Integral Equations and Fast Algorithms

with applications

Courant Institute
New York University

Michael O'Neil[1]

Fall 2017

[1]Department of Mathematics, Courant Institute, New York University, New York, NY. E-mail: oneil@cims.nyu.edu.

Contents

1 Introduction

Before describing the current lay of the land in modern computational science (at least with regard to PDEs and classical physics-driven problems), it is worth pointing out what is necessary to understand the following lecture notes. We will assume that the reader has a strong grasp of linear algebra and multi-variable calculus, a working knowledge of the basic theorems and ideas in complex analysis, and at least some exposure to ordinary and partial differential equations. Of course, proficiency in computer programming (preferably `C`, `Fortran`, or `Matlab`) will prove invaluable if any of the algorithms in these notes are to be implemented. Remember, the level of work required to implement a fast algorithm *is not* an excuse to avoid implementing it!

Keep in mind that these notes are far from finished! Please send suggestions and errors to `oneil@cims.nyu.edu`.

1.1 Motivation: electrostatics

As motivation for the use of integral equations and fast algorithms over the usual PDE formulations (and solvers) of many physical systems, we first examine a special case of Maxwell's equations: electrostatics. In an isotropic medium with constant magnetic permeability μ and electric permittivity ϵ, the fully time-dependent Maxwell equations governing the propagation of electric and magnetic fields, $\mathcal{E}$ and $\mathcal{H}$, are given by:

$$\begin{aligned} \nabla \times \mathcal{E} &= -\mu \frac{\partial \mathcal{H}}{\partial t}, & \nabla \times \mathcal{H} &= \varepsilon \frac{\partial \mathcal{E}}{\partial t} + \mathcal{J}, \\ \nabla \cdot \mathcal{E} &= \frac{\rho}{\varepsilon}, & \nabla \cdot \mathcal{H} &= 0, \end{aligned} \tag{1.1}$$

where $\mathcal{J}$ and ρ are the electric current and charge, respectively. For various important physical devices, it suffices to study the above equations in the time harmonic case. Assuming an implicit time dependence of $e^{-i\omega t}$ on $\mathcal{E}$, $\mathcal{H}$, $\mathcal{J}$, and ρ, we have what are known as the time-harmonic Maxwell's equations:

$$\begin{aligned} \nabla \times \boldsymbol{E} &= i\omega\mu \boldsymbol{H}, & \nabla \times \boldsymbol{H} &= -i\omega\varepsilon \boldsymbol{E} + \boldsymbol{J}, \\ \nabla \cdot \boldsymbol{E} &= \frac{\rho}{\varepsilon}, & \nabla \cdot \boldsymbol{H} &= 0, \end{aligned} \tag{1.2}$$

where script quantities have been replaced by bold quantities to show only spatial dependence. Furthermore, letting $\omega \to 0$ we arrive at the equations of electro- and magnetostatics:

$$\begin{aligned} \nabla \times \boldsymbol{E} &= \boldsymbol{0}, & \nabla \times \boldsymbol{H} &= \boldsymbol{J}, \\ \nabla \cdot \boldsymbol{E} &= \frac{\rho}{\varepsilon}, & \nabla \cdot \boldsymbol{H} &= 0. \end{aligned} \tag{1.3}$$

In this regime, we see that the electric and magnetic fields have completely decoupled. Examining the equations for $\boldsymbol{E}$ for a moment, the first implies that the electric field must be a gradient. We then write:

$$\boldsymbol{E} = -\nabla\varphi, \tag{1.4}$$

where φ will be referred to as the electric potential. Inserting this representation into the divergence equation, we have a Poisson problem for the potential function:

$$-\Delta\varphi = \frac{\rho}{\varepsilon}. \tag{1.5}$$

That is to say, for a known charge distribution ρ, the calculation of the electric field requires the solution to the above Poisson problem.

Standard PDE discretizations of the Poisson problem (finite difference, finite element, etc.) result in sparse linear systems which can be solved relatively efficiently using sparse linear algebra methods (nested dissection, etc.) or iterative methods (e.g. conjugate gradients). Often solvers which scale near linearly in the number of unknowns can be derived. However, if the domain of interest is unbounded (i.e. all of $\mathbb{R}^3$) then a prohibitive amount of discretization is required, or such methods as Perfectly Matched Layers (PMLs) must be used. Lastly, derivatives of the solution φ must be calculated numerically, which can lead to severe loss of precision if a very fine discretization mesh was used.

On the other hand, using a little bit of physics and mathematics, we can directly write down the solution to equation (1.5) using Coulomb's Law:

$$\varphi(\boldsymbol{x}) = \int \frac{\rho(\boldsymbol{x}')}{4\pi\epsilon|\boldsymbol{x}-\boldsymbol{x}'|}\, dV(\boldsymbol{x}'). \tag{1.6}$$

Mathematically, this corresponds to the use of the Green's function for the Laplace operator:

$$\Delta g = \delta, \tag{1.7}$$

with $g(\boldsymbol{x},\boldsymbol{x}') = -1/4\pi|\boldsymbol{x}-\boldsymbol{x}'|$. Note that by writing down the solution φ in integral form, we have eliminated the solve step! Only an evaluation of the integral in (1.6) is required. In avoiding having to actually solve a linear system, however, we have introduced various numerical analysis and algorithmic issued that need to be overcome. First, the singular kernel in this expression needs to be integrated accurately. This requires the development of specialized quadrature rules for singular integrals. And second, if support of ρ is discretized at N points $\boldsymbol{x}_j$, and the potential φ is required at each of these points, denoting our quadrature weights by w_j, we are left with the *dense* sum:

$$\varphi(\boldsymbol{x}_i) \approx \frac{1}{\epsilon}\sum_{j\neq i} \frac{w_j\,\rho(\boldsymbol{x}_j)}{4\pi|\boldsymbol{x}_i-\boldsymbol{x}_j|}. \tag{1.8}$$

It is the rapid evaluation of sums such as these that precipitated the development, originally, of what are known as *tree codes*. These algorithms are almost always low-order accurate, and scale as $O(N\log N)$. Later on, in the late 80s, more efficient algorithms known as *fast multipole methods* (FMMs) which scale linearly in N were developed to optimally accelerate the computation of N-body sums such as (1.8).

Despite these numerical and algorithmic requirements in order to efficiently use integral methods, the benefits far outweigh work required to implement them. For example, derivatives

of φ can be computed analytically, and the potential can be computed at locations $\boldsymbol{y}$ outside the support of ρ straightforwardly – that is to say, merely the sum above is evaluated with bx_i replaced by $\boldsymbol{y}$. Lastly, any numerical conditioning issues in the evaluation of the solution are usually tied to underlying physical considerations, and not artifacts of numerical differentiation or other various instabilities.

It is with examples such as these that the following lecture notes were developed. While fast multipole methods are most commonly associated with the evaluation of N-body sums, their true power lies in their ability to accelerated integral equation methods. This feature has revolutionized the field of computational electromagnetics, in particular. Analogous methods are used every day in large-scale calculations in fluid flow, acoustic wave propagation, and elastodynamics simulations. We will address many of these applications in the following chapters, with a focus on modern developments and thorough treatment of both the integral equation theory and the implementation of the related algorithms.

1.2 The state of computing

For the past several decades, Moore's Law regarding the growth of computational power has approximately held true: the number of transistors in integrated circuits has roughly doubled every two years. This behavior has resulted in unprecedented growth in computational power, allowing for computations that previously required entire buildings of computers to now be performed on laptops. While Moore's Law cannot continue to hold true indefinitely, it will most likely persist for some time.

This being said, it is more common today for computing environments to obtain more flops per second by adding parallelism in the form of networked machines or dedicated compute cards such as Graphics Processing Units (GPUs) or specialized co-processors, instead of relying on an increase in clock speed or density of transistors. Many of the following fast algorithms, since they are often built on hierarchical data structures, are compatible with such compute architecture.

The question that remains is: what happens to computational science when the exponential growth in compute power dies off? Or rather, what types of algorithms will be compatible with limited computing resources?

While not complete by any means, there are several characteristics that all the fast analysis-based algorithms shared that will be covered in these notes. These algorithms differ from many of the classical algorithms rooted in theoretical computer science (sorting, etc.).

> **Asymptotic scaling**: In part because Moore's Law has held true, asymptotic scaling of fast computational algorithms is perhaps their most important characteristic. For example, a standard matrix-vector multiplication requires $O(n^2)$ operations. If for twice the money, a computer can be purchased which is twice as fast and has twice as much memory (this is not exactly true), then only a matrix of dimension $\sqrt{2} \approx 1.4$ times larger can be applied to vectors of length N in the amount of time. This is not a sustainable scaling of cost vs. computational ability in the long-term. Going forward, by *fast algorithm* we mean one whose computational complexity scales as $O(N \log^p N)$, for some reasonably small p, as the size N of the inputs grows.

Controlled precision: Analysis-based algorithms strongly take advantage of the fact that modern day computers operate natively as finite precision machines. For example, the Fast Fourier Transform relies on particular algebraic observations concerning complex exponentials, as well as relying on the data to be sampled at equispaced intervals. No numerical approximation is made in the design of the algorithm; it is exact in infinite precision. This being said, often times that degree of precision is not needed. The algorithms detailed in this course often scale as $O(N \log^p N \log^q 1/\epsilon)$, where ϵ is the precision to which the output of the algorithm is desired. Obtaining more correct digits in the output only affects the constant implicit in the $O(\cdot)$ notation, *not* the asymptotic scaling with N. Because of only the need for finite precision solutions, many ideas in approximation theory can be used to accelerate and control intermediate calculations.

Technical implementations: Most, if not all of the algorithms described in these notes rely on hierarchical data structures, efficient linear algebra routines, and/or efficient *and* accurate evaluation of many of the special functions of classical mathematical physics (Bessel functions, orthogonal polynomials, etc.). Each of these tasks can be an entire project in and of themselves, but once implemented, become black box subroutines that can be relied on. In fact, the efficient and accurate evaluation of certain classes of special functions is an ongoing research topic in several groups.

The above characteristics of many fast analysis-based algorithms have some interesting consequences. For example, as in the FFT, evaluating N-body interactions (for very large N) using the fast multipole method is often *more* accurate that doing the direct calculation via matrix-vector multiplication. The reason for this is the inherent round-off error from computing the dot-product of two vectors of length N. In fact, the round-off error in computing $\boldsymbol{x} \cdot \boldsymbol{y}$ can be shown to be:

$$\epsilon \leq N \, \epsilon_{mach} \sum_j |x_j||y_j|. \tag{1.9}$$

Since fast multipole methods never add up N numbers for each data point, the round-off error is often much smaller.

1.3 Notes on the notes

In the following manuscript, we have explicitly avoided any such discuss of function spaces. When necessary, and of interest, references are provided to direct the reader to sources that may clarify the requirements on functions, kernels, operators, etc. For the most part, unless otherwise noted, boundaries are assumed to be smooth and kernels and functions are assumed to be square integrable. This assumption naturally corresponds to the classic L^2 theory of integral operators, as is commonly seen in works by Fredholm, Hilbert, and Riesz.

2 The Laplace equation

In the introduction, we saw that Poisson's equation arises naturally from physical considerations in electrostatics merely by using Coulomb's law for the force exerted on charged particles. Similarly, Laplace's equation with boundary conditions (and Poisson's equation) also arise naturally in fluid flow and elasticity. These problems will be treated in later chapters. We first discuss the simplest case, that of Laplace's equation in two dimensions.

2.1 Two dimensions

In two dimensions, the Green's function g for Laplace's equation satisfies

$$\Delta g(\boldsymbol{x}, \boldsymbol{x}') = \delta(\boldsymbol{x} - \boldsymbol{x}'). \tag{2.1}$$

Since the right hand side is only a function of r, we can assume that so is the Green's function $g = g(r)$. With this in mind, and switching to polar coordinates, we see that the Green's function, away from $r = 0$ must satisfy:

$$\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r}\right) g(r) = 0. \tag{2.2}$$

The solution to this ODE is $g(r) = c_1 \log r + c_2$. Taking $c_2 = 0$, the constant c_1 can be determined by integrating both sides of (2.1) over some ball of radius R and applying the Divergence Theorem. We have that

$$\begin{aligned}
\iint_{B(0,R)} \Delta g(r)\, dV(r) &= \iint_{B(0,R)} \nabla \cdot \nabla g(r)\, dV(r) \\
&= \int_{\partial B(0,R)} \boldsymbol{n} \cdot \nabla g(r)\, dA(r) \\
&= \int_0^{2\pi} \left(\frac{\partial}{\partial r} \log r\right) |_{r=R}\, R\, d\phi \\
&= 2\pi.
\end{aligned} \tag{2.3}$$

This result is independent of R, and since $\iint \delta = 1$, by definition the definition of the δ function, we have that

$$g(\boldsymbol{x}, \boldsymbol{x}') = g(\boldsymbol{x} - \boldsymbol{x}') = \frac{1}{2\pi} \log |\boldsymbol{x} - \boldsymbol{x}'|. \tag{2.4}$$

Using this Green's function the solution to the Poisson problem in all of $\mathbb{R}$,

$$\Delta u = f, \tag{2.5}$$

can explicitly be written as:

$$u(\boldsymbol{x}) = \frac{1}{2\pi} \iint_{\mathbb{R}^2} \log|\boldsymbol{x} - \boldsymbol{x}'|\, f(\boldsymbol{x}')\, dV(\boldsymbol{x}'). \tag{2.6}$$

The log kernel is absolutely integrable, and therefore u is well-defined everywhere.

For functions σ and μ supported on the boundary Γ of some open region $\Omega \subset \mathbb{R}^2$, we define the *single* and *double* layer potential operator:

$$\begin{aligned} \mathcal{S}\sigma(\boldsymbol{x}) &= \int_\Gamma g(\boldsymbol{x}, \boldsymbol{x}')\, \sigma(\boldsymbol{x}')\, ds(\boldsymbol{x}') \\ \mathcal{D}\mu(\boldsymbol{x}) &= \int_\Gamma \left(\frac{\partial}{\partial n'} g(\boldsymbol{x}, \boldsymbol{x}') \right) \mu(\boldsymbol{x}')\, ds(\boldsymbol{x}'), \end{aligned} \tag{2.7}$$

where $\partial/\partial n = \boldsymbol{n} \cdot \nabla$ and it is assumed that $\boldsymbol{x} \notin \Gamma$. It is worth noting that no matter what σ and μ are, the functions $\mathcal{S}\sigma$ and $\mathcal{D}\mu$ are harmonic *everywhere* by construction.

In the limit as $\boldsymbol{x}$ approaches the boundary Γ, special case must be made to in order to obtain the correct limit in these integrals. Since the Green's function g exhibits only a logarithmic singularity, it is absolutely integrable and we have that

$$\lim_{h \to 0^\pm} \mathcal{S}\sigma(\boldsymbol{x} + h\boldsymbol{n}) = \mathcal{S}\sigma(\boldsymbol{x}). \tag{2.8}$$

That is to say, the function $\mathcal{S}\sigma$ is continuous across the boundary Γ. On the other hand, it can be shown that the double layer potential $\mathcal{D}\mu$ exhibits a jump of size μ across Γ:

$$\lim_{h \to 0^\pm} \mathcal{D}\mu(\boldsymbol{x} + h\boldsymbol{n}) = \pm\frac{1}{2}\mu(\boldsymbol{x}) + \mathcal{D}\mu(\boldsymbol{x}), \tag{2.9}$$

where for $\boldsymbol{x} \in \Gamma$, in somewhat of an abuse of notation, $\mathcal{D}\mu(\boldsymbol{x})$ is interpreted in its principal value sense:

$$\mathcal{D}\mu(\boldsymbol{x}) = \lim_{\epsilon \to 0} \int_{\Gamma \setminus B(\boldsymbol{x}, \epsilon)} \frac{-\boldsymbol{n}' \cdot (\boldsymbol{x} - \boldsymbol{x}')}{2\pi|\boldsymbol{x} - \boldsymbol{x}'|^2} \mu(\boldsymbol{x}') ds(\boldsymbol{x}'), \tag{2.10}$$

where $B(\boldsymbol{x}, \epsilon)$ is the ball of radius ϵ centered at $\boldsymbol{x}$. A careful calculation along the smooth curve Γ can show that the integral over $\Gamma \cap B(\boldsymbol{x}, \epsilon)$ is merely $\pm\mu/2$:

$$\lim_{\epsilon \to 0} \lim_{h \to 0^\pm} \int_{\Gamma \cap B(\boldsymbol{x}, \epsilon)} \frac{-\boldsymbol{n}' \cdot (\boldsymbol{x} + h\boldsymbol{n} - \boldsymbol{x}')}{2\pi|\boldsymbol{x} + h\boldsymbol{n} - \boldsymbol{x}'|^2} \mu(\boldsymbol{x}') ds(\boldsymbol{x}') = \pm\frac{1}{2}\mu(\boldsymbol{x}). \tag{2.11}$$

Furthermore, the double layer kernel is actually continuous at $\boldsymbol{x} = \boldsymbol{x}'$, with

$$\lim_{\boldsymbol{x} \to \boldsymbol{x}'} \frac{-\boldsymbol{n}' \cdot (\boldsymbol{x} - \boldsymbol{x}')}{2\pi|\boldsymbol{x} - \boldsymbol{x}'|^2} = \kappa(\boldsymbol{x}), \tag{2.12}$$

where κ denotes the signed curvature of Γ. See [**?**] for a full derivation. Related formulas in the complex analytic case, known as Sokhotski-Plemelj formulae, can easily be derived using the residue theorem.

Lemma 1 (Plemelji formula)**.** *Let f be a complex analytic function in a neighborhood of the curve Γ which bounds a simply connected region $\Omega \subset \mathbb{C}$. Then for $w \in \Gamma$,*

$$\lim_{\xi \to w^{\pm}} \frac{1}{2\pi i} \int_{\Gamma} \frac{f(z)}{\xi - z}\, dz = \pm \frac{1}{2} f(w) + \text{PV} \int_{\Gamma} \frac{f(z)}{\xi - z}\, dz, \tag{2.13}$$

where + denotes the limit from the exterior ($\mathbb{C} \setminus \Omega$), − the limit from the interior, and PV *the principal value.*

Proof. We prove the case for the interior limit only, as the exterior limit is virtually identical. To this end, let $w \in \Gamma$, and let $\Gamma_\epsilon = \Gamma \cap B(w, \epsilon)$. Denote by $\partial B^+(w, \epsilon)$ the part of the boundary of this ball that lies exterior to Ω. The interior limit corresponds to the limit as $\epsilon \to 0$ in Figure 2.1. The direction (counter-clockwise) of integration is also shown in the figure. The integral can therefore be split as follows:

$$\int_{\Gamma} \frac{f(z)}{w - z}\, dz = \lim_{\epsilon \to 0} \int_{\partial B^+(w,\epsilon)} \frac{f(z)}{w - z}\, dz + \text{PV} \int_{\Gamma} \frac{f(z)}{w - z}\, dz. \tag{2.14}$$

Under the assumption that f is analytic in a neighborhood of Γ, ϵ can be chosen small enough so that f can be written in $B(w, \epsilon)$ as

$$f(z) = f(w) + f'(w)(z - w) + \frac{f''(w)}{2}(z - w) + \ldots. \tag{2.15}$$

Inserting this expansion into the first integral on the right of (2.14), we have (as in a residue calculation):

$$\begin{aligned} \int_{\partial B^+(w,\epsilon)} \frac{f(z)}{w - z}\, dz &= \int_{\partial B^+(w,\epsilon)} \frac{f(w)}{w - z}\, dz + \int_{\partial B^+(w,\epsilon)} \frac{f'(w)\,(z - w)}{w - z}\, dz + \ldots \\ &= f(w) \int_{\theta_1}^{\theta_2} \frac{-i\epsilon e^{i\theta}}{\epsilon e^{i\theta}}\, d\theta + f'(w) \int_{\partial B^+(w,\epsilon)} \frac{i\epsilon^2 e^{i2\theta}}{\epsilon e^{i\theta}}\, d\theta + \ldots \\ &= -if(w) \int_{\theta_1}^{\theta_2} d\theta + i\epsilon f'(w) \int_{\partial B^+(w,\epsilon)} e^{i\theta}\, d\theta + \ldots \end{aligned} \tag{2.16}$$

where the curve has been parameterized according to $w - z = \epsilon e^{i\theta}$. Taking the limit as $\epsilon \to 0$, we see that $\theta_2 - \theta_1 \to \pi$, and we have that

$$\lim_{\epsilon \to 0} \int_{\partial B^+(w,\epsilon)} \frac{f(z)}{w - z}\, dz = -i\pi f(w). \tag{2.17}$$

□

A similar calculation can be done for the real-valued double layer kernel, as it can be shown that

$$\Re\left(\frac{1}{2\pi i} \frac{dz'}{z - z'}\right) = \frac{-\boldsymbol{n}' \cdot (\boldsymbol{x} - \boldsymbol{x}')}{|\boldsymbol{x} - \boldsymbol{x}'|^2}\, ds(\boldsymbol{x}'), \tag{2.18}$$

where we have associated the complex values z, z' with the real vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively.

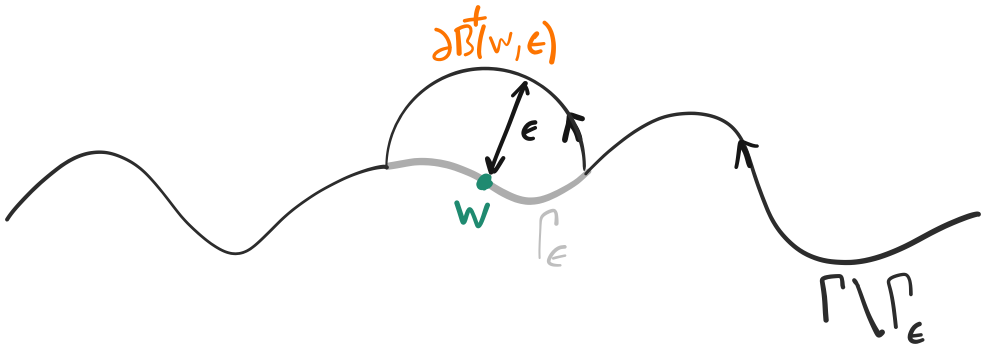

Figure 2.1: Splitting of the boundary curve in order to calculate the jump conditions layer potential operators and the Cauchy integral for values on the curve.

To summarize, we have the following limited behaviors for the single and double layer potentials:

$$\begin{aligned} \lim_{h\to 0^{\pm}} \mathcal{S}\sigma(\boldsymbol{x}+h\boldsymbol{n}) &= \mathcal{S}\sigma(\boldsymbol{x}), \\ \lim_{h\to 0^{\pm}} \mathcal{D}\mu(\boldsymbol{x}+h\boldsymbol{n}) &= \pm\frac{1}{2}\mu(\boldsymbol{x}) + \mathcal{D}\mu(\boldsymbol{x}). \end{aligned} \tag{2.19}$$

The kernel of $\mathcal{S}$ is singular, but integrable, and the kernel of $\mathcal{D}$ is continuous (and therefore integrable). Analogous results can be shown for their normal derivatives along Γ, denoted by $\mathcal{S}'$ and $\mathcal{D}'$:

$$\begin{aligned} \lim_{h\to 0^{\pm}} \mathcal{S}'\sigma(\boldsymbol{x}+h\boldsymbol{n}) &= \mp\frac{1}{2}\sigma(\boldsymbol{x}) + \mathcal{S}'\sigma(\boldsymbol{x}), \\ \lim_{h\to 0^{\pm}} \mathcal{D}'\mu(\boldsymbol{x}+h\boldsymbol{n}) &= \mathcal{D}'\mu(\boldsymbol{x}). \end{aligned} \tag{2.20}$$

The operator $\mathcal{D}'$ is known as a hypersingular operator, and has to be interpreted as a finite part integral. That is to say, for $\boldsymbol{x} \in \Gamma$,

$$\text{FP}\,\mathcal{D}'\mu(\boldsymbol{x}) = \lim_{\epsilon\to 0}\left(\int_{\Gamma\setminus B(\boldsymbol{x},\epsilon)} \frac{\partial^2 g(\boldsymbol{x},\boldsymbol{x}')}{\partial n\,\partial n'}\mu(\boldsymbol{x}')\,ds(\boldsymbol{x}') - \frac{2\mu(\boldsymbol{x})}{\epsilon}\right). \tag{2.21}$$

As before, principal value and finite part integrals are implied when the argument is located on the boundary of integration.

2.1.1 The Interior Dirichlet Problem

The canonical boundary value problem for the Laplace equation is that of the Interior Dirichlet problem. In this section we derive a corresponding integral equation for this boundary value problem, prove that it has a unique solution, and then discuss discretization and quadrature techniques in the next section. Other boundary value problems are treated in subsequent sections, but the techniques are very similar.

For some connected bounded region Ω with smooth boundary Γ, and f defined on Γ, the interior Dirichlet boundary value problem is given as:

$$\begin{aligned} \Delta u &= 0, && \text{in } \Omega, \\ u &= f, && \text{on } \Gamma. \end{aligned} \tag{2.22}$$

This boundary value problem arises naturally in electrostatic scattering problems from perfect electric conductors. The uniqueness of u is established by Green's First Identity (see Appendix). If u_1 and u_2 were two different solutions to (2.22), then the difference $u = u_1 - u_2$ would satisfy the interior Dirichlet problem with $f = 0$. Setting $v = u$ in Green's First Identity (A.2), we have that

$$\int_\Omega |\nabla u|^2 dV = 0. \tag{2.23}$$

Therefore, u must be a constant. But since $u = 0$ on Γ, $u = 0$ everywhere.

Instead of numerically solving (2.22) in differential form, we now change variables and represent the solution u in Ω in terms of a double layer potential due to an unknown distribution σ (motivation to be discussed later on). For $\boldsymbol{x}$ in the interior of Ω, we write:

$$\begin{aligned} u(\boldsymbol{x}) &= \mathcal{D}\sigma(\boldsymbol{x}) \\ &= \int_\Gamma \left(\frac{\partial}{\partial n'} \frac{1}{2\pi} \log|\boldsymbol{x} - \boldsymbol{x}'| \right) \sigma(\boldsymbol{x}')\, ds(\boldsymbol{x}'). \end{aligned} \tag{2.24}$$

Note that since the Green's function was used in the representation of u, we *automatically* have that $\Delta u = 0$ in Ω. It merely remains to satisfy the boundary condition on Γ that $u = f$. Using the limiting values of the layer potential operators discussed in the previous section, enforcing the boundary condition $u = f$ on Γ yields the following integral equation for σ:

$$-\frac{1}{2}\sigma + \mathcal{D}\sigma = f, \tag{2.25}$$

where the limit to the boundary has been taken from the domain of interest (i.e., the *interior*, inside Ω). Since $\mathcal{D}\sigma$ defines a harmonic function everywhere inside Ω, any such solution to the above integral equation will yield a solution u to the interior Dirichlet problem by setting $u = \mathcal{D}\sigma$. It remains to be shown that there is a unique solution to this integral equation.

Theorem 1. *The integral equation $(-\mathcal{I}/2 + \mathcal{D})\sigma = f$ on Γ has a unique solution σ for any right-hand side f. The function $u = \mathcal{D}\sigma$ then uniquely solves the interior Dirichlet problem in Ω with boundary data f.*

Proof. By the Fredholm Alternative (see Appendix B), since the operator $\mathcal{D}$ is compact, to show the uniqueness of σ it suffices to show that the homogeneous integral equation

$$\left(-\frac{1}{2}\mathcal{I} + \mathcal{D} \right) \sigma = 0 \tag{2.26}$$

only has the solution $\sigma = 0$. To this end, define the function $u^{in}(\boldsymbol{x}) = \mathcal{D}\sigma(\boldsymbol{x})$ for $\boldsymbol{x} \in \Omega$. By the jump properties of the double layer, we have that for $\boldsymbol{x} \in \Gamma$:

$$\begin{aligned} \lim_{h \to 0^-} u^{in}(\boldsymbol{x} + h\boldsymbol{n}) &= -\frac{\sigma(\boldsymbol{x})}{2} + \mathcal{D}\sigma(\boldsymbol{x}) \\ &= 0. \end{aligned} \tag{2.27}$$

This means that u^{in} satisfies the interior Dirichlet problem with zero boundary data. This implies that $u^{in} = 0$ in Ω, and therefore, also that for $\boldsymbol{x} \in \Gamma$:

$$\lim_{h \to 0^-} \frac{\partial u^{in}}{\partial n}(\boldsymbol{x} + h\boldsymbol{n}) = 0. \tag{2.28}$$

Next, for $\boldsymbol{x} \in \mathbb{R}^2 \setminus \bar{\Omega}$, set $u^{out}(\boldsymbol{x}) = \mathcal{D}\sigma(\boldsymbol{x})$. Since $\mathcal{D}'\sigma$ is continuous across the boundary Γ, we have that for $\boldsymbol{x} \in \Gamma$,

$$\frac{\partial u^{out}}{\partial n} = \frac{\partial u^{in}}{\partial n} = 0. \tag{2.29}$$

That is to say, u^{out} satisfies that boundary value problem:

$$\begin{aligned} \Delta u &= 0, && \mathbb{R}^2 \setminus \bar{\Omega}, \\ \frac{\partial u^{out}}{\partial n} &= 0, && \text{on } \Gamma. \end{aligned} \tag{2.30}$$

We will now show that this implies that $u^{out} = 0$ in $\mathbb{R}^2 \setminus \bar{\Omega}$. We begin by applying Green's First Identity in the region contained between Γ and a very large circle of radius R centered at the origin. We will denote this region by $D = B(0, R) \setminus \Omega$:

$$\int_D \left(u^{out} \Delta u^{out} + |\nabla u^{out}|^2 \right) \, dv = \int_{\partial D} u^{out} \frac{\partial u^{out}}{\partial n} \, da. \tag{2.31}$$

Since u^{out} is harmonic, and $\partial u^{out}/\partial n = 0$ on Γ, we have that:

$$\int_D |\nabla u^{out}|^2 \, dv = \int_{\partial B(0,R)} u^{out} \frac{\partial u^{out}}{\partial n} \, da. \tag{2.32}$$

But for sufficiently large R, $u^{out} \sim O(1/R)$ on $\partial B(0, R)$ since:

$$\begin{aligned} u^{out}(\boldsymbol{x}) &= \mathcal{D}\sigma(\boldsymbol{x}) \\ &= \int_\Gamma \frac{1}{2\pi} \left(\frac{\partial}{\partial n'} \log |\boldsymbol{x} - \boldsymbol{x}'| \right) \sigma(\boldsymbol{x}') \, da \\ &= -\int_\Gamma \frac{1}{2\pi} \frac{\boldsymbol{n}' \cdot (\boldsymbol{x} - \boldsymbol{x}')}{|\boldsymbol{x} - \boldsymbol{x}'|} \sigma(\boldsymbol{x}') \, da \\ &\sim O\left(\frac{1}{|\boldsymbol{x}|} \right) \text{ as } \boldsymbol{x} \to \infty. \end{aligned} \tag{2.33}$$

Taking the normal derivative, $\partial/\partial r$, of this asymptotic behavior in u^{out} we have that as $|\boldsymbol{x}| \to \infty$:

$$\frac{\partial u^{out}}{\partial n}(\boldsymbol{x}) \sim \frac{1}{|\boldsymbol{x}|^2}. \tag{2.34}$$

Using this behavior in (2.32), we have that

$$\begin{aligned} \lim_{R\to\infty} \int_D |\nabla u^{out}|^2 \, dv &= \lim_{R\to\infty} \int_{\partial B(0,R)} u^{out} \frac{\partial u^{out}}{\partial n} \, da \\ &\sim \lim_{R\to\infty} \int_{\partial B(0,R)} O\left(\frac{1}{r^3} \right) \, r \, dr \\ &\sim \lim_{R\to\infty} O\left(\frac{1}{R^2} \right) \\ &= 0. \end{aligned} \tag{2.35}$$

This implies that u^{out} is a constant, but since $u^{out} \to 0$ as $|\boldsymbol{x}| \to \infty$ it must be that $u^{out} = 0$. We now have both that $u^{in} = 0$ and that $u^{out} = 0$. By the jump conditions of $\mathcal{D}\sigma$, we have that

$$0 = \lim_{h \to 0^+} \left(u^{out}(\boldsymbol{x} + h\boldsymbol{n}) - u^{in}(\boldsymbol{x} - h\boldsymbol{n}) \right) = \sigma(\boldsymbol{x}). \tag{2.36}$$

This proves that if $(-\mathcal{I}/2 + \mathcal{D})\,\sigma = 0$ then $\sigma = 0$.

□

2.2 Three dimensions

3 Discretization and quadrature

This chapter will describe modern techniques for discretizing continuous boundary integral equations into finite dimension linear systems that can be solved using linear algebra techniques. Technically, discretization and quadrature can be viewed as two separate topics, but it is often convenient to treat them together as the easiest to use quadrature rule often depends on the method by which functions are discretized. We treat both the discretization of integrals and functions along curves in two dimensions and surfaces in three dimensions. The analogous volume discretizations are not discussed, but similar methods immediately extend to those cases.

In keeping with the theme of these lecture notes, we will assume that the operators to be discretized are viewed as maps from $L^2 \to L^2$, and that functions (unknowns and right-hand sides) to be discretized are elements of L^2. With this in mind, there are basically two options for both discretizing functions and enforcing the discrete approximation of the continuout integral equation: point values and moments (function expansions). That is to say, a function f (or class of functions) can be discretized using N point values $f(\boldsymbol{x}_j)$ at nodes $\boldsymbol{x}_j$ with weights w_j such that

$$\int_\Gamma |f(\boldsymbol{x})|^2 \, da(\boldsymbol{x}) = \sum_{j=1}^{N} w_j \, |f(\boldsymbol{x}_j)|^2. \tag{3.1}$$

If the above formula merely holds to accuracy ϵ, then we say that the nodes and weights $\boldsymbol{x}_j$, w_j provide an ϵ-discretization of f in L^2. (This formula may hold to a particular *order of approximation* as well, but we choose to use ϵ-discretizations instead because of the forthcoming relationship with matrix decompositions and approximations.) Values of f at locations other than the nodes $\boldsymbol{x}_j$ can be obtained via interpolation, which may or may not yield accurate results based on the choice of node locations. We will discuss this matter later on.

On the other hand, instead of using point values of f, one may wish to instead express f in terms of its coefficients with respect to some set of N basis functions φ_j:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j \, \varphi_j(\boldsymbol{x}). \tag{3.2}$$

Once again, if the above approximation is valid to accuracy ϵ, we say that φ_j provides an ϵ-discretization of f.

With these two methods of discretization in mind, we can discuss the three major methods for discretizing integral equations: the Nyström method, collocation, and Galerkin discretization. In short, the discretized linear systems of the continous integral equation $(\mathcal{I} + \mathcal{K})\sigma = f$ for each

of these methods is given by:

$$\begin{aligned}
\text{Nyström:} \qquad & \sigma_i + \sum_j w_j\, k(\boldsymbol{x}_i, \boldsymbol{x}_j)\, \sigma_j = f(\boldsymbol{x}_i) \\
\text{Collocation:} \qquad & \sum_j \alpha_j \left(\varphi_j(\boldsymbol{x}_i) + \mathcal{K}\varphi_j(\boldsymbol{x}_i)\right) = f(\boldsymbol{x}_i) \\
\text{Galerkin:} \qquad & \sum_j \alpha_j \left((\psi_i, \varphi_j) + (\psi_i, \mathcal{K}\varphi_j)\right) = (\psi_i, f),
\end{aligned} \tag{3.3}$$

where (f, g) denotes the usual L^2 inner product on Γ and ψ_i, φ_j are some set of suitable *testing* and *expansion* functions. The solution to the Nyström discretization is an approximation to actual functions values of σ, $\sigma_i \approx \sigma(\boldsymbol{x}_i)$, whereas the unknowns in both the collocation and Galerkin methods are the expansion coefficients of σ, given by α_i. While the linear systems due to collocation and Galerkin methods tend to be *slightly* better conditioned (heuristically) due to the extra integration, the Nyström method has several algorithmic advantages, namely that each matrix element $\mathbf{K}_{ij}$ in the discretized system $(\mathbf{I} + \mathbf{K})\boldsymbol{\sigma} = \mathbf{f}$ is merely a kernel evaluation times a weight function. This is in contrast to, for example, the Galerkin discretization wherein each matrix entry is an inner product: $(\psi_i, \mathcal{K}\varphi_j)$. As a result, matrix assembly for the Nyström method is often much faster than for the Galerkin method, and is directly compatible with the standard form of fast multipole methods for computing N-body simulations. It is for this reason that we mainly focus on Nyström-type discretizations going forward.

Within the technique of Nyström discretization, there is a subtle, but very practical scaling of the discretized system which ensures that (on the order of the underlying quadrature) that the finite dimensional operator on ℓ_2 captures the behavior of the infinite-dimensional operator on L^2. That is to say, with the proper scaling of unknowns, the eigenvalues of $\mathbf{I} + \mathbf{K}$ converge to the eigenvalues of the continuous operator $\mathcal{I} + \mathcal{K}$ as the number of discretization points tends to infinity. A thorough treatment of this can be found in [1, 2], but we outline the scaling here.

First, we note that given a nyström discretization of a function σ defined along a curve (or surface) Γ (i.e. values and weights), its L^2 norm can be approximated as:

$$\begin{aligned}
||\sigma||_2^2 &= \int_\Gamma |\sigma(\boldsymbol{x})|^2\, da(\boldsymbol{x}) \\
&\approx \sum_j w_j\, |\sigma(\boldsymbol{x}_j)|^2 \\
&= \sum_j |\sqrt{w_j}\sigma(\boldsymbol{x}_j)|^2.
\end{aligned} \tag{3.4}$$

Declaring the vector $\boldsymbol{\sigma} = (\sigma_1 \cdots \sigma_n)$ to be discretized version of σ means that the usual ℓ_2 norm of $\boldsymbol{\sigma}$ does *not* converge to the L^2 norm of σ, since

$$\begin{aligned}
||\boldsymbol{\sigma}||_2^2 &= \sum_j |\sigma_j|^2 \\
&\neq \sum_j \sum_j w_j\, |\sigma_j|^2 \\
&\approx \int_\Gamma |\sigma|^2\, da.
\end{aligned} \tag{3.5}$$

We denote the discretization of $\sigma \rightarrow \sqrt{w_j}\sigma(\boldsymbol{x}_j)$ as an L^2 embedding of σ, also known as an inner-product preserving discretization as the norm in ℓ_2 converges to the norm in L^2.

Likewise, the usual Nyström discretization of $\mathcal{K}$ with matrix entries $\mathbf{K}_{ij} = w_j\, k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ results in a matrix with eigenvalues which, in general, have nothing to do with the eigenvalues of the continuous operator $\mathcal{K}$. The proper scaling can be derived by noting the effect of using the inner product preserving discretization of σ in the discretized integral equation instead of the usual one. The ith equation becomes:

$$\begin{aligned} \sigma_i + \sum_j w_j\, k(\boldsymbol{x}_i, \boldsymbol{x}_j)\sigma_j &= f_i \\ \sigma_i + \sum_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\, \sqrt{w_j}\, \sqrt{w_j}\sigma_j &= f_i \\ \sqrt{w_i}\sigma_i + \sum_j \sqrt{w_i} k(\boldsymbol{x}_i, \boldsymbol{x}_j)\, \sqrt{w_j}\, \sqrt{w_j}\sigma_j &= \sqrt{w_i} f_i \end{aligned} \tag{3.6}$$

Therefore, the discretization $\sqrt{w_i w_j}k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the corresponding inner product preserving discretization of $\mathcal{K}$. These scalings are particularly important for adaptive discretizations, especially those along geometries with corners and edges as the integral operators are not compact on L^2 anymore but merely bounded. In particular, these weightings can have a large effect on the resulting condition number of the discretized system.

3.1 Curves in two dimensions

3.1.1 Discretization

While Nyström method *truists* may claim that discretizing a function at point values is just that – samples of a function at particular locations, and nothing more – the fact is that each such discretization *implies* an underlying representation of the function (as opposed to the explicit representation of the function in Galerkin methods).

Global periodic discretizations

Locally adaptive discretizations

3.1.2 Quadrature

Generally, in two dimensions, the integrals of interest have kernels with logarithmic singularities (absolutely integrable) or normal derivatives thereof (weakly-singular, defined in the principal value sense). The singularity of these kernels along a curve is *generic* in the sense that it can be shown to only depend on the distance of the source and target in *parameter-space*, not in $\mathbb{R}^2$.

For example,

$$\begin{aligned}\mathcal{S}\sigma(\boldsymbol{x}) &= \int_\Gamma \log|\boldsymbol{x}-\boldsymbol{x}'|\,\sigma(\boldsymbol{x}')\,ds(\boldsymbol{x}') \\ &= \int_0^1 \log|\boldsymbol{x}(\tau)-\boldsymbol{x}'(t)|\,\sigma(\boldsymbol{x}'(t))\,ds(\boldsymbol{x}'(t)) \\ &= \int_0^1 \log|\boldsymbol{x}(\tau)-\boldsymbol{x}'(t)|\,\sigma(t)\,\frac{ds}{dt}(t)\,dt \\ &= \int_0^1 \left(\log\left|\frac{\boldsymbol{x}(\tau)-\boldsymbol{x}'(t)}{\tau-t}\right| - \log|\tau-t|\right)\,\sigma(t)\,\frac{ds}{dt}(t)\,dt.\end{aligned} \tag{3.7}$$

The above expression contains no geometry information in the singular part, only the parameterization variables. Therefore, it suffices to design quadratures for polynomials times log on the line.

Adaptive quadrature

For example, if σ (the unknown solution to an integral equation) is discretized on a panel at k Legendre nodes in the parameterization domain of the panel, then the diagonal block in the resulting discretized linear system can be constructed as follows. Sampling σ at k Legendre points implies that there exists a degree $k-1$ interpolant on the same panel, given as:

$$\sigma(t) = \sum_{\ell=0}^{k-1} c_\ell\, P_\ell(t). \tag{3.8}$$

Inserting this interpolant into integral (3.7), for example, we have

$$\begin{aligned}\phi_i &= \int_\Gamma \log|\boldsymbol{x}_i-\boldsymbol{x}'|\,\sigma(\boldsymbol{x}')\,ds(\boldsymbol{x}') \\ &= \int_{-1}^1 \log|\boldsymbol{x}_i-\boldsymbol{x}(t)|\,\left(\sum_{\ell=0}^{k-1} c_\ell\, P_\ell(t)\right)\,\frac{ds}{dt}(t)\,dt.\end{aligned} \tag{3.9}$$

Product quadratures

Corrected-trapezoidal rules

Generalized Gaussian rules

3.2 Surfaces in three dimensions

4 The 3D Laplace Fast Multipole Method

4.1 Tree-codes

Early algorithms of the 1980's based on various oct-tree or kd-tree divisions of space, and used mainly for molecular dynamics or cosmological (galaxy formation) simulations, were known as tree-codes. These algorithms [**?**, **?**, **?**] were low-order (usually only using approximations equivalent to zeroth-order multipole expansions) and did not make use of systematic error analysis to guarantee precision upon completion.

In the work of Appel [**?**], a kd-tree was used to hierarchically compute the gravitational n-body problem for galaxy formation. The drawbacks of using a kd-tree, as opposed to the oct-tree that was later introduced by Barnes and Hut [**?**], is that it makes no assumption on the *physical* distribution of points. For uniformly distribution points, the two trees are *almost* equivalent in practice. Keeping track of the physical separation and structure points in an n-body calculation turns out to be necessary if one is to maintain control over the accuracy of the computation.

We now introduce the basic Barnes-Hut algorithm here (in terms of the potential function formulation), and discuss the error and higher-order extensions.

4.1.1 Barnes-Hut

The goal of many galaxy formation calculations is to simulation, in time, the movement and coagulation (or dispersion) of point-masses in the universe. Basic time integrators for this system require that, on every step, the following n-body calculation be done for all point-masses $\boldsymbol{x}_j$ in the system, with masses denoted by m_j:

$$\begin{aligned} \boldsymbol{F}_i &= \sum_{j \neq i} \frac{G m_i m_j (\boldsymbol{x}_i - \boldsymbol{x}_j)}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^3} \\ &= -G m_i \nabla_i \sum_{j \neq i} \frac{m_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|} \\ &= -G m_i \nabla_i \phi_i \end{aligned} \tag{4.1}$$

where ∇_i is the gradient on the *target*, $\boldsymbol{x}_i$. For simplicity, we will now address the problem of computing the sum of the potential point interactions, ϕ_i, and *not* the field $\boldsymbol{F}$. The field can be obtained by differentiation of the proceeding formulas.

The driving motivation for the following algorithm is that from far away, the gravitational field due to a collection of point masses located at $\boldsymbol{x}_j$ is very close to that due to a *single* point mass, located at the center of mass $\boldsymbol{x}_c$ of the $\boldsymbol{x}_j$'s. Using this idea hierarchically (i.e. replacing

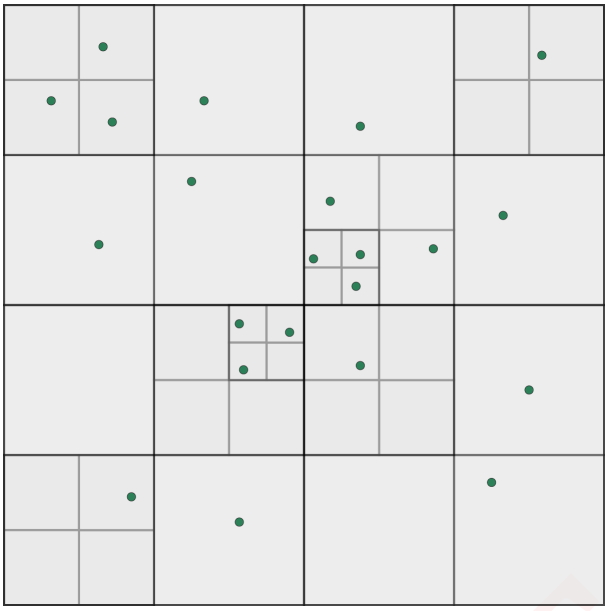

Figure 4.1: An oct-tree with one particle per box. The green dots can denote either original particles, or centers of mass for boxes on finer levels.

centers of mass with other centers of mass on larger scales), we have the following algorithm, known as Barnes-Hut.

THE ORIGINAL BARNES-HUT ALGORITHM

Step 0 Divide the computational box, containing all point masses, using an oct-tree such that at most one point is contained in each leaf node. Set all the ϕ_i's to zero, initialize threshold parameter $\theta < 1$.

Step 1 Starting at the leaf nodes, recursively compute the center of mass and total mass for all particles contained in each box, all the way to the root.

Step 2 Now, for each target particle $\boldsymbol{x}_i$, starting at the root, traverse the tree toward the leafs, processing one box B (that is not marked **done**) at a time. Let:

$$\begin{aligned} r_B &= \text{the radius of box } B \\ \boldsymbol{x}_{B,c} &= \text{the center of mass for all particles contained in } B \\ m_B &= \text{the total mass of all particles in box } B \\ r_{B,i} &= \text{the distance from } \boldsymbol{x}_{B,c} \text{ to } \boldsymbol{x}_i \end{aligned}$$

If $r_B/r_{B,i} < \theta$, mark this box as **done** and increment ϕ_i:

$$\phi_i = \phi_i + \frac{m_B}{\|\boldsymbol{x}_i - \boldsymbol{x}_{B,c}\|}.$$

If $r_B/r_{B,i} \geq \theta$, then skip this box and process its children. Repeat for all boxes.

Step 3 For all boxes B_j on the leaf level that were not marked **done**, add their contribution directly:

$$\phi_i = \phi_i + \frac{m_{B_j}}{\|\boldsymbol{x}_i - \boldsymbol{x}_{B_j}\|}.$$

The potential ϕ_i has now been computed.

Let us now discuss these steps one-by-one in more detail.

Complexity analysis

Step 0: This step can be considered pre-computation, as it merely constructs the data structure which keeps track of centers of mass and total mass for each box. The oct-tree data structure is hierarchical data structure that *evenly divides* space on each level into eight equal cubes (the two-dimensional version is a quad-tree, and the one-dimensional version is a binary tree) [**?**]. Its construction requires $O(n \log n)$ operations, where n is the total number of particles being sorted. The parameter θ that is set will be proportional to the global accuracy of the proceeding algorithm.

Step 1: The center of mass and total mass for each box is computed. For a box B with masses $m_j > 0$ located at $\boldsymbol{x}_j$, the center of mass is calculated as:

$$\boldsymbol{x}_{B,c} = \frac{\sum_j m_j \boldsymbol{x}_j}{\sum_j m_j}. \tag{4.2}$$

We will discuss the matter of non-positive masses later on when detailing multipole methods for electrostatics (in that case, electric charge takes the place of mass). The center of mass for parent boxes can be computed directly from the centers of mass of their children. Since on each level of the oct-tree, each box performs an $O(1)$ calculation to compute its center of mass and total mass (from its children), the complexity of this step is proportional to the total number of boxes, which is $O(n)$ in an oct-tree. See Figure 4.1 for a depiction of a two-dimensional oct-tree (quad-tree).

Step 2: This step is the important part of the Barnes-Hut algorithm. If the center of mass $\boldsymbol{x}_{B,c}$ of a particular box B is far away enough from the target particle $\boldsymbol{x}_i$ (relative to the size of the box containing the center of mass), then add this contribution to the potential for target i. In particular, if $r_B/r_{B,i} < \theta$ then this means that $r_{B,i} > r_B/\theta$, i.e. that particle i located at $\boldsymbol{x}_i$ is $1/\theta$ box-lengths away from the center of mass for box B (where one unit of box length is the box radius r_B). See Figure 4.2 for a graphical depiction of dimensions discussed.

As the tree is traversed, starting at the root, the values of r_B and $\max_B r_{B,i}$ decrease by the same constant factor (since each box is always divided evenly into eight children). Because of this fact, there are at most $O(1)$ boxes that will contribute to the potential ϕ_i on every one of $O(\log n)$ levels. Therefore, the computational cost for this step for one particle is $O(\log n)$, and for all particles is $O(n \log n)$.

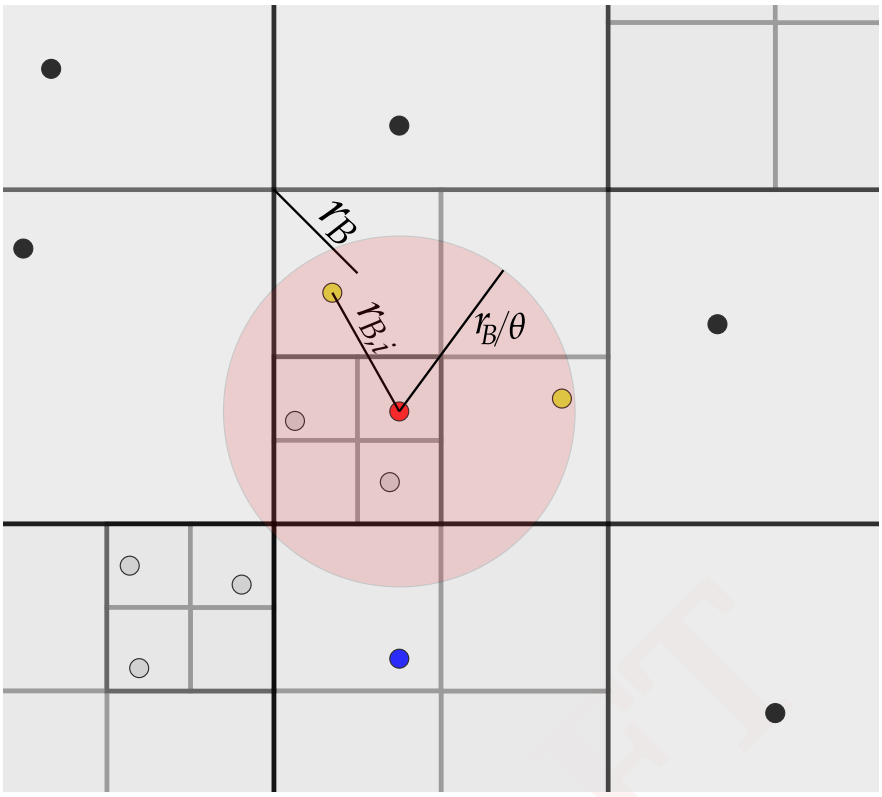

Figure 4.2: The region of interaction for a target particle $\boldsymbol{x}_i$ in the Barnes-Hut style tree-code algorithm. The red dot is the target, the yellow dots do not contribute, only the blue dot does. The grey dots are on levels have have not been processed, and the black dots have already been processed.

Step 3: Unless θ is very large, for a given particle $\boldsymbol{x}_i$ there will be a finite number of nearby leaf boxes whose center of mass (i.e. the location of the actual particle) will not satisfy the criterion $r_B/r_{B,i} \geq \theta$. The contribution to ϕ_i from these boxes is now added directly. For each particle, this occurs for $O(1)$ boxes, depending on the value of θ. The total cost for this step (for all particles $\boldsymbol{x}_i$) is $O(n)$.

Error analysis

In the center-of-mass Barnes-Hut algorithm, the only parameter that can be changed to increase the accuracy of the computation is θ. We will now derive an expression in θ based on multipole expansions for an estimate of the error for Barnes-Hut [**?**].

The main approximation made in the original Barnes-Hut algorithm was the center of mass approximation. In its simplest form, we are interested in the error in the following approximation:

$$\begin{aligned} \phi(\boldsymbol{x}_i) &= \frac{m_a}{\|\boldsymbol{x}_i - \boldsymbol{x}_a\|} + \frac{m_a}{\|\boldsymbol{x}_i - \boldsymbol{x}_a\|} \\ &= \frac{m_a + m_b}{\left\|\boldsymbol{x}_i - \frac{m_a \boldsymbol{x}_a + m_b \boldsymbol{x}_b}{m_a + m_b}\right\|} + \epsilon \end{aligned} \tag{4.3}$$

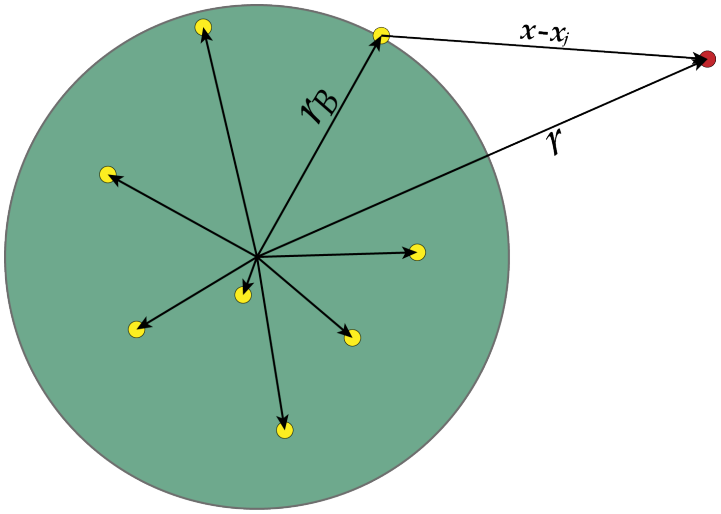

Figure 4.3: Source and target particles for a multipole expansion.

as a function of the distance from $\boldsymbol{x}_i$ to the center of mass of $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$. In order to estimate this error, we need an expression for $1/\|\boldsymbol{x}-\boldsymbol{x}'\|$ in terms of multipoles. As detailed in Appendix **??**, if $\|\boldsymbol{x}'\| < \|\boldsymbol{x}\|$, then this fundamental solution can be expanded as:

$$\frac{1}{\|\boldsymbol{x}-\boldsymbol{x}'\|} = \sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{\ell}\frac{r'^{\ell}}{r^{\ell+1}}\,Y_\ell^{-m}(\theta',\varphi')\,Y_\ell^m(\theta,\varphi) \tag{4.4}$$

where the spherical coordinates of $\boldsymbol{x}$ are r,θ,φ and similarly for $\boldsymbol{x}'$. Truncating this expansion at $\ell = p$ yields the following error estimate:

$$\left|\frac{1}{\|\boldsymbol{x}-\boldsymbol{x}'\|} - \sum_{\ell=0}^{p}\sum_{m=-\ell}^{\ell}\frac{r'^{\ell}}{r^{\ell+1}}\,Y_\ell^{-m}(\theta',\varphi')\,Y_\ell^m(\theta,\varphi)\right| \leq \frac{1}{r-r'}\left(\frac{r'}{r}\right)^{p+1}, \tag{4.5}$$

which is straightforward to derive using the fact that when properly normalized, $|P_\ell^m| \leq 1$. A sufficient normalization is given in the appendix. By the principle of superposition, the multipole expansion for a collection of point sources with masses m_j and locations r_j,θ_j,φ_j with $r_j < r$ is

$$\begin{aligned}\phi(\boldsymbol{x}) &= \sum_j \frac{m_j}{\|\boldsymbol{x}-\boldsymbol{x}_j\|} \\ &= \sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{\ell}\frac{M_{\ell m}}{r^{\ell+1}}\,Y_\ell^m(\theta,\varphi),\end{aligned} \tag{4.6}$$

with

$$M_{\ell m} = \sum_j m_j\, r_j^\ell\, Y_\ell^{-m}(\theta_j,\varphi_j). \tag{4.7}$$

Likewise, by interchanging the order of summation, the error in truncating this expansion at

$\ell = p$ is given by

$$\left| \phi(\boldsymbol{x}) - \sum_{\ell=0}^{p} \sum_{m=-\ell}^{\ell} \frac{M_{\ell m}}{r^{\ell+1}} Y_\ell^m(\theta, \varphi) \right| \leq \frac{\sum_j m_j}{r - r_B} \left(\frac{r_B}{r} \right)^{p+1}, \tag{4.8}$$

where $r_B = \max_j r_j$, see Figure 4.3.

We now apply this error estimate to the geometry governing particle interactions in the Barnes-Hut algorithm. Examining Figure 4.2, we see that when processing a box B, the center of mass $\boldsymbol{x}_{B,c}$ can be located *anywhere* inside B. Therefore, the radius of the smallest ball, centered at $\boldsymbol{x}_{B,c}$ that encloses all sources contained in box B is bounded by $2r_B$, where r_B is the diagonal radius of the box. Using this observation, we see that the Barnes-Hut potential at location $\boldsymbol{x}_i$ due to all sources in box B is given by equation (4.3) is nothing more than a zeroth-order multipole expansion, centered at $\boldsymbol{x}_{B,c}$:

$$\begin{aligned} \phi(\boldsymbol{x}_i) &\approx \frac{m_B}{r_{B,i}} \\ &= \frac{\sum_j m_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_{B_c}\|} \end{aligned} \tag{4.9}$$

where the sum is over all masses m_j contained in box B. For the Barnes-Hut parameter $\theta \leq 1/2$, the error in the zeroth-order multipole expansion is then given by:

$$\begin{aligned} \left| \phi(\boldsymbol{x}_i) - \frac{m_B}{r_{B,i}} \right| &\leq \frac{m_B}{r_{B,i} - 2r_B} \left(\frac{2r_B}{r_{B,i}} \right) \\ &\leq \frac{2m_B}{r_{B,i} - 2r_B} \theta \\ &\leq \frac{m_B}{r_B} \frac{2\theta^2}{1 - 2\theta}. \end{aligned} \tag{4.10}$$

At first glance, this error estimate (which is an absolute accuracy statement) grows linearly with the size of the mass contained in each box. However, multipole estimates are inherently a *relative precision* statement with respect to the scale of the potential, $\sum_j m_j$. Similar observations hold when applying multipole estimates not to point masses, but rather to *mass densities* (e.g. in the case of solving a Poisson problem, where multipole methods are used to evaluate integrals).

In order to improve the accuracy of the Barnes-Hut algorithm, we must use higher order multipole expansions for evaluate the potential due to a collection of sources in a particular box. In the next section, we will describe the calculations for these *high-order tree-codes*.

4.1.2 High-order tree-codes

In the first upward-pass through the oct-tree in the original Barnes-Hut algorithm, the total mass and center of mass for each box was calculated. These quantities are then used to approximate the potential due to all the sources in a particular box, regardless of its depth in the oct-tree. This center of mass approximation is nothing more than a zeroth-order multipole expansion, centered at the center of mass, with a multipole coefficient equal to the total mass contained in the box (as was shown in the previous section). In order to improve the accuracy of this $O(n \log n)$ scheme, the zeroth-order multipole approximation can be replaced with one of higher order. In doing so, we now require one additional piece of mathematics: a procedure for translating and merging the

multipole expansions of child boxes into one for their parent. In particular, we can now describe a higher-order version of the previous Barnes-Hut scheme, with some additional modifications that will enable better error approximation.

As before, we first informally describe the algorithm and then examine details of the complexity and accuracy.

A p^{th} ORDER TREE-CODE

Step 0 Divide the computational box, containing all points, using an oct-tree such that there is at most one point contained in each leaf node. Set all the ϕ_i's to zero, initialize the order of the multipole expansion to $p > 0$.

Step 1 Process all the leaf nodes. For each box B on the leaf level, L, form the order p multipole expansion due to the source contained in B, centered about the center of box B.

Step 2 For levels $L-1$ to 0, recursively compute the order p multipole expansion centered about the center of a box B by translating and merging the multipole expansions for the children of box B.

Step 3 For each target particle $\boldsymbol{x}_i = (r_i, \theta_i, \phi_i)$, starting at the root, traverse the tree toward the leafs, processing one box B (that is not marked **done**) at a time. Let:

$$\begin{aligned} r_B &= \text{the radius of box } B \\ \boldsymbol{x}_B &= \text{the center of box } B \\ M_{\ell m}^B &= \text{the multipole coefficients for box } B \\ r_{B,i} &= \|\boldsymbol{x}_b - \boldsymbol{x}_i\|, \text{ the distance from } \boldsymbol{x}_B \text{ to } \boldsymbol{x}_i \\ (\rho, \alpha, \beta) &= \boldsymbol{x}_i - \boldsymbol{x}_B, \text{ the coordinates of } \boldsymbol{x}_i \text{ relative to } \boldsymbol{x}_B \end{aligned}$$

If box B is *well-separated* from the target $\boldsymbol{x}_i$, mark this box as **done** and increment ϕ_i:

$$\phi_i = \phi_i + \sum_{\ell=0}^{p} \sum_{m=-\ell}^{\ell} \frac{M_{\ell m}^B}{\rho^{\ell+1}} Y_\ell^m(\alpha, \beta)$$

By well-separated, we mean that the target $\boldsymbol{x}_i$ is separated from box B by at least one box of the same size as B. See Figure **??**. If $\boldsymbol{x}_i$ is not well-separated, then skip box B and process its children using the same procedure. Repeat for targets $\boldsymbol{x}_i$.

Step 4 For any boxes B on the leaf level that were not marked **done**, add the contributions from the sources $\boldsymbol{x}_j$ located in them directly:

$$\phi_i = \phi_i + \frac{m_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}.$$

The potential ϕ_i, due to all sources other than $\boldsymbol{x}_i$, has now been computed.

Note: Unlike the previous tree-code, where the inclusion of a potential contribution was determined by a separation parameter θ, which in turn determined the accuracy of the calculation, this scheme does *not* contain an analogous parameter. All multipole-target interactions are determined by the oct-tree structure. This allows for more efficient computation and data structure development. We shall see that it is the multipole order p which directly determines the accuracy of the calculation.

It remains to be discussed how to *translate* a multipole expansion, as in **Step 2** of the p^{th} order tree-code described above. We can formulate this computational task as follows: let $M'_{\ell m}$ be the multipole coefficients for an expansion centered at $\boldsymbol{x}' = (r', \theta', \varphi')$, valid at any point $\boldsymbol{x} = (r, \theta, \varphi)$ such that $\|\boldsymbol{x} - \boldsymbol{x}'\| > a$. Let the coordinates of $\boldsymbol{x} - \boldsymbol{x}'$ by given as (ρ, α, β), i.e. these are the coordinates of $\boldsymbol{x}$ relative to $\boldsymbol{x}'$. See Figure 4.4. Also assume that $r > r' + a$. The potential at $\boldsymbol{x}$ is then given by:

$$\phi(\boldsymbol{x}) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \frac{M'_{\ell m}}{\rho^{\ell+1}} Y_\ell^m(\alpha, \beta), \qquad \text{for } \|\boldsymbol{x} - \boldsymbol{x}'\| > a. \tag{4.11}$$

The computational task in translating a multipole expansion to the origin $\mathbf{0}$ is to construct a new expansion for the potential at $\boldsymbol{x}$, valid outside of a ball of radius $r' + a$, centered at the origin, in terms of the absolute coordinates of $\boldsymbol{x}$:

$$\phi(\boldsymbol{x}) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \frac{M_{\ell m}}{r^{\ell+1}} Y_\ell^m(\theta, \varphi), \qquad \text{for } \|\boldsymbol{x}\| > r' + a, \tag{4.12}$$

and where the new multipole coefficients $M_{\ell m}$ depend only on $\boldsymbol{x}'$ and the $M'_{\ell m}$'s. We will now derive a formula for the coefficients $M_{\ell m}$, i.e. the *translation to the origin* operator. This operator is often referred to as an `M2M` or `multipole_to_multipole` operator, when thought of as a subroutine.

Equivalently, we wish to derive an expression for $Y_\ell^m(\alpha, \beta)/\rho^{\ell+1}$ in terms of the functions $Y_\ell^m(\theta, \varphi)/r^{\ell+1}$. The coefficients in such an expression will yield the translation operator. In one dimension, these so-called *addition formulas* are best illustrated in terms of complex exponentials:

$$\begin{aligned} e^{i(\alpha+\beta)} &= e^{i\alpha} e^{i\beta} \\ &= (\cos\alpha + i\sin\alpha)(\cos\beta + i\sin\beta) \\ &= (\cos\alpha\cos\beta - \sin\alpha\sin\beta) + i\,(\cos\alpha\sin\beta - \sin\alpha\cos\beta) \end{aligned} \tag{4.13}$$

which yield the sum-of-angles and difference-of-angles formulas for sin and cos. The angle $\theta = \alpha + \beta$ can also be given in terms of an angle *relative to* α, which is β. We merely seek analogous formulas for the functions $Y_\ell^m(\alpha, \beta)/\rho^{\ell+1}$, also known as *outgoing solid harmonics*.

We begin with a lemma which gives each of the solid harmonics, defined in spherical coordinates, in terms of *cartesian* derivatives of the function $1/\|\boldsymbol{x}\|$. A derivation of the following formulas can be found in [].

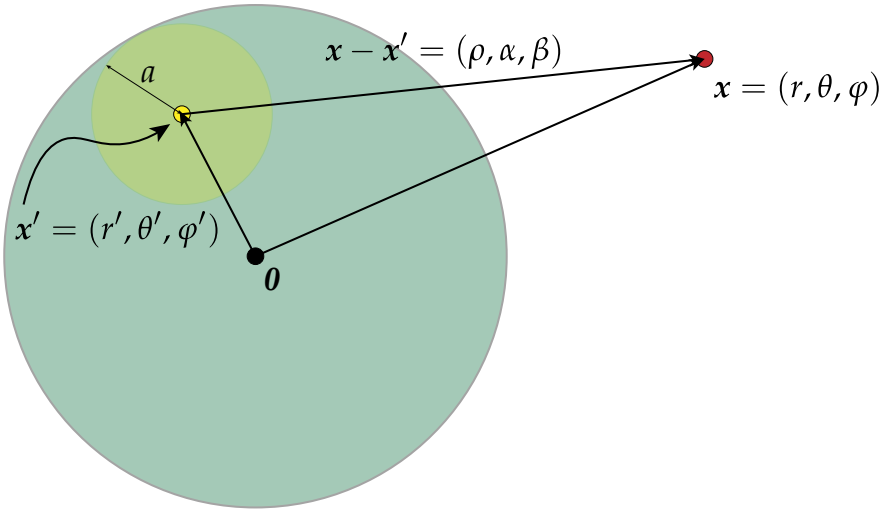

Figure 4.4: The geometry used in constructing the addition formula for Y_ℓ^m.

Lemma 2. *Let the spherical coordinates of the point* $\boldsymbol{x}$ *be given by* (r, θ, φ)*, with* $r^2 = x^2 + y^2 + z^2$*,* θ *the angle from the z-axis, and* φ *the azimuthal angle. Then for all* $\ell \geq m \geq 0$*,*

$$\frac{Y_\ell^{\pm m}(\theta, \varphi)}{r^{\ell+1}} = A_{\ell m} \left(\frac{\partial}{\partial x} \pm i \frac{\partial}{\partial y} \right)^m \left(\frac{\partial}{\partial z} \right)^{\ell - m} \left(\frac{1}{r} \right), \tag{4.14}$$

with

$$A_{\ell m} = \frac{(-1)^\ell}{\sqrt{(\ell - m)!\,(\ell + m)!}}. \tag{4.15}$$

Keep in mind that the above definition of $A_{\ell m}$ depends directly on the normalization used in defining the associated Legendre functions P_ℓ^m. The above definition is consistent with the normalization for P_ℓ^m and Y_ℓ^m in Appendix **??**. Often, the differential operators given in Lemma 2 will be abbreviated as:

$$\partial_+ = \frac{\partial}{\partial x} + i \frac{\partial}{\partial y}, \qquad \partial_- = \frac{\partial}{\partial x} - i \frac{\partial}{\partial y}, \qquad \partial_z = \frac{\partial}{\partial z}. \tag{4.16}$$

These operators, $\partial_\pm$, ∂_z are related to the raising and lowering operators (ladder operators) in quantum mechanics, see [**?**] for a nice discussion. Furthermore, if a function u is harmonic, then we will use the fact that

$$\partial_+ \partial_- u = -\partial_z^2 u. \tag{4.17}$$

To derive the multipole-to-multipole addition formula, we begin by recalling the multipole expansion of a point source at $\boldsymbol{x}'$:

$$\begin{aligned} \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}'\|} &= \frac{1}{\rho} \\ &= \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \frac{r'^\ell}{r^{\ell+1}} Y_\ell^{-m}(\theta', \varphi')\, Y_\ell^m(\theta, \varphi), \end{aligned} \tag{4.18}$$

and inserting the representation in (4.14), we have that

$$\frac{1}{\rho} = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} r'^{\ell}\, Y_{\ell}^{-m}(\theta', \varphi')\, A_{\ell m}\, \partial_{\mathrm{sgn}(m)}^{|m|}\, \partial_z^{\ell-|m|} \left(\frac{1}{r}\right), \tag{4.19}$$

where $\mathrm{sgn}(m) = 1$ if $m \geq 0$, and -1 otherwise. We handle the cases for $m' \geq 0$ and $m' < 0$ separately for the sake of clarity. For $\ell' \geq m' \geq 0$, if we apply the operator $\partial_+^{m'}\, \partial_z^{\ell'-m'}$ to both sides of this expression, and use (4.14), we have that

$$\frac{1}{A_{\ell' m'}} \frac{Y_{\ell'}^{m'}(\alpha, \beta)}{\rho^{\ell'+1}} = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} r'^{\ell}\, Y_{\ell}^{-m}(\theta', \varphi')\, A_{\ell m}\, \partial_+^{m'}\, \partial_{\mathrm{sgn}(m)}^{|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right). \tag{4.20}$$

Now, note that for the terms in the previous sum corresponding to those with $m \geq 0$, we have that

$$\begin{aligned} \partial_+^{m'}\, \partial_{\mathrm{sgn}(m)}^{|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right) &= \partial_+^{m'+m}\, \partial_z^{\ell+\ell'-(m+m')} \left(\frac{1}{r}\right) \\ &= A_{(\ell+\ell')(m+m')} \frac{Y_{\ell+\ell'}^{m+m'}(\theta, \varphi)}{r^{\ell+\ell'+1}}. \end{aligned} \tag{4.21}$$

For those terms in (4.20) with $m < 0$, i.e. $\mathrm{sgn}(m) = -1$,

$$\partial_+^{m'}\, \partial_{\mathrm{sgn}(m)}^{|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right) = \partial_+^{m'}\, \partial_-^{|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right). \tag{4.22}$$

Now, since $1/r$ is a harmonic function, and if $m' \geq |m|$, then we can eliminate the $\partial_-^{|m|}$ term by using (4.17):

$$\begin{aligned} \partial_+^{m'}\, \partial_-^{|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right) &= \partial_+^{m'-|m|}\, \partial_+^{|m|}\, \partial_-^{|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right) \\ &= \partial_+^{m'-|m|}\, (-1)^{|m|}\, \partial_z^{2|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right) \\ &= (-1)^{|m|}\, \partial_+^{m'-|m|}\, \partial_z^{\ell+\ell'+|m|-m'} \left(\frac{1}{r}\right) \\ &= (-1)^{|m|}\, \partial_+^{m'+m}\, \partial_z^{\ell+\ell'-(m'+m)} \left(\frac{1}{r}\right) \\ &= (-1)^{|m|}\, A_{(\ell+\ell')(m+m')} \frac{Y_{\ell+\ell'}^{m+m'}(\theta, \varphi)}{r^{\ell+\ell'+1}}. \end{aligned} \tag{4.23}$$

Likewise, if $m' < |m|$, then similarly the term $\partial_+^{m'}$ can be eliminated, giving:

$$\partial_+^{m'}\, \partial_-^{|m|}\, \partial_z^{\ell+\ell'-|m|-m'} \left(\frac{1}{r}\right) = (-1)^{m'}\, A_{(\ell+\ell')(m+m')} \frac{Y_{\ell+\ell'}^{m+m'}(\theta, \varphi)}{r^{\ell+\ell'+1}}. \tag{4.24}$$

Defining the functions

$$J_+^{M2M}(m', m) = \begin{cases} 1, & \text{if } m \geq 0, \\ (-1)^{\min(m', |m|)}, & \text{if } m < 0, \end{cases} \tag{4.25}$$

and

$$J_{-}^{M2M}(m',m) = \begin{cases} 1, & \text{if } m \leq 0, \\ (-1)^{\min(|m'|,m)}, & \text{if } m > 0, \end{cases} \tag{4.26}$$

we have that for $m' \geq 0$,

$$\frac{Y_{\ell'}^{m'}(\alpha,\beta)}{\rho^{\ell'+1}} = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \frac{J_{+}^{M2M}(m',m)\, A_{\ell' m'}\, A_{\ell m}\, r'^{\ell}\, Y_{\ell}^{-m}(\theta',\varphi')}{A_{(\ell+\ell')(m+m')}} \frac{Y_{\ell+\ell'}^{m+m'}(\theta,\varphi)}{r^{\ell+\ell'+1}}. \tag{4.27}$$

A virtually identical calculation shows that for any $m' < 0$,

$$\frac{Y_{\ell'}^{m'}(\alpha,\beta)}{\rho^{\ell'+1}} = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \frac{J_{-}^{M2M}(m',m)\, A_{\ell' m'}\, A_{\ell m}\, r'^{\ell}\, Y_{\ell}^{-m}(\theta',\varphi')}{A_{(\ell+\ell')(m+m')}} \frac{Y_{\ell+\ell'}^{m+m'}(\theta,\varphi)}{r^{\ell+\ell'+1}}. \tag{4.28}$$

The only difference between these addition formulas is the sign function $J_{\pm}^{M2M}$, which depends on the sign of m'. We will refer to formulas (4.27) and (4.28) as *the first addition formula* for solid harmonics, i.e. that which operates on the outgoing (or multipole) functions. Having derived the multipole-to-multipole translation operator directly, we can simplify the remaining discussion by defining scaled versions of the solid harmonics, which we will refer to as the *inner* and *outer* functions:

$$\begin{aligned} I_{\ell}^{m}(r,\theta,\varphi) &= i^{-|m|}\, A_{\ell m}\, r^{\ell}\, Y_{\ell}^{m}(\theta,\varphi), \\ O_{\ell}^{m}(r,\theta,\varphi) &= \frac{(-1)^{\ell}\, i^{|m|}}{A_{\ell m}} \frac{Y_{\ell}^{m}(\theta,\varphi)}{r^{\ell+1}}. \end{aligned} \tag{4.29}$$

In this basis, multipole expansion for $1/\|\boldsymbol{x}-\boldsymbol{x}'\|$ becomes

$$\frac{1}{\|\boldsymbol{x}-\boldsymbol{x}'\|} = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} (-1)^{\ell}\, I_{\ell}^{-m}(\boldsymbol{x}')\, O_{\ell}^{m}(\boldsymbol{x}), \tag{4.30}$$

where, of course we assume that $\|\boldsymbol{x}\| > \|\boldsymbol{x}'\|$. Likewise, the first addition formula for the outgoing solid harmonics is given by:

$$O_{\ell}^{m}(\boldsymbol{x}-\boldsymbol{x}') = \sum_{\ell'=0}^{\infty} \sum_{m'=-\ell'}^{\ell'} (-1)^{\ell'}\, I_{\ell'}^{-m'}(\boldsymbol{x}')\, O_{\ell+\ell'}^{m+m'}(\boldsymbol{x}), \tag{4.31}$$

Remark 1. It is worth pointing out that using the inner and outer functions I_{ℓ}^{m} and O_{ℓ}^{m} simplifies the resulting formulas considerably, however, they are not well-scaled functions. Each includes the $A_{\ell m}$ coefficients, which contain factorial terms. For this reason, it is often numerically more stable to deal with the standard solid harmonics functions when implementing any translation algorithm. The inner and outer functions are certainly preferable for formula derivation.

We can now state the multipole translation theorem precisely.

Theorem 2 (Multipole-to-multipole translation)**.** *Suppose that some collection of sources $\boldsymbol{x}_j'$ with strengths q_j are located inside the sphere of radius a centered at $\boldsymbol{x}'$. For targets $\boldsymbol{x}$ outside this sphere, the potential is given by the multipole expansion*

$$\phi(\boldsymbol{x}) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \frac{M_{\ell m}'}{\rho^{\ell+1}} Y_{\ell}^{m}(\alpha,\beta), \tag{4.32}$$

where the multipole coefficients $M'_{\ell m}$ are given as in (4.7)*, and* $(\rho, \alpha, \beta) = \boldsymbol{x} - \boldsymbol{x}'$*. Then, outside of the sphere centered at the origin with radius* $r' + a$*, the potential is given as*

$$\phi(\boldsymbol{x}) = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} \frac{M_{\ell m}}{r^{\ell+1}} Y_\ell^m(\theta, \varphi) \tag{4.33}$$

with multipole coefficients given as:

$$M_{\ell m} = \sum_{\ell'=0}^{\infty} \sum_{m=-\ell'}^{\ell'} \frac{M'_{\ell-\ell', m-m'} \, J^{M2M}(m', m-m') \, A_{\ell' m'} \, A_{\ell-\ell', m-m'} \, r'^{\ell'} \, Y_{\ell'}^{-m'}(\theta', \varphi')}{A_{\ell m)}} \tag{4.34}$$

and

$$J^{M2M}(j,k) = \begin{cases} (-1)^{\min(|j|,|k|)}, & \text{if } j \cdot k < 0, \\ 1, & \text{otherwise.} \end{cases} \tag{4.35}$$

Furthermore, if the multipole expansion centered at $\boldsymbol{x}'$ *satisfied the bound*

$$\left| \phi(\boldsymbol{x}) - \sum_{\ell=0}^{p} \sum_{m=-\ell}^{\ell} \frac{M'_{\ell m}}{\rho^{\ell+1}} Y_\ell^m(\alpha, \beta) \right| \leq \frac{\sum_j |q_j|}{\rho - a} \left(\frac{a}{\rho} \right)^{p+1} \tag{4.36}$$

then the translated multipole expansion, when evaluated outside the disk of radius $r' + a$*, satisfies the error estimate*

$$\left| \phi(\boldsymbol{x}) - \sum_{\ell=0}^{p} \sum_{m=-\ell}^{\ell} \frac{M_{\ell m}}{r^{\ell+1}} Y_\ell^m(\theta, \varphi) \right| \leq \frac{\sum_j |q_j|}{r - (r' + a)} \left(\frac{r' + a}{r} \right)^{p+1}. \tag{4.37}$$

Proof. The proof of this is straightforward from the first addition theorem for multipole expansions. The bound is obtained directly from the fact that multipole expansions are unique, and therefore the translated expansion must be the unique expansion about $\mathbf{0}$. □

We will refer to the linear operator which maps the multipole coefficients $M'_{\ell m}$ to the multipole coefficients $M_{\ell m}$ as $\mathcal{T}^{M2M}$, which when viewed as a second-order tensor (properly ordered matrix) has elements equal to

$$T^{M2M}_{\ell m, \ell' m'} = \tag{4.38}$$

Computational cost

The computational cost of the high-order tree-code described in the previous section, based on direct multipole-to-multipole translation operators is straightforward to derive. While the asymptotic cost of the upward-pass and evaluation stay the same as for the original Barnes-Hut algorithm, $O(n)$ and $O(n \log n)$, respectively, we can also include an explicit dependence on p, the order of the multipole expansion used.

During the upward-pass stage of the high-order tree-code, the additional work that each box must perform (except for the leaf level) is translating and combining each of the p^{th}-order multipole expansions from each of its children into a single p^{th}-order expansion about its center. As we saw in the previous section, the direct multipole-to-multipole translation operator is

a dense, linear, convolution-type operator and therefore costs $O(p^4)$ operations to translate a single p-term expansion. This is akin to multiplication by a dense $p^2 \times p^2$ matrix. Since each box has eight children, this brings the cost of the upward pass up to $O(8p^4 n)$. We shall see in later sections that bringing the cost of these translations down from $O(p^4)$ is one of the main accelerations that can be made.

An order p multipole expansion includes $(p+1)^2 = O(p^2)$ terms, and assuming that each term can be numerically calculated using a fixed number of floating point operations, the cost of evaluating a p^{th} order expansion is $O(p^2)$. Therefore, the cost of the evaluation stage of a high-order tree-code is $O(p^2\, n \log n)$.

Error analysis

As mentioned before, the high-order tree-code described earlier does not depend on a parameter like θ in the original Barnes-Hut algorithm that determined when a box contributes to the potential at a target on a given level. Instead, all interactions are determined via the oct-tree structure (Figure **??**). The accuracy of the resulting algorithm is determined by the order of the expansion p. Using the fact that boxes only interact if they are well-separated, and the error bound on multipole expansions, we see that $p \sim O(\log_{\sqrt{3}} 1/\epsilon)$. This number of terms can be derived based on the separation distance between a sphere enclosing a box, and the nearest box that is well-separated.

Implementation details

Despite the simplicity of the previous multipole and translation formulas, constructing an accurate and efficient numerical implementation can seem daunting. Here we discuss a couple aspects of the implementation of tree-codes.

Building the oct-tree:

Numerical computation of Legendre functions: The Legendre polynomials satisfy the three-term recurrence relation:

$$P_{\ell+1}(x) = \frac{2\ell+1}{\ell+1} x\, P_\ell(x) - \frac{\ell}{\ell+1} P_{\ell-1}(x), \tag{4.39}$$

with

$$P_0(x) = 1, \quad \text{and} \quad P_1(x) = x. \tag{4.40}$$

For $x \in (-1, 1)$, this recurrence is *upward-stable* [**?**]. Furthermore the associate Legendre functions satisfy:

$$P_0^0(x) = 1, \quad P_1^0(x) = x, \quad \text{and} \quad P_1^1(x) = -\sqrt{1-x^2}, \tag{4.41}$$

and

$$P_\ell^{-m}(x) = (-1)^m. \tag{4.42}$$

These function obey the following (relatively) stable three-term recurrence in the *order m*:

$$P_\ell^{m+1} = \tag{4.43}$$

4.2 The 3D Laplace fast multipole method

At this point in the development of fast n-body codes, there are two routes to take: the reduction of the $O(p^4)$ cost for multipole translation, and the reduction of the overall complexity from $O(n \log n)$ to $O(n)$. We will first discuss the reduction in the overall scaling with n, and then address the scaling with the expansion order p (since it is a dominant cost, independent of the scaling with n). As before, we first give an informal description of the algorithm known as the Fast Multipole Method (FMM) for the 3D Laplace kernel, and then work through the necessary pieces of machinery needed to implement the algorithm. The main additional piece of technology that allows for a reduction in the asymptotic complexity are what are known as *local expansions*. Local expansions allow for information contained in multipole expansions to be shipped more efficiently to boxes on finer levels in the tree hierarchy.

THE FAST MULTIPOLE METHOD

Step 0 Divide the computational box, containing all points, using an oct-tree such that there are at most n_{fine} points contained in each leaf node. Set all the ϕ_i's to zero, initialize the order of the multipole expansion to $p > 0$.

Step 1 Process all the leaf nodes. For each box B on the leaf level, L, form the order p multipole expansion due to the sources contained in B, centered about the center of box B.

Step 2 For levels $L-1$ to 0, recursively compute the order p multipole expansion centered about the center of a box B by translating and merging the multipole expansions for the children of box B.

Step 3 For levels 2 through L, for each box B on each level, translate its multipole expansion to boxes in List 2 (also known as box B's interaction list).

Step 4 For each target particle $\boldsymbol{x}_i = (r_i, \theta_i, \phi_i)$, starting at the root, traverse the tree toward the leafs, processing one box B (that is not marked **done**) at a time. Let:

$$\begin{aligned} r_B &= \text{the radius of box } B \\ \boldsymbol{x}_B &= \text{the center of box } B \\ M_{\ell m}^B &= \text{the multipole coefficients for box } B \\ r_{B,i} &= \|\boldsymbol{x}_b - \boldsymbol{x}_i\|, \text{ the distance from } \boldsymbol{x}_B \text{ to } \boldsymbol{x}_i \\ (\rho, \alpha, \beta) &= \boldsymbol{x}_i - \boldsymbol{x}_B, \text{ the coordinates of } \boldsymbol{x}_i \text{ relative to } \boldsymbol{x}_B \end{aligned}$$

If box B is *well-separated* from the target $\boldsymbol{x}_i$, mark this box as **done** and increment ϕ_i:

$$\phi_i = \phi_i + \sum_{\ell=0}^{p} \sum_{m=-\ell}^{\ell} \frac{M_{\ell m}^B}{\rho^{\ell+1}} Y_\ell^m(\alpha, \beta)$$

By well-separated, we mean that the target $\boldsymbol{x}_i$ is separated from box B by at least one box of the same size as B. See Figure **??**. If $\boldsymbol{x}_i$ is not well-separated, then skip box B and process its children using the same procedure. Repeat for targets $\boldsymbol{x}_i$.

Step 4 For any boxes B on the leaf level that were not marked **done**, add the contributions from the sources $\boldsymbol{x}_j$ located in them directly:

$$\phi_i = \phi_i + \frac{m_j}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}.$$

The potential ϕ_i, due to all sources other than $\boldsymbol{x}_i$, has now been computed.

Computational cost

Error analysis

4.2.1 Point and shoot translation

As we have seen several times before, the main computational cost in the FMM (as it is formulated here) is the multipole-to-local translations. Done directly, this

4.2.2 Diagonal plane-wave translation operators

Using the point-and-shoot method of the previous section, it is possible to reduce the cost of the multipole and local translations from $O(p^4)$ to $O(p^3)$ by first rotating the coordinate system, translating along the z-axis, and the rotating back. However, due to the anisotropy of spherical coordinate systems, there is still coupling between the θ and φ components in the rotation. In the case of multipole-to-local translations, it is possible to reduce this cost further, down to $O(p^2)$, by switching to a *different* representation of the potential in Cartesian coordinates: plane-wave representations.

By Fourier transforming Laplace's equation for the Green's function

$$\triangle G(\boldsymbol{x}) = \delta(\boldsymbol{x}), \tag{4.44}$$

we see that

$$\widehat{G}(\boldsymbol{\xi}) = \tag{4.45}$$

and then performing contour integration in the ξ_3 variable, it is possible to write:

$$\frac{1}{\|\boldsymbol{x} - \boldsymbol{x}'\|} = \tag{4.46}$$

4.2.3 Accelerations

5 The Helmholtz equation

6 Electromagnetics

7 Elasticity

8 Fluids

9 Parabolic theory

10 Fast multipole methods

10.1 The 3D fast multipole method for Helmholtz potentials

10.1.1 Addition formulas

10.1.2 Exponential translations

10.1.3 Implementation caveats

10.2 3D Stokesian potentials

11 Discrete Fourier Transforms

Entire courses could be taught only covering Discrete Fourier Analysis and its applications in methods for differential equations, signal processing, etc. It is merely our goal to cover some basic, very useful properties of the Discrete Fourier Tranform (denote by DFT from here on out), and then to discuss algorithms for applying the operator rapidly – both in the standard case of the widely know Fast Fourier Transform (FFT), and in the case where the sample in time and modes in frequency are *not* equispaced or integers, respectively.

In these notes, we define the n-point *forward* DFT to be the finite-dimensional linear operator $\mathbf{F}_n : \mathbb{C}^n \to \mathbb{C}^n$ defined by:

$$\begin{aligned} \widehat{\mathbf{u}} &= \mathbf{F}_n \mathbf{u} \\ \widehat{u}_k &= \sum_{j=1}^{n} u_j \, e^{-2\pi i (j-1)(k-1)/n} \end{aligned} \tag{11.1}$$

Likewise, it is easy to show that the inverse of $\mathbf{F}_n$ is given by $\mathbf{F}_n^{-1} = \frac{1}{n}\mathbf{F}_n^*$, where $\mathbf{F}_n^*$ is the adjoint matrix to $\mathbf{F}_n$, also referred to as the *backward* DFT. Therefore, in infinite precision arithmetic

$$\mathbf{u} = \frac{1}{n}\mathbf{F}_n^* \mathbf{F}_n \mathbf{u}. \tag{11.2}$$

Keep in mind that there are *many* other definitions of the DFT, including almost all permutations of $\pm 2\pi i$ in the exponent, scaling by $1/n$ or $1/\sqrt{n}$, summing over $-n/2 + 1$ to $n/2$, etc. We stick with the above definition of $\mathbf{F}_n$ as it is consistent with almost all FFT software packages.

Quadrature and aliasing

The DFT can also be though of as a discretization of a series of Fourier integrals using the trapezoidal rule:

$$\begin{aligned} \widehat{f}(k) &= \frac{1}{2\pi} \int_0^{2\pi} f(x) \, e^{-ixk} \, dx \\ &= \int_0^1 f(2\pi x) \, e^{-2\pi i x k} \, dx \\ &\approx \frac{1}{n} \sum_{j=1}^{n} f\left(\frac{2\pi(j-1)}{n}\right) \, e^{-2\pi i (j-1) k/n}, \end{aligned} \tag{11.3}$$

where we assume that k is an integer. The approximation of the previous integral is in fact *exact* for functions of the form

$$f(x) = \sum_{k=-m}^{m} \alpha_k \, e^{ixk}, \tag{11.4}$$

with $n > 2m$. The integer m (or $2\pi m$ in some references) is known as the *bandwidth* of the function f.

If the size of the quadrature rule n (or, it will turn out, the size of the FFT that we compute) is not sufficiently larger, then we will observe what is known as *aliasing*. When sampled discretely on an equispaced grid in x, exponentials $e^{2\pi ikx}$ will *alias* to exponentials with a smaller integer frequency $k \in (-n/2, n/2)$. This is because these functions have exactly the same values on under-resolved grids. This is the whole intuition behind the Nyquist sampling rate. Two sample points per wavelength are able to uniquely resolve the function.

11.1 The Fast Fourier Transform

It is useful to define the factor

$$\omega_n = e^{2\pi i/n}. \tag{11.5}$$

This factor, as well as $\omega_{n/2}$, etc., will be the basic building block for what is known as the radix-2 split FFT. Assuming that the $n = 2^d$, for some positive integer d, we describe one level of the FFT and leave the rest as a straightforward exercise in induction.

Written out as in equation (11.1), the DFT is given, for $k = 0, \ldots, n-1$ by:

$$\begin{aligned} \widehat{u}_k &= \sum_{j=0}^{n-1} u_j \, e^{-2\pi ijk/n} \\ &= \sum_{j=0}^{n-1} u_j \, \omega_n^{-jk}. \end{aligned} \tag{11.6}$$

This sum can then be split into two pieces – those corresponding to *even* indices j and those corresponding to odd indices j – and properly re-ordered into *two* DFTs of size $n/2$:

$$\begin{aligned} \widehat{u}_k &= \sum_{j=0}^{n/2-1} u_{2j} \, e^{-2\pi i(2j)k/n} \; + \; \sum_{j=0}^{n/2-1} u_{2j+1} \, e^{-2\pi i(2j+1)k/n} \\ &= \sum_{j=0}^{n/2-1} u_{2j} \, e^{-2\pi ijk/(n/2)} \; + \; e^{-2\pi ik/n} \sum_{j=0}^{n/2-1} u_{2j+1} \, e^{-2\pi ijk/(n/2)} \\ &= \sum_{j=0}^{n/2-1} u_{2j} \, \omega_{n/2}^{-jk} \; + \; \omega_n^{-k} \sum_{j=0}^{n/2-1} u_{2j+1} \, \omega_{n/2}^{-jk} \end{aligned} \tag{11.7}$$

Introducing a new frequency variable, $\tilde{k} = 0, \ldots, n/2-1$, we have that:

$$\begin{aligned} \widehat{u}_{\tilde{k}} &= \sum_{j=0}^{n/2-1} u_{2j}\, \omega_{n/2}^{-j\tilde{k}} + \omega_n^{-\tilde{k}} \sum_{j=0}^{n/2-1} u_{2j+1}\, \omega_{n/2}^{-j\tilde{k}} \\ \widehat{u}_{n/2+\tilde{k}} &= \sum_{j=0}^{n/2-1} u_{2j}\, \omega_{n/2}^{-j\tilde{k}} + \omega_n^{-n/2-\tilde{k}} \sum_{j=0}^{n/2-1} u_{2j+1}\, \omega_{n/2}^{-j\tilde{k}}. \end{aligned} \tag{11.8}$$

There is one final step in simplifying these relationships, which hinges on the identity that $\omega_n^{n/2} = e^{\pi i} = -1$. We now have:

$$\begin{aligned} \widehat{u}_{\tilde{k}} &= \sum_{j=0}^{n/2-1} u_{2j}\, \omega_{n/2}^{-j\tilde{k}} + \omega_n^{-\tilde{k}} \sum_{j=0}^{n/2-1} u_{2j+1}\, \omega_{n/2}^{-j\tilde{k}} \\ \widehat{u}_{n/2+\tilde{k}} &= \sum_{j=0}^{n/2-1} u_{2j}\, \omega_{n/2}^{-j\tilde{k}} - \omega_n^{-\tilde{k}} \sum_{j=0}^{n/2-1} u_{2j+1}\, \omega_{n/2}^{-j\tilde{k}}. \end{aligned} \tag{11.9}$$

Each of these four sums can then be split again, into their relatively odd and even parts. The resulting scheme is the FFT. Efficiently implementing the FFT involves an a priori reordering of the coefficients u_j so that neighboring coefficients in memory contribute to the same sums. The particular re-ordering required is a *bit-reversal* re-ordering, as u_j will be permuted to location $\bar{j}$, where $\bar{j}$ is the integer obtained by reversing the binary representation of the integer j.

Computational cost

It is easy to show that when $n = 2^d$, the DFT sum can be split $d = \log_2 n$ times. The resulting coefficients $\widehat{u}_k$ are then each the sum of d terms. This gives an overall running time of $O(n \log_2 n)$, and the intricate details are omitted [**?**].

Accuracy

Unlike virtually every other algorithm that is discussed in these lecture notes, the FFT is an *exact* algorithm when performed in infinite precision. There are no approximations made in the algorithm, and the only source of error in the scheme arises from floating point arithmetic (and is virtually non-existent for all practical purposes because the number of operations done per output is $\log_2 n$). A precise discussion of this error is beyond the scope of these notes [**?**].

General radix schemes

The FFT is generally fastest when applied to vectors whose length is the product of many small primes because of the fine-scale splitting that can be accomplished. We previously described the most common case, when $n = 2^d$ for some positive integer d. If, for example, n is divisible

by 3, then one step of the FFT will contain a radix-3 split:

$$\sum_{j=0}^{n-1} u_j\, e^{-2\pi ijk/n} = \sum_{j=0}^{n/3-1} u_{3j}\, e^{-2\pi ijk/(n/3)} + e^{-2\pi ik/(n/3)} \sum_{j=0}^{n/3-1} u_{3j+1}\, e^{-2\pi ijk/(n/3)} + e^{-2\pi ik/(2n/3)} \sum_{j=0}^{n/3-1} u_{3j+2}\, e^{-2\pi ijk/(n/3)}, \quad (11.10)$$

or using the previous notation,

$$\sum_{j=0}^{n-1} u_j\, \omega_n^{-jk} = \sum_{j=0}^{n/3-1} u_{3j}\, \omega_{n/3}^{-jk} + \omega_{n/3}^{-1} \sum_{j=0}^{n/3-1} u_{3j+1}\, \omega_{n/3}^{-jk} + \omega_{n/3}^{-2} \sum_{j=0}^{n/3-1} u_{3j+2}\, \omega_{n/3}^{-jk}. \quad (11.11)$$

There are several methods for FFTs on vectors whose length is a large prime number, including incremental and absolute zero-padding, etc. These are beyond the scope of this discussion, as well as efficient methods for implementing FFTs. See resources related to the software package `fftw`.

11.1.1 As a matrix factorization

An extremely powerful interpretation of the FFT is that presented by Van Loan in [**?**] as a recursive matrix factorization. We briefly discuss this interpretation here, as it is often overlooked and very insightful. To simplify the following discussion of the length-n FFT in factorized matrix form, let us first define the number

$$\omega_n = e^{2\pi i/n}. \quad (11.12)$$

Then the DFT can be written as

$$\widehat{u}_j = \sum_{j=0}^{n-1} u_j\, \omega^{-jk}, \quad (11.13)$$

for $k = 0, \ldots, n-1$. Note here that for the case of simplicity, we have defined the DFT using slightly different indicies than in (11.1). Under this notation, the length-4 DFT matrix $\mathbf{F}_4$ can be written as:

$$\mathbf{F}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix}. \quad (11.14)$$

Our goal will be to write this matrix as the product of $\log_2 n$ sparse matrices, each of size $n \times n$. First, we examine this factorization in detail in the case of the length-4 transform.

As described earlier, we can use the radix-2 splitting algorithm to write the length-4 FFT as the sum of two length-2 FFTs:

$$\mathbf{F}_4\mathbf{u} = \begin{bmatrix} \mathbf{F}_2\mathbf{u}_e + \Omega_2\mathbf{F}_2\mathbf{u}_o \\ \mathbf{F}_2\mathbf{u}_e - \Omega_2\mathbf{F}_2\mathbf{u}_o \end{bmatrix}, \quad (11.15)$$

where the matrix $\dot{}_m$ is the diagonal matrix:

$$\dot{}_m = \begin{bmatrix} 1 & & & \\ & \omega_{2m}^{-1} & & \\ & & \ddots & \\ & & & \omega_{2m}^{-(m-1)} \end{bmatrix}. \tag{11.16}$$

The relation inside the matrix in (11.15) is known as a *butterfly relation*. Explicitly expanding (11.15) we have:

$$\mathbf{F}_4\mathbf{u} = \begin{bmatrix} \mathbf{F}_2 & \dot{}_2\mathbf{F}_2 \\ \mathbf{F}_2 & -\dot{}_2\mathbf{F}_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_e \\ \mathbf{u}_o \end{bmatrix}. \tag{11.17}$$

We can play the same trick again with the matrices $\mathbf{F}_2$:

$$\mathbf{F}_2\mathbf{v} = \begin{bmatrix} \mathbf{F}_1 & \dot{}_1\mathbf{F}_1 \\ \mathbf{F}_1 & -\dot{}_1\mathbf{F}_1 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}. \tag{11.18}$$

Since $\mathbf{F}_1 = 1$ and $\Omega_1 =$, we are done. Combing these two previous formulas, we have that:

$$\mathbf{F}_4\mathbf{u} = \begin{bmatrix} \mathbf{I}_2 & \dot{}_2 \\ \mathbf{I}_2 & -\dot{}_2 \end{bmatrix} \begin{bmatrix} \mathbf{I}_1 & \dot{}_1 & & \\ \mathbf{I}_1 & -\dot{}_1 & & \\ & & \mathbf{I}_1 & \dot{}_1 \\ & & \mathbf{I}_1 & -\dot{}_1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & & 1 & \\ & 1 & & \\ & & & 1 \end{bmatrix} \mathbf{u}, \tag{11.19}$$

where, as is standard, the matrix $mtxI_m$ is the $m \times m$ identity matrix. Performing a similar factorization for larger size FFTs quickly becomes unmanageable, and therefore it becomes very useful to denote by the *size-m butterfly matrix* $\mathbf{B}_m$:

$$\mathbf{B}_m = \begin{bmatrix} \mathbf{I}_{m/2} & \dot{}_{m/2} \\ \mathbf{I}_{m/2} & -\dot{}_{m/2} \end{bmatrix}. \tag{11.20}$$

Furthermore, the *Kronecker product* of two matrices $\mathbf{A}$ and $\mathbf{B}$ will be given as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{n1}\mathbf{B} & \cdots & a_{nn}\mathbf{B} \end{bmatrix} \tag{11.21}$$

Obviously, if $\mathbf{A} \in \mathbb{C}^{n\times n}$ and $\mathbf{B} \in \mathbb{C}^{m\times m}$, then $\mathbf{A} \otimes \mathbf{B} \in \mathbb{C}^{(n+m)\times(n+m)}$. See [**?**] for a thorough discussion of this operation, its properties, and its usefulness. Lastly, the *bit-reversal* matrix $\mathbf{W}_n$ is given as ...

Combining all these notations together, we have

$$\mathbf{F}_4 = (\mathbf{I}_1 \otimes \mathbf{B}_4)(\mathbf{I}_2 \otimes \mathbf{B}_2)\mathbf{W}_n. \tag{11.22}$$

Based on the sparsity pattern of the above factorization, it is obvious that $\mathbf{F}_4$ can be applied in $O(n \log_2 n)$ operations (assuming, of course, that sparse matrix-vector products are performed). In general, for a DFT of size $n = 2^d$ we have:

$$\mathbf{F}_n = (\mathbf{I}_1 \otimes \mathbf{B}_n)\left(\mathbf{I}_2 \otimes \mathbf{B}_{n/2}\right)\left(\mathbf{I}_4 \otimes \mathbf{B}_{n/4}\right)\cdots\left(\mathbf{I}_{n/2} \otimes \mathbf{B}_2\right)\mathbf{W}_n. \tag{11.23}$$

This is a very powerful framework in which to view the FFT. Similar factorizations for other radix FFTs exist, but the notation becomes slightly more complicated.

11.2 The non-uniform FFT

The entire previous discussion of the FFT was predicated on the fact that we are computing inner products of the vector $\mathbf{u}$ with *equi-spaced* samples in $x \in [0, 2\pi]$ of the basis functions $e^{2\pi i x k}$, with k an integer. We now address the problem of computing similar sums when the samples in space *and/or* frequency are not equispaced (or integers). To this end, let's begin by stating the precise problem, which can be split into three distinct cases. In what follows, let $x_1, \ldots, x_n \in [0, 2\pi]$ be arbitrary points.

The **Type I DFT** is given by:

$$F(k) = \sum_{j=1}^{n} f_j \, e^{-ikx_j}, \quad \text{for } k = -\frac{m}{2}, \ldots, \frac{m}{2}. \tag{11.24}$$

The **Type II DFT** is given by:

$$f(x_j) = \sum_{k=-m/2}^{m/2} F_k \, e^{ikx_j}, \quad \text{for } j = 1, \ldots, n. \tag{11.25}$$

The **Type III DFT** is given by:

$$\varphi_k = \sum_{j=1}^{n} u_j \, e^{\pm i s_k x_j}, \quad \text{for } k = 1, \ldots, m, \tag{11.26}$$

with $s_k \in \mathbb{R}$ arbitrary. In some cases, these sums correspond to the evaluation of an expansion in complex exponentials (Type II), or discretization of an integral (Type I). The problems of efficiently discretizing Fourier integrals and evaluating the above *discrete* sums should not be conflated – the sums above do not hold any physical meaning unless attached to an outer problem (as in the case of FMMs, etc.). We merely address the task of efficiently computing these sums in near linear time.

We have listed the one-dimensional versions of each of these schemes, but they are readily extended to sums based on higher dimensional frequency vectors $\boldsymbol{k}$ and spatial locations $\boldsymbol{x}_j$. For example, in two dimensions, the Type I DFT is defined as:

$$F(\boldsymbol{k}) = \sum_{j=1}^{n} f_j \, e^{-i\boldsymbol{k}\cdot\boldsymbol{x}_j}, \quad \text{for } \boldsymbol{k} = (-m/2, -m/2), \ldots, (m/2, m/2). \tag{11.27}$$

The algorithms which allow for the rapid computation of these sums are of a different nature than those upon which FMMs are designed. FMMs exist because after separating *near-field* from *far-field* calculations, it can be shown that the far-field part is a low-rank operation. Using this idea hierarchically, one can build a fast algorithm.

Similar addition formulas for complex exponentials exist, for example:

$$e^{i\rho t} = J_0(\rho) + \sum_{n=1}^{\infty} 2i^n \, J_n(\rho) \, T_n(t), \tag{11.28}$$

where J_n is the Bessel function of order n, T_n is the degree n Chebyshev polynomial and $t \in [-1, 1]$. However, this addition formula does *not* directly yield a fast algorithm because there

is no obvious low-rank observation. We will discuss other FFT-like fast algorithms in the next chapter, where we develop what are known as *butterfly algorithms*.

The schemes described below are a combination of interpolation, application of standard FFT methods, and rigorous error analysis. We will now discuss an algorithm for the Type I NUFFT.

11.2.1 The Type I NUFFT

The Type I NUFFT corresponds to evaluating the sums:

$$F(k) = \sum_{j=1}^{n} f_j \, e^{-ikx_j}, \quad \text{for } k = -\frac{m}{2}, \ldots, \frac{m}{2}. \tag{11.29}$$

We first describe an algorithm, and then analyze its accuracy and efficiency.

THE TYPE I NUFFT

Step 0 Choose absolute error tolerance ϵ, set width of convolution kernel to be $\tau = \tau(\epsilon)$ (see accuracy discussion below).

Step 1 Compute the Fourier coefficients

$$\widehat{f_\tau}(k) = \frac{1}{2\pi} \int_0^{2\pi} f_\tau(x) \, e^{-ikx} \, dx \tag{11.30}$$

of the function

$$f_\tau(x) = \sum_{j=1}^{n} f_j \sum_{\ell=-\infty}^{\infty} e^{-(x-x_j-2\ell\pi)^2/4\tau}. \tag{11.31}$$

This can be done using a sufficiently large FFT.

Step 2 Evaluate F as

$$F(k) = \sqrt{\frac{\pi}{\tau}} \, e^{k^2\tau} \, \widehat{f_\tau}(k). \tag{11.32}$$

The previous algorithm is very straightforward, all that remains is to prove some identities, error bounds, and choices for the parameters τ and the size of the FFT to take of f_τ. First, we explain the algorithm in some more detail.

First, we note that $F(k)$ in (11.29) is the *exact* Fourier coefficient of the 2π-periodic function:

$$f(x) = 2\pi \sum_{\ell=-\infty}^{\infty} \sum_{j=1}^{n} f_j \, \delta(x - x_j - 2\pi\ell) \tag{11.33}$$

That is, we have that

$$\begin{aligned} F(k) &= \frac{1}{2\pi}\int_0^{2\pi} f(x)\, e^{-ikx}\, dx \\ &= \sum_{j=1}^{n}\int_0^{2\pi} f_j\, \delta(x-x_j)\, dx \\ &= \sum_{j=1}^{n} f_j\, e^{-ikx_j} \end{aligned} \tag{11.34}$$

If the function f was well-resolved by equispaced sampling on $[0, 2\pi]$, then these coefficients could be directly computed (as in the case of smooth periodic functions on this interval). Unfortunately, no number of equispaced points would resolve the δ-functions. In order to take advantage of this fact, we first convolve f with a smooth kernel function (the Gaussian), and then scale the resulting Fourier coefficients (deconvolve).

Define the kernel function g_τ by:

$$g_\tau(x) = e^{-x^2/4\tau}, \tag{11.35}$$

and the convolution over the entire real line f_τ as:

$$\begin{aligned} f_\tau(x) &= (f * g_t)(x) \\ &= \int_{-\infty}^{\infty} f(y)\, g_\tau(x-y)\, dy. \end{aligned} \tag{11.36}$$

It is easy to confirm that f_τ is a smooth, C^∞ function with period 2π (due to the smoothness of g_τ and the periodicity of the function f). The exact Fourier coefficients of f_τ are then given by $\widehat{f_\tau}$:

$$\widehat{f_\tau}(k) = \frac{1}{2\pi}\int_0^{2\pi} f_\tau(x)\, e^{-ikx}\, dx. \tag{11.37}$$

It is obvious that if the coefficients $\widehat{f_\tau}$ were known exactly, the using then properties of the Fourier transform,

$$\widehat{f_\tau}(k) = F(k)\cdot\widehat{g_\tau}(k) \qquad \Longleftrightarrow \qquad F(k) = \frac{\widehat{f_\tau}(k)}{\widehat{g_\tau}(k)}, \tag{11.38}$$

where, as before, F is the exact Fourier transform of f in (11.33). The computational task, then, is to efficiently evaluate the terms $\widehat{f_\tau}(k)$, since the Fourier transform of g_τ is known analytically:

$$\widehat{g_\tau}(k) = \sqrt{\frac{\tau}{\pi}}e^{-k^2\tau}. \tag{11.39}$$

These integrals defining $\widehat{f_\tau}$ can be evaluated with spectral accuracy using an N-point trapezoidal rule, where N is chosen in order to sufficiently resolve Gaussians placed at each of the locations x_j. That is to say, $N \sim O(1/\sqrt{\tau})$. Details of this analysis follow.

Accuracy

The formula for computing $F(k)$ is given in equation (11.38), and the accuracy is completely controlled by the evaluation of $\widehat{f_\tau}(k)$ since $\widehat{g_\tau}(k)$ is known analytically. If a maximum $k = m/2$ is needed in equation (11.37), then at least $N = 2m$ points are needed in its discretization (to resolve the $m/2$ mode in f_τ and in the exponential). Note that the formula in (11.38) is exact – that is, it is true for *any* value of τ. It is merely a point of computational complexity and accuracy to choose τ.

Computational cost

Here we briefly detail the computational cost, which if we assume $m \sim n$, is $O(m \log m + n)$.

Step 1: This step consists of Gaussian gridding, followed by an FFT of length $O(m)$.

Step 2: Evaluating $F(k)$ is merely a diagonal scaling of $\widehat{f_\tau}$, and costs $O(m)$ operations.

Caveats

Here we briefly discuss regimes in which the previous scheme does *not* yield a sufficiently fast or efficient algorithm.

11.2.2 The Type II NUFFT

Accuracy

11.2.3 The Type III NUFFT

Accuracy

12 Butterfly algorithms

12.1 Analogy with the FFT

12.2 Applications

12.3 An algorithm

12.4 Applications

13 Randomized linear algebra

14 Fast direct solvers

14.1 Nested dissection

14.2 Hierarchically structured matrices

A Green's Identities

All three Green's identities are a direct consequence of the Divergence Theorem (valid in any dimension):

$$\int_\Omega \nabla \cdot \boldsymbol{F}\, dV = \int_\Gamma \boldsymbol{n} \cdot \boldsymbol{F}\, dA, \tag{A.1}$$

where Ω is a bounded (assumed to be open) region with boundary Γ, and $\boldsymbol{F}$ is a smooth vectorfield. Green's First Identity can be derived by setting $\boldsymbol{F} = u\nabla v$, where u and v are smooth functions defined on the closure of Ω:

$$\int_\Omega (\nabla u \cdot \nabla v + u\Delta v)\, dV = \int_\Gamma u \frac{\partial v}{\partial n}\, dA, \tag{A.2}$$

where as always, $\partial/\partial n$ denoted differentiation in the direction outward from Ω.

Green's Second Identity is derived by setting $\boldsymbol{F} = v\nabla u$, and subtracting the resulting equation from that in (A.2):

$$\int_\Omega (u\Delta v - v\Delta u)\, dV = \int_\Gamma \left(u\frac{\partial v}{\partial n} - v\frac{\partial u}{\partial n} \right) dA. \tag{A.3}$$

Finally, Green's Third Identity is obtained by setting $v(\boldsymbol{x}') = g(\boldsymbol{x}, \boldsymbol{x}')$, where g is the Green's function for the Laplace operator, and integrating in the $\boldsymbol{x}'$ variable:

$$\begin{aligned} u &= \int_\Omega g(\cdot, \boldsymbol{x}')\, \Delta u(\boldsymbol{x}')\, dV(\boldsymbol{x}') + \int_\Gamma \left(u(\boldsymbol{x}') \frac{\partial g(\cdot, \boldsymbol{x}')}{\partial n'} - g(\cdot, \boldsymbol{x}') \frac{\partial u(\boldsymbol{x}')}{\partial n'} \right) dA(\boldsymbol{x}') \\ &= \mathcal{V}\Delta u + \mathcal{D}u - \mathcal{S}\frac{\partial u}{\partial n}. \end{aligned} \tag{A.4}$$

If u happens to be a harmonic function, this identity reduces to what is known as Green's Reproducing Identity:

$$u = \mathcal{D}u - \mathcal{S}\frac{\partial u}{\partial n}. \tag{A.5}$$

For values of $\boldsymbol{x}$ on the boundary of Ω, the limiting form of the above reproducing formula is:

$$u = \pm\frac{1}{2}u + \mathcal{D}u - \mathcal{S}\frac{\partial u}{\partial n}, \tag{A.6}$$

where the sign depends on which side the limit is taken from, and the integral operator $\mathcal{D}$ is implicitly interpreted in its principal value sense (even though it has a continuous kernel).

B Fredholm Theory

The standard treatments often paid to integral equation theory and Fredholm theory in most texts are needlessly complicated for most applications. In particular, most of the time for most applications (at least in mathematical physics), boundary integral equations of concern are being solving on curves in 2D or surfaces in 3D (usually smooth curves and surfaces, maybe with a few corners) and contain as kernels the standard single- and double-layer kernels corresponding to the appropriate Green's function. What this means, mathematically, is that almost all of the standard integral operators can be treated as operators acting from L^2 to L^2. This greatly simplifies much of the analysis, and has a direct correspondence to the numerical methods often used to solve them: iterative linear algebraic solvers, generalized Fourier series, etc. It is with this in mind that we provide the overview of what we will call *Fredholm theory*.

We first classify integral equations into one of two kinds. In the simplest case, let $\mathcal{K}$ be a compact integral operator acting on some curve in 2D or surface in 3D, denoted by Γ, with kernel k, integral equations of the first and second kind are then:

$$\begin{aligned} \text{First-kind:} \qquad & \int_\Gamma k(\boldsymbol{x}, \boldsymbol{x}')\, \varphi(\boldsymbol{x}')\, da(\boldsymbol{x}') = f(\boldsymbol{x}) \\ \text{Second-kind:} \qquad & \varphi(\boldsymbol{x}) + \int_\Gamma k(\boldsymbol{x}, \boldsymbol{x}')\, \varphi(\boldsymbol{x}')\, da(\boldsymbol{x}') = g(\boldsymbol{x}) \end{aligned} \tag{B.1}$$

A third class of integral equation exists, known as *Volterra equations*, and arise naturally in the theory of parabolic PDEs and the associated integral equations. We omit a discussion of these types for now.

Definition 1 (Weakly-singular integral operators)**.** The integral operator $\mathcal{K}$ with kernel $k(\boldsymbol{x}, \boldsymbol{y})$ is a *weakly-singular* integral operator on a curve in 2D if:

$$k(\boldsymbol{x}, \boldsymbol{y}) \leq C \frac{1}{|\boldsymbol{x} - \boldsymbol{y}|^\alpha}, \tag{B.2}$$

where $\alpha \in [0, 1)$. Along surfaces in 3D, the condition can be relaxed to $\alpha \in [0, 2)$ due to the change in metric (and therefore conditions on integrability).

It can be shown that weakly singular integral operators along smooth boundaries are compact operators acting from $L^2 \to L^2$. If $\mathcal{K}$ is a compact operator, then it can be approximated to any precision ϵ by a finite rank operator, $\mathcal{K}_n$:

$$||\mathcal{K} - \mathcal{K}_n||_2 \leq \epsilon \tag{B.3}$$

where n denotes the rank and depends on ϵ. The smallest n such that the above equation holds will be referred to as the ϵ-rank of the operator $\mathcal{K}$.

Theorem 3 (The Fredholm Alternative)**.** *For a compact integral operator* $\mathcal{K} : L^2 \to L^2$*,*

1. *the equation* $(\mathcal{I} + \mathcal{K})\sigma = f$ *has a solution for any* f *if and only if* $(\mathcal{I} + \mathcal{K})\sigma = 0$ *has only* $\sigma = 0$ *as a solution. If so, the operator* $(\mathcal{I} + \mathcal{K})^{-1}$ *is in fact bounded. Otherwise,*

2. *if* $(\mathcal{I} + \mathcal{K})\sigma = 0$ *has non-zero solutions, then* $\dim \text{Null}(\mathcal{I} + \mathcal{K}) = \dim \text{Null}(\mathcal{I} + \mathcal{K}^*)$*, and* $(\mathcal{I} + \mathcal{K})\sigma = f$ *is solvable if and only if* f *is orthogonal to* $\text{Null}(\mathcal{I} + \mathcal{K}^*)$*.*

Proof. We omit a proof, as various proofs can be found an any integral equations textbook or functional analysis textbook. □

C Iterative Solvers

Bibliography

[1] J. Bremer. On the Nyström discretization of integral equations on planar curves with corners. *Appl. Comput. Harm. Anal.*, 32:45–64, 2012.

[2] J. Bremer and Z. Gimbutas. A Nyström method for weakly singular integral operators on surfaces. *J. Comput. Phys.*, 231:4885–4903, 2012.