solve it.

Vectors and Norms. A vector is an ordered collection of real numbers, typically denoted as $x = [x_1, x_2, \cdots, x_n]$, and in the context of discretised PDEs, each component, x_i, corresponds to the approximation of the solution at a particular internal point in the grid.

To measure the size and magnitude of a vector, we use norms. A **vector norm** is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$, that satisfies the following properties.

- Nonnegativity and definiteness: $\|x\| > 0 \ \ \forall x \in \mathbb{R}^n$ and $\|x\| = 0$ if and only if $x = 0$
- Homogeneity: $\|\alpha x\| = |\alpha| \cdot \|x\| \ \ \forall \alpha \in \mathbb{R}$ and $x \in \mathbb{R}^n$
- Triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$

The norms most commonly used include the **Euclidean norm**, or **2-norm (l_2)**, the **1-norm (l_1)** and the **infinity norm (l_∞)**, defined as

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2},$$

$$\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

and

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

respectively .
Each of the norms mentioned above plays a distinct role in the context in which they are applied. For example, the 2-norm is commonly used to compute the relative error

of a solution, while $\infty -$ norm can become helpful when enforcing bounds or studying stability. Furthermore, norms are essential for defining convergence criteria for iterative solvers, where the residual vector, $r = b - Aw$, is monitored until its norm becomes sufficiently small.

Coefficient matrix and linear system. When discretising partial differential equations such as the Laplace equation using finite difference methods, the continuous PDE is transformed into a linear system of the form

$$Au = b,$$

where $A \in \mathbb{R}^{n\times n}$ is the coefficient matrix resulting from the stencil applied to the interior grid points, $u \in \mathbb{R}^n$ is the vector of unknown values at the interior grid points, and $b \in \mathbb{R}^n$ incorporates the effect of boundary conditions.

With an $N \times N$ interior grid, $u \in \mathbb{R}^N$, with N^2, stacks the interior unknown. The coefficient matrix, $A \in \mathbb{R}^{n\times n}$, that results from applying the five-point stencil to the grid points is sparse. Each row of A couples one node with a small stencil of neighbours.
Matrix Norms. In analysis, matrix norms, such as the infinity norm, can be used to estimate error propagation or study convergence. If $\|\cdot\|$ is a vector norm in $\mathbb{R}^n$, then

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

is a matrix norm.

Matrix norms defined by vector norms are called the **natural**, or induced, **matrix norm** associated with the vector norm and are defined as follows.

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$$

Some examples include the **1-norm (l_1)**, **infinity norm (l_∞)**, and **2-norm (l_2)**, or otherwise known as the spectral norm, defined as

$$\|A\|_1 = \max_{i\leq j\leq n} \sum_{i=1} |a_{ij}|,$$

$$\|A\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}|,$$

and

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^\top A)},$$

where $\lambda_{\max}$ is the largest eigenvalue of $A^\top A$ [10, 11]. This norm measures how much A can stretch a vector and is widely used in error and stability analysis.

Lastly, the **Frobenius Norm** is defined as follows [11].

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

The Frobenius norm is analogous to the Euclidean norm of vectors, is computationally convenient, and satisfies the following property.

$$\|A\|_F = \sqrt{\text{trace}(A^\top A)}$$

Although not always computed directly, matrix norms can be used to understand and study the conditioning and stability of the system.

Properties. The coefficient matrices that arise in this context exhibit special structural properties due to the nature of discretisation and the symmetry that the problem often has. These properties are described in the following.

Sparsity. The finite difference scheme uses only local interactions, which means that each equation explicitly depends on a few neighbouring grid points, resulting in a coefficient matrix, A, which is sparse. For example, in the five-point stencil used for the discretisation of the two-dimensional Laplace equation, each row of A contains at most five nonzero entries, corresponding to the central node and its four immediate neighbours.

Symmetry. If the grid is uniform and the discretisation is centred, the coefficient matrix, A, is symmetric, which means that $A = A^\top$, where $A^\top$ is the transpose of A, or equivalently $a_{ij} = a_{ji} \quad \forall i, j$. Symmetry is a desirable property, as it allows the use of efficient numerical algorithms that leverage symmetry and converge to a solution faster.

Diagonal Dominance. The coefficient matrix, A, is often diagonally dominant, especially when the interior points outweigh the boundary effects. A matrix is said to be strictly

diagonally dominant if the following condition is satisfied.

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \forall i$$

This property greatly improves the convergence behaviour of classic iterative methods such as Jacobi and Gauss-Seidel.

Positive Definiteness. In the case of the Laplace equation with Dirichlet boundary conditions, the resulting coefficient matrix is not only symmetric but also positive definite, that is, for all nonzero vectors, $x \in \mathbb{R}^n$, $x^\top Ax > 0$.

Positive definiteness guarantees that the linear system has a unique solution and ensures the convergence of Krylov methods such as the Conjugate Gradient method when used without preconditioning.

Chapter 3

Graph Theoretic Interpretation

The discretisation of partial differential equations, particularly those involving the Laplace operator, results in a system matrix that is large, sparse, symmetric, and diagonally dominant. Although such linear systems are mostly studied through a numerical linear algebra lens, an alternative perspective emerges when the system matrix is analysed from a graph-theoretic point of view.

More precisely, the system matrix can be seen as the Laplacian of a structured graph, where each unknown of the problem corresponds to a vertex, and edges encode the finite difference interactions between the vertices. This chapter adopts a graph-theoretic perspective to explore the structure of the linear system resulting from the finite difference discretisation of the two-dimensional Laplace equation with Dirichlet boundary conditions.

3.1 Graphs and Laplacians

3.1.1 Basic graph theory and notation

A **graph** is a mathematical structure that is used to represent pairwise relations between objects. Formally, a graph is an ordered pair $G = (V, E)$ that consists of a set of vertices V and a set of edges $E \subset \{\{i, j\} \mid i, j \in V, i \neq j\}$, where each edge connects two distinct vertices. This type of object is called an **undirected graph**. In an undirected graph, the edges link two vertices symmetrically.
We may also have **directed graphs**, where the edges link two vertices asymmetrically. A directed graph is defined as an ordered pair $G = \{V, E\}$ that comprises a set of vertices, V, and a set of edges $E \subseteq \{\{i, j\} \mid (i, j) \in V^2 \text{ and } i \neq j\}$ [12, 13].

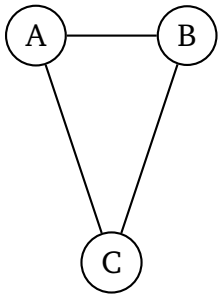


Figure 3.1:
An undirected graph with three nodes.

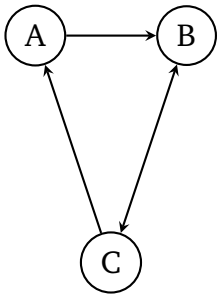


Figure 3.2:
A directed graph with three nodes.

The set of vertices in both graphs in the figures 3.1-3.2 is $V = \{A, B, C\}$, and the edge sets are $E = \{\{A, B\}, \{B, C\}, \{A, C\}\}$ and $E = \{(A, B), (B, C), (C, B), (C, A)\}$ for the undirected and directed graphs, respectively.

Under the definitions given above, multiple edges that connect the same vertices are not allowed. In a more general sense of allowing multiple edges, we may define a graph as an ordered triple $G = (V, E, \phi)$, where V and E are the vertices and edges sets, respectively, and $\phi : E \to \{\{i, j\} \mid i, j \in V, i \neq j\}$ is an **incidence function** which maps each edge to an unordered pair of vertices; that is, an edge is associated with two distinct vertices. This type of graph is called an **undirected multigraph**.
Under the same concept, we may define a **directed multigraph**, which has an incident function $\phi : E \to \{(i, j) \mid (i, j) \in V^2, i \neq j\}$.

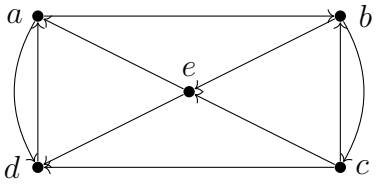


Figure 3.3:
A directed multigraph with no loops.

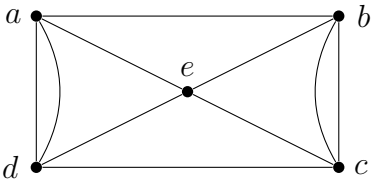


Figure 3.4:
An undirected multigraph with no loops.

Lastly, we define a **loop** as an edge that connects a vertex to itself. In an undirected graph, a loop is an edge of the form $\{i, i\} \in E$ and in a directed graph $(i, i) \in E$. In matrix terms, a loop appears as a nonzero diagonal entry of the adjacency matrix, $A_{ii} \neq 0$.

In this chapter, we focus on **undirected** and **simple graphs**, which means that edges have no orientation, no loops, and there is at most one edge between two distinct vertices.

Give a vertex $i \in V$, the **neighbourhood** of i is defined as $N(i) = \{j \in V \mid \{i, j\} \in E\}$, that is, $N(i)$ consists of all vertices directly connected to i. The **degree** of the vertex, i, denoted as $\deg(i)$, is the number of vertices adjacent to i, so $\deg(i) = |N(i)|$. For a directed graph, the degree is the number of edges that enter or exit the vertex.

For an undirected graph, the sum of degrees of all vertices equals twice the number of edges that the graph contains: $\sum_{i \in V} \deg(i) = 2|E|$.

Adjacency and Degree matrices

Graphs can be conveniently represented by matrices, thus allowing the use of linear algebra to analyse their structure and properties. Two matrices are often used to represent a graph, the **adjacency**, and the **degree** matrices.

For a weighted graph $G = (V, E, w)$, the adjacency matrix is defined as

$$A = \begin{cases} w_{ij}, & \text{if } \{i, j\} \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Since G is an undirected graph, A is symmetric; that is, $A_{ij} = A_{ji}, \ \ \forall i, j$.

The degree matrix $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix in which each diagonal entry is the degree of the corresponding vertex. The degree matrix is defined as follows.

$$D = \begin{cases} d_i, & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

The weighted degree, d_i, of a vertex, i, is defined as

$$d_i = \deg(i) = \sum_{j : \{i,j\} \in E} w_{i,j}.$$

The degree matrix encodes the number of edges incident to each vertex. Being diagonal, it is easy to compute and invert when needed, under the condition that no vertex has degree 0.

For an unweighted graph $G = (V, E)$ with $|V| = n$, the adjacency matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$A = \begin{cases} 1, & \text{if } \{i, j\} \in E, \\ 0, & \text{otherwise.} \end{cases}$$

In the case of directed graphs, we define the **out-** and **in-degrees** as

$$\deg_{out}(i) = \sum_{j=1}^{n} A_{ij}$$

and

$$\deg_{in}(i) = \sum_{j=1}^{n} A_{ji},$$

and the corresponding diagonal matrices D_{out} and D_{in}.

3.1.2 The Graph Laplacian

The **graph Laplacian** is the discrete analogue of the continuous Laplace operator and arises naturally in various physical and computational models. For a graph $G = (V, E, w)$ with adjacency and degree matrices A and D, respectively, its (unnormalised) graph Laplacian is defined as follows [14, 15].

$$L = \begin{cases} d_i - w_{ij} & \text{if } i = j \\ -w_{ij} & \text{if } i \neq j \text{ and } \{i, j\} \in E, \\ 0 & \text{otherwise} \end{cases}$$

The matrix form is

$$L = D - A.$$

The graph Laplacian, L, has nonpositive off-diagonal elements and is weakly diagonally dominant, meaning that

$$|L_{ii}| = deg(i) \geq \sum_{j \neq i} |L_{ij}| = \#\{j : \{i, j\} \in E\}.$$

In a PDE setting, the inequality is strict at any vertex adjacent to a boundary node of the grid.

Since D is diagonal and A symmetric, L will be symmetric. Furthermore, L is **positive**

semidefinite, that is, $\forall x \in \mathbb{R}^n$,

$$x^\top Lx = x^\top Dx - x^\top Ax = \sum_i \Big(\sum_j w_{ij}\Big) x_i^2 - \sum_{i,j} w_{ij} x_i x_j = \frac{1}{2}\sum_{i,j} w_{ij}(x_i - x_j)^2 = \sum_{\{i,j\}\in E} w_{ij}(x_i - x_j)^2 \geq 0.$$

For an unweighted graph, the graph Laplacian is defined as

$$L = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

which is also positive semidefinite. Therefore, $\forall x \in \mathbb{R}^n$,

$$x^\top Lx = \sum_{i=1}^{n} \deg(i)\, x_i^2 - 2\sum_{\{i,j\}\in E} x_i x_j = \sum_{\{i,j\}\in E} (x_i - x_j)^2 \geq 0.$$

3.2 Connection to discretised partial differential equations

In the context of numerically solving the two-dimensional Laplace equation with Dirichlet boundary conditions, the graph Laplacian plays a central role. As we analysed in section 2.2, to approximate the solution to the Dirichlet boundary value problem, we use an adaptation of the finite difference method in two dimensions. Recall that the discrete two-dimensional Laplace equation can be written as

$$\Delta u_{i,j} \approx \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = 0,$$

or equivalently

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0. \tag{3.1}$$

for $i, j \in \{1, \ldots, n\}$, where $n = N - 1$ is the dimension of the inner grid, resulting in $m = n^2$ unknown interior points. If we further analyse the above equation from a graph-theoretic perspective, we will see that it mirrors the definition of the combinatorial graph Laplacian we gave before.

To numerically approximate the solution to the Dirichlet boundary value problem, we must first define a Cartesian grid within the domain of the problem. Suppose that after discretising the domain, we get the grid shown in Figure 3.5.
Translating the entire grid into a graph, $\widehat{G} = (V, E)$, each point of the grid can be seen as

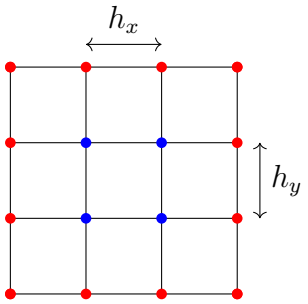


Figure 3.5: An $n = 2$ interior grid (blue) with boundary nodes (red) and $h_x = h_y = h$.

a vertex, and the interactions between a point and its neighbours can be seen as edges. We assume that the weight in all directions is 1.

Let $P = \{(i, j) \mid i, j \in \{0, \dots, N\}\}$ be the set of the coordinates of all points in the grid. The set of interior points, I, is defined as $I = \{(i, j) : 1 \leq i \leq n, 1 \leq j \leq n\}$, and the set of boundary points is defined as $B = P \setminus I$. We may also define the set of corner points $C = \{(0, 0), (0, N), (N, 0), (N, N)\}$.
The adjacency and degree matrices of $\widehat{G}$ are defined as

$$\widehat{A}\big((i, j), (i', j')\big) = \begin{cases} 1, & |i - i'| + |j - j'| = 1, \\ 0, & \text{otherwise} \end{cases}$$

and

$$\widehat{D}\big((i, j), (i', j')\big) = \begin{cases} \deg(i, j), & (i', j') = (i, j) \\ 0, & \text{otherwise} \end{cases}$$

where $\deg(i, j) = \sum_{(i',j')} \widehat{A}\big((i, j), (i', j')\big)$. Therefore, the degree is 4 for points that belong in the interior, I, 3 for points on the edges, $B \setminus C$, and 2 for the corner points, C.

Another way in which we can write $\widehat{A}$ is

$$\widehat{A} = \begin{bmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{bmatrix},$$

where the vertices have been ordered so that the interior node indices come first and the boundary node indices second, with $|I| = m = n^2$ and $|B| = |P| - m$. Then

$$A_{II} \in \mathbb{R}^{m \times m}, \quad A_{IB} \in \mathbb{R}^{m \times |B|}, \quad A_{BI} \in \mathbb{R}^{|B| \times m}, \quad A_{BB} \in \mathbb{R}^{|B| \times |B|},$$

where

A_{II} : interior-interior adjacency ,
A_{BI} : boundary-interior adjacency ,
A_{IB} : interior-boundary adjacency ($A_{BI}^{\top}$), and
A_{BB} : boundary-boundary adjacency.

Formally, since I and B are the sets of interior and boundary indices, respectively, we have $(A_{II})_{pq} = \widehat{A}(p,q)$ for $p,q \in I$, $(A_{IB})_{pq} = \widehat{A}(p,q)$ for $p \in I, q \in B$, and similarly for A_{BI} and A_{BB}.
Similarly, we can write $\widehat{D}$ as

$$\widehat{D} = \begin{bmatrix} D_{II} & 0 \\ 0 & D_{BB} \end{bmatrix}.$$

The graph Laplacian of $\widehat{G}$ is defined as

$$\widehat{L}\big((i,j),(i',j')\big) = \widehat{D} - \widehat{A} = \begin{cases} \deg(i,j), & (i',j') = (i,j), \\ -1, & |i-i'| + |j-j'| = 1, \\ 0, & \text{otherwise} \end{cases}$$

or equivalently

$$\widehat{L} = \begin{bmatrix} D_{II} - A_{II} & -A_{IB} \\ -A_{BI} & D_{BB} - A_{BB} \end{bmatrix} = \begin{bmatrix} L_{II} & L_{IB} \\ L_{BI} & L_{BB} \end{bmatrix}.$$

Let $\widehat{u} = [u_I, u_B]^{\top}$ be a vector that contains the grid values of all nodes, where u_I and $u_B = g$ are the values in the interior and the prescribed Dirichlet vertices, respectively. To build the Laplacian of $\widehat{u}$, we apply $\widehat{L}$ to $\widehat{u}$ and get

$$\widehat{L}\,\widehat{u} = \begin{bmatrix} D_{II} - A_{II} & -A_{IB} \\ -A_{BI} & D_{BB} - A_{BB} \end{bmatrix} \begin{bmatrix} u_I \\ g \end{bmatrix} = \begin{bmatrix} (D_{II} - A_{II})\,u_I \;-\; A_{IB}\,g \\ -A_{BI}\,u_I \;+\; (D_{BB} - A_{BB})\,g \end{bmatrix}.$$

For the Dirichlet boundary problem, the boundary values are not unknowns, $u_B = g$, so only the **interior equations** must vanish. Hence, the discrete Laplace equation reduces to

$$(D_{II} - A_{II})\,u_I \;-\; A_{IB}\,g = 0 \Longleftrightarrow (D_{II} - A_{II})\,u_I = A_{IB}\,g.$$

On the full grid, each interior vertex has degree four, so $D_{II} = 4I$. Therefore,

$$(4I - A_{II})u_I = A_{IB}\,g \tag{3.2}$$

Defining $L_{II} \coloneqq 4I - A_{II}$ and $b \coloneqq A_{IB}g$, we obtain the standard linear system $L_{II}u_I = b$, which yields the solution to the Dirichlet boundary problem. Each entry of b is the sum

of the Dirichlet values at the boundary neighbours of the interior nodes.

If we further analyse Equation 3.2, we get

$$4\,u_I - A_{II}\,u_I = A_{IB}\,g,$$

or equivalently for each interior node $(i,j) \in I$,

$$4\,u_{i,j} - \sum_{\substack{(i',j')\in I \\ |i-i'|+|j-j'|=1}} u_{i',j'} = \sum_{\substack{(i',j')\in B \\ |i-i'|+|j-j'|=1}} g_{i',j'}.$$

This is exactly the five-point stencil of Equation 3.1, with all boundary-neighbour contributions collected on the right-hand side.
Originally, the five-point formula comes from the Taylor series expansion of the second derivatives, introducing a factor $\frac{1}{h^2}$ in each direction. Therefore, the interior equations with Dirichlet data are

$$\frac{1}{h^2}\Big(4\,u_I - A_{II}u_I\Big) \;=\; \frac{1}{h^2}\,A_{IB}\,g.$$

Multiplying by h^2 yields the combinatorial form $4u_I - A_{II}u_I = A_{IB}g$, used above. For the homogeneous Laplace equation, scaling has no effect on the solution, u_I; it only scales both sides. In the case of the Poisson equation, the right-hand side must be scaled consistently, that is $b_h = h^2 f_I + A_{IB}g$ when a source is present. If the grid spacing differs across axes, the edges are weighted, thus getting a weighted graph Laplacian.

Summarising, the five-point finite-difference stencil is the Dirichlet (restricted) graph Laplacian of the full grid, restricted to interior rows and columns (the principal submatrix).
Starting from the full grid Laplacian, $\widehat{L} = \widehat{D} - \widehat{A}$, and imposing $u_B = g$, the boundary entries are removed, leaving the interior submatrix $L_{II} = D_{II} - A_{II} = 4I - A_{II}$. The discrete system is

$$(4I - A_{II})u_I = A_{IB}\,g \Longleftrightarrow L_{II}\,u_I = b$$

with the right-hand side $b = A_{IB}\,g$, collecting the contributions of the boundary neighbours.

3.3 Graph Properties and Matrix Structure

3.3.1 Properties

Sparsity. For unknown interior points, u_I, in an interior grid $n \times n$, with $m = n^2$ nodes, the matrix $L_{II} = 4I - A_{II}$ is **sparse**. Due to the five-point pattern, the system matrix, L_{II}, has at most five nonzero entries per row, that is, one diagonal entry equal to four, and up to four off-diagonal entries for each of the four neighbours. Vertices that belong to the inner ring have two-three interior neighbours, so these rows have off-diagonal entries less than four.

Diagonal Dominance. Each row i of L_{II} has a diagonal element, $L_{II}(i,i)$, equal to 4, and off-diagonal elements equal to -1 for each interior neighbour. If $d_i^{\text{int}} \in \{2,3,4\}$ is the number of interior neighbours of node i, then

$$|L_{II}(i,i)| \geq \sum_{j \neq i} |L_{II}(i,j)| = d_i^{\text{int}},$$

thus, making matrix L_{II} **(weakly) diagonally dominant**. Strict diagonal dominance fails at interior nodes with four interior neighbours where $d_i^{\text{int}} = 4 = |L_{II}(i,i)|$. For all other nodes $d_i^{\text{int}} < |L_{II}(i,i)|$.

Because the interior graph is connected, L_{II} is an **irreducibly diagonally dominant** matrix, which implies nonsingularity. Combined with symmetry and nonpositive off-diagonal entries, L_{II} is a **Z-matrix** [16, 17]. Writing $L_{II} = 4I - A_{II}$ with $A_{II} \geq 0$ irreducible and $\rho(A_{II}) < 4$, strict since some interior points have $d_i^{\text{int}} \leq 3$, shows that L_{II} is a **nonsingular M-matrix**. Since L_{II} is symmetric, all its eigenvalues are real and strictly positive. Hence, L_{II} is **symmetric positive definite (SPD)**.

Discrete Maximum Principle. The **Discrete Maximum Principle (DMP)** is the lattice analogue of the continuous maximum principle for harmonic functions. Recall that the combinatorial (unscaled) finite-difference Laplacian is

$$4u_{i,j} = u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}, \quad \forall (i,j) \in I.$$

Thus, every interior value is the unweighted average of its four neighbours. Consequently, the interior extrema will not exceed the boundary range. More precisely, if the boundary values g, imposed by the Dirichlet conditions, satisfy $m \leq g \leq M$, then the interior solution, u_I, of the system $L_{II}\, u_I = b$, where $b = A_{IB}\, g$, also satisfies $m \leq u_I \leq M$

component-wise.
Indeed, $u_I = L_{II}^{-1} A_{IB}\, g$ and for the five-point Dirichlet Laplacian L_{II} is a nonsingular M-matrix, $L_{II}^{-1} \geq 0$, while $A_{IB} \geq 0$. Thus, u_I is a nonnegative linear combination of boundary values and is, consequently, bounded by m and M. The averaging relation rules out strict interior extrema unless the interior is constant.
In the case of the Poisson equation, $\Delta u = f$, the five-point formula gives

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = h^2 f_{i,j},$$

and after moving boundary neighbours to the right-hand side we obtain

$$L_{II} u_I \;=\; A_{IB} g \;-\; h^2 f_I.$$

If $f_I \geq 0$ ($f_I \leq 0$), that is, f_I is subharmonic (superharmonic), the maximum (minimum) is attained on the boundary. Using the alternative convention $-\Delta u = f$, the linear system becomes $L_{II} u_I = A_{IB} g + h^2 f_I$, and the corresponding extrema statements are flipped accordingly.

3.3.2 Incidence factorisation $L = BB^\top$

Let $G = (V, E, w)$ be a weighted graph with $|V| = n$ and $|E| = m$. Giving each undirected edge, $e = \{i, j\}$, an arbitrary orientation, say $i \to j$, the vertex-edge incidence matrix, $B \in \mathbb{R}^{n \times m}$ is defined as follows.

$$B_{v,e} = \begin{cases} -\sqrt{w_{i,j}}, & \text{if } v = i \text{ (tail)} \\ \sqrt{w_{i,j}}, & \text{if } v = j \text{ (head)} \\ 0, & \text{otherwise} \end{cases},$$

Each column of B corresponds to an edge and has exactly two nonzero entries: $-\sqrt{w_{i,j}}$ at the tail and $\sqrt{w_{i,j}}$ at the head. Reversing the orientation of an edge essentially multiplies its column by -1. Consequently, $BB^\top$ is independent of the orientation of the edges.

Using the incidence matrix, we can construct the combinatorial (weighted) graph Laplacian as

$$L = BB^\top$$

[13, 14].

According to the above definition of the Laplacian matrix, for any vertices $u, v \in V$,

$$(BB^\top)_{u,v} = \sum_{e \in E} B_{u,e} B_{v,e}.$$

If $u = v$, only edges incident to u contribute, each with $B_{u,e}^2$, giving

$$(BB^\top)_{uu} = \sum_{e \sim u} w_e = D_{uu}.$$

Regarding off-diagonal entries, where $u \neq v$, if $\{u, v\} \notin E$, then no column has both u and v nonzero, thus $(BB^\top)_{uv} = 0$. On the other hand, if $\{u, v\} \in E$, then exactly one column contributes with product $-\sqrt{w_{u,v}} \cdot \sqrt{w_{u,v}} = -w_{u,v}$. Therefore,

$$(BB^\top)_{u,v} = \begin{cases} -w_{u,v}, & \{u, v\} \in E, \\ 0, & \text{otherwise.} \end{cases}$$

It is now obvious that $(BB^\top)$ has diagonal and off-diagonal entries D and $-A$, respectively. Therefore,

$$BB^\top = D - A = L.$$

In our case, since the graph is unweighted, the incidence matrix is defined as

$$B_{v,e} = \begin{cases} -1, & \text{if } v = i \text{ (tail)} \\ 1, & \text{if } v = j \text{ (head)} \\ 0, & \text{otherwise} \end{cases}$$

and, consequently,

$$(BB^\top)_{u,v} = \begin{cases} -1, & \{u, v\} \in E, \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad (BB^\top)_{uu} = \deg(u).$$

From an operator point of view, $B^\top$ acts as a **discrete gradient** on edges, while B corresponds to a **discrete divergence** at vertices.
In calculus, the gradient ∇f measures how the function changes in space. On a graph, for a vertex function $f : V \to \mathbb{R}$ and an orientated edge, $e = (i \to j)$, the gradient ∇f is

$$(\nabla f)(e) = f(\text{head of } e) - f(\text{tail of } e) = f_j - f_i,$$

which is encoded algebraically as $\nabla f = B^\top f$.

If $u \in \mathbb{R}^n$ stores a scalar at each vertex, then for an edge $e = (i \to j)$,

$$(B^\top u)_e = \sum_{v \in V} B_{v,e} u_v = \sqrt{w_{i,j}}(u_j - u_i),$$

which is essentially the weighted signed difference along e.

The divergence $\nabla \cdot \mathbf{F}$ measures the net outflow. On a graph, if F is an edge function, $\mathbf{F} : E \to \mathbb{R}$, then the divergence at each node, u, is defined as

$$(\text{div } \mathbf{F})(u) = \sum_{\text{edges } e \text{ leaving } u} \mathbf{F}(e) - \sum_{\text{edges } e \text{ entering } u} \mathbf{F}(e).$$

If $y \in \mathbb{R}^m$ stores an edge flow that is positive in the orientation of the edge, then for each vertex i

$$(By)_i = \sum_{e \in E_{\text{in}}(i)} \sqrt{w_e} y_e - \sum_{e \in E_{\text{out}}(i)} \sqrt{w_e} y_e$$

is the net inflow at vertex i, where

$$E_{\text{out}}(i) = \{e = (i \to j)\} \text{ (outgoing edges)},$$
$$E_{\text{in}}(i) \ = \{e = (j \to i)\} \text{ (incoming edges)}.$$

Consequently, the discrete divergence (net outflow) is $-By$. Composing the "divergence of gradient", we get $Lu = BB^\top = B(B^\top u) = -\nabla \cdot (\nabla u)$ up to this sign convention. Componentwise,

$$(Lu)_i = \sum_{j : \{i,j\} \in E} w_{i,j} \, (u_i - u_j),$$

which for unit weights on the four-neighbour grid is exactly the five-point stencil

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}.$$

3.3.3 Grid/Kronecker form

Kronecker product. If $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$, their Kronecker product $A \otimes B$ is the block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{(mp) \times (nq)}.$$

Several useful properties include:

- **Replication of blocks with identities**.
  - $I_n \otimes B$ is block-diagonal with B repeated n times
  - $A \otimes I_p$ is each entry of A turned into a $p \times p$ scalar multiple of I_p
- **Vectorisation identity**: If $X \in \mathbb{R}^{n \times n}$ and vec(X) stacks its columns, then vec$(UXV^\top) =$ $(V \otimes U)$vec(X) [18, 19]. Two special cases will be used in the following, for which $T = T^\top$ are:
  - $(I \otimes T)$vec$(X) =$ vec(TX)
  - $(T \otimes I)$vec$(X) =$ vec(XT)
- **Kronecker sum**: For two square matrices A and B of size n, $A \oplus B \coloneqq A \otimes I_n + I_n \otimes B$. If A and B are Laplacian matrices, then $A \otimes B$ is also a Laplacian.
- **Spectrum**: If A and B are square matrices of size n and m, respectively, and $\{\lambda_i\}$ and $\{\mu_j\}$ are the eigenvalues of A and B, respectively, then $\{\lambda_i \mu_j\}$ are the eigenvalues of $A \otimes B$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

Two-dimensional Laplacian via Kronecker sum. Let $T = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{n \times n}$ be the one-dimensional Laplacian of n interior points with Dirichlet boundary conditions excluding boundary points. This encodes the three-point stencil $(Tu)_i = 2u_i - u_{i-1} - u_{i+1}$ for $i = 1, \ldots, n$. For a two-dimensional grid with interior grid $n \times n$, the Laplacian is

$$L_{II} = T \otimes I_n + I_n \otimes T,$$

with $L \in \mathbb{R}^{n^2 \times n^2}$ [18, 19].

Let $U \in \mathbb{R}^{n \times n}$ be the array of values in the interior grid, with entries $U_{i,j} = u_{\phi(i,j)}$ for $i, j = 1, \ldots, n$, where $\phi(i, j) = (j - 1)n + i$ is a column stack mapping. Applying T by rows and columns, we get

$$(TU)_{i,j} = 2U_{i,j} - U_{i-1,j} - U_{i+1,j} \tag{3.3}$$

and

$$(UT)_{i,j} = 2U_{i,j} - U_{i,j-1} - U_{i,j+1}, \tag{3.4}$$

respectively.

Adding the above expressions yields

$$(TU + UT)_{i,j} = 4U_{i,j} - U_{i-1,j} - U_{i+1,j} - U_{i,j-1} - U_{i,j+1},$$

which is the five-point stencil before excluding boundary neighbours to the right-hand side.

Applying $\text{vec}(AXB^\top) = (B \otimes A)\,\text{vec}(X)$ and $T = T^\top$ on equations 3.3 and 3.4, we get

$$\text{vec}(TU) = (I \otimes T)u \quad \text{and} \quad \text{vec}(UT) = (T \otimes I)u,$$

hence

$$\text{vec}(TU + UT) = (I \otimes T + T \otimes I)u.$$

Therefore, the interior two-dimensional Laplacian matrix is

$$L_{II} = \frac{1}{h^2}(I \otimes T + T \otimes I) \in \mathbb{R}^{n^2 \times n^2}$$

and

$$L_{II}u = \frac{1}{h^2}\text{vec}(TU + UT).$$

Generalising, we can form the Laplacian in higher dimensions as

$$L_d \;=\; \sum_{k=1}^{d} \frac{1}{h_k^2} \left(I_{n_d} \otimes \cdots \otimes I_{n_{k+1}} \otimes T_k \otimes I_{n_{k-1}} \otimes \cdots \otimes I_{n_1} \right) \;\in\; \mathbb{R}^{N \times N}.$$

For $d = 3$, we would obtain the Laplacian matrix for the three-dimensional problem:

$$L_3 = \frac{1}{h_x^2}(T_x \otimes I_{n_y} \otimes I_{n_z}) + \frac{1}{h_y^2}(I_{n_x} \otimes T_y \otimes I_{n_z}) + \frac{1}{h_z^2}(I_{n_x} \otimes I_{n_y} \otimes T_z)$$

If the grid is isotropic and the sizes match, then it reduces to

$$L_3 = \frac{1}{h^2}(T \otimes I \otimes I \;+\; I \otimes T \otimes I \;+\; I \otimes I \otimes T).$$

3.3.4 Dirichlet and Neumann block structures

Dirichlet boundary data. As mentioned in previous subsections, the Laplacian matrix of the full grid graph $\widehat{G} = (V, E)$ can be written as

$$\widehat{L} = \widehat{D} - \widehat{A} = \begin{bmatrix} L_{II} & L_{IB} \\ L_{BI} & L_{BB} \end{bmatrix},$$

where $L_{II} = D_{II} - A_{II}$, $L_{IB} = -A_{IB}$, $L_{BI} = -A_{BI}$, and $L_{BB} = D_{BB} - A_{BB}$.

Dirichlet boundary conditions fix the boundary: $u_B = g$. Substituting into $\widehat{L}u = 0$ and keeping only the points of the inner grid, we obtain the following.

$$L_{II}u_I + L_{IB}g = 0 \iff L_{II}u_I = -L_{IB}g = A_{IB}g = b.$$

The system matrix is the principal interior submatrix of $\widehat{L}$, $L_{II} = 4I - A_{II}$, which is **symmetric positive definite (SPD)**, that is, the interior is grounded by Dirichlet nodes ruling out the constant null mode. Furthermore, L_{II} is a nonsingular M-matrix, hence $L_{II}^{-1} \geq 0$, and the discrete maximum principle holds as discussed earlier.
To summarise, Dirichlet boundary conditions eliminate boundary rows/columns, and yield a solution to the initial boundary-value problem by solving the SPD system

$$L_{II}u = b.$$

Neumann boundary data. Neumann conditions, instead of fixing boundary values, prescribe a normal derivative on the boundary, that is, a flux. In finite differences, there are two common ways to enforce Neumann conditions.

- **Boundary point values as unknowns (no elimination).** Boundary nodes are kept in the unknown vector. For each boundary node, its system matrix row is replaced by a discrete Neumann row using a one-sided or central difference for the normal derivative. The interior rows remain intact.
- **Interior only unknowns.** The unknown vector contains only interior points. At each interior node that neighbours the boundary domain, a "ghost value" is introduced, and the Neumann condition is used to express that value in terms of the interior values and the prescribed flux. The resulting expression is then substituted into the five-point stencil.

Consider the one-dimensional case on a uniform grid with spacing h. Let the left boundary at $x = 0$ satisfy $u_x(0) = g$. The interior three-point stencil for the Laplacian is

$$\frac{2u_i - u_{i-1} - u_{i+1}}{h^2} = f_i,$$

or equivalently in combinatorial form

$$2u_i - u_{i-1} - u_{i+1} = h^2 f_i.$$

for $i = 1, \ldots, N-1$.

Boundary point values as unknowns. As we mentioned above, the boundary nodes are kept in the unknown vector, and their corresponding row in the Laplacian matrix is substituted with the derivative that enforces $u_x(0) = g$.
The first-order (forward) Neumann row at $x = 0$ is

$$\frac{u_1 - u_0}{h} = g \implies -u_0 + u_1 = hg$$

with local truncation error $O(h)$.

The second-order one-sided derivative is

$$\frac{-3u_0 + 4u_1 - u_2}{2h} = g.$$

with local truncation error $O(h^2)$.

If the first-order approximation is used, the boundary point is only first-order accurate and can reduce the global order. The second-order approximation maintains $O(h^2)$ at the boundary that matches the interior stencil. The resulting global system matrix from this approach is not symmetric, and, with only Neumann boundary conditions, is singular (constant nullspace).

Eliminating boundary unknowns. The boundary points appearing in the three-point stencil are substituted with their second-order centred approximation:

$$\frac{u_2 - u_0}{2h} = g \implies u_0 = u_2 - 2hg$$

Substituting into the interior row, $i = 1$, gives

$$2u_1 - 2u_2 = h^2 f_1 - 2hg.$$

In this way, the interior neighbour carries the coefficient -2 instead of -1, and the prescribed flux contributes $-2hg$ to the right-hand side. We could also use the first-order approximation $\frac{u_1 - u_0}{h} = g \implies u_0 = u_1 - hg$, which yields $u_1 - u_2 = h^2 f_1 - hg$, but is only $O(h)$ accurate at the boundary.

With this elimination approach, the interior matrix retains the symmetry. If Dirichlet nodes are present, the reduced system is SPD. With pure Neumann data on all sides,

the matrix is symmetric positive semidefinite (SPSD) with a one-dimensional constant nullspace. Uniqueness must be enforced by imposing a zero-mean constraint, that is, $1^\top u = 0$, or pinning one node.

Mixed boundary conditions. If $\partial\Omega = \Gamma_D \cap \Gamma_N$ with Dirichlet and Neumann data $u = g_B$ and $\frac{\partial u}{\partial n} = g_N$ on Γ_D and Γ_N, respectively, then we treat the two parts as follows.

- Γ_D: boundary nodes are eliminated by substituting their values and dropping their unknowns. Each Dirichlet boundary neighbour contributes its value to the right-hand side of the adjacent interior point equation.
- Γ_N: for each interior node adjacent to a Neumann side, the boundary is eliminated by using a second-order midpoint relation and substituting into the five-point, or three-point for one-dimensional problems, stencil.

Robin boundary data. If a boundary segment imposes a condition of the form $\alpha u + \beta\frac{\partial u}{\partial n} = r$, then it can be eliminated using the relation

$$u = \frac{r - \beta\frac{\partial u}{\partial n}}{a}.$$

$\frac{\partial u}{\partial n}$ is substituted using the same midpoint difference as above. Robin conditions result in a small diagonal shift caused by αu.

3.4 Spectral properties

3.4.1 Kernel, connectivity, and spectral gap.

Dirichlet energy. For interior node values $u \in \mathbb{R}^n$ and interior combinatorial Laplacian L_{II}, the *discrete Dirichlet energy* is defined as

$$E(u) = \frac{1}{2}u^\top L_{II} u = \frac{1}{2}\sum_{\{i,j\}\in E_{\text{int}}} (u_i - u_j)^2,$$

and measures how much u varies between neighbouring grid points. Low energy implies that u is smooth, while high energy indicates that u exhibits oscillatory behaviour.
The induced energy norm of u is

$$||u||_{L_{II}} = \sqrt{u^\top L_{II} u} = \sqrt{\sum_{\{i,j\}\in E_{\text{int}}} (u_i - u_j)^2}.$$

On a uniform grid with spacing h and for sufficiently smooth u, the discrete quantities are second-order accurate:

$$E(u) \;=\; \tfrac{1}{2}\int_\Omega \|\nabla u\|^2\,dx\,dy \;+\; O(h^2), \qquad \|u\|_{L_{II}} \;=\; \Big(\int_\Omega \|\nabla u\|^2\,dx\,dy\Big)^{1/2} \;+\; O(h^2).$$

Connectivity. Let $G_{int} = (I, E_{\text{int}})$ be the **interior** grid graph, where I is the set of interior nodes and E_{int} is the set of undirected edges, $\{i, j\}$, connecting interior nearest-neighbour pairs. Two interior nodes are **connected** if there exists a chain of interior edges connecting them. This partitions I into connected components $C_1, \ldots, C_c$. For the rectangular domains used in this dissertation, G_{int} is connected ($c = 1$).
If a mask cuts the interior grid graph into "islands" (blocks/subgraphs), then after rearranging unknowns by components, the interior Dirichlet matrix becomes block diagonal:

$$P^\top L_{II} P = \text{blockdiag}\left(L^{(1)}, \ldots, L^{(c)}\right)$$

The interior points removed are treated as Dirichlet nodes with prescribed values, typically 0. Since the boundary nodes are not vertices of G_{int}, any path that would have to walk through them to connect two interior points on different islands does not count; therefore, the blocks decouple algebraically.
Each block remains **SPD** and the problem is split into separate sub-problems that can be solved independently, or in parallel, without affecting definiteness.

Kernel (nullspace). The kernel (nullspace) of a matrix M is $\ker(M) = \{x : Mx = 0\}$. In the case of the interior Dirichlet Laplacian L_{II}, we have $\ker(L_{II}) = \{0\}$.
Let $d_\partial(i)$ be the number of Dirichlet boundary neighbours of interior node i. The quadratic form of L_{II} can be written as

$$u^\top L_{II} u = \tfrac{1}{2} \sum_{\{i,j\}\in E_{\text{int}}} (u_i - u_j)^2 \;+\; \sum_{i\in I} d_\partial(i)\, u_i^2,$$

where the first sum iterates over undirected edges of the interior graph and is counted once, and the second sum term results from edges that connect interior nodes to Dirichlet nodes. If $u \in \ker(L_{II})$, then $u^\top L_{II} u = 0$. The first sum gives $u_i = u_j$ on every interior edge; therefore, u is constant along any chain of interior neighbours. From the second sum, any interior node with $d_\partial(i) > 0$ must satisfy $u_i = 0$. Starting from any interior node, this chain reaches a boundary-adjacent interior node; hence the constant must be 0 and $u = 0$.

Dirichlet spectral gap. Let the eigenvalues of the interior Dirichlet Laplacian L_{II} be

$0 < \lambda_1 \leq \lambda_2 \cdots \leq \lambda_n$. The smallest eigenvalue,

$$\lambda_1 = \min_{u \neq 0} \frac{u^\top L_{II} u}{\|u\|_2^2},$$

is the **Dirichlet spectral gap**. It yields the discrete Poincaré inequality,

$$u^\top L_{II} u \geq \lambda_1 \|u\|_2^2 \quad \forall u \in \mathbb{R}^n,$$

so the energy controls the interior ℓ^2-size.
Explicit formulas for the spectrum on rectangular grids and their consequences for the condition number $\kappa(L_{II})$ are given in 3.4.2.

3.4.2 Eigenpairs for two-dimensional finite differences.

Isotropic grids. On an $n_x \times n_y$ interior grid with uniform spacing h, the interior Dirichlet matrix is the Kronecker sum of one-dimensional Dirichlet Toeplitz matrices (3.3.3):

$$L_{II} = T_{n_x} \otimes I_{n_y} + I_{n_x} \otimes T_{n_y}, \quad T_n = \text{tridiag}(-1, 2, -1).$$

Separation of variables results in a complete orthogonal basis of eigenvectors expressed as

$$u_{i,j}^{(k,\ell)} = \sin\left(\frac{ki\pi}{n_x + 1}\right) \sin\left(\frac{\ell j \pi}{n_y + 1}\right), \quad i = 1, \ldots, n_x, \quad j = 1, \ldots, n_y$$

with eigenvalues

$$\lambda_{k,\ell} = 4 \sin^2\left(\frac{k\pi}{2(n_x + 1)}\right) + 4 \sin^2\left(\frac{\ell \pi}{2(n_y + 1)}\right), \quad k = 1, \ldots, n_x, \quad \ell = 1, \ldots, n_y,$$

which correspond to the **Discrete Sine Transform of type I (DST-I)** at each coordinate. If S_x, S_y are the one-dimensional DST-I orthonormal transforms, then $(S_x \otimes S_y)^\top L_{II} (S_x \otimes S_y)$ is diagonal with diagonal entries $\lambda_{k,\ell}$.

Scaled and anisotropic variants. For the $1/h^2$-scaled operator, $L_{II,h} = (1/h^2) L_{II}$, the eigenvalues are

$$\lambda_{k,\ell}^{(h)} = \frac{\lambda_{k,\ell}}{h^2}.$$

If the grid parameters differ across axes, that is, $h_x \neq h_y$, then the scaled Laplacian operator is

$$L_{II,h} = \frac{1}{h_x^2} T_{n_x} \otimes I + \frac{1}{h_y^2} I \otimes T_{n_y},$$

and the eigenvalues are

$$\lambda_{k,\ell} = \frac{4}{h_x^2}\sin^2\left(\frac{k\pi}{2(n_x+1)}\right) + \frac{4}{h_y^2}\sin^2\left(\frac{\ell\pi}{2(n_y+1)}\right).$$

With $L_x = (n_x+1)h_x$, $L_y = (n_y+1)h_y$ fixed and $h_x, h_y \to 0$,

$$\lambda_{k,\ell}^{(h_x,h_y)} \to \left(\frac{k\pi}{L_x}\right)^2 + \left(\frac{\ell\pi}{L_y}\right)^2.$$

Conditioning (rectangles). For the combinatorial operator, the smallest and largest eigenvalues occur at $(k,\ell) = (1,1)$ and $(k,\ell) = (n_x, n_y)$:

$$\lambda_{\min} = \lambda_{1,1} = 4\sin^2\!\left(\tfrac{\pi}{2(n_x+1)}\right) + 4\sin^2\!\left(\tfrac{\pi}{2(n_y+1)}\right),$$

and

$$\lambda_{\max} = 4\sin^2\!\left(\tfrac{n_x\pi}{2(n_x+1)}\right) + 4\sin^2\!\left(\tfrac{n_y\pi}{2(n_y+1)}\right) \;\nearrow\; 8 \quad (n_x, n_y \to \infty).$$

The condition number is $\kappa(L_{II}) = \frac{\lambda_{\max}}{\lambda_{\min}}$.

Using $\sin x \approx x$ and writing $L_x = (n_x+1)h$ and $L_y = (n_y+1)h$, the eigenvalues are rewritten as

$$\lambda_{\min} \;=\; \pi^2\!\left(\tfrac{h}{L_x}\right)^2 + \pi^2\!\left(\tfrac{h}{L_y}\right)^2 + O(h^4),$$

which yields $\kappa(L_{II}) = \Theta(h^{-2})$ [20].

On a unit square $L_x = L_y = 1$ and $h \approx 1/(n+1)$, therefore

$$\lambda_{\min} \approx 2\pi^2 h^2$$

and

$$\kappa(L_{II}) \approx \frac{8}{2\pi^2 h^2} = \frac{4}{\pi^2}\,h^{-2} \approx 0.405\,(n+1)^2.$$

For the $1/h^2$-scaled operator $L_{II,h}$ we have

$$\lambda_{\max}(L_{II,h}) \approx \frac{8}{h^2},$$

and

$$\lambda_{\min}(L_{II,h}) \approx \left(\frac{\pi}{L_x}\right)^2 + \left(\frac{\pi}{L_y}\right)^2,$$

with $\kappa(L_{II,h}) = \Theta(h^{-2})$.

In the case of anisotropic spacings, that is, $h_x \neq h_y$, $\kappa = \Theta\big(\max\{h_x^{-2},\, h_y^{-2}\}\big)$.

3.4.3 Rayleigh quotient

Rayleigh quotient and energy. For any nonzero vector u, the Rayleigh quotient is

$$\mathcal{R}(u) = \frac{u^\top L_{II} u}{\|u\|_2^2}.$$

According to the min-max principle, the eigenvalues of L_{II} are the extreme values of $\mathcal{R}$, and the corresponding eigenvectors are exactly the sine modes mentioned in 3.4.2; that is, $\lambda_1 \leq \mathcal{R}(u) \leq \lambda_{\max}$, with equality iff u is an eigenvector of L_{II}.
Let $\{v^{(k,\ell)}\}$ be the orthonormal sine eigenbasis of 3.4.2. Expanding u, we get that

$$u = \sum_{k=1}^{n_x}\sum_{\ell=1}^{n_y} a_{k,\ell} v^{(k,\ell)}, \quad \|v^{(k,\ell)}\|_2 = 1,$$

therefore,

$$u^\top L_{II} u \;=\; \sum_{k=1}^{n_x}\sum_{\ell=1}^{n_y} \lambda_{k,\ell}\, \alpha_{k,\ell}^2, \qquad \|u\|_2^2 \;=\; \sum_{k=1}^{n_x}\sum_{\ell=1}^{n_y} \alpha_{k,\ell}^2.$$

The Rayleigh quotient can be written as

$$\mathcal{R}(u) = \frac{\sum_{k=1}^{n_x}\sum_{\ell=1}^{n_y} \lambda_{k,\ell}\, \alpha_{k,\ell}^2}{\sum_{k=1}^{n_x}\sum_{\ell=1}^{n_y} \alpha_{k,\ell}^2}.$$

It is obvious that $\mathcal{R}(u)$ is a weighted average of eigenvalues. Large (small) eigenvalues indicate high (low) spatial frequency and increase (decrease) $\mathcal{R}(u)$.

The energy and Euclidean norms are linked by

$$\lambda_1 \|u\|_2^2 \;\leq\; u^\top L_{II} u \;\leq\; \lambda_{\max} \|u\|_2^2 \quad\Longleftrightarrow\quad \sqrt{\lambda_1}\|u\|_2 \;\leq\; \|u\|_{L_{II}} \;\leq\; \sqrt{\lambda_{\max}}\|u\|_2,$$

where $\|u\|_{L_{II}} = \sqrt{u^\top L_{II} u}$. The left inequality is the *discrete Poincaré inequality*:

$$u^\top L_{II} u \geq \lambda_1 \|u\|_2^2, \;\; \forall u \in \mathbb{R}^n,$$

with equality iff u is an eigenvector associated with λ_1.

Finally, solving $L_{II}\, u = f$ is equivalent to minimising the strictly convex quadratic function $\Phi(u) = \frac{1}{2} u^\top L_{II} u - f^\top u$, with $\nabla\Phi(u) = L_{II} u - f$, whose unique minimiser, u^*,

satisfies $\nabla\Phi(u^*) = 0 \iff L_{II}u^* = f$.

The consequences of these spectral properties for iterative methods (Jacobi, Gauss–Seidel, SOR, CG, PCG) are discussed in 4.1, where iteration counts per method and parameter choices are presented in more detail.

Chapter 4

Solvers for linear systems

4.1 Iterative algorithms

In numerical linear algebra, there are two primary categories of methods of solving linear systems: direct and iterative. Direct methods, such as Gaussian elimination or LU decomposition, aim to compute the exact solution in a finite number of operations. Even though they are robust and reliable for small to moderately sized problems, their computational cost and memory requirements become prohibitive for large sparse systems, especially those forming from the discretisation of partial differential equations. In contrast, iterative solvers start with an initial guess and gradually refine the solution until converging to the true solution within a prescribed tolerance [21, 20, 10].
In the following subsections, three fundamental iterative methods are presented: the Jacobi method, the Gauss-Seidel method, and the Successive Over-Relaxation (SOR) method. These methods generate the next iterate using a fixed update formula; thus, they are characterised as stationary. Furthermore, matrix splitting is described as a general framework for deriving and analysing these schemes.

4.1.1 Matrix Splitting

Matrix splitting is a technique in numerical linear algebra that is used to represent a given matrix as a sum or a difference of matrices. This decomposition is used to simplify the solution of the linear system or to analyse the properties of the original coefficient matrix. Several iterative methods are based on the idea of matrix splitting.
Consider the linear system

$$Au = b \tag{4.1}$$

be a system of linear equations, where $A \in \mathbb{R}^{n\times n}$ is a non-singular matrix and u, b are vectors in $\mathbb{R}^n$.

Matrix A can be split as

$$A = \Gamma - \Delta$$

where Γ is chosen in a way that is (easily) invertible and Δ is the remainder. The system can be rewritten as

$$(\Gamma - \Delta)u = b \implies \Gamma u = \Delta u + b,$$

and the associated fixed-point iteration formula is

$$u^{(k)} = \Gamma^{-1}\Delta u^{(k-1)} + \Gamma^{-1}b.$$

Equivalently, we can write

$$u^{(k)} = Tu^{(k-1)} + c,$$

where $T = \Gamma^{-1}\Delta$ is the **iteration matrix**, and $c = \Gamma^{-1}b$. The method requires an initial, usually random, guess $u^{(0)}$ to start.

It is often useful to write the update in the following form:

$$T = \Gamma^{-1}\Delta = I - \Gamma^{-1}A,$$

so

$$u^{(k)} = u^{(k-1)} + \Gamma^{-1}\big(b - Au^{(k-1)}\big) = u^{(k-1)} + \Gamma^{-1}r^{(k-1)},$$

where $r^{(k-1)} = b - Au^{(k-1)}$ is the **residual** [17, 21].
If $e^{(k)} = u - u^{(k)}$ is the error, then

$$e^{(k)} = T\,e^{(k-1)}, \qquad r^{(k)} = Ae^{(k)}.$$

Convergence criterion. The fixed-point iteration $u^{(k)} = T\,u^{(k-1)} + c$, with $T = \Gamma^{-1}\Delta$ and $c = \Gamma^{-1}b$, converges to the unique solution $u = A^{-1}b$ for every initial guess $u^{(0)}$ iff$\rho(T) = \rho(\Gamma^{-1}\Delta) < 1$, where $\rho(T)$ is the **spectral radius** of the iteration matrix. The spectral radius of T is defined as follows [17, 21].

$$\rho(T) = \max_{i=1,2,\ldots,n} |\lambda_i| = \max\{|\lambda_1|, |\lambda_2|, \ldots, |\lambda_n|\}$$

For the error $e^{(k)} = u - u^{(k)}$, we have that

$$e^{(k)} = T^k\,e^{(0)} \implies \|e^{(k)}\| \leq \|T\|^k\|e^{(0)}\|. \tag{4.2}$$

Therefore, if $\rho(T) < 1$ there exists a matrix norm with $\|T\| < 1$ and the convergence is

geometric.

For large systems, directly computing $\rho(T)$ is often infeasible. Therefore, convergence is usually guaranteed by the structural properties of the coefficient matrix depending on the method used, such as diagonal dominance or positive definiteness.

Stopping criterion. For all iterative methods described below, convergence is determined by monitoring the norm of the *relative residual* [21]:

$$r_{\text{rel}}^{(k)} = \frac{\|Au^{(k)} - b\|_2}{\|b\|_2}.$$

The process ends when $r_{\text{rel}}^{(k)} < \epsilon$, where ϵ is a predefined tolerance.

If the system is homogeneous, that is, $\|b\|_2 = 0$, $\max\{1, \|b\|_2\}$ is used for the denominator. Alternatively, the norm of the absolute residual can be used: $\|Au^{(k)} - b\|_2$.

4.1.2 Jacobi Method

The **Jacobi method** is a classical stationary iterative algorithm for solving linear systems $Au = b$. It is defined for any square matrix A, and *converges* under standard sufficient conditions, for example, when A is *strictly (or irreducibly) diagonally dominant* by rows. In our two-dimensional Dirichlet Laplacian model, A is an M-matrix and is strictly diagonally dominant; therefore, Jacobi converges.
Let

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

and let $u^{(k)}$ denote the $k-$th iterate. Isolating u_i in the i-th equation gives

$$u_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} u_j \right), \quad a_{ii} \neq 0.$$

Each component u_i is updated using the values from the previous iterate (u_j), that is,

$$u_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} u_j^{(k-1)} \right), \quad i = 1, \ldots, n.$$

Matrix form and splitting. A can be written as $A = D + L + U$, where $D = \mathrm{diag}(A)$, L is strictly lower and U strictly upper. Then

$$u^{(k)} = D^{-1}\left(b - (L+U)u^{(k-1)}\right) = (I - D^{-1}A)u^{(k-1)} + D^{-1}b,$$

and the Jacobi iteration matrix is $T_J = I - D^{-1}A = -D^{-1}(L+U)$.

Convergence check. Let $e^{(k)} = u - u^{(k)}$. Then $e^{(k)} = T_J e^{(k-1)}$, hence $e^{(k)} = T_J^k e^{(0)}$ and $\|e^{(k)}\| \leq \|T_J\|^k \, \|e^{(0)}\|$. The Jacobi method converges for every initial guess $u^{(0)}$ iff $\rho(T_J) < 1$. Sufficient conditions include strict or irreducible diagonal dominance, or A being an M-matrix [17, 16].

Algorithm 4.1 Jacobi algorithm

Require: $A \in \mathbb{R}^{n\times n}$ (with nonzero diagonal), $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{u}^{(0)}$, tolerance ε, maximum iterations $N_{\max}$
Ensure: Approximate solution $\mathbf{u}$

$\mathbf{u} \leftarrow \mathbf{u}^{(0)}$
for $i = 1$ to n **do**
  if $A_{ii} = 0$ **then**
    error(``Jacobi: zero diagonal entry at row i'')
  end if
  $D_i^{-1} \leftarrow 1/A_{ii}$ ▷ store diagonal inverse
end for
$\beta \leftarrow \max\{1, \|\mathbf{b}\|_2\}$ ▷ robust scaling for relative residual
$r \leftarrow \|A\mathbf{u} - \mathbf{b}\|_2/\beta$
if $r < \varepsilon$ **then**
  return $\mathbf{u}$
end if
for $k = 1$ to $N_{\max}$ **do**
  $\mathbf{u}_{\text{next}} \leftarrow \mathbf{0}$
  for $i = 1$ to n **do**
    $s \leftarrow 0$
    for $j = 1$ to n **do**
      if $j \neq i$ **then**
        $s \leftarrow s + A_{ij}\, u_j$
      end if
    end for
    $u_{\text{next},i} \leftarrow D_i^{-1}\left(b_i - s\right)$

```
    end for
    r ← ‖A u_next − b‖_2 / β
    if r < ε then
        return u_next
    end if
    u ← u_next
end for
return u
```

For the five-point stencil interior Laplacian L_{II}, $D = 4I$, so

$$T_J = I - \frac{1}{4} L_{II}.$$

The eigenvectors of T_J are

$$\mu_{k,\ell} = 1 - \frac{1}{4}\lambda_{k,\ell} \in (-1, 1),$$

so $\rho(T_J) = \max_{k,\ell} |\mu_{k,\ell}| = 1 - \frac{1}{4}\lambda_{\min} < 1$ [21].

As the mesh is refined, that is, $h \to 0$, the smallest Laplacian eigenvalue is $\lambda_{\min} \approx \pi^2\left(\frac{h}{L_x}\right)^2 + \pi^2\left(\frac{h}{L_y}\right)^2$, so the Jacobi convergence factor behaves as

$$\rho(T_J) \approx 1 - \tfrac{1}{4}\lambda_{\min} \approx 1 - \frac{\pi^2}{4}\left(\tfrac{h}{L_x}\right)^2 - \frac{\pi^2}{4}\left(\tfrac{h}{L_y}\right)^2$$

[10, 20].
On a unit square ($L_x = L_y = 1$) it simplifies to $\rho(T_J) \approx 1 - \frac{\pi^2}{2} h^2$.

The Jacobi method eventually converges slowly on fine meshes. The iteration count to reduce the error by a factor ϵ scales like $\Theta(h^{-2} \log(1/\epsilon))$ [20].

4.1.3 Gauss-Seidel Method

The **Gauss-Seidel method** is an improvement over the Jacobi method, using the most recent updates within the same iteration to update the current approximation of the solution.
Matrix A can be decomposed as

$$A = D + L + U,$$

and the system can be rewritten as

$$(D + L)u = b - Uu$$

or

$$u^{(k)} = (D + L)^{-1} \left(b - Uu^{(k-1)}\right).$$

Equivalently, the correction form is

$$u^{(k)} \;=\; u^{(k-1)} + (D + L)^{-1}\big(b - A\,u^{(k-1)}\big).$$

The iteration matrix is defined as

$$T_{\text{GS}} = -(D + L)^{-1}U = I - (D + L)^{-1}A.$$

The element-wise formula is as follows.

$$u_i^{(k)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} u_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} u_j^{(k-1)} \right)$$

The two summations present in the above formula can be expressed as one summation, $\sum_{j \neq i} a_{ij} u_j$, that utilises the most recent computed values of u_j from the same iteration.

Gauss-Seidel converges for every initial guess iff $\rho(T_{\text{GS}}) < 1$). Sufficient conditions are:

- A is symmetric positive-definite
- A is strictly or irreducibly diagonally dominant

For an SPD matrix A, Gauss-Seidel monotonically decreases the quadratic function $\Phi(u) = \frac{1}{2}u^\top Au - b^\top u$ and converges for any initial guess [17, 21].

Algorithm 4.2 Gauss–Seidel algorithm (in-place)

Require: $A \in \mathbb{R}^{n \times n}$ (with nonzero diagonal), $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{u}^{(0)}$, tolerance ε, maximum iterations N_{max}

Ensure: Approximate solution $\mathbf{u}$

1: $\mathbf{u} \leftarrow \mathbf{u}^{(0)}$
2: **for** $i = 1$ to n **do**
3: **if** $A_{ii} = 0$ **then**
4: **error**(``Gauss--Seidel: zero diagonal entry at row i'')
5: **end if**

```
    D_i^{-1} ← 1/A_ii
end for
β ← max{1, ‖b‖_2}                          ▷ robust scaling for relative residual
r ← ‖Au − b‖_2/β
if r < ε then
    return u
end if
for k = 1 to N_max do
    for i = 1 to n do
        s ← 0
        for j = 1 to n do
            if j ≠ i then
                s ← s + A_ij u_j            ▷ uses updated u_j if j < i, old if j > i
            end if
        end for
        u_i ← D_i^{-1} (b_i − s)            ▷ in-place update
    end for
    r ← ‖Au − b‖_2/β
    if r < ε then
        return u
    end if
end for
return u
```

As mentioned above, the Gauss-Seidel method uses the elements of $u^{(k)}$ and only the elements of $u^{(k-1)}$ that have not been computed in the k-th iteration. This means that only one storage vector is required since the elements can be overwritten during computation (in-place updates), unlike the Jacobi method, which stores the solution vector of the previous iteration. This can be advantageous for large systems where space becomes critical. However, the Gauss-Seidel method is much harder to parallelise due to its sequential nature; that is, each $u_i^{(k)}$ depends on previously updated values in the same iteration.

Dirichlet Laplacian. The general formula of the five-point stencil including a source term is

$$\frac{-u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} + 4u_{i,j}}{h^2} = f_{i,j},$$

or equivalently,

$$4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2 f_{i,j} + b_{i,j},$$

where $b_{i,j}$ is the sum of the prescribed Dirichlet values adjacent to the interior grid points.

Following a lexicographic sweep, that is, left-to-right within each row, and bottom-to-top over rows, Gauss-Seidel uses newly updated neighbours, when they are available, on the sweep direction:

$$u_{i,j}^{(k)} = \frac{1}{4}\left(h^2 f_{i,j} + b_{i,j} + u_{i-1,j}^{(k)} + u_{i+1,j}^{(k-1)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k-1)}\right).$$

In our model, $D = 4I$ and the iteration matrix is $T_{\text{GS}} = I - (D + L)^{-1}A$. As the mesh is refined, the spectral radius satisfies $\rho(T_{\text{GS}}) = 1 - \Theta(h^2)$, so the number of sweeps required to reduce the error by a factor ϵ scales like $k = \Theta\big(h^{-2}\log(1/\varepsilon)\big)$.
For reference, the one-dimensional factors are explicit:

$$\rho(T_J) = \cos\Big(\frac{\pi}{n+1}\Big), \qquad \rho(T_{\text{GS}}) = \cos^2\Big(\frac{\pi}{n+1}\Big) \;\approx\; 1 - \pi^2 h^2.$$

Therefore, Gauss-Seidel damps the error roughly twice as fast as Jacobi, requiring fewer sweeps [17]. The same h^2-scaling holds for the two-dimensional case as well.

4.1.4 Successive Over-Relaxation Method

The **Successive Over-Relaxation (SOR) method** is a variant of the Gauss-Seidel method, which displays faster convergence by introducing a relaxation parameter, $\omega \in (0, 2)$. Like Gauss-Seidel, the solution is updated sequentially, but each new value incorporates a weighted average between the newly computed and the previous iterate.

$$u^{(k)} = (1 - \omega)u^{(k-1)} + \omega \cdot \tilde{u}^{(k)} \tag{4.3}$$

where $\tilde{u}^{(k)}$ is the usual Gauss-Seidel update.
Like in the Jacobi method, the coefficient matrix can be decomposed as follows.

$$A = D + L + U.$$

The system may be rewritten as

$$u = (D + L)^{-1}b - (D + L)^{-1}Uu$$

resulting in the fixed-point update formula

$$u^{(k)} = -(D+L)^{-1}Uu^{(k-1)} + (D+L)^{-1}b.$$

This formula corresponds to the update formula of the Gauss-Seidel method. Therefore,

$$\tilde{u}^{(k)} = (D+L)^{-1}\big(b - U\,u^{(k-1)}\big). \tag{4.4}$$

If we plug in equation 4.4 in equation 4.3 we obtain the following fixed-point update formula.

$$u^{(k)} = (1-\omega)u^{(k-1)} + \omega(D+L)^{-1}(b - Uu^{(k-1)})$$

Multiplying both sides by $(D+\omega L)$ we get

$$(D+\omega L)u^{(k)} = -[\omega U + (\omega-1)D]u^{(k-1)}) + \omega b.$$

Thus, the iterate update formula is as follows.

$$u^{(k)} = (D+\omega L)^{-1}(\omega b - [\omega U + (\omega-1)D]u^{(k-1)}) \tag{4.5}$$

The iteration matrix and constant are

$$T_{\text{SOR}}(\omega) = (D+\omega L)^{-1}\big[(1-\omega)D - \omega U\big], \qquad c_{\text{SOR}}(\omega) = \omega(D+\omega L)^{-1}b.$$

The algorithm of the SOR method is presented below.

Algorithm 4.3 Successive Over-Relaxation (SOR) — in-place

Require: $A \in \mathbb{R}^{n\times n}$ (nonzero diagonal), $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{u}^{(0)}$, relaxation $\omega \in (0,2)$, tolerance ε, max iters $N_{\max}$

Ensure: Approximate solution $\mathbf{u}$

$\mathbf{u} \leftarrow \mathbf{u}^{(0)}$
if $\omega \le 0$ **or** $\omega \ge 2$ **then**
    error("SOR: relaxation ω must be in (0,2)")
end if
for $i = 1$ **to** n **do**
    if $A_{ii} = 0$ **then**
        error("SOR: zero diagonal at row i")
    end if
    $D_i^{-1} \leftarrow 1/A_{ii}$ ▷ precompute diagonal inverse
end for

$\beta \leftarrow \max\{1, \|\mathbf{b}\|_2\}$ ▷ robust scaling for relative residual
$r \leftarrow \|A\mathbf{u} - \mathbf{b}\|_2 / \beta$
if $r < \varepsilon$ **then**
    return $\mathbf{u}$
end if
for $k = 1$ **to** $N_{\max}$ **do**
    for $i = 1$ **to** n **do**
        $s \leftarrow 0$
        for $j = 1$ **to** n **do**
            if $j \neq i$ **then**
                $s \leftarrow s + A_{ij}\, u_j$ ▷ uses updated u_j if $j < i$, old if $j > i$
            end if
        end for
        $u_i \leftarrow (1 - \omega)\, u_i \; + \; \omega\, D_i^{-1} \left(b_i - s\right)$ ▷ SOR in-place update
    end for
    $r \leftarrow \|A\mathbf{u} - \mathbf{b}\|_2 / \beta$
    if $r < \varepsilon$ **then**
        return $\mathbf{u}$
    end if
end for
return $\mathbf{u}$

The elements of $u^{(k)}$ can be computed sequentially if the triangular form of $(D + \omega L)$ utilised via forward substitution.

$$u_i^{(k)} = (1-\omega)u_i^{(k-1)} + \frac{\omega}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij} u_j^{(k)} - \sum_{j=i+1}^{n} A_{ij} u_j^{(k-1)} \right), \quad i = 1, \ldots, n.$$

The expression above can be written in matrix-vector form as

$$u^{(k)} = (1-\omega)u^{(k-1)} + \omega D^{-1}(b - Lu^{(k)} - Uu^{(k-1)}) \tag{4.6}$$

without inverting $(D + \omega L)$.

The SOR method converges for any $u^{(0)}$ iff $\rho(T_{\text{SOR}}) < 1$, where T_{SOR} is the iteration matrix. Furthermore, diagonal dominance and symmetric positive definiteness of the coefficient matrix are sufficient conditions to guarantee convergence. For SPD matrices, convergence is guaranteed for any relaxation factor $\omega \in (0, 2)$.

The choice of relaxation factor depends on the spectral properties of the system matrix and is often determined experimentally. For structured problems, such as the Laplace equation, we can perform Fourier analysis to calculate the optimal ω.
For an interior grid $n \times n$, ω can be calculated according to the following expression:

$$\omega_{\text{opt}} \approx \frac{2}{1 + \sin(\frac{\pi}{n+1})}$$

[17, 22].

For $\omega = 1$ the SOR method reduces to the Gauss-Seidel method, while for $\omega < 1$ and $\omega > 1$ we have under-relaxation, where the method is more stable but slow to converge, and over-relaxation, which is potentially faster but may diverge if ω exceeds 2, respectively.

For the two-dimensional Dirichlet Laplacian on an $n \times n$ interior grid, the optimal relaxation parameter, ω_{opt} yields a spectral radius $\rho\big(T_{\text{SOR}}(\omega_{\text{opt}})\big) = 1 - \Theta(h) = 1 - \Theta\big(1/n\big)$. Using the error recursion formula (4.2), we find that the number of sweeps that the SOR method needs to reduce the error by a factor ϵ is

$$k = \Theta\big(h^{-1} \log(1/\varepsilon)\big) = \Theta\big(n \log(1/\varepsilon)\big)$$

[17, 20].
For comparison, on the same grid, Jacobi and Gauss-Seidel have $\rho = 1 - \Theta(h^2)$, hence $k = \Theta\big(h^{-2} \log(1/\varepsilon)\big) = \Theta\big(n^2 \log(1/\varepsilon)\big)$.

4.2 Krylov subspace methods

While the stationary iterative methods analysed above can effectively solve certain classes of problems (e.g. well-structured models that result in diagonally dominant matrices), they often display slow convergence on large sparse systems. In addition, their dependence on matrix splitting limits the structural information they can exploit, and convergence may be sensitive parameter choices and the initial guess.

Krylov subspace methods provide a more general framework. They build approximate solutions by exploring a subspace generated by successive applications of the system matrix to the initial residual and do not rely on matrix splitting or local smoothing. In

contrast to stationary iterative methods that apply local corrections, Krylov subspace methods build a global search space that captures matrix-wide behaviour, by performing simple updated based on local or component-wise corrections. This leads to significantly faster convergence, especially when the system matrix is symmetric positive definite, as in the case of the discretised two-dimensional Laplace equation with Dirichlet data [21, 23, 24].

Mathematical insights

Definition 4.1 (Krylov subspace)**.** *The order-m Krylov subspace generated by $A \in \mathbb{R}^{n\times n}$, and $b \in \mathbb{R}^n$, is defined as the linear subspace spanned by the images of b under the first powers of A, starting from $A^0 = I$:*

$$K_m(A, b) = \mathit{span}\{b, Ab, A^2b, \ldots, A^{(m-1)}b\},$$

with the nesting $K_1 \subseteq K_2 \subseteq K_3 \subseteq \ldots$ [21, 24].
The order, m, of the subspace refers to the (maximal) number of linearly independent basis vectors that the subspace contains.

Let $A \in \mathbb{R}^{n\times n}$ be a large, sparse, and nonsingular matrix, $b \in \mathbb{R}^n$ the right-hand side of the linear system $Au = b$, $u^{(0)}$ the initial guess of the solution of the system, and $r^{(0)} = b - Au^{(0)}$ the initial residual. Krylov methods construct approximations of the solution using the *residual* Krylov subspace:

$$K_m(A, r^{(0)}) \coloneqq \text{span}\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \ldots, A^{(m-1)}r^{(0)}\}.$$

The approximate solution, $u^{(m)}$, is then sought in the affine space

$$u^{(m)} \in u^{(0)} + K_m(A, r^{(0)}),$$

and is defined by an optimality condition. This condition typically involves minimising the residual in some norm or enforcing orthogonality conditions.

The majority of Krylov methods are **projection methods**; that is, they seek an approximate solution, $u^{(m)} \in u^{(0)} + K_m(A, r^{(0)})$, such that

$$\langle r^{(m)}, w\rangle = 0 \quad \forall w \in L_m,$$

where $r^{(m)} = b - Au^{(m)}$ is the residual at iteration m, and $L_m \subseteq \mathbb{R}^n$ is a test (constraint) subspace of dimension m [23, 21]. Here, K_m is the search space. In many Krylov meth-

ods we also have $r^{(m)} \in K_{m+1}(A, r^{(0)})$.

The concept behind Krylov subspace methods is to compute a sequence of approximate solutions, $u^{(m)}$, such that their corresponding residuals, $r^{(m)}$, converge to zero. In exact arithmetic, if m reaches the degree of the minimal polynomial of A with respect to $r^{(0)}$, the residual vanishes, and $u^{(m)}$ is the exact solution.

For **non-symmetric square** systems, popular choices include **GMRES**, **BiCG**, and **BiCGSTAB**. For **symmetric (possibly indefinite)** systems, **MINRES** applies. For least-squares/rectangular problems, **LSQR** or **LSMR** are used. The convergence behaviour depends on spectral features (eigenvalue clustering, condition number) and, for non-symmetric problems, on the field of values.

4.2.1 Conjugate Gradient

The **Conjugate Gradient method (CG)** [25, 21, 10, 26], is a Krylov subspace method suitable for solving systems that are symmetric positive definite (SPD) or Hermitian positive definite. In this project, the discretisation of the two-dimensional Laplace equation with Dirichlet boundary conditions over a rectangular domain yields a large, sparse, symmetric, SPD system

$$Au = b,$$

as discussed earlier.
The CG method is a powerful alternative to the classical stationary methods analysed above. Because CG exploits symmetry and positive definiteness, it ensures convergence in at most n steps in exact arithmetic, where n is the dimension of the system.

In terms of memory and computational overhead, CG is well suited for large-scale problems, especially in a two- or three-dimensional setting. Unlike Jacobi or Gauss-Seidel, which apply local row-wise updates, CG builds global search directions from Krylov subspaces and naturally exploits sparsity, making it a standard choice for the sparse matrices arising from elliptic partial differential equations and quadratic optimisation.

CG as a minimisation problem. Given that A is SPD, solving the system $Au = b$ is equivalent to minimising the strictly convex quadratic function $\phi(u) = \frac{1}{2}u^\top Au - b^\top u$ [25, 23].
Its gradient is $\nabla\phi(u) = Au - b$, and the unique minimiser, $\tilde{u}$, satisfies $Au = b$. The CG method constructs a sequence of approximations, $u^{(k)}$, by minimising ϕ over expanding

Krylov subspaces:

$$u^{(k)} \in u^{(0)} + K_k(A, r^{(0)}),$$

where $K_k(A, r^{(0)}) = \text{span}\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \ldots, A^{(k-1)}r^{(0)}\}$ and $r^{(0)} = b - Au^{(0)}$.

The level sets of ϕ are ellipsoids. A convenient description is

$$\phi(x) = \frac{1}{2}x^\top Ax - b^\top x = C, \tag{4.7}$$

whose contours shrink to the centre at the minimiser.

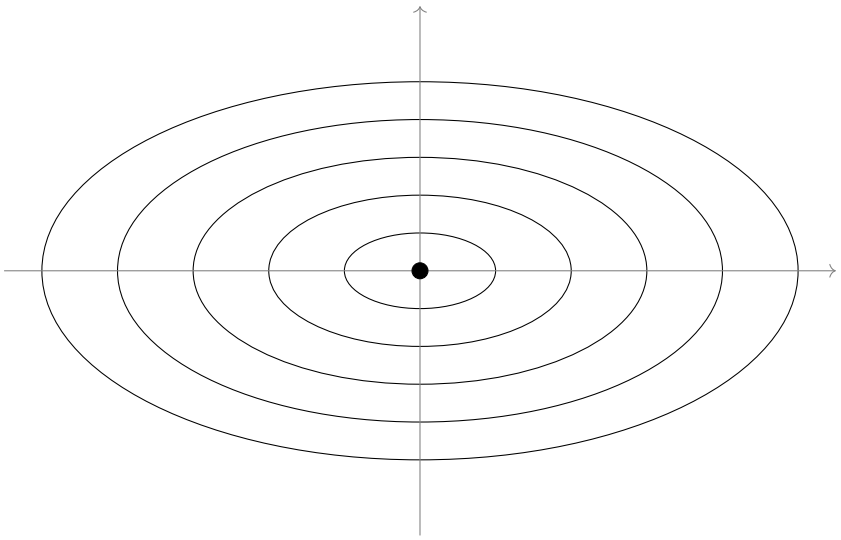


Figure 4.1: Level sets of the quadratic function $x^\top Ax - 2b^\top x = C$.

In plain terms, minimising the quadratic function ϕ reduces to "walking downhill" toward the centre (global minimiser) of its elliptical level sets. At iteration k, CG moves along the affine line

$$u^{(k)} + ad^{(k)}, \quad a \in \mathbb{R},$$

and selects the point on that line with the smallest value of ϕ:

$$a^{(k)} = \arg\min_{a \in \mathbb{R}} \phi(u^{(k)} + ad^{(k)}).$$

The selected point is the point where the new gradient is orthogonal to the search direction (optimality condition):

$$\nabla\phi(u^{(k+1)})^\top d^{(k)} = 0.$$

We start from an arbitrary initial guess, $u^{(0)}$, and set the initial residual (which is equal to $-\nabla\phi(u^{(0)})$) to $r^{(0)} = b - Au^{(0)}$. The first direction is $d^{(0)} = r^{(0)}$. The exact line search

gives the closed-form search step

$$\alpha^{(k)} = \frac{\left(r^{(k)}\right)^\top r^{(k)}}{\left(d^{(k)}\right)^\top A\, d^{(k)}}$$

[25, 26].
The iterate and residual are then updated:

$$u^{(k+1)} = u^{(k)} + \alpha^{(k)} d^{(k)}, \qquad r^{(k+1)} = r^{(k)} - \alpha^{(k)} A d^{(k)}.$$

New directions, $d^{(k)}$, are built to be A-conjugate to the previous ones, and not all directions a priori.

$$d^{(k+1)} = r^{(k+1)} + \beta^{(k)} d^{(k)},$$

where $\beta^{(k)} = \frac{(r^{(k+1)})^\top r^{(k+1)}}{(r^{(k)})^\top r^{(k)}}$ is a scalar chosen to enforce conjugacy. The directions satisfy $(d^{(i)})^\top A d^{(j)} = 0$ for $i \neq j$, and the residuals are mutually orthogonal, that is, $\left(r^{(i)}\right)^\top r^{(j)} = 0$ for $i \neq j$.

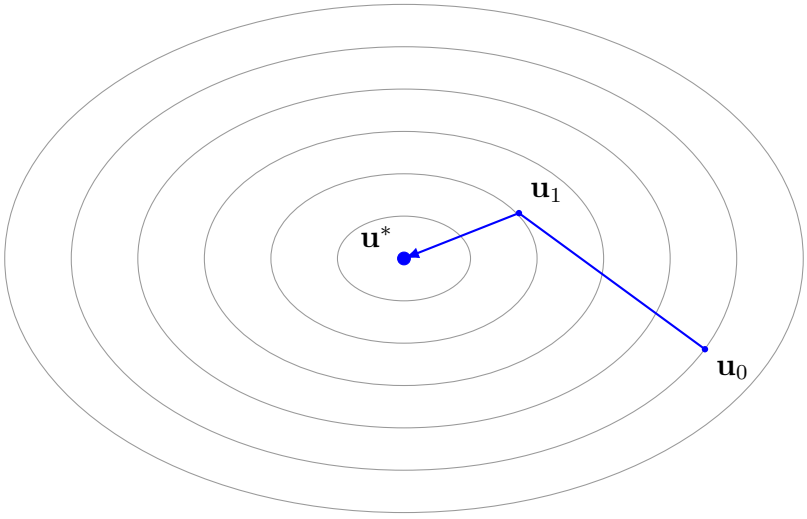


Figure 4.2: Conjugate Gradient steps on elliptical level sets of a convex quadratic function.

Algorithm 4.4 Conjugate Gradient Method

Require: $A \in \mathbb{R}^{n\times n}$ (SPD), $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{u}^{(0)}$, tolerance ε, maximum iterations $N_{\max}$

Ensure: Approximate solution $\mathbf{u}$

1: $\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\mathbf{u}^{(0)}$
2: $\mathbf{d}^{(0)} \leftarrow \mathbf{r}^{(0)}$
3: res $\leftarrow \|\mathbf{r}^{(0)}\|_2 / \max\{1, \|\mathbf{b}\|_2\}$

if res $< \varepsilon$ **then**
    return $\mathbf{u}^{(0)}$
end if
for $k = 0$ to $N_{\max} - 1$ **do**
    $\mathbf{q}^{(k)} \leftarrow A\mathbf{d}^{(k)}$
    $\alpha^{(k)} \leftarrow \frac{\left(\mathbf{r}^{(k)}\right)^\top \mathbf{r}^{(k)}}{\left(\mathbf{d}^{(k)}\right)^\top \mathbf{q}^{(k)}}$
    $\mathbf{u}^{(k+1)} \leftarrow \mathbf{u}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$
    $\mathbf{r}^{(k+1)} \leftarrow \mathbf{r}^{(k)} - \alpha^{(k)}\mathbf{q}^{(k)}$
    res $\leftarrow \|\mathbf{r}^{(k+1)}\|_2 / \max\{1, \|\mathbf{b}\|_2\}$
    if res $< \varepsilon$ **then**
        return $\mathbf{u}^{(k+1)}$
    end if
    $\beta^{(k)} \leftarrow \frac{\left(\mathbf{r}^{(k+1)}\right)^\top \mathbf{r}^{(k+1)}}{\left(\mathbf{r}^{(k)}\right)^\top \mathbf{r}^{(k)}}$
    $\mathbf{d}^{(k+1)} \leftarrow \mathbf{r}^{(k+1)} + \beta^{(k)}\mathbf{d}^{(k)}$
end for
return $\mathbf{u}^{(N_{\max})}$

Convergence behaviour. Given that $A \in \mathbb{R}^{n\times n}$ is SPD, CG minimises the $A-$norm of the error over $u^{(0)} + K_m(A, r^{(0)})$ and satisfies the bound

$$\|u^{(m)} - \tilde{u}\|_A \leq \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^m \|u^{(0)} - \tilde{u}\|_A,$$

where $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$ [23, 21, 26].
The inequality above suggests faster convergence when κ is small or the eigenvalues of A are tightly clustered. A practical iteration count estimate is

$$\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^m \leq \varepsilon \implies m \gtrsim \tfrac{1}{2}\sqrt{\kappa}\log\frac{2}{\varepsilon}.$$

For the two-dimensional Dirichlet Laplacian on a rectangle, both the combinatorial, L_{II}, and scaled, $L_{II,h} = \frac{1}{h^2}L_{II}$, matrices have $\kappa = \Theta(h^{-2}$. On a unit square, we have $\kappa(L_{II,h}) \approx \frac{4}{\pi^2}h^{-2}$, which results in $m \approx \frac{1}{\pi}h^{-1}\log\frac{2}{\varepsilon}$ [10, 20]. Therefore, CG requires $O(h^{-1}\log(1/\varepsilon))$ iterations. In practice, CG often outperforms these bounds, and convergence is faster, especially when effective preconditioning is applied.

4.3 Preconditioning

The convergence of iterative solvers, and particularly Krylov subspace methods, is strongly dependent on the spectral properties of the system matrix. Even if the system matrix is SPD, the problem may be **ill-conditioned**, leading to slow convergence. To address this, we apply a transformation to the system known as **preconditioning**. A **preconditioner** is a matrix $M \approx A$ that is easy to apply and for which the preconditioned operator $M^{-1}A$ is better conditioned [21, 27].

There are three standard ways to use a preconditioner:

- **Left preconditioning**: $M^{-1}Au = M^{-1}b$
- **Right preconditioning**: $AM^{-1}z = b$ with $u = M^{-1}z$
- **Symmetric (split) preconditioning**: if M is SPD with $M = LL^{\top}$, then $L^{-1}AL^{-T}y = L^{-1}b$ with $u = L^{-T}y$.

In practice, we never explicitly form M^{-1}. We apply M^{-1} to a vector by solving a small structured system involving M. The system is

$$M^{-1}Au = M^{-1}b.$$

Alternatively, we apply M^{-1} to the residual vector, $Au - b$, at each iteration, and solve for $z = M^{-1}(Au - b) \implies z = M^{-1}r$.

A good preconditioner should meet the following requirements:

1. The preconditioned system should be easy to solve.
2. The preconditioner should be inexpensive to construct and apply.

4.3.1 Preconditioners

Jacobi/ Diagonal preconditioner. The preconditioner is chosen to be the diagonal of the system matrix; that is, $M = \text{diag}(A)$. M^{-1} is defined as $\left[\frac{1}{A_{ii}}\right]$, so applying M^{-1} means scaling the residual, $r = Au - b$, element-wise:

$$z = M^{-1}r \iff z_i = \frac{r_i}{A_{ii}}$$

[21].

Successive Over-Relaxation (SSOR) preconditioner. Let $A = D+L+U$, where D, L, U are the diagonal, strictly lower, and strictly upper triangular components of A, respectively. The SSOR preconditioner is defined as

$$M_{\text{SSOR}} = \frac{\omega}{2-\omega} \left(D + \omega L\right) D^{-1} \left(D + \omega U\right)$$

[21, 17].
The application of the preconditioner is performed by solving two triangular systems instead of explicitly computing M_{SSOR}^{-1}.

Let $\tilde{L} \coloneqq D+\omega L$ and $\tilde{U} = D+\omega U$ be the lower and upper triangular matrices, respectively. If we rewrite $M_{\text{SSOR}} z = r$ we get

$$\frac{\omega}{2-\omega} \tilde{L} D^{-1} \tilde{U} z = r.$$

It is convenient to first solve the unscaled system and then apply the scaling at the end. After dropping the factor $\frac{2-\omega}{\omega}$ we obtain the system

$$\tilde{L} D^{-1} \tilde{U} z = r.$$

We define an intermediate unknown vector a and we have

$$\tilde{L} a = r.$$

Since $\tilde{L}$ and r are known, we can solve for a using forward substitution. After scaling a by $D = [A_{ii}]$ we get a second intermediate variable, $b = Da$.

To find z, we solve the system

$$\tilde{U} z = b$$

using backward substitution and scaling the result by $\frac{2-\omega}{\omega}$, thus getting

$$z \coloneqq \frac{2-\omega}{\omega} z.$$

Incomplete Cholesky factorisation preconditioner. For an SPD matrix, A, the **Incomplete Cholesky** factorisation computes a sparse lower triangular matrix, $\tilde{L}$, such that

$$A \;\approx\; \tilde{L}\, \tilde{L}^{\top},$$

where fills outside a chosen sparsity pattern are dropped during factorisation [21, 27].

Common drop strategies include

- **IC(0)**: the sparsity pattern of A is preserved
- **IC(k)**: allows up to level-of-fill k beyond the pattern of A
- **Threshold IC (ICT/IC-τ)**: entries whose magnitude is below the tolerance, τ, are dropped during factorisation

A higher level-of-fill (smaller drop tolerance) typically reduces iterations, but increases the setup time, memory, and the cost of each triangular solve.
The preconditioner is

$$M = \tilde{L}\tilde{L}^{\top}.$$

To compute $z = M^{-1}r$ we solve two triangular systems. We start by rewriting $Mz = r$ as

$$(\tilde{L}\tilde{L}^{\top})z = r \implies \tilde{L}^{\top}z = \tilde{L}^{-1}r.$$

We define an intermediate vector, a, such that

$$a = \tilde{L}^{-1}r \implies \tilde{L}a = r.$$

Since r and L are known, we solve for a using forward substitution. After determining a, we can solve $\tilde{L}^{\top}z = a$ for z using backward substitution and obtain z.

4.3.2 Preconditioned Conjugate Gradient

The **Preconditioned Conjugate Gradient (PCG) method** is an adaptation of the standard Conjugate Gradient method, which solves the left-preconditioned linear system $Au = b$ as $M^{-1}Au = M^{-1}b$, where A is SPD, and the preconditioner, M, is SPD and is inexpensive to apply [21, 23].

PCG builds iterates in the preconditioned Krylov space

$$u^{(k)} \in u^{(0)} + \mathcal{K}_k\big(M^{-1}A,\ M^{-1}r^{(0)}\big), \qquad r^{(0)} = b - Au^{(0)}.$$

The geometry of PCG mirrors that of the CG method: it minimises the quadratic function $\frac{1}{2}u^{\top}Au - b^{\top}u$ over that affine subspace.

From an algorithmic point of view, PCG only differs from CG in the *preconditioned residual*,

$$r^{(k)} = b - Au^{(k)}, \qquad \zeta^{(k)} = M^{-1}r^{(k)},$$

which appears in the step and direction updates.

We begin by initialising $\zeta^{(0)} = M^{-1}r^{(0)}$ and $d^{(0)} = \zeta^{(0)}$.

For in-loop updates, the step size is computed as

$$\alpha^{(k)} = \frac{(r^{(k)})^\top \zeta^{(k)}}{(d^{(k)})^\top q^{(k)}},$$

where $q^{(k)} = Ad^{(k)}$.
The iterate and residual are updated as

$$u^{(k+1)} = u^{(k)} + \alpha^{(k)} d^{(k)}$$

and

$$r^{(k+1)} = r^{(k)} - \alpha^{(k)} q^{(k)},$$

respectively.
The preconditioned residual, $\zeta^{(k+1)}$, and the new search direction, $d^{(k+1)}$, are computed as

$$\zeta^{(k+1)} = M^{-1}r^{(k+1)}, \quad d^{(k+1)} = \zeta^{(k+1)} + \beta^{(k)} d^{(k)},$$

where $\beta^{(k)} = \frac{(r^{(k+1)})^\top \zeta^{(k+1)}}{(r^{(k)})^\top \zeta^{(k)}}$. The residuals are mutually orthogonal in the $M^{-1}-$inner product and the directions remain $A-$conjugate, that is, $(d^{(i)})^\top A d^{(j)} = 0$ for $i \neq j$.

Algorithm 4.5 Preconditioned Conjugate Gradient Method

Require: $A \in \mathbb{R}^{n\times n}$ (SPD), $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{u}^{(0)}$, preconditioner M^{-1}, tolerance ε, maximum iterations $N_{\max}$
Ensure: Approximate solution $\mathbf{u}$

$\mathbf{r}^{(0)} \leftarrow \mathbf{b} - A\mathbf{u}^{(0)}$
$\boldsymbol{\zeta}^{(0)} \leftarrow M^{-1}\mathbf{r}^{(0)}$
$\mathbf{d}^{(0)} \leftarrow \boldsymbol{\zeta}^{(0)}$
$s \leftarrow \max\{1, \|\mathbf{b}\|_2\}$ ▷ robust scaling for relative residual
if $\|\mathbf{r}^{(0)}\|_2 / s < \varepsilon$ **then**
    return $\mathbf{u}^{(0)}$
end if

for $k = 0$ to $N_{\max} - 1$ **do**
  $\mathbf{q} \leftarrow A\mathbf{d}^{(k)}$
  $\alpha \leftarrow \frac{\mathbf{r}^{(k)} \cdot \boldsymbol{\zeta}^{(k)}}{\mathbf{d}^{(k)} \cdot \mathbf{q}}$
  $\mathbf{u}^{(k+1)} \leftarrow \mathbf{u}^{(k)} + \alpha \mathbf{d}^{(k)}$
  $\mathbf{r}^{(k+1)} \leftarrow \mathbf{r}^{(k)} - \alpha \mathbf{q}$
  if $\|\mathbf{r}^{(k+1)}\|_2 / s < \varepsilon$ **then**
    return $\mathbf{u}^{(k+1)}$
  end if
  $\boldsymbol{\zeta}^{(k+1)} \leftarrow M^{-1}\mathbf{r}^{(k+1)}$
  $\beta \leftarrow \frac{\mathbf{r}^{(k+1)} \cdot \boldsymbol{\zeta}^{(k+1)}}{\mathbf{r}^{(k)} \cdot \boldsymbol{\zeta}^{(k)}}$
  $\mathbf{d}^{(k+1)} \leftarrow \boldsymbol{\zeta}^{(k+1)} + \beta \mathbf{d}^{(k)}$
end for
return $\mathbf{u}^{(N_{\max})}$

Convergence behaviour. The convergence of PCG depends on the spectrum of the preconditioned operator. With A and M SPD the symmetric preconditioned matrix is defined as

$$\widetilde{A} \;=\; M^{-1/2}\, A\, M^{-1/2},$$

with condition number $\kappa(\widetilde{A}) = \kappa(M^{-1}A) = \frac{\lambda_{\max}(\widetilde{A})}{\lambda_{\min}(\widetilde{A})}$ and bounds

$$\|u^{(m)} - \tilde{u}\|_A \;\le\; \left(\frac{\sqrt{\kappa(\widetilde{A})} - 1}{\sqrt{\kappa(\widetilde{A})} + 1} \right)^m \|u^{(0)} - \tilde{u}\|_A$$

for the error and

$$\|r^{(m)}\|_2 \;\le\; \left(\frac{\sqrt{\kappa(\widetilde{A})} - 1}{\sqrt{\kappa(\widetilde{A})} + 1} \right)^m \|r^{(0)}\|_2$$

for the residual [21, 23].
The number of iterations to reduce the residual by a factor ϵ is $m \gtrsim \frac{1}{2}\sqrt{\kappa(\widetilde{A})} \log \frac{2}{\varepsilon}$. In exact arithmetic, PCG terminates in at most n steps, but in practice it converges in fewer steps when $\kappa(\tilde{A})$ is small. The ideal scenario occurs when $M = A$, which gives $\tilde{A} = I$ and convergence in just one step. Good preconditioners aim to approximate this effect at low initialisation and application cost.

Geometric insights. The CG method moves on the quadratic $\phi(u) = \frac{1}{2}\, u^\top A u - b^\top u$ minimising ϕ over an expanding Krylov subspace. The directions are $A-$conjugate, that is, $(d^{(i)})^\top A\, d^{(j)} = 0$ for $i \neq j$, and the residuals are orthogonal: $(r^{(i)})^\top r^{(j)} = 0$ for $i \neq j$.

The PCG algorithm operates on the transformed SPD system

$$\tilde{A} \;=\; M^{-1/2} A M^{-1/2}, \qquad \tilde{b} \;=\; M^{-1/2} b, \qquad y \;=\; M^{1/2} u,$$

minimising $\tilde{\phi}(y) = \frac{1}{2}\, y^{\top} \tilde{A}\, y - \tilde{b}^{\top} y$.
The method now searches for a solution in a preconditioned Krylov subspace and enforces the orthogonality of the residuals under the M^{-1}-inner product, that is, $(r^{(i)})^{\top} M^{-1} r^{(j)} = 0$ for $i \neq j$. PCG maintains $(d^{(i)})^{\top} A d^{(j)}$ for $i \neq j$.

Geometrically, the level sets of ϕ, which are ellipses, are changed by $M^{-1/2}$. If M makes $\tilde{A}$ closer to the identity, the ellipses become more circular, so line searches point more directly toward the minimiser for ϕ, and convergence accelerates.

Chapter 5

Experimental evaluation

5.1 Experimental setup

For the experimental part, we solve

$$u_{xx} + u_{yy} = 0, \quad (x, y) \in (0, 1) \times (0, 1),$$

given the boundary conditions:

$$\begin{cases} u(0, y) = u(1, y) = 0, & 0 \leq y \leq 1 \\ u(x, 0) = sin(\pi x), & 0 \leq x \leq 1 \\ u(x, 1) = 4sin(3\pi x), & 0 \leq x \leq 1 \end{cases}$$

The analytic solution to the above problem is

$$u(x, y) = \frac{1}{\sinh(\pi)} \sinh[\pi(1 - y)] \sin(\pi x) + \frac{4}{\sinh(3\pi)} \sinh(3\pi y) \sin(3\pi x).$$

This case is used to verify the correctness and, when reported, to compute the error norms.

Grid sequence and system sizes. Grid sizes, including boundaries, are swept as $N_{out} = \{10, 20, 30, \ldots, 200\}$, so the number of interior points per axis are $N = N_{out} - 2$. The size of the mesh is $h = 1/(N + 1)$, and the dimension of the system is $n = N^2$.

The grid used to solve the equation was discretised using the 5-point finite differences

stencil defined by Equation 2.20:

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0.$$

Solvers. Jacobi, Gauss-Seidel (GS), Successive Over-Relaxation (SOR), Conjugate Gradient (CG), and Preconditioned Conjugate Gradient (PCG) with Jacobi (diagonal), SSOR, and IC(0) preconditioners are compared. All runs start from $u^{(0)} = 0$. IC(0) is applied via triangular solves (no explicit inverses).

Stopping rules and caps. A run converges when the relative residual, $\rho_k = \frac{\|r_k\|_2}{\|b\|_2}$, drops below a tolerance, τ, reported with the results. Each solver has a method-specific maximum iteration cap; reaching the cap is logged as non-converged.

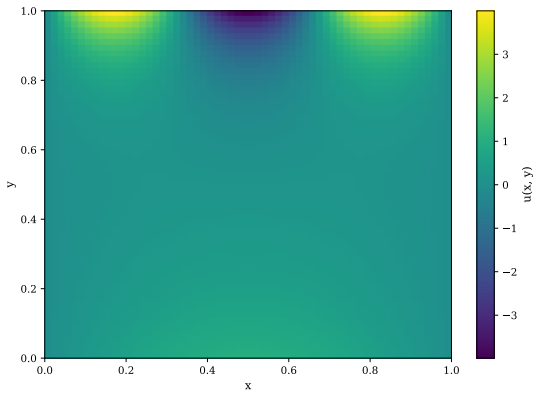


Figure 5.1: Heatmap of the analytical solution, $u(x, y)$.

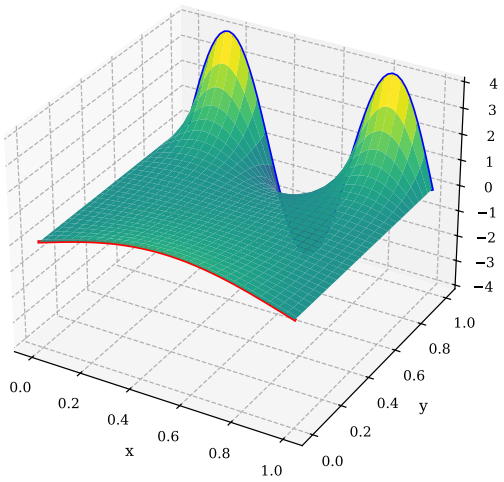


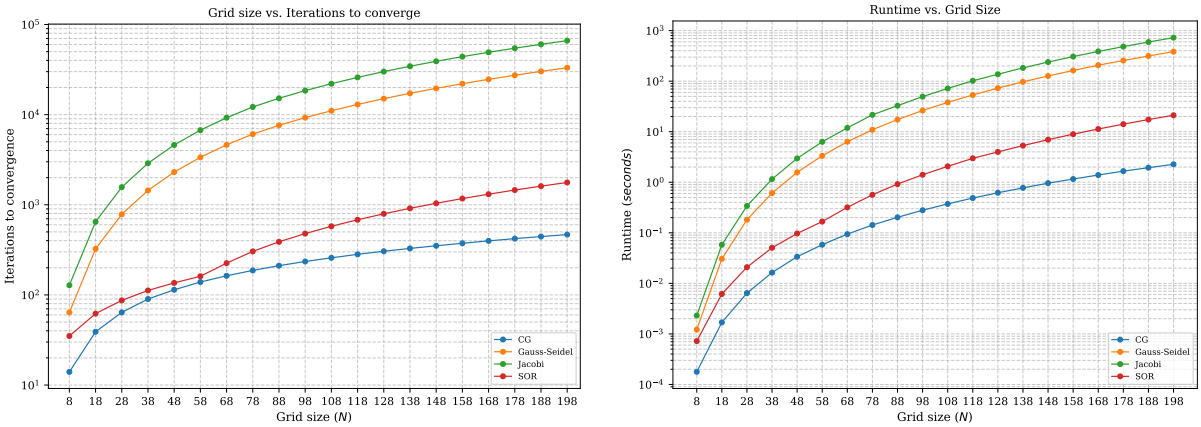
Figure 5.2: Surface plot of the analytical solution, $u(x, y)$. Top and bottom boundary conditions are coloured blue and red, respectively.

5.2 Solver performance evaluation

We compare Jacobi, Gauss-Seidel (GS), Successive Over-Relaxation (SOR), and Conjugate Gradient (CG). All runs start from $u^{(0)} = 0$ and terminate when $\|r_k\|_2/\|b\|_2 \leq \tau$. The discretisation and problem family are as specified in the Experimental Setup subsection.

Iteration growth vs. grid size. Figure 5.3a shows iterations with respect to the interior grid size, $N = \{8, 18, \ldots, 198\}$. The trends align with theory for the 2D Laplacian; For Jacobi and GS iteration counts grow like $\Theta(N^2)$ consistent with $\rho = 1 - \Theta(h^2)$. SOR reduces iteration growth significantly between linear and quadratic, consistent with the $1 - \Theta(h)$ factor for near-optimal ω. Lastly, in the case of CG the iterations grow almost linearly with N, reflecting $k = O(\sqrt{\kappa}) = O(N)$ for SPD Laplacians.

Runtime scaling. Figure 5.3b shows wall-clock time versus N. Since each iteration costs $O(\text{nnz}(A)) = \Theta(N^2)$, time $\approx \Theta(N^2 \times \text{iters})$. CG is the fastest of all methods on all sizes and scales close to $O(N^3)$. SOR is the second fastest, and even though the cost per iteration is still low, it performs more iterations than CG even for near-optimal ω. GS and Jacobi scale much worse, empirically near $O(N^4)$, becoming impractical for large N. At the largest grid size, $N = 198$, Table 5.1 reports the following runtimes (s): CG = 2.27, SOR = 21.18, GS = 383.71, Jacobi = 725.12.



(a) Iterations required to reach the tolerance as a function of the grid size, N (interior points per axis; system size $n = N^2$).

(b) Wall-clock time to convergence versus interior grid size N.

Figure 5.3: Iterations to converge (left) and wall-clock time (right) for all solvers across all interior grid dimensions.

Residual history at $N_{out} = 100$ $(N = 98)$. Figure 5.4 shows $\rho_k = \|r_k\|_2/\|b\|_2$ versus the number of iterations to convergence (log-log scale). CG exhibits the steepest decay with the characteristic small "saw-tooth" oscillations in $\|r_k\|_2$, while the $A-$norm of the error

is monotone. SOR follows with a smooth, nearly geometric decay, whereas GS and Jacobi are much slower to converge. The flat bottom near 10^{-7} reflects the stopping threshold. At this size, the measured iteration counts are: $CG = 235$, $SOR = 479$, $GS = 9260$, and $Jacobi = 18468$.

CG drops the fastest because it minimises the error in the $A-$norm over growing Krylov spaces. With $\kappa(A) \approx \Theta(N^2)$ for the discrete Laplacian, the classic bound, $\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k$, implies $k = O(\sqrt{\kappa}) = O(N)$ iterations to reach τ. SOR follows CG with a near-optimal relaxation parameter $\omega \approx 2/(1+\sin(\pi/(N+1)))$ and a convergence factor that behaves like $1 - \Theta(h)$. SOR removes low-frequency errors significantly faster than Jacobi and GS. Lastly, we have GS followed by Jacobi. GS improves over Jacobi because it uses freshly updated values from the same iterate, thus converging faster. For this stencil, one effectively gets $\rho(GS) \approx \rho(J)^2$. Nevertheless, both remain slow and require $\Theta(h^{-2}) = \Theta(N^2)$ iterations to converge.

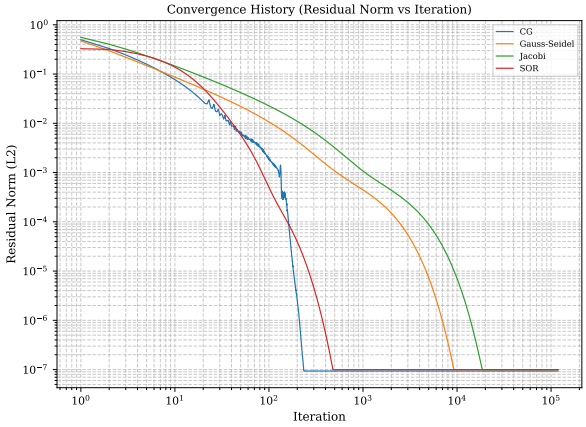


Figure 5.4: Relative residual $\rho_k = \|r_k\|_2/\|b\|_2$ vs iterations at $N = 98$.

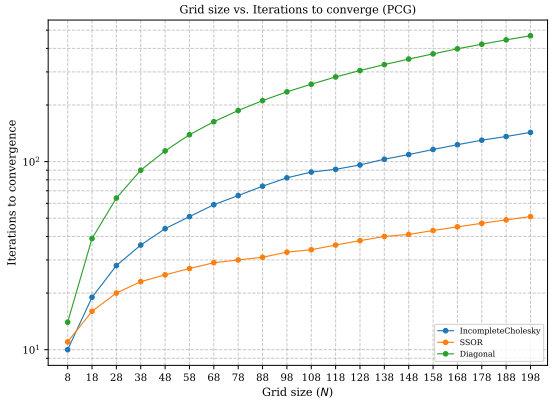
Table 5.1: Runtime (seconds) and iterations to convergence for each solver.

N	CG		Gauss–Seidel		Jacobi		SOR	
	Time [s]	Iters	Time [s]	Iters	Time [s]	Iters	Time [s]	Iters
8	0.000177959	14	0.00121517	64	0.00230983	128	0.000719833	35
18	0.00169129	39	0.0305138	324	0.0580044	648	0.00616004	62
28	0.0064165	64	0.181097	784	0.33995	1568	0.020887	87
38	0.0163401	90	0.610652	1444	1.15322	2888	0.0505257	112
48	0.0336446	114	1.57087	2304	2.96421	4608	0.0968123	136
58	0.0581803	139	3.33086	3364	6.31558	6728	0.167059	161
68	0.0941607	163	6.33088	4624	11.9047	9248	0.31874	225
78	0.142707	187	10.9254	6084	21.5861	12168	0.565452	304
88	0.202431	211	17.3948	7614	32.7111	15181	0.920513	388
98	0.279877	235	26.2664	9260	49.4331	18468	1.40855	479
108	0.373134	258	38.0045	11048	71.6863	22040	2.06226	577
118	0.486984	282	53.3277	12977	101.682	25893	2.97758	682
128	0.62029	305	72.7969	15043	137.16	30020	3.98379	794
138	0.774516	328	96.9209	17244	182.858	34417	5.3102	913
148	0.960303	351	126.571	19577	238.902	39078	6.95592	1039
158	1.15827	374	162.963	22041	306.409	44001	8.94578	1172
168	1.39062	398	207.408	24633	387.293	49181	11.2971	1311
178	1.66018	421	255.68	27352	482.928	54613	14.1125	1457
188	1.94244	444	315.11	30196	594.65	60296	17.4149	1609
198	2.2656	467	383.712	33163	725.124	66225	21.181	1768

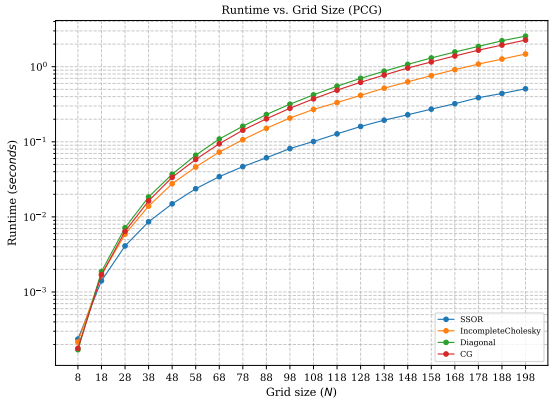
5.3 Preconditioners evaluation

Iterations to converge. Figure 5.5a shows iteration counts for all interior grid dimensions, $N = \{8, 18, \ldots, 198\}$. The iteration counts grow roughly linearly with N for all PCG variants consistent with the bound $k = O(\sqrt{\kappa(M^{-1}A)}\log(1/\tau))$, with different slopes. SSOR requires the fewest iterations across the grid sequence, IC(0) sits in the middle, while Jacobi (Diagonal) matches CG, confirming that diagonal scaling does not improve asymptotic behaviour.

Runtime. Figure 5.5b reports the wall-clock time for all system dimensions. Since a single iteration costs $\Theta(\text{nnz}(A)) = \Theta(N^2)$, the total time is $\Theta(N^2 \times \text{iters})$ in addition to any preconditioner overhead. We see that for small N all methods yield runtimes that are really close. As N increases, PCG-SSOR becomes the fastest of all, PCG-IC(0) comes second, and PCG-Jacobi follows the CG baseline, sometimes marginally due to the additional scaling the preconditioner performs.



(a) Iterations required to reach the tolerance as a function of the grid size, N (interior points per axis; system size $n = N^2$) for all preconditioners.



(b) Wall-clock time to convergence versus interior grid size N for all preconditioners.

Figure 5.5: Iterations to converge (left) and wall-clock time (right) for all preconditioners across all interior grid dimensions.

PCG convergence history at $N = 98$. Figure 5.6 plots the relative residual, $\rho_k = \|r_k\|_2/\|b\|_2$ against iteration. It is clear from the curves that PCG-SSOR decays almost geometrically and is steepest. With ω near its optimal value, SSOR damps the slow, low-frequency error components that dominate as $h \to 0$. PCG-IC(0) is evidently faster than CG, but not as steep as SSOR in the configuration used for the current experimental evaluation (zero-fill, natural ordering), which limits eigenvalue clustering. Lastly, PCG-Jacobi closely overlaps CG, as diagonal scaling normalises the diagonal but does not affect the asymptotic conditioning of the Laplacian matrix.

At this size ($N = 98$), Table 5.2 reports: CG = 235 iters, PCG-Jacobi = 235 iters, PCG-IC(0) = 82 iters, and PCG-SSOR = 33 iters, and wall-clock time $0.2799\,s$, $0.3172\,s$, $0.2072\,s$, and $0.0812\,s$, respectively.

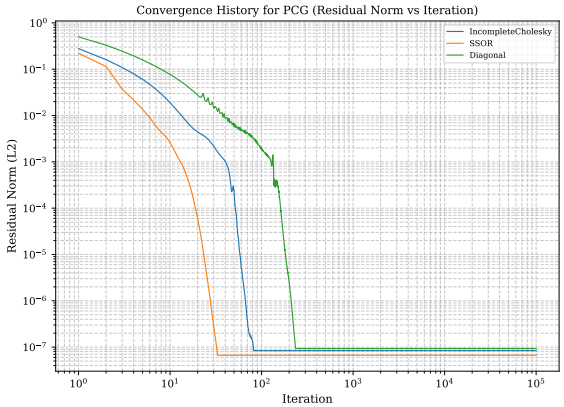


Figure 5.6: Relative residual $\rho_k = \|r_k\|_2/\|b\|_2$ vs iterations at $N = 98$.

Table 5.2: Preconditioned CG on the $N^2 \times N^2$ Laplace system: runtime (seconds) and iteration counts.

N	IC(0)		SSOR		Jacobi (Diagonal)		CG (no prec.)	
	Time [s]	Iters	Time [s]	Iters	Time [s]	Iters	Time [s]	Iters
8	0.0002155	10	0.000236042	11	0.000170167	14	0.000177959	14
18	0.00169937	19	0.00140813	16	0.000187083	39	0.00169129	39
28	0.00587671	28	0.0041035	20	0.00716217	64	0.0064165	64
38	0.0139075	36	0.00860375	23	0.0183873	90	0.0163401	90
48	0.0276257	44	0.0149268	25	0.0370299	114	0.0336446	114
58	0.0462384	51	0.0237212	27	0.0663408	139	0.0581803	139
68	0.0729098	59	0.0343322	29	0.108867	163	0.0941607	163
78	0.10657	66	0.0467103	30	0.161362	187	0.142707	187
88	0.151092	74	0.0612588	31	0.230465	211	0.202431	211
98	0.207162	82	0.0812248	33	0.317184	235	0.279877	235
108	0.269849	88	0.100979	34	0.421586	258	0.373134	258
118	0.333021	91	0.127538	36	0.555006	282	0.486984	282
128	0.415464	96	0.159979	38	0.700240	305	0.620290	305
138	0.517511	103	0.194146	40	0.871368	328	0.774516	328
148	0.628583	109	0.228946	41	1.074490	351	0.960303	351
158	0.760941	116	0.272377	43	1.308120	374	1.158270	374
168	0.911905	123	0.321688	45	1.570770	398	1.390620	398
178	1.083930	130	0.386357	47	1.863220	421	1.660180	421
188	1.264150	136	0.440259	49	2.215880	444	1.942440	444
198	1.476350	143	0.507789	51	2.552160	467	2.265600	467

Across all system dimensions, SSOR accomplishes the fewest number of iterations and the lowest runtime. Although IC(0) reduces the iteration count and time compared to CG,

it is not as aggressive as SSOR. The Jacobi (Diagonal) preconditioner leaves the number of iterations identical to CG for all N and changes the runtime due to the tiny overhead of the scaling at each step.

5.3.1 Spectral analysis

When analysing the spectrum, we examine how preconditioning changes the properties of the system matrix, by inspecting the spectra of the preconditioned operator $P = M^{-1/2}AM^{-1/2}$. The CG iteration count depends on the spectrum of P via

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq 2\left(\frac{\sqrt{\kappa(P)}-1}{\sqrt{\kappa(P)}+1}\right)^k, \quad \kappa(P) = \frac{\lambda_{\max}(P)}{\lambda_{\min}(P)}.$$

Reducing $\kappa(P)$ and clustering the eigenvalues of P away from zero reduce the number of Krylov steps.

Baseline conditioning. For the 2D five-point Laplacian on an interior grid $N \times N$ with $h = 1/(N+1)$, the eigenpairs are

$$\lambda_{p,q}(A) = \frac{4}{h^2}\left[sin^2\left(\frac{p\pi}{2(N+1)}\right) + sin^2\left(\frac{q\pi}{2(N+1)}\right)\right],$$

for $p, q = 1, \ldots, N$. Hence $\kappa(A) = \Theta(N^2)$. Figure 5.7 confirms that all calculated $\kappa(A)$ lie on the asymptote $\frac{4}{\pi^2}(N+1)^2$ for all N.

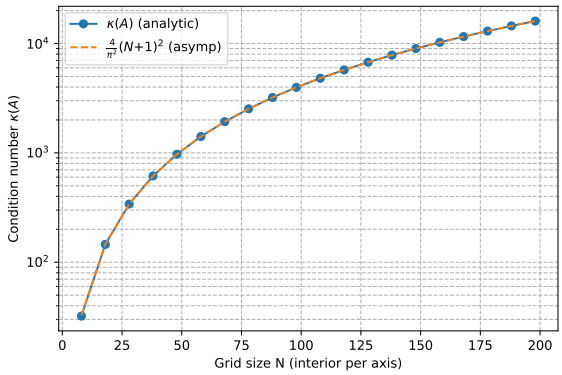
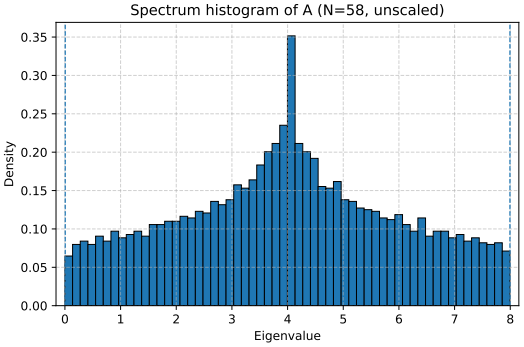


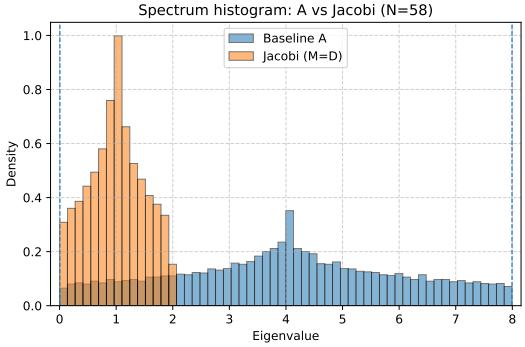
Figure 5.7: Condition number, $\kappa(A)$, of the Laplacian grows like $\kappa(A) \approx \frac{4}{\pi^2}(N+1)^2$, matching $\Theta(N^2)$.

Eigenvalue distributions at $N = 58$. Figure 5.8 shows the spectral densities for $N = 58$. The spectrum of the initial matrix A (Figure 5.8a) is a broad distribution with a substantial mass near the origin, which implies poor preconditioning. Figure 5.8b shows

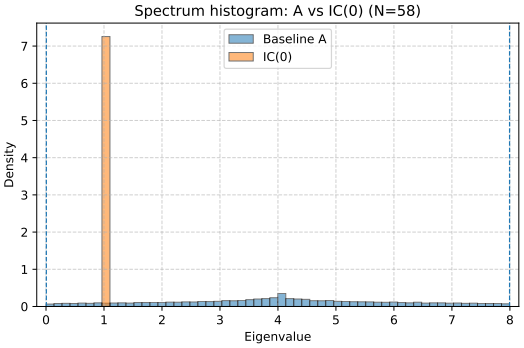
the spectrum of the Jacobi preconditioned operator: $P_J = D^{-1/2}AD^{-1/2}$. Jacobi preconditioning only normalises the axes and does not change the eigenvalue behaviour. Essentially, diagonal preconditioning results in a 'shrinked' distribution that is now centred around 1 and not 4 as before. Asymptotically, $\kappa(P_J) \approx \kappa(A)$. The application of IC(0) reports a clear tightening relative to A; many eigenvalues cluster in a compact band away from zero, although very few near-zero values remain with zero fill and natural ordering. Lastly, Figure 5.8d displays the spectrum of the SSOR preconditioned operator. In contrast to diagonal scaling, SSOR lifts the low end of the spectrum and, more importantly, gathers the bulk into a tighter cluster. As a result, the preconditioned matrix has a smaller condition number, as shown in Figure 5.9, and a more compact eigenvalue distribution. Although there are a few SSOR eigenvalues closer to zero than IC(0), the overall cluster is narrower and $\lambda_{\max}$ is smaller for SSOR.



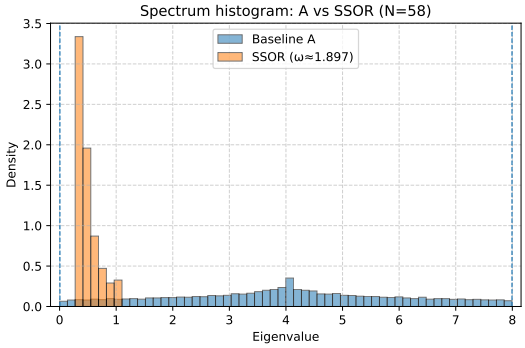
(a) A (unscaled): broad distribution with substantial mass near the origin.



(b) Jacobi preconditioning: diagonal scaling, asymptotically $\kappa(P_J) = \kappa(A)$.



(c) IC(0): eigenvalues cluster in a compact band separated from zero.



(d) SSOR ($\omega = 1.897$): tight clustering, yields the smallest κ and fastest PCG.

Figure 5.8: Spectrum histograms at $N = 58$.
All spectra are shown in normalised units $\hat{\lambda} = h^2\lambda$, so values lie in $(0, 8]$.

Sorted spectra for $N = 58$. Reading the eigenvalues in increasing order highlights clustering and spread. The baseline Laplacian, A, displays a long near-zero tail and a steep rise close to 8, a wide interval, and poor conditioning. Jacobi ($D^{-1/2}AD^{-1/2}$) is

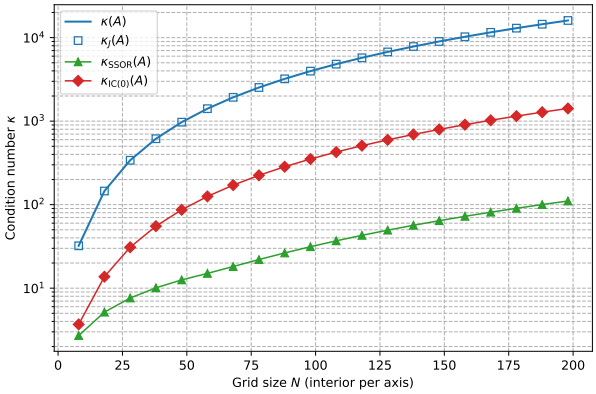
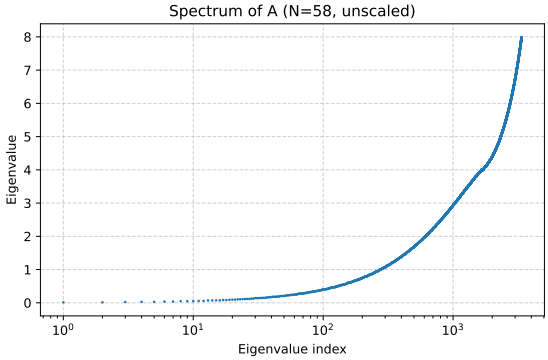
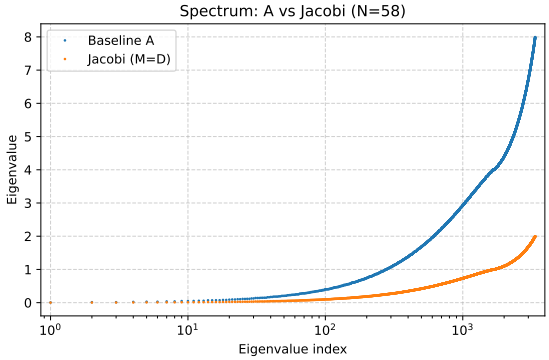


Figure 5.9: Condition number for all system dimensions of the preconditioned operators.

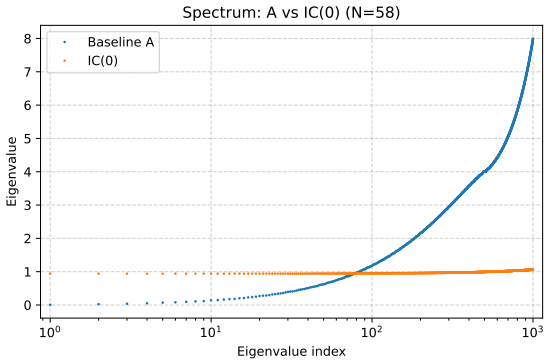
essentially rescaling the spectrum: the top end is capped ≈ 2 with the left tail around zero still present. IC(0) compresses most eigenvalues in a short, nearly flat band around 1 with few to no outliers, providing a clear constant-factor reduction in κ (see Figure 5.9) and steps. SSOR ($\omega \approx 1.897$) produces the most compact spectrum. The bulk forms a very tight cluster with a trimmed upper tail and a lifted left end relative to A.



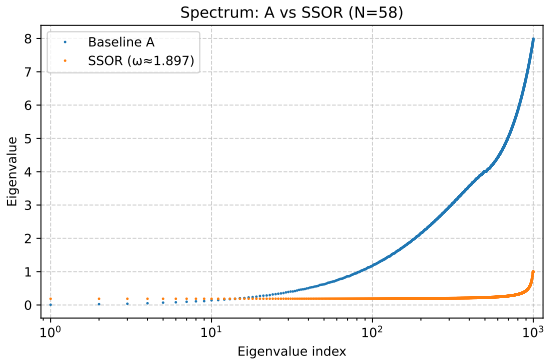
(a) A (unscaled): wide spread from small eigenvalues up to 8; small flat plateau at 4.



(b) A vs Jacobi: orange curve is a rescaled version of A, with the plateau near 1.



(c) A vs IC(0): flattens the spectrum; most eigenvalues lie in a narrow band near 1.



(d) A vs SSOR ($\omega = 1.897$): tightest, most compact band; yields the smallest κ.

Figure 5.10: Sorted spectra at $N = 58$ for all preconditioned operators (orange) vs A (unscaled).

Chapter 6

Conclusions and Recommendations for future work

6.1 Conclusions

We studied the 2D Laplace equation defined in a unit square, $(x, y) \in [0, 1] \times [0, 1]$ with Dirichlet boundary conditions, using the 5-point stencil of the finite differences method to form, and eventually solved the resulting SPD systems with Jacobi, Gauss-Seidel (GS), SOR, and PCG (Jacobi, SSOR, IC(0)).
The results show that

- **CG vs. stationary methods:** CG consistently outperforms Jacobi and GS. Iteration growth matches theory; CG grows $\sim O(N)$, while Jacobi and GS grow $\sim O(N^2)$. SOR improves substantially over Jacobi and GS, often reaching τ in hundreds rather than tens of thousands of steps.

- **PCG:** Among the preconditioners tested, the performance-based ordering is **PCG-SSOR** $>$ **PCG-IC(0)** $>>$ **PCG-Jacobi** $\approx$ **CG**. At $N = 198$, the iteration counts are CG = 467, PCG-Jacobi = 476, PCG-IC(0) = 143, and PCG-SSOR = 51. The corresponding runtimes (s) are 2.27, 2.55, 1.48, and 0.51.

- **Spectral view:** The discrete Laplacian has $\kappa(A) \approx \frac{4}{\pi^2}(N + 1)^2$. Diagonal scaling (Jacobi) normalises but does not change this asymptotic conditioning; therefore, the iteration count remains the same for all N. SSOR and IC(0) both narrow the spectrum and lift the left tail. SSOR produced the tightest cluster of eigenvalues and the smallest condition number, and consequently the fewest CG steps.

- **Cost per iteration:** SSOR is cheaper to apply than IC(0) even where IC(0) approaches SSOR in step counts. SSOR requires two triangular solves, while IC(0)

requires an additional one-time factorisation.

6.2 Limitations and next steps

Limitations. The present study is confined to the 2D Laplace equation with Dirichlet boundary conditions on a unit square, a uniform grid, and the classical five-point stencil. Variable coefficients, anisotropy, non-rectangular domains, and Neumann/Robin boundary conditions were not considered. The preconditioners choices were deliberately minimal, Jacobi, SSOR with a fixed ω, and IC(0) with zero fill. Convergence was assessed using only a residual criterion, with limited reporting of error norms, and the spectral analysis focused on the condition number, κ, and visual clustering.

Next steps. A natural extension of this assignment is to strengthen preconditioning. We could introduce IC with drop tolerance or ILU(κ)/ILUT(τ) combined with RCM (Reverse Cuthill–McKee) or nested dissection orderings, test SPAI (Sparse Approximate Inverse)/FSAI (Factorised Sparse Approximate Inverse) or diagonally shifted IC for robustness, and add geometric or algebraic multigrid as a preconditioner in PCG to target mesh-independent convergence. Generalising the PDE suite to Poisson with sources, variable coefficient, and anisotropic cases, and to Neumann/Robin conditions as well as to non-uniform meshes, would test robustness. It would also help to report setup versus solve time separately and include accuracy metrics, such as A-norm and L_∞, alongside residuals while varying tolerance.

Bibliography

[1] W. A. Strauss, *Partial Differential Equations: An Introduction*, 2nd ed. John Wiley & Sons, 2008.

[2] L. C. Evans, *Partial Differential Equations*, 2nd ed., ser. Graduate Studies in Mathematics. American Mathematical Society, 2010, vol. 19.

[3] M. E. Taylor, *Partial Differential Equations I: Basic Theory*, 2nd ed., ser. Applied Mathematical Sciences. Springer, 2011, vol. 115.

[4] R. Haberman, *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*, 5th ed. Pearson, 2013.

[5] N. H. Asmar, *Partial Differential Equations with Fourier Series and Boundary Value Problems*, 3rd ed. Prentice Hall, 2005.

[6] D. Gilbarg and N. S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, reprint of the 1998 2nd ed. ed., ser. Classics in Mathematics. Springer, 2001.

[7] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, 2nd ed. Society for Industrial and Applied Mathematics, 2004.

[8] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. Society for Industrial and Applied Mathematics, 2007.

[9] R. Courant, K. O. Friedrichs, and H. Lewy, "On the partial difference equations of mathematical physics," *IBM Journal of Research and Development*, vol. 11, no. 2, pp. 215–234, 1967, english translation of [28].

[10] L. N. Trefethen and I. Bau, David, *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[11] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge University Press, 2012.

[12] B. Bollobás, *Modern Graph Theory*, ser. Graduate Texts in Mathematics. Springer, 1998, vol. 184.

[13] N. Biggs, *Algebraic Graph Theory*, 2nd ed. Cambridge University Press, 1993.

[14] F. R. K. Chung, *Spectral Graph Theory*, ser. CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997, vol. 92.

[15] R. Merris, "Laplacian matrices of graphs: A survey," *Linear Algebra and its Applications*, vol. 197–198, pp. 143–176, 1994.

[16] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, ser. Classics in Applied Mathematics. SIAM, 1994, vol. 9.

[17] R. S. Varga, *Matrix Iterative Analysis*, 2nd ed. Springer, 2009.

[18] C. F. V. Loan, "The ubiquitous kronecker product," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1–2, pp. 85–100, 2000.

[19] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 4th ed. Johns Hopkins University Press, 2013.

[20] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, 2nd ed. Springer, 2016.

[21] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, 2003.

[22] D. M. Young, "Iterative methods for solving partial difference equations of elliptic type," *Transactions of the American Mathematical Society*, vol. 76, no. 1, pp. 92–111, 1954.

[23] A. Greenbaum, *Iterative Methods for Solving Linear Systems*. SIAM, 1997.

[24] J. Liesen and P. Strakoš, *Krylov Subspace Methods: Principles and Analysis*. Oxford University Press, 2013.

[25] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436, 1952.

[26] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Carnegie Mellon University, Tech. Rep., 1994, updated versions available online.

[27] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[28] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen der mathematischen physik," *Mathematische Annalen*, vol. 100, pp. 32–74, 1928.