Chapter 1

Scope and aims

This is a book about how to efficiently solve elliptic partial differential equations (PDEs) numerically. What makes these equations challenging is that they model interactions that are global rather than local, so that every point in a computational domain interacts with every other point. If you model gravitational interactions in a system of stars, then each star will feel the pull of every other star in the system; if you place a weight on a bridge, then the entire bridge will deform in response; and so on. Interactions like these do decay in magnitude over distances, but so slowly that if one were to set them to zero at some “safe” distance, then this would often distort the mathematical model to the point where the answer would not correctly represent the physics of the problem. However, while the magnitudes of these fields decay only slowly, the amount of information that they carry decays enormously fast. To illustrate this idea, suppose that you wanted to computationally determine the gravitational field caused by the Moon. An accurate mathematical model of this problem would be to formulate a Poisson equation $-\Delta u = g$, where the source term $g$ encodes the mass density of the Moon and $u$ is the resulting gravitational field. Now, if you are only interested in the gravitational forces evaluated on Earth, then the solution to the Poisson equation would be almost completely indistinguishable from the gravitational field caused by a single point charge placed at the center of gravity of the Moon. In effect, the field can accurately be encoded by four scalar numbers (three to specify the center of mass and one for the total mass). A central theme of this book is how the observation that only small amounts of information survive across long distances can be used to build solvers for elliptic PDEs that are fast, robust, and highly accurate.

1.1 • Direct and iterative solvers

Let us start by considering an archetypal elliptic PDE, the Poisson equation,

$$\begin{cases}
-\Delta u(x) = g(x) & \text{for } x \in \Omega, \\
u(x) = f(x) & \text{for } x \in \Gamma,
\end{cases} \quad (1.1)$$

involving a body load $g$, a boundary data function $f$, a domain $\Omega \subseteq \mathbb{R}^d$ with boundary $\Gamma$, and the Laplace operator

$$\Delta u = \sum_{j=1}^d \frac{\partial^2 u}{\partial x_j^2}.$$
Standard practice for numerically solving an equation such as (1.1) is to use a technique such as the finite element or finite difference method to form a linear system

$$Au = b,$$  

(1.2)

where $b$ encodes the given data functions $g$ and $f$ and $u$ represents an approximation to the solution $u$ of (1.1). The matrix $A$ can be viewed as a discrete representation of the differential operator. It is typically a sparse matrix, which is to be expected since the differential operator is local. Since $A$ is sparse, it is inexpensive to apply it to a vector, which makes it very natural to solve (1.2) using an iterative solver that constructs a sequence of successively more accurate solutions by applying $A$ to a sequence of vectors. Solvers of this type form the backbone of many codes in scientific computing and have reaped great success over the past few decades. We will at times discuss solvers of this type throughout this book, but they do not play a major role in our treatment since they are already well covered in the literature. Instead, our focus will be on methods based on the observation that the solution operator to (1.1) takes a particularly nice mathematical form. It consists of two integral operators that map the two given data functions $g$ and $f$ to the solution

$$u(x) = \int_{\Omega} G(x, y)g(y)dy + \int_{\Gamma} F(x, y)f(y)ds(y).$$  

(1.3)

These integral operators are stable and bounded, and are often regularizing in the sense that the function $u$ is smoother than the input functions $f$ and $g$. For simple geometries, we can sometimes write down exact analytic formulas for the kernel functions $F$ and $G$, and then the task of solving (1.1) becomes the task of applying the integral operators in (1.3). But even in such a simple case, an obstacle arises: the integral operators are global objects in which every point of the domain communicates with every other point. From a numerical point of view, the result is that if we discretize the domain using $N$ degrees of freedom, then the discrete equivalent of applying an integral operator is a matrix-vector product involving an $N \times N$ coefficient matrix that is typically \textit{dense}. If we use straightforward techniques for this task, the cost is $O(N^2)$. A central theme of this book is to describe techniques for reducing this cost to linear $O(N)$ complexity. We will go beyond the task of applying analytically known integral operators, however, and also describe techniques for how to numerically build approximations to the solution operators in (1.3) in cases involving complex geometries and a range of different elliptic operators, including ones that have variable coefficients.

### 1.2 Rank-structured matrices and the concept of “data sparsity”

The property that enables us to develop fast algorithms for the dense matrices associated with integral operators such as those in (1.3) is that their off-diagonal blocks have low numerical rank. To be precise, it is often possible to tessellate the matrix into $O(N)$ blocks in such a way that the rank of each block does not depend on $N$, or depends only very weakly on $N$. Figure 5.1 illustrates a representative tessellation pattern. For a matrix that is “rank structured” in this sense, it is possible to devise very efficient algorithms for performing a matrix-vector multiplication, and sometimes for other algebraic operations as well, such as inverting or factorizing the matrix. We will explore different ways to tessellate the matrix and different ways to represent the low-rank subblocks. In the literature, the taxonomy of rank-structured matrices includes terms such as $\mathcal{H}$- and $\mathcal{H}^2$-matrices, hierarchically block separable matrices, hierarchically off-diagonal low-rank matrices, and many more.
We will explore how they relate to one another and determine which are best suited to which environments.

Some of the matrices that we work with have more complicated internal structures, such as a mixture of sparsity in the traditional sense, and the rank-structured property described in the previous paragraph. An example would be the L and U factors that arise when computing an LU factorization of a sparse matrix obtained from the discretization of an elliptic PDE; cf. Chapter 21. There are other formats that are generalizations of the fast Fourier transform (FFT) that also do not fall exactly into what we define as rank-structured matrices. To define this broader class of structured matrices, let us paraphrase a common informal definition of a sparse matrix (often attributed to Wilkinson [62, 39]) and say that a matrix is *data sparse* if it has enough internal structure that it pays to take advantage of it.

**Remark 1.1 (quantifying the amount of information loss).** *A recurring theme in this book is estimating how much information is communicated between different regions of a computational domain. From a linear algebraic point of view, this question is answered by determining how quickly the singular values of the corresponding off-diagonal blocks in the dense matrices decay. (The faster they decay, the lower the numerical rank.) The reader may find it of interest to peek ahead and look at plots of the singular values of some representative matrices in Figures 9.3, 14.7, 21.2, and 25.4. These matrices come from different contexts, but they all show the very rapid exponential decay that enables all the linear complexity methods described.*

### 1.3 Why work with global operators?

In situations where an explicit solution operator is known analytically, the case for working with fast algorithms for global operators is straightforward. Application of the solution operator is simply a far more mathematically benign operation than approximating an unbounded operator and then solving a necessarily ill-conditioned linear system. In particular, it becomes much easier to implement high-order accurate methods, since the only discretization that is required is a quadrature. Many of these advantages carry over to the larger class of problems where we do not have an explicit solution operator available, but where we can formulate the physical problem that we seek to solve as a well-conditioned integral equation, since such an equation can often be solved through a rapidly converging iteration.

While techniques based on combining integral equation formulations with iterative solvers and fast algorithms for applying an integral operator with a known kernel have been very successful, there remain important situations where even the best iterative methods that we know of converge slowly or not at all, and standard preconditioners do not improve the situation. Acoustic and electromagnetic scattering problems with oscillatory solutions are the classical examples here. For problems such as these, the ability to *invert* data-sparse matrices becomes invaluable, as such methods enable us to solve problems that are intractable to traditional methodologies. We refer to these new solvers, which seek to directly build an approximation to solution operators such as those in (1.3), as *fast direct solvers.*

The new fast direct solvers offer compelling advantages even in many situations for which iterative methods converge reasonably rapidly. One reason is because once a solution operator has been built, it is typically extremely fast to apply it to a vector. This means that in situations where one needs to solve a sequence of equations involving the same differential operator, we are able to greatly accelerate the computational simulations. A closely related situation is when we solve a sequence of equations where each problem...
that we seek to solve is only slightly different from the previous one, as happens in, e.g.,
optimal design or many algorithms for solving inverse scattering problems. Fast direct
solvers have also proved particularly suitable for parallelization.

Fast direct solvers have important limitations as well. They tend to require substantially
more memory per degree of freedom than a standard iterative solver for a sparse linear
system. In situations where iterative methods for sparse linear systems converge rapidly,
they are highly competitive in terms of both speed and memory requirements, and the
effort required to implement a fast direct solver generally only pays off in situations when
the cost of building the solution operator can be amortized over many solves.

1.4 • Scope

My objective in writing this book has been to convey the fundamental ideas and techniques
that underlie fast algorithms for global operators, while keeping the text as readable as
possible. I have often refrained from giving the most rigorous or general mathematical
statements in favor of discussing examples that illustrate the essential concepts. Technical
challenges that arise in realistic computations that do not have a broader bearing on the
subject as a whole are mentioned, but only to alert the reader to their existence and to
provide pointers to papers that discuss them in full detail.

In order to keep the narrative focused, for the most part I describe how the methods
apply to the Laplace and Helmholtz equations only. All techniques can be applied far more
broadly and can be used to handle a large class of elliptic PDEs that includes the equations
of linear elasticity, the Stokes equation, the time-harmonic Maxwell equations, and many
more. I have also limited the focus almost entirely to the two-dimensional (2D) case. How-
ever, I only discuss techniques that have three-dimensional (3D) analogues that are compu-
tationally efficient, and that can be derived from the 2D methods through straightforward
modifications. Invariably, there are significant complications in extending a method from
two to three dimensions, and these are described in the text, with references provided for
the reader seeking to learn more.

1.5 • Target audience

This book is written to be readily accessible to readers from diverse backgrounds. The es-
sential prior knowledge required is a familiarity with topics from linear algebra, such as the
singular value decomposition, Schur complements, the QR factorization, etc. I also assume
that the reader is familiar with the basic properties of elliptic PDEs, such as the Laplace
and Helmholtz equations. What is primarily required is a knowledge of topics from poten-
tial theory, such as fundamental solutions, Fourier analysis, and the Green’s identities. In
contrast, the book is deliberately written to not require detailed knowledge about advanced
topics from PDE theory. (Sobolev spaces make no appearance, for instance.)

My intention is for the book to be accessible to a graduate student in mathematics or ap-
plied mathematics, as well as to advanced undergraduate students who have completed the
coursework mentioned in the previous paragraph. The text is also intended to be accessible
to practitioners in scientific computing with a background in physics or engineering.

1.6 • Organization

The book begins with two introductory chapters, and the remaining chapters are organized
into five principal parts:
1.7. Ways to read the book

1. Linear algebra (Chapters 3–5). Chapter 3 is a survey of concepts from linear algebra. Many readers will be familiar with much of this material, except perhaps the interpolative decompositions described in section 3.5. Chapter 4 surveys recently developed randomized techniques for computing factorizations of matrices, and Chapter 5 is an introduction to rank structured hierarchical matrices.

2. The fast multipole method (Chapters 6–9). These chapters describe a classical method for rapidly evaluating all pairwise interactions between \( N \) particles in \( O(N) \) operations. This is a key building block for applying many of the standard integral operators of potential theory. This technique is also the source of many of the concepts that are used in fast direct solvers.

3. Integral equation methods (Chapters 10–12). It is in many environments advantageous to model physical phenomena such as electromagnetic scattering or deformations of solid bodies using integral equations rather than PDEs. These chapters show how this can be done through some simple examples and describe how the resulting integral equations can be discretized to high accuracy.

4. Fast direct solvers for integral equations (Chapters 13–18). When an integral equation is discretized, it often results in a linear system \( A\sigma = f \), where \( A \) is a dense matrix that is “rank structured” or “data sparse” in the sense described in section 1.2. These chapters describe how to exploit this structure in \( A \) not only to apply \( A \) to vectors but also to solve the linear system directly.

5. Fast direct solvers for sparse matrices (Chapters 19–26). A linear system \( Au = f \) resulting from the discretization of an elliptic PDE via a finite element or finite difference method has a sparse coefficient matrix. It turns out that there are ways to order its columns and rows that make it particularly efficient to compute its LU factorization, typically using \( O(N^2) \) operations for problems in 3D. By exploiting not only the sparsity but also the physical properties of the underlying equations, one can further reduce the asymptotic complexity to \( O(N^{4/3}) \) or even better. These chapters describe both the classical techniques based on sparsity and the more recent methods that rely on exploiting “data sparsity” in certain dense matrices that arise.

About half of the book covers well-established material that has by now been thoroughly integrated into the standard toolkit of scientific computing. This includes most of the discussion of linear algebraic techniques, the material on the fast multipole method (FMM) and integral equations, and the first part of the discussion about solvers for sparse systems (Chapters 1–12, 19, and 20). The remaining chapters on accelerated direct solvers concern material that at the time of writing (2019) remains an active area of research.

In writing this book, I have made every effort to make the different parts as independent of one another as possible. My objective has been to make the book accessible to readers who may be interested in certain aspects of the material but not necessarily others. The drawback of this approach is that I have at times found it necessary to repeat myself. For instance, hierarchical trees play a fundamental role in all parts, and are introduced several times throughout the book. When the quest for modularity comes into conflict with the desire for elegance and economy of writing, I have sacrificed the latter.

1.7 Ways to read the book

This book would serve well in a graduate-level course on numerical methods for elliptic PDEs. Such a course could be designed to be accessible to advanced undergraduate stu-
udents who have taken basic courses in matrix computations, PDEs and potential theory, and numerical analysis. The core material for the course would consist of Chapters 1–12, as well as Chapters 19 and 20. Including Chapters 13, 14, 21, and 22 would introduce the key ideas of fast direct solvers, without getting overly mired in technical details. For a course of this nature, it would be very natural to devote substantial time to iterative solvers as well (multigrid, preconditioned Krylov methods, etc.) and to complement the material in this book with some of the excellent texts that exist on these topics.

The material on the fast multipole method (Chapters 6–9), and on integral equation methods (Chapters 10–12) can be read on a completely standalone basis, and could be incorporated as modules into a variety of graduate-level or advanced undergraduate-level courses.

For a practitioner or a student seeking just an introduction to fast direct solvers, it is possible to skip much of the material on fundamentals early in the book. I would recommend any reader to at least quickly review the material in Chapters 3–5 on matrix computations, as these techniques are foundational to much of what follows later. The reader interested in direct solvers for sparse systems can then skip ahead and read Chapters 19–22 for an introduction to the key concepts, and tips on where to go next for additional details. A reader interested in direct solvers for the dense systems arising from discretization of boundary integral equations could in principle skip directly ahead to Chapters 13–18 if he or she is already familiar with FMMs and integral equation-based techniques. I would still recommend reading Chapter 9 since it introduces some less well known ideas that are important in the development of direct solvers, and would generally suggest that even experienced readers glance over the material in Chapters 6–12 to familiarize themselves with the notation used.