An Efficient and High-Order Accurate Boundary Integral Solver for the Stokes Equations in Three Dimensional Complex Geometries

by

Lexing Ying

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May 2004

Denis Zorin

Acknowledgments

This dissertation would not have been possible without the help and support from many people to whom I am greatly indebted.

First of all, I would like to thank my advisor Denis Zorin for his guidance, encouragement and collaboration. It is Denis who taught me how to do research and his influence will be timeless. He has demonstrated great devotion to the sciences and also great care for his students.

I am grateful to my collaborator George Biros for being a great teacher and a constant supporter.

I would like to thank Leslie Greengard, Michael Shelley and Olof Widlund for serving on my thesis committee and for providing valuable discussion and suggestion.

I wish to thank the people I have worked with at MRL for making a wonderful place to work. Special thanks to Henning Biermann, Eitan Grinspun, Aaron Hertzmann, Harper Langston, Jianbo Peng and Elif Tosun.

I would like to thank the faculty and staff of the computer science department for providing a great environment for academic research.

I am greatly indebted to Yi Zhou for her care, support and encouragement. Without her, my days as a graduate student would have been colorless and boring.

Finally, I want to thank my parents, Lizhen Ying and Lihua Wang, for their love, care and encouragement. They provide me with the best education possible. Without them, I would be nothing.

Abstract

This dissertation presents an efficient and high-order boundary integral solver for the Stokes equations in complex 3D geometries. The targeted applications of this solver are the flow problems in domains involving moving boundaries. In such problems, traditional finite element methods involving 3D unstructured mesh generation experience difficulties. Our solver uses the indirect boundary integral formulation and discretizes the equation using the Nyström method.

Although our solver is designed for the Stokes equations, we show that it can be generalized to other constant coefficient elliptic partial differential equations (PDEs) with non-oscillatory kernels.

First, we present a new geometric representation of the domain boundary. This scheme takes quadrilateral control meshes with arbitrary geometry and topology as input, and produces smooth surfaces approximating the control meshes. Our surfaces are parameterized over several overlapping charts through explicit nonsingular C^∞ parameterizations, depend linearly on the control points, have fixed-size local support for basis functions, and have good visual quality.

Second, we describe a kernel independent fast multipole method (FMM) and its parallel implementation. The main feature of our algorithm is that it is based only on kernel evaluation and does not require the multipole expansions of the underlying kernel. We have tested our method on kernels from a wide range of elliptic PDEs. Our numerical results indicate that our method is efficient and accurate. Other advantages include the simplicity of the implementation and its immediate extension to other elliptic PDE kernels. We also present an MPI based parallel implementation which scales well up to thousands of processors.

Third, we present several algorithms to evaluate the singular integrals in our solver. A singular integral is decomposed into a smooth far field part and a local part that contains the singularity. The smooth part of the integral is integrated using the trapezoidal rule over overlapping charts, and the singular part is integrated in the polar coordinates which removes or decreases the order of singularity. We also describe a new algorithm to integrate the nearly singular integrals coming from the evaluation at points close to the boundary.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

1.1 Motivation

The development of fast and robust numerical solvers for the boundary value problems of the elliptic partial differential equations (PDEs) on complex three dimensional domains has been one of the most important tasks in computational science and applied mathematics. The applications of such solvers can be found in almost every corner of the engineering sciences. One of the most important equations is the Stokes equation:

$$\begin{aligned} -\mu\Delta u + \nabla p &= 0 \\ \operatorname{div} u &= 0 \end{aligned}$$

which governs the behavior of a incompressible viscous fluid with velocity u and pressure p.

The most popular methods for such problems use finite element discretization on unstructured meshes. Multigrid methods [15, 86] and domain decomposition methods [17, 18, 82] are often used to precondition the linear system resulting from the finite element discretization. Nevertheless, applications of these methods to problems with complex geometry have two major difficulties: the generation of the unstructured mesh and the construction of efficient preconditioners.

First, 3D unstructured mesh generation is algorithmically difficult and computationally intensive. Although a lot of great work has been done in this direction [48, 74, 79], it still remains an area of active and ongoing research. For large-scale problems, parallel computing is often required in order to obtain sufficient accuracy within reasonable time period, therefore the unstructured mesh generation often needs to be done in parallel as well. This poses an even more challenging problem.

Second, most preconditioners, such as multigrid methods and domain decomposition methods, are multilevel techniques which necessitates construction of one or more progressively coarser versions of the finest unstructured mesh used in the computation. However, the difficulty in automatic generation of such unstructured mesh hierarchies greatly restricts us from applying these efficient preconditioners to general problems.

Along with the increase in computational power, researchers have begun to study time-dependent multi-physics problems with deforming or moving boundaries, such as the fluid-fluid interaction [78] and fluid-structure interaction [29] problems. At every time step, usually one or more elliptic PDE boundary value problems need to be solved on the updated domain. The finite element based algorithms are not optimal for these problems, since they often require unstructured mesh generation whenever the existing unstructured mesh becomes distorted.

Another class of methods use boundary integral equation formulations, in which, the boundary value problem is written as an integral equation form, involving only the quantities defined on the boundary. Compared to the finite element method, the major advantage of this approach is that there is no need for unstructured volume mesh generation. The discretization of the integral equations also result in a much smaller number of unknowns. Moreover, the resulting linear system often has much nicer spectral properties, which enable the algorithms based on the integral formulation to demonstrate optimal complexity. These properties make the boundary integral formulation an attractive and promising approach to model problems with complex or deforming boundaries.

The boundary integral formulations have three major disadvantages. First, the linear system resulting from discretization is always dense, which makes it computationally expensive to perform matrix vector multiplication. Second, the integrals in these formulations are often singular, which makes it difficult to evaluate these integrals accurately. Third, the integrals involved are different for various elliptic PDEs, which often require different implementations for the singular integration of each equation.

This thesis presents an efficient and high-order boundary integral equation solver for the Stokes equations. It solves the dense linear system efficiently and integrates the singular integrals with high-order accuracy. In addition, the algorithmic components of this solver have been extended to other constant coefficient elliptic PDEs with minimal modification.

1.2 Problem Statement

The goal of this thesis is to develop an *efficient* and *highly accurate* solver for the boundary integral equation of the Stokes equation with complex geometry.

The problems with non-smooth boundaries are more complicated and not addressed in this thesis, although the techniques and algorithms we developed can be extended to address the non-smooth case. We explain our solver for the Stokes Dirichlet problem. The boundary integral formulation for the Neumann boundary condition is similar and we comment on their differences in Section 1.4.

The Stokes equations for a Dirichlet boundary value problem can be stated as follows:

$$\begin{aligned} -\mu\Delta u + \nabla p &= 0 \quad \text{in} \quad \Omega, \\ \operatorname{div} u &= 0 \quad \text{in} \quad \Gamma, \\ u &= f \quad \text{on} \quad \Gamma, \end{aligned} \tag{1.1}$$

where Ω is the 3D fluid domain, Γ the domain boundary, u the velocity field, p the pressure, and f the Dirichlet boundary condition defined on Γ.

A boundary integral formulation writes the velocity u in Ω as

$$u(x) = \int_\Gamma D(x,y)\varphi(y)\,\mathrm{d}s(y) \tag{1.2}$$

[69, 70]. Here φ is a function defined on Γ and called the *double layer density*. D is the *double layer kernel* for velocity and defined by

$$D(x,y) = -\frac{6}{8\pi}\frac{(r\otimes r)(r\cdot n(y))}{|r|^5},$$

where $x \in \Omega$ is the observation point, $y \in \Gamma$ is the source point, $r = x - y$ and $n(y)$ is the outward normal direction of the boundary surface at point y. By introducing the following operator notation

$$(C\varphi)(x) := \int_\Gamma C(x,y)\varphi(y)\,\mathrm{d}s(y),$$

we can write the integral representation as

$$u(x) = (D\varphi)(x).$$

As $x' \in \Omega$ approaches a boundary point $x \in \Gamma$, the limit of $u(x')$, which is usually denoted as $u_+(x)$, is given by:

$$u_+(x) = \lim_{x' \to x} (D\varphi)(x') = \frac{1}{2}\varphi(x) + (D\varphi)(x). \tag{1.3}$$

Here $\varphi(x)$ is also equal to the jump at x: $[[u]](x) = u_+(x) - u_-(x)$ where $u_-(x)$ is the limit of (1.2) at x from the exterior of the domain. Given the boundary condition f on Γ, we obtain the double layer boundary integral equation for the Dirichlet problem of the Stokes equation [69, 70]:

$$\frac{1}{2}\varphi(x) + (D\varphi)(x) = f(x). \tag{1.4}$$

A boundary integral equation accomplishes two tasks. First, it solves for φ using (1.4). Second, it evaluates $u(x)$ and other related quantities (such as pressure and stress) at an arbitrary point x in Ω or Γ. For u we can use (1.2) or (1.3) depending on whether x is in Ω or on Γ. For pressure p and stress $s = -pI + \mu(\nabla u + \nabla u^t)$, the equation for x in Ω is given by:

$$p(x) = (K\varphi)(x) = \int_\Gamma K(x, y)\varphi(y)\,\mathrm{d}s(y),$$

and

$$s(x) = (T\varphi)(x) = \int_\Gamma T(x, y)\varphi(y)\,\mathrm{d}s(y),$$

where the pressure kernel K and the stress T are given in the appendix. For a point x on Γ, the equations are correspondingly

$$p(x) = \frac{1}{2}[[p]](x) + (K\varphi)(x),$$

and

$$s(x) = \frac{1}{2}[[s]](x) + (T\varphi)(x),$$

where $[[p]]$ and $[[s]]$ are the jumps for the pressure and stress. It is important to point out that, in the equations for x on Γ, both integrals have an $\frac{1}{|r|^3}$ singularity and are to be understood in the *Hadamard* sense [39].

There are several advantages of using the integral formulation instead of the differential formulation in the numerical computation. First, as we mentioned, both the constraints and the unknowns are restricted to the boundary Γ. Therefore, there is no need for unstructured mesh generation, and the number of unknowns to be solved is much smaller than the number for a finite element solver. Second, (1.4) is an integral operator of the second kind. The Fredholm alternative states that the spectrum of such operators are nicely bounded. Therefore, the algebraic system resulting from the discretization of such equations can be solved using an iterative solver like GMRES with only a few iterations.

However, boundary integral equations also pose several new challenges as we pointed out in the previous section. First, $D(x, y)$ is non-zero for all pairs of x and y, which means that the linear system to be solved is a dense matrix, and each matrix-vector multiplication in the iterative solver can be costly. Second, the operators $(D\varphi)(x)$ and $(K\varphi)(x)$ in (1.4) are either weakly-singular or singular integrals since x belongs to Γ. Our numerical scheme should be able to integrate these singular integrals with high accuracy. Third, for $x \in \Omega$ the operator $(D\varphi)(x)$ for the evaluation of $u(x)$ can have arbitrary sharp peaks as x approaches the boundary Γ. Developing a scheme to integrate such integrals, independent of the distance between x and Γ, is far from trivial.

1.3 Approach

The approach taken in this thesis is as follows. First, the possibility of a high-order solver hinges on a high-order smooth surface representation for the domain boundary [16, 34]. Most of the available boundary integral solvers use piecewise linear or piecewise quadratic representation for the boundary, which is acceptable if the solver itself is only first or second order accurate. To get high-order accuracy, the boundary representation needs to have at least the same order of accuracy as the targeted order of accuracy of the boundary integral solver. Moreover, in order to represent deforming objects, the surface representation should also be free-form and able to model boundaries with complex topology and geometry. This thesis first presents a free-form parametric surface representation for modeling smooth manifolds with arbitrary geometry and topology. We generate the surface based on a set of control points. The resulting surface is C^k continuous with explicit C^k parameterizations ($k \geq 2$ or $k = \infty$) over a set of bounded open domains in $\mathbf{R}^2$, and has an explicitly constructed C^∞ smooth partition of unity.

We discretize the boundary integral equation using the Nyström method and use a GMRES solver to solve the resulting linear algebraic system. In a Nyström method, the integral operator is approximated by a quadrature formula, and the solution of the integral equation is approximated by the solution of matching constraints at the quadrature points [2, 47]. Since the essential step of the GMRES solver is the matrix vector multiplication (in our case, the evaluation of the boundary integral at the Nyström points using the double layer density at the same points), the success of this approach depends on the ability to integrate the singular kernel accurately. To construct an efficient and high-order accurate integrators, we face two challenges:

evaluating smooth non-adjacent interactions efficiently and evaluating singular adjacent interactions accurately. This thesis overcomes the first challenge by developing a kernel-independent fast multipole method (FMM). In attempting to handle the second problem, we develop a method which is similar to, but more general than [16]. The C^k parameterizations and the smooth partition of unity of the boundary surface are essential to the accurate and efficient integration of our Nyström method.

Finally, we develop a scheme to evaluate the solution at points in the domain or on the boundary. For points on the boundary, we use the same integrator we used in the Nyström solver to evaluate the singular integral. We also derive the formulas for the jumps and use fast Fourier transform (FFT) to evaluate them. To evaluate the solution everywhere in the domain, we partition the points in the domain into different regions depending on their distance to the boundary. For points which are adequately separated from the boundary, we evaluate the solution with an integrator using a trapezoidal quadrature rule and FMM acceleration. For points which are very close to the boundary, we compute the integral by first evaluating the field in several nearby points and then interpolating the solution at the target point.

1.4 Neumann Boundary Condition and Other PDEs

We briefly sketch the boundary integral representation for the Neumann problem here and comment on how the approach described in the previous section can be used. The Neumann boundary condition specified in terms of the stress tensor $s = -pI + \mu(\nabla u + \nabla u^t)$ is

$$s \cdot n = g \quad \text{on} \quad \Gamma,$$

where g is a smooth function defined on Γ. In the Neumann case, the velocity field u is represented using the single layer formulation by

$$u(x) = (S\varphi)(x) = \int_\Gamma S(x,y)\varphi(y)\,\mathrm{d}s(y),$$

where φ is the single layer density and S the singular layer kernel

$$S(x,y) = \frac{1}{8\pi\mu}(\frac{1}{|r|}I + \frac{r\otimes r}{|r|^3}),$$

where again $r = x - y$. The integral equation is given in terms of the singular layer density φ and the boundary condition g by:

$$\begin{aligned} g(x) &= \frac{1}{2}\varphi(x) + \int_\Gamma \frac{\partial S(x,y)}{\partial n(x)}\varphi(y)\,\mathrm{d}s(y) \\ &= \frac{1}{2}\varphi(x) + \int_\Gamma \left(-\frac{6}{8\pi}\frac{(r\otimes r)(r\cdot n(x))}{|r|^5}\right)\varphi(y)\,\mathrm{d}s(y). \end{aligned}$$

The singularity of this kernel is of the same type as the one for D. As in the case of D, this equation is again of the second kind. Therefore, the same approach we discussed in Section 1.3 can be used to solve this integral equation.

We can extend our approach to several other elliptic PDEs, including the Laplace equation, the Navier equation and all their modified versions. For all of these equations, the boundary integral equation formulation takes the same form as (1.4), only with different double layer kernels $D(x,y)$. The kernels are given in the appendix.

However, for the Navier equation and its modified version, the equations are no longer integral equations of the second kind anymore, due to the fact that the double layer kernel D for the Navier equation is singular (instead of just weakly singular in the case of the Stokes equation). The operator D needs to be interpreted in the *Cauchy* sense and is not a compact operator anymore, which means that we cannot directly apply the Fredholm alternative on such integral equation. However, a similar version of the Fredholm alternative can be applied [63] and the same numerical algorithm adopted in this thesis for the Stokes equations can be utilized on the Navier equation.

1.5 Contributions and Thesis Organization

This thesis develops an efficient and high-order boundary integral solver for the Stokes equation with complex geometries. The main contributions of this thesis are:

- A free-form method to model high-order smooth surfaces with arbitrary complex geometry and topology. Surfaces constructed by this method have an explicit smooth parameterization. They linearly depend on the control points, have fixed-size support for basis functions, and demonstrate good visual quality.

- A kernel-independent adaptive fast multipole method in 2D and 3D. This method is adaptive and uses only kernel evaluation. It exhibits good accuracy and efficiency, and has proved error bounds. We also develop a parallel version of this algorithm which demonstrates good scalability.

- General schemes to evaluate singular integrals and nearly singular integrals. The scheme for singular integral integration improves upon previous research. These schemes are independent of the specific kernel, efficient and high-order accurate.

- All the algorithmic components are kernel-independent and the boundary integral solvers for other equations can be constructed from them in a blackbox fashion.

This thesis is organized as follows. Chapter 2 describes the new high-order surface representation. Chapters 3 and 4 present the kernel-independent fast multipole method and its parallel implementation. Chapter 5 describes the Nyström discretization based on our surface representation and the general schemes for evaluating singular integrals and nearly singular integrals. Chapter 6 presents the numerical results and applications.

1.6 Background

Boundary integral formulation has been used to investigate a lot of physics problems involving second order elliptic partial differential equations.

For the fluid problems, the boundary integral formulation in prime variables for the Stokes equations can be found in [46, 69, 70]. In [31, 51, 68], the homogeneous Stokes problem is solved using a boundary integral representation combined with multipole-like far-field expansions to accelerate the matrix-vector multiplications. The Stokes equation can also be posed as a biharmonic problem. Related formulations can be found in [53, 54]. In [34] the homogeneous Stokes problem is solved for both interior and exterior problems using this formulation. In [42, 78], the velocity potential formulation is used to solve intefacial flow problems.

The boundary integral formulation has also been used widely to solve physics problems in electrostatics [44, 45], electromagnetics [16, 55], elastics [63] and scattering theory [11, 16, 83].

In the rest of this section, we review the related research on the numerical solutions of the boundary integral formulation of the second kind for elliptic PDEs. The general form of the equation is:

$$\frac{1}{2}\varphi(x) + \int_\Gamma D(x,y)\varphi(y)\,\mathrm{d}s(y) = f(x) \quad \text{or} \quad \frac{1}{2}\varphi + D\varphi = f. \tag{1.5}$$

1.6.1 Discretization Methods

The general theory on the integral equations of the second kind has been developed in the early 20th century. Some good references are [39, 47, 53, 54]. Most of the numerical methods fall into two categories: projection methods and quadrature (Nyström) methods. The projection methods solve the integral equation by choosing a finite dimensional space Φ of functions defined on Γ which is constructed to contain a good approximation $\bar{\varphi}$ to the true solution φ, and solving for $\bar{\varphi}$ by satisfying (1.5) in an approximate sense. The finite dimensional function space Φ is a linear span of a set of basis functions $\{\varphi_1, \cdots, \varphi_n\}$. The basis functions are usually locally supported, in which cases the numerical methods are generally called boundary element methods. Sometimes, the basis functions are globally supported, such as polynomials, spherical polynomials and trigonometric polynomials, and these type of methods are called spectral element methods. We can write $\bar{\varphi}$ as

$$\bar{\varphi} = \sum_{i=1}^{n} c_i \varphi_i,$$

and solving the integral equation numerically is equivalent to finding the coefficients c_i.

Depending on the sense in which (1.5) is approximated, the projection methods can be further classified into collocation methods and Galerkin methods. In the collocation methods, the residual

$$r = f - (\frac{1}{2}\bar{\varphi} + D\bar{\varphi})$$

is required to vanish at a set of points $\{x_i, i = 1, \cdots, n\}$ on Γ. This leads to the linear system:

$$\sum_{j=1}^{n} c_j \left(\frac{1}{2}\varphi_j(x_i) + \int_{\Gamma} D(x_i, y)\varphi_j(y)\,\mathrm{d}s(y) \right) = f(x_i), \quad i = 1, 2, \cdots, n.$$

In the Galerkin methods, the residual r is required to be orthogonal to a finite dimensional space Ψ of functions with bases $\{\psi_1, \cdots, \psi_n\}$. This new space is often chosen to be the same as Φ. This leads to the following linear system:

$$\sum_{j=1}^{n} c_j \left(\frac{1}{2}(\psi_i, \varphi_j) + (\psi_i, D\varphi_j) \right) = (\psi_i, f), \quad i = 1, 2, \cdots, n.$$

There have been numerous algorithms and implementations of collocation and Galerkin methods for integral equations in both 2D and 3D. We refer to [2, 3, 39, 47, 54, 81] for general descriptions of the collocation and Galerkin methods. For applications of boundary element methods in 3D case, good references include [14, 19, 39, 92]. Examples of using spectral element methods in 3D applications can be found in [2, 21].

The quadrature or Nyström methods approximate the integral operator in (1.5) by numerical integration. Given a quadrature rule with points $x_1, \cdots, x_n$ on Γ and weights $\alpha_1, \cdots, \alpha_n$, we approximate the integral $D\varphi(x)$ with

$$\bar{D}\varphi(x) = \sum_{j=1}^{n} \alpha_j(x) D(x, x_j)\varphi(x_j),$$

where $\alpha_i(x)$ can be functions of x though they are mostly constants. The solution of (1.5) is approximated by the solution of the following operator equation

$$\frac{1}{2}\bar{\varphi} + \bar{D}\bar{\varphi} = f.$$

Writing the equation at every quadrature point x_i leads to a finite dimensional linear system

$$\frac{1}{2}\bar{\varphi}_i + \sum_{j=1}^{n} \alpha_{ij} D(x_i, x_j)\bar{\varphi}_j = f(x_i),$$

where $\alpha_{ij} = \alpha_j(x_i)$ and $\bar{\varphi}_i = \bar{\varphi}(x_i)$. Nyström methods are widely used for 2D problems, in which case the integral in (1.5) is on a 1D periodic domain. High-order integrators with exponentially accurate trapezoidal rule and other high-order quadrature rules are the optimal tools to treat these boundary integrals. Detailed treatments can be found in [2, 3, 39, 47]. The situation changes drastically if we move to 3D problems, in which case there are no straightforward quadrature rules available for the integrals in (1.5). There are two difficulties: First, the integrand is always singular. Second, even for smooth functions, trapezoidal rules cannot be used, and the development of other high-order quadrature rules is non-trivial. Only recently, [16] presents an algorithm to solve the 3D acoustic scattering problem.

1.6.2 Techniques for Fast Solvers

The discretization of the boundary integral equations always leads to dense matrices. The matrix-vector multiplication, if carried out in the direct way, requires $O(N^2)$ operations, where N is the number of unknowns. In this case, even the iterative solvers, such as GMRES, can be quite expensive for real applications. For large-scale problems, we need to develop efficient schemes to perform the matrix vector multiplication. The basic idea is to exploit the fact that the kernel $D(x, y)$ is very smooth when x and y are well separated. Most of the methods fall into three classes.

The first class is based on the fast Fourier transform (FFT). These methods embed the boundary into a Cartesian grid. The sources at the discretization nodes (usually the basis function in projection methods or the Nyström points in Nyström methods) are transfered onto the Cartesian grid in a way that the potentials produced by the original sources and the ones on Cartesian grids are sufficiently close for the region far away from the sources. Then the far interaction by the grid sources is naturally formulated as a 3D convolution and FFT is used to compute this convolution efficiently. These methods are easy to implement and work for different PDEs without much modification. The major disadvantage of this method is that it is not adaptive due to the use of the Cartesian grid. Examples of this class include [16, 55].

The second class of methods [6, 22] is based on wavelet decomposition. The idea is to use a redundant wavelet representation which automatically zeros out the far field interaction. These methods are adaptive and have optimal complexity. However, the difficulty with these methods is the construction of appropriate wavelet bases on the boundaries of the complex domains. Moreover, the application of these methods is usually restricted only to Galerkin methods.

The third class of methods is based on the fast multipole method (FMM) [20, 35, 36]. The idea is to construct efficient representations for the potential generated by a cluster of sources and also efficient translations between these representations. Moreover, this is done in a hierarchical fashion so that the algorithm has optimal complexity $O(N)$. This algorithm is fully adaptive and highly accurate. The disadvantage is that it requires different implementations for different equations. Chapter 3 of this thesis addresses this issue. Related research about the FMM is reviewed in that chapter.

Chapter 2

High-order Surface Representation Scheme

This chapter describes a high-order surface representation scheme which is used to model the boundaries of the computational domains. These boundaries are required to have high-order smooth parameterizations in order to make it possible for the solver to achieve high-order accuracy.

2.1 Introduction

Much of the work on smooth surface representations, excluding variational surfaces, is based on the paradigm of stitching polynomial patches together. While subdivision surfaces are generally defined as limits of recursive refinement algorithms, the surfaces produced by most popular schemes can still be interpreted as infinite collections of stitched spline patches.

In this chapter we take a different and often neglected approach based on the manifold construction of [38]. We demonstrate how this approach can produce with relative ease a number of desirable properties which are hard to achieve simultaneously with polynomial patches, subdivision surfaces or variational surfaces. Specifically, our surfaces are C^∞-continuous with explicit nonsingular C^∞ parameterizations, are at least 3-flexible (i.e., can have arbitrary derivatives of order up to three) at control vertices, depend linearly on control points, have fixed-size local support for basis functions, and have good visual quality.

This surface representation was developed with its application to the boundary integral equations. It provides a high-order, smooth and nonsingular parametrization to ensure fast convergence of quadrature rules on surfaces, i.e., for the parametrization to have good *mathematical quality*; at the same time, it is essential to be able to model objects of arbitrary shape and obtain good *visual quality* without additional processing.

We have observed that even all existing flexible C^2 constructions are quite complex, and while higher-order constructions exist, despite having nice mathematical properties few were ever fully implemented and visual surface quality was typically inferior to lower-order schemes.

Subdivision surfaces are a notable exception: they were not constructed to satisfy a specific set of requirements. Rather, it was observed that the subdivision algorithms for splines generalize well to arbitrary control meshes, and the visual quality of the surface is adequate in an intuitive sense, except near high-valence vertices. Analysis of properties came later and is quite complex: even obtaining a C^1 nonsingular parametrization is nontrivial and requires inversion of the characteristic map [84].

In our approach, we relax the requirement of representing surfaces using polynomial patches to simplify the construction needed to achieve good mathematical quality, and we ensure that our surfaces approximate closely the shape of subdivision surfaces to achieve acceptable visual quality for a similar range of vertex valences.

We believe that due to the properties enumerated above, representations of this type provide the most convenient basis for “black-box” surface approximation software: the user provides an input control mesh, and the surface and its parametric derivatives of any order can be evaluated at any point. While the black-box approach is not the most efficient, it is the most convenient and reliable one for applications requiring a large variety of algorithms to operate on surfaces.

2.2 Related Work

We are not aware of any C^∞ constructions for surfaces with general control meshes. The work on computational representations of surfaces based on manifolds is relatively limited. The idea was introduced in [38]. More recently, [58] have proposed a C^k construction based only on polynomials.

Extensive literature exists on spline-based constructions of different types; C^k constructions include S-patches [49], DMS splines [76], freeform splines [71] and TURBS [72]. DMS splines were developed to the greatest extent; as a large number of knots and control points need to be introduced for each triangle, an additional algorithm is needed to position these points if only an initial mesh is given. TURBS are based on singular parameterizations; both for free-form splines and TURBS, additional degrees need to be set which can be done using fairing functionals and precomputed matrices, similar to the technique that we use. There are numerous C^2 constructions, e.g. [37], [88], [60], [41],[12], and most recently [62]. There is an even larger number of C^1-constructions; excluding subdivision surfaces, which are

C^2 away from isolated points, these constructions rarely yield surfaces of acceptable quality. In all cases, the required polynomial degree and number of additional patch control points to be set rapidly grows with smoothness.

Unfortunately, in most cases it is impossible to compare the visual quality of the resulting surfaces to our construction. The implementation is complex and only few images of simple objects are provided in the papers, as the stated goal in most cases is to obtain surfaces satisfying a specific mathematical condition.

The literature related to subdivision surfaces is extensively reviewed in the book [89] and in the course notes [96]. Direct fixed-time evaluation of subdivision surfaces was introduced in [84], which is the only known approach to directly cast subdivision surfaces in parametric form. [61] describes a technique for approximating Catmull-Clark surfaces with a collection of bicubic patches joined with C^1 continuity.

2.3 Construction

In this section, we start with the basic definitions of manifolds. We then give an overview of our construction, followed by three key components of the construction.

2.3.1 Manifold Structure

We consider meshes consisting of quadrilaterals, although this is not critical for our construction: it can be carried out in a similar way using triangle meshes and Loop subdivision surfaces [96], for example. We focus on the quadrilateral case as it has more relevance for geometric modeling applications.

The foundation of our approach is a simple construction of a C^∞-manifold associated with a mesh. The basic definition of a manifold is as follows: a set M

has 2D manifold structure, if a collection of charts (C_i, χ_i) is defined, where C_i are open domains in the plane, χ_i are one-to-one maps $C_i \to M$, such that the images $P_i = \chi_i(C_i)$ cover all of M. M is a C^∞ manifold if the transition maps from chart to chart

$$t_{ji} = \chi_j^{-1} \circ \chi_i : \chi_i^{-1}(P_i \cap P_j) \to \chi_j^{-1}(P_i \cap P_j)$$

defined for pairs of charts for which $\chi_i(C_i)$ and $\chi_j(C_j)$ intersect, are C^∞. Detailed discussion of manifolds is given in [9, 13, 23, 38]. Notice that in our definition, the charts are the maps from open sets in $\mathbf{R}^2$ to the manifold, while in most differential geometry books they are defined the other way around. The reason is that, in this thesis, we focus on the smooth parameterization of the surface (which is essential to the accurate integration) rather than the global intrinsic properties of the manifold (which are the main problems in differential geometry).

In our construction, we use the control mesh as the domain M. For this we need to assume that the mesh has no self-intersections. This assumption is not crucial (we can construct the domain in a more abstract manner) but simplifies explanations. It has no implications for implementation.

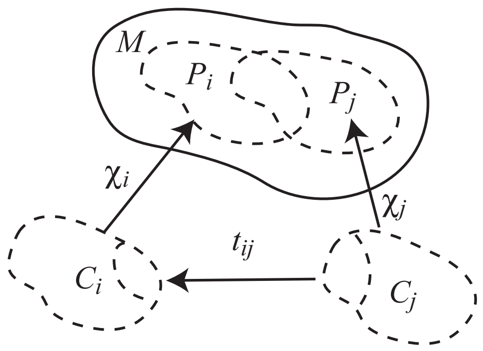


Figure 2.1: Basic definitions of manifold.

Another important idea related to manifolds is the *partition of unity* (POU). A set of smooth functions w_i each defined on M, and having compact support, is called a partition of unity, if $\sum_i w_i(m) = 1$ for $m \in M$. In our case, we require the support of w_i to be contained inside P_i. Therefore, $w_i \circ \chi_i$ is a smooth function defined on C_i and vanishes on the boundary of C_i.

2.3.2 Overview of the Construction

The general approach is close to the one in [38]. We construct functions $g_i^l : C_i \to \mathbf{R}^3$, defining the local geometry on each chart; then, we use a partition of unity to define the global geometry. The complete surface at m in M is defined by

$$\sum_i w_i(m) \cdot (g_i^l \circ \chi_i^{-1})(m)$$

where i ranges over the charts such that $m \in \chi_i(C_i)$. However, in practice it is evaluated on individual charts C_i via

$$g_i(x) = \sum_j w_j(\chi_i(x)) \cdot g_j^l(t_{ji}(x)) \quad x \in C_i. \tag{2.1}$$

where j ranges over the charts which overlap with C_i, and g_i defines the global geometry in contrast to g_i^l which defines the local geometry.

Note that the complexity of evaluation of this expression is determined by three factors: complexity of transition maps t_{ij}, weights w_j and geometry functions g_j^l. In our case, the transition maps can be expressed in complex form as z^α (up to a rotation), the weights are piecewise exponential and C^∞, and the geometry functions are polynomials of degrees proportional to the valence of vertices corresponding to the charts. Another important observation is that $g_i(x)$ is C^∞ if all components are C^∞. Next, we discuss each component separately.

2.3.3 Charts and Transition Maps

As a basis for our construction, we use the conformal atlas for meshes. While its variations can be found in the literature (e.g. [24] in the context of parametrization), a complete description is not easily available, and we present it here. We define charts per vertex. Each chart domain is a curved star shape D_i, shown in Figure 2.2. The overlap region between the images of two charts in the control mesh is two faces of the mesh. Rather than constructing the maps χ_i we construct the maps χ_i^{-1}. The chart construction proceeds in two steps: first, the faces adjacent to a given vertex are mapped piecewise bilinearly to the plane (maps L_i to domains S_i). Then a transformation c_i is applied to each wedge of the regular star S_i; c_i squeezes it so that it becomes a conformal image of square. Maps c_i have simple explicit expressions for each wedge. As illustrated in Figure 2.3 for the shown choice of coordinate system these maps are compositions of a linear map l_{k_i} with matrix

$$\begin{pmatrix} \cos(\pi/4)/\cos(\pi/k_i) & 0 \\ 0 & \sin(\pi/4)/\sin(\pi/k_i) \end{pmatrix}$$

where k_i is the valence of D_i and a simple map g_{k_i}, which using standard identification of the plane with complex numbers $z = x + iy$, can be written as z^{4/k_i}. The chart maps χ_i^{-1} are compositions $c_i \circ L_i$.

This atlas has an important property: *all transition maps are conformal, in particular, C^∞.* In fact, the transition maps, for a certain choice of the coordinate systems can be written as z^{k_1/k_2}. The fact that transition maps have simple expressions is very important; it allows as to define the geometry in an efficiently computable way. The proof of C^∞ continuity for the transition map is outlined as follows:

Let's fix coordinate systems in the domains S_i, $i = 1, 2$ depicted in Figure 2.2 with x direction along the common edge of two shaded wedges. Let L_i', $i = 1, 2$ be

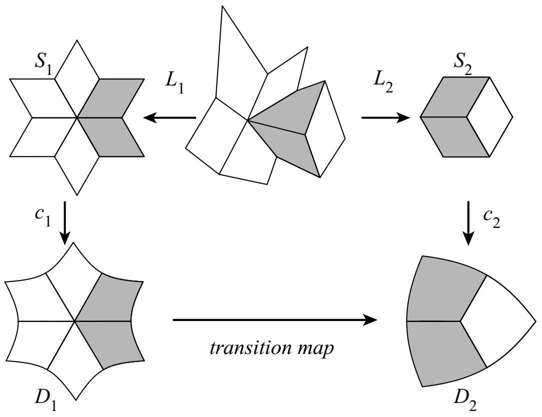


Figure 2.2: Construction of the charts. The maps L_i, $i = 1, 2$ are piecewise bilinear; the maps c_i are constructed on individual wedges as shown in Figure 2.3.

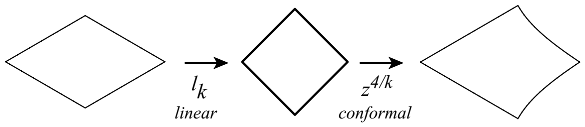


Figure 2.3: On each wedge, the map c_i is a composition of a linear map and the map z^{4/k_i}.

the piecewise linear maps from a pair of adjacent unit squares. We also assume the coordinate system x axis to be along the common edge. A similar choice is made for the domains D_i. The maps c_i for these coordinates can be written as $R(\pi/k_i) \circ g_{k_i} \circ l_{k_i} \circ R(-\pi/k_i)$ for the top quad and $R(\pi/k_i) \circ g_{k_i} \circ l_{k_i} \circ R(-\pi/k_i)$, where k_i is the valence of the corresponding vertex, $R(\alpha)$ is the rotation by the angle α, and the maps l_{k_i} and g_{k_i} are defined in Section 2.3.

The transition map $c_2 \circ L_2 \circ L_1^{-1} \circ c_1^{-1}$ can then be written as $c_2 \circ L_2' \circ {L_1'}^{-1} \circ c_1^{-1}$, with the mesh maps itself eliminated. This can be done, as the composition of a linear and a bilinear map is bilinear, and the maps are defined uniquely by the correspondence of the domain corners, $L_i = L_i' \circ L$, so the bilinear part L is factored out.

Next, we observe that the two piecewise linear parts of L'_i are equivalent to $R(\pi/k_i) \circ l_{k_i}^{-1} \circ R(-\pi/4)$ and $R(-\pi/k_i) \circ l_{k_i}^{-1} \circ R(\pi/4)$ in the chosen coordinate system. Therefore, the transition map can be rewritten, for example on the top square, as $R(\pi/k_2) \circ g_{k_2} \circ g_{k_1}^{-1} \circ R(-\pi/k_1)$. In the complex form, the rotation is just a multiplication by $\exp(i\alpha)$. The transition map is $(\exp(-i\pi/k_1)z)^{k_1/k_2} \circ \exp(i\pi/k_2) = z^{k_1/k_2}$. Exactly the same can be shown for the bottom square.

2.3.4 Partition of Unity

The partition of unity is a crucial element of our construction: the quality of surface is defined not only by the quality of the geometry functions but also how well they are blended. Our empirical observations are that the partition of unity cannot have transition regions which are too steep, and, even more importantly, its support shape should match the shape of the star-like shape of the corresponding chart.

We build the partition of unity from identical pieces defined initially on the standard square $[0, 1]$ as a product of two identical one-dimensional functions $\eta(u)\eta(v)$. The function η is defined as follows [16]:

$$\eta(t) = \begin{cases} 1 & : 0 \leq t \leq \delta \\ \frac{h((t-\delta)/a)}{(h((t-\delta)/a)+h(1-(t-\delta)/a))} & : \delta < t < 1-\delta \\ 0 & : 1-\delta \leq t \leq 1 \end{cases}$$

where $a = 1 - 2\delta$ and $h(s) = \exp(2\exp(-1/s)/(s-1))$. The resulting function is quite close in appearance to a Hermite spline (Figure 2.4).

We set $\delta > 0$ for the following reason: When $\delta = 0$, the transition maps has unbounded derivatives at the boundary of the overlapping charts. While it is possible that the composition of the transition map and the partition of unity has bounded

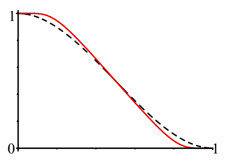


Figure 2.4: The solid line is the function $\eta(t)$ used in the construction of the partition of unity. The dashed line is a Hermite spline which is close to $\eta(t)$.

derivatives if the partition of unity has sufficiently fast decay, we simply choose the partition of unity to be constant near the boundary. In our implementation, we use $\delta = 1/8$.

Once the function is defined on the square, we obtain a weight, defined on the whole chart as follows. First, we use a rotation by $\pi/4$ combined with the map $g_k^{-1} = z^{k/4}$ to remap $\eta(u)\eta(v)$ to a single wedge. The function is defined by rotational symmetry on the rest of the chart. Finally, we use the chart to move the resulting function onto M to get the partition of unity.

The resulting POU function is C^∞ on the whole chart. One can easily verify that derivatives of all orders of $\eta(u)\eta(v)$, defined in Section 2.3 involving v, are zero at the boundary $v = 0$, and same is true for $u = 0$ by symmetry. Remapping to a wedge using a non-degenerate C^∞ map does not change the fact that all derivatives vanish identically, except derivatives along the boundary as the map that we use may not be differentiable at zero; however, in a δ-neighborhood of zero the function $\eta(u)\eta(v)$ is constant. By symmetry, these match at all orders after rotations extending the map to all wedges.

2.3.5 Defining geometry.

We define geometry using polynomials. The basic idea is to apply several subdivision steps to define the overall coarse shape of the surface, and use polynomials in the chart to fit this shape in the least square sense. As the fit is linear and the control points of refined subdivision mesh depend linearly on the control points of the original mesh, the transformation matrix converting control points to the polynomial coefficients can be precomputed. Thus, in practice the process is reduced to assembling a vector of control points and multiplying them by a matrix.

Every control point of the refined mesh after two Catmull-Clark subdivision steps can be assigned to the points with bilinear coordinates $(i/4, j/4)$ in each sector of the star S_k. For each vertex v, we remap these points in S_k to the chart domain D_k by using the map c_i. There are $m = 12k + 1$ points *inside* D_k which we denote $x_0, \dots x_{m-1}$. We compute 3D limit positions for these points in the same order, and denote them $s_0, \dots, s_{m-1}$. Our goal is to define a geometry function g^l such that differences $g^l(x_i) - s_i$ are minimized in the least squares sense.

We use the monomials of degree $\leq d = \lfloor \mathbf{min}(14, \sqrt{2(12k+1)}) \rfloor$ as the basis functions in the fitting process. The choice of 14 as the maximal degree is empirical: using higher-order polynomials results in lower quality surfaces for high valences. We denote these monomials $p_0, \dots, p_{n-1}$ where $n = d(d+1)/2$ is the number of monomials used in the fitting. We use the least square fit to solve for the basis coefficients a_j, such that $g^l = \sum_{j=0}^{n-1} a_j p_j$. Let a be the vector of coefficients a_j, s be the vector of values s_i and U be the $m \times n$ matrix of monomial values $p_j(x_i)$ at points x_i. Then the least squares fit minimizing $||Ua - s||^2$ is given by

$$a = U^+ s$$

where $(\cdot)^+$ denotes pseudoinverse. The $n \times m$ matrix U^+ only depends on the valence k since x_i and p_j depend only on k. Therefore, it can be precomputed once and used for all charts with the same valence.

Flexibility of the surface at vertices in the center of the charts is easy to show, as one can construct specific control point configurations yielding various low-degree polynomials in a direct form.

We note that the above construction is the simplest among those we have tried; its disadvantage is the relatively large size of U_k, which can be reduced by using a more careful choice of polynomial bases and the singular value decomposition (SVD) from n to $3k+1$ without loosing surface quality.

2.3.6 Alternatives

Although the construction described in the previous sections generates C^∞ continuous surfaces with good visual quality (Section 2.4), alternative choices for every step of the construction are available and can potentially generate better surfaces.

In the chart definition step, instead of using the conformal map z^{4/k_i} in the construction of the map c_i from S_i to D_i, we can use any function of the form

$$|z|^p (\frac{z}{|z|})^{4/k_i},$$

where p is positive. The chart maps and transition maps associated with this function are still C^∞ continuous although not conformal any more. One preferable choice of p is $\log_2(1/\lambda_{k_i})$ where λ_{k_i} is the second largest eigenvalue of the subdivision matrix of the Catmull-Clark subdivision scheme at valence k_i. This choice makes the domain D_i very close to the shape of the characteristic map [96] of the Catmull-Clark scheme and therefore the resulting map g_i are very close to the standard C^1 parameterization of the subdivision surface.

In cases when the C^∞ property of the generated surface is not important, we can also use spline functions as function η to construct the partition of unity (Figure 2.4). The resulting surfaces will have the same order of continuity as the spline function assuming the rest parts of the construction remain unmodified.

Using polynomial bases is only one way to define the local geometry. More advanced methods include using trigonometric splines [75], radial basis functions [87] and other interpolation methods. Another idea is to approximate the product of subdivision surface geometry and the POU in each domain C_i, since only their product is used for the construction of the global geometry. In this case, 2D trigonometric polynomials can be used as the approximation bases.

2.4 Results

Implementing our scheme is relatively simple: our basic implementation has 1,500 lines of code including subdivision but excluding SVD code.

In most images, we use a reflection map on a part of the surface to show the surface quality. Figure 2.5 shows a detailed comparison of the surfaces with Catmull-Clark surfaces near valance 5, valence 8 and valence 12 vertices. The visual quality of our surfaces is close to the one of Catmull-Clark surfaces, except in the immediate neighborhood of the vertex, where reflection lines show lack of C^2-continuity of Catmull-Clark. This can also be observed in parametrization images, where a uniformly spaced checkerboard in the parametric domain is mapped to the surface.

Figure 2.6 shows the sum of the magnitudes of the derivatives of the parametrization on a chart, to demonstrate the variation. We note that starting from fourth derivatives the behavior is dominated by the behavior of the derivatives of the partition of unity functions.

Figure 2.7 shows the principal curvature directions, Gaussian curvature, mean curvature at various charts on several examples.

Figure 2.8 shows several examples of surfaces obtained from various control meshes. In all cases, overall quality is quite similar to Catmull-Clark surfaces; as expected, with smoother reflection lines near extraordinary vertices as in Figure 2.5.

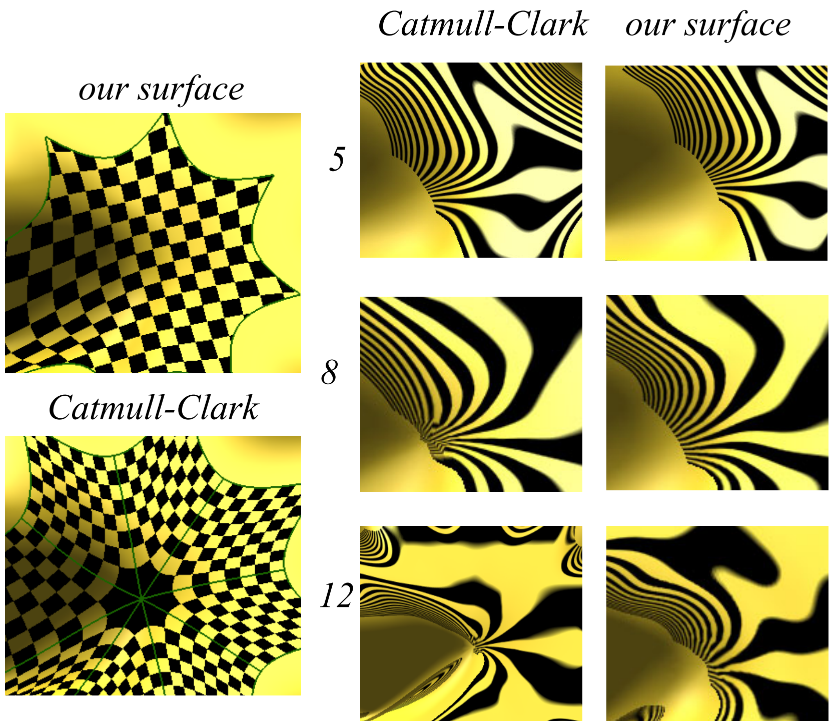


Figure 2.5: Left: comparison of parameterizations: chart for our surface and Stam's for Catmull-Clark. Right: Surface behavior near extraordinary points for valence 5,8,12.

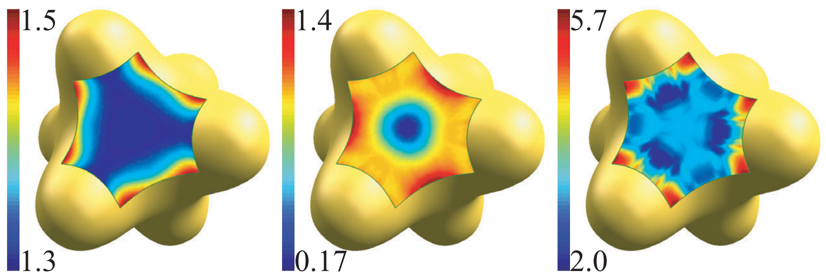


Figure 2.6: Maps of the total derivative magnitudes for the first, second and third derivatives.

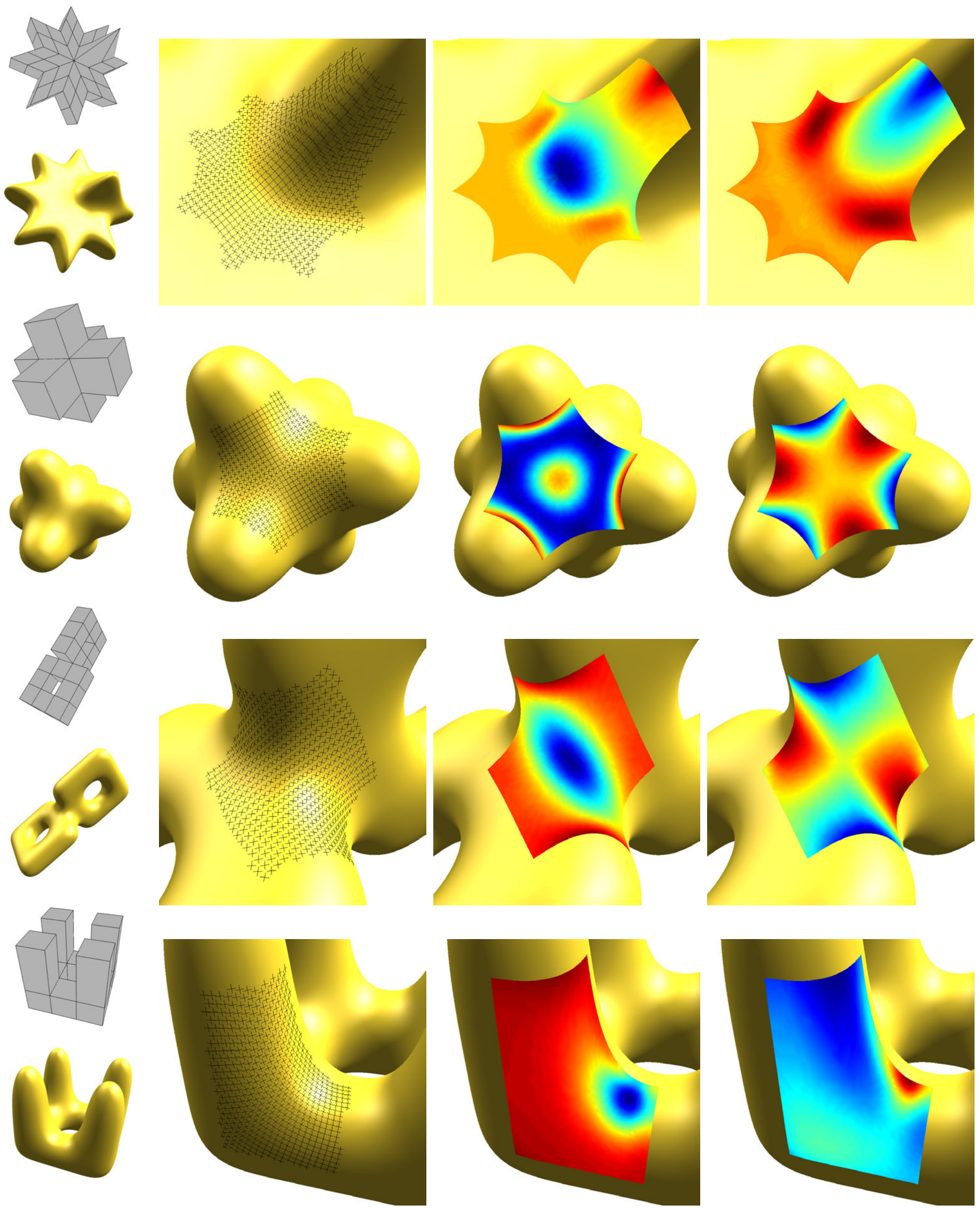


Figure 2.7: Principal curvature directions, Gaussian curvature and mean curvature. For each example, the first figure shows the principal curvature directions in the chart of an extraordinary vertex. Every cross represents the two principal curvature directions. Notice the smooth transition of the directions (away from the umbilical points). The second and third figures show the Gaussian and the mean curvatures.

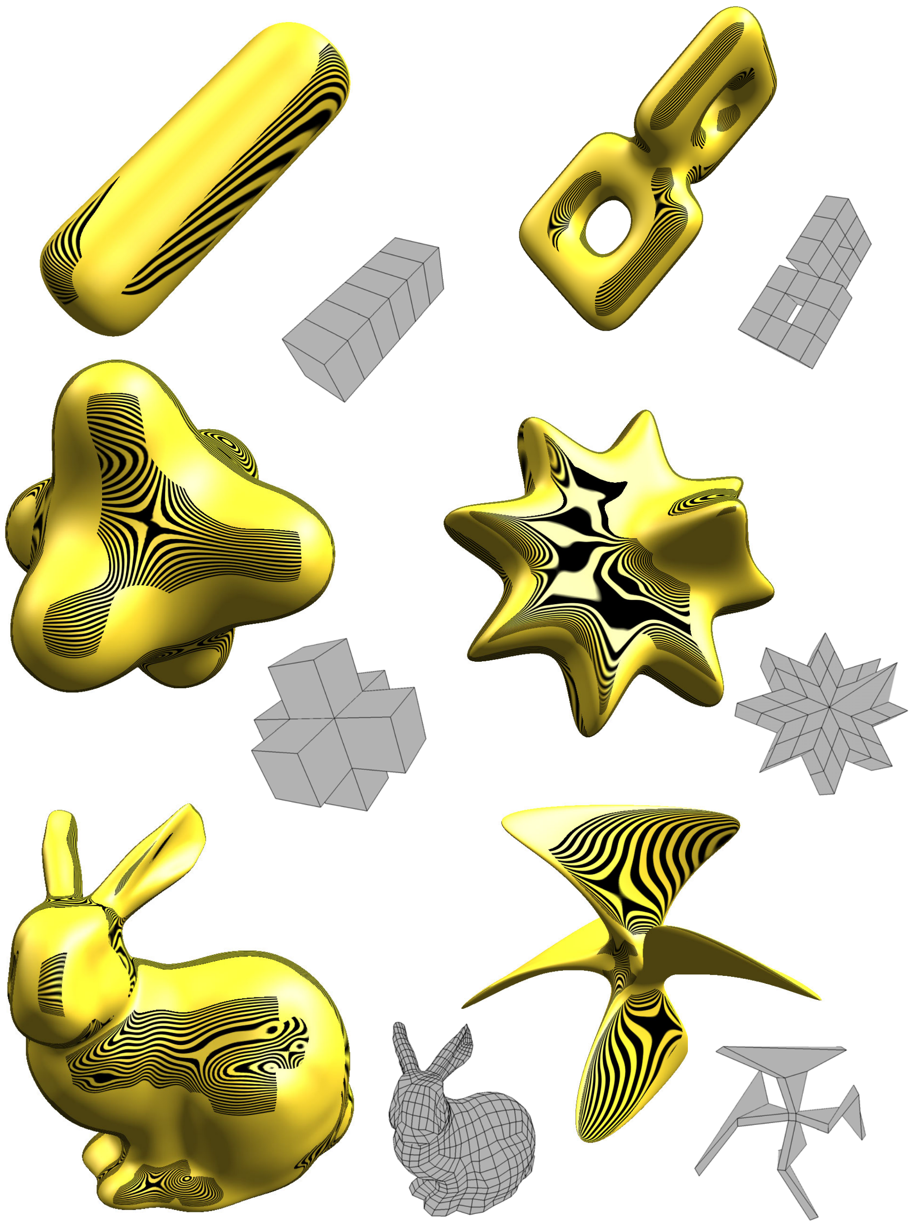


Figure 2.8: Several examples of surfaces produced with our method.

2.5 Summary

The development of the construction described in this paper was driven by the need for high-order surface representation from the the boundary integral solver for the Stokes equation. It nicely meets its needs while being a completely general tool. The surfaces generated by our construction are C^∞ smooth with explicit C^∞ parameterization, depend linearly and locally on control points and exhibits good visual quality.

Chapter 3

Kernel Independent Fast Multipole Method

This chapter describes a kernel-independent fast multipole method (FMM). This algorithm is used to evaluate the non-adjacent interaction in the Nyström solver of the boundary integral formulation of the Stokes equation.

3.1 Introduction

In attempting to evaluate the integrals deriving from the boundary integral formulation of the Stokes equations, we need to evaluate the non-adjacent interaction efficiently without compromising the accuracy of the integrator. After discretization, this problem becomes a special case of a more general problem: the evaluation of pairwise interaction on a large set of particles, where the interaction corresponds to the potential related to the fundamental solution of elliptic partial differential equations. Many other methods in computational physics (e.g., vortex methods, molecular dynamics) are also based on the evolution of particle systems with this pairwise interaction. The most important among these kernels is the single-layer Laplacian. Other kernels include the the kernels of the Helmholtz and Navier operators, their modified versions, and their derivatives (double-layer and hypersingular kernels).

Particle formulations result in dense linear algebraic systems because all pairwise interactions have to be computed. This is a significant bottleneck since for N particles it results in a $O(N^2)$ computation. In order to make large scale problems tractable it is essential to efficiently compute these interactions. A number of algorithms have been proposed for this purpose. The fast multipole method (FMM) has been one of the most successful, especially for non-uniform particle distributions.

The method presented in this chapter is a new kernel-independent FMM-like algorithm. Our algorithm has the structure of the adaptive FMM algorithm [32] but requires only the kernel evaluations, and it does not sacrifice the efficiency of the original algorithm. The crucial element of our approach is to replace the analytic expansions and translations with *equivalent density* representations. These representations are computed by solving local exterior and interior problems on circles (2D), spheres or cubes (3D) using the integral equation formulations. We demonstrate the efficiency of our method in both 2D and 3D for many kernels: the single and double layer potentials of the Laplacian, the modified Laplacian, the Navier, the Stokes, and their modified variants. Our method has $O(N)$ asymptotic complexity, and, like analytic FMM, works well for non-uniform particle distributions.

Synopsis of the new method. The basic structure of our method follows [35], the original fast multipole method, which we briefly review in Section 3.2. FMM consists of the following steps:

1. generation of a hierarchical tree partitioning of the computational domain;
2. accumulation of the multipole expansions for the far field by a postorder traversal of the tree;
3. translation of the multipole moments to the local expansions;
4. construction of local expansions by a preorder traversal of the tree;
5. evaluation of the far field action on the particles using local expansions;
6. evaluation of the near field interactions.

The same steps are used in our algorithm. However in the postorder traversal of the tree, the multipole expansion construction is replaced by solving local exterior inverse problems. To represent the potential generated by particles inside a box, we use a continuous distribution of an equivalent density on a surface enclosing the box. To find this equivalent density on the surface, we match its potential to the potential of the original sources at another surface in the far field. The translations are done by direct evaluation on the far field, sparsified with SVD or FFT. During the preorder traversal of the tree, we evaluate the far field interaction on a surface enclosing a target box, and solve an interior Dirichlet-type integral equation to compute an equivalent density. Then we use this density to represent the potential inside a target box.

Our method does not require implementation of analytic expansions for the kernel, it only requires their existence, and exclusively uses kernel evaluations. Like FMM, our algorithm is recursive and has an $O(N)$ complexity. Additional properties like scale invariance and rotational symmetries of kernels can be used to further accelerate the translation step, as in the case of the standard FMM.

Related work. The description of the original fast multipole algorithm can be found in [35], and [73]. Although the method is highly successful in two dimensions, the three-dimensional version of the original method was inefficient. Efficient extensions in three dimensions were realized only recently [20]. For these reasons many researchers tried to devise algorithms which were hybrids of tree codes and FMM, in order to combine the high accuracy of FMM methods with the simplicity of tree codes. In addition, the extension of the FMM to more general kernels like the modified Laplacian [33], the Stokes [28] , and the Navier [27, 95] operators can be quite cumbersome, due to the need to implement efficient translation operators. Below, we only review algorithms that could be used to develop kernel independent methods.

The idea of using a set of equivalent sources was first introduced in [1]. In that paper, the far field is represented as the solution to an exterior Dirichlet problem on a ball surrounding the particles using the exact Green's function (Poisson formula) for Laplacian. The method is somewhat easier than FMM to implement, but requires the analytic form of the Green's function for each kernel, which may not be available in the general case.

In [5] instead of using the exact Green's function, a number of equivalent densities are placed on a Cartesian grid in each source box; these densities are computed analytically by matching a number of multipole moments in the multipole expansion series of the original source densities. An important feature of this method is the fact that the Cartesian grid allows the use of FFT to accelerate the multipole to local-expansions translations. However, the method is not kernel-independent since for different kernels different expansions have to be constructed. The same idea is used in [50], and like in Anderson's method the densities are distributed over a ball containing the source box.

The idea of equivalent densities is also used in the precorrected FFT method, [66]. The equivalent densities are distributed over a regular grid, so that the far field convolutions can be computed with FFT instead of FMM. The term “precorrected” is related to the computation of the local interactions: the subtraction of the local influence of the equivalent densities and the addition of the near field interactions. The regular grid sources are computed by matching the field at selected checking points, usually located on a ball enclosing the original sources. In [16], a precorrected FFT method is applied to the Helmholtz kernel, but the equivalent sources are distributed along the faces of an enclosing cube, and three FFTs along the coordinate system planes are used to compute the far interaction. FFT-based methods are very efficient, often faster than FMM due to much smaller constants. For uniform distributions of particles FFT is likely to be preferable and it is kernel-independent. However, in the case of highly irregular particle distributions FMM is more efficient.

A hybrid method for kernel independent matrix-vector multiplication algorithm was proposed in [44] and [45]. Based on the fact that large blocks of the particle interaction matrix are low rank, this method uses singular value decomposition to sample and sparsify these blocks. It can be applied recursively and attains a $O(N \log N)$ complexity. We have applied this method on the Stokes and Navier operators [8] [7] with very satisfactory results in both accuracy and speed. One serious shortcoming of this method is the high setup cost. For problems with static particle distributions this is not a concern, but it becomes a bottleneck for problems with time evolving particles. The SVD approach was been further explored in series of papers [93, 94, 30] to obtain a kernel-independent method that does not require the kernel to be a solution of an elliptic PDE or a convolution. However, due to its generality, as the authors of these papers assert, the method does not achieve the efficiency of FMM for kernels that are related to fundamental solutions of PDEs.

Another method for fast matrix multiplication is based on higher-order Taylor expansions in Cartesian coordinates. This approach is not suitable for high accuracy computations because is computationally expensive (for p_{th}-order accuracy it requires $O(p^d)$ expansion terms). However, it is a kernel-independent method (the higher-order expansions can be easily obtained by differentiation). For example, it has been used to accelerate problems with the Stokes kernel [67].

As we pointed out in Chapter 1, another class of kernel-independent approaches used in solving boundary integral equations is based on wavelet decompositions, combined with a Galerkin scheme. This approach is quite promising, since it has the same complexity with FMM, and allows the constructions of efficient preconditioners for the resulting systems. However, it is hard to compare directly to FMM, as different trade-offs are made: FMM is a “bottom-up” approach, and is relatively insensitive to the distribution of samples. Adaptive wavelet methods are “top-down” but require samples to be located on a surface satisfying certain assumptions, which may not hold in the general case.

The rest of this chapter is organized as follows. In Section 3.2 we briefly review the classical FMM algorithm for the two dimensional Laplacian. In Section 3.3 we present the new algorithm and its implementation; in Section 3.4 we present an error analysis for the algorithm, and in Section 3.5 we present numerical results for several different scalar and vector kernels in two and three dimensions.

3.2 Review of the Fast Multipole Method

Given N source densities $\{\varphi_i\}$ located at N points $\{y_i\}$ in $\mathbf{R}^d$ $(d = 2, 3)$, we want to compute the potential $\{u_i\}$ at N points $\{x_i\}$ induced by a kernel G (single layer, double layer or other kernels of a elliptic PDE) using the following relation:

$$u_i = u(x_i) = \sum_{j=1}^{N} G(x_i, y_j)\varphi(y_j) = \sum_{j=1}^{N} G_{ij}\varphi_j,\ i = 1, \cdots, N.$$

We use x to refer to target locations and y to refer to source locations, but in general $\{x_i\}$ and $\{y_i\}$ can be the same set of points.

Direct implementation of this summation gives an $O(N^2)$ algorithm. For a large class of kernels and under reasonable assumptions on the particle distribution, FMM requires $O(N)$ work to compute an approximate potential with a prescribed relative error, [55], [20]. The constant in the complexity estimate depends on the relative error (the absolute error of the potential is bounded by the product of the relative error and the total charge).

We will use the single layer Laplacian kernel to describe FMM. In two dimensions we have $G(x, y) = -\frac{1}{2\pi}\log\rho$, with $r = x - y$, and $\rho = |r|$. In the FMM context it is convenient to use $G(x, y) = Re(\log(z_x - z_y))$ where z_x and z_y are complex numbers corresponding to x (target) and y (source) points on the plane. The idea of FMM is to encode the potentials of a set of source densities using the *multipole expansion* and *local expansion* at places far away from these sources.

Multipole expansion. Suppose the m source densities $\{\varphi_j\}$ located at $\{z_j\}$, with $|z_j - z_C| < r$, then for any z with $|z - z_C| > R$, the induced potential $u(z)$ can be approximated by:

$$u(z) = a_0 \log(z - z_C) + \sum_{k=1}^{p} \frac{a_k}{(z - z_C)^k} + O(\frac{r^p}{R^p}) \tag{3.1}$$

where $\{a_k, 0 \le k \le p\}$ satisfies

$$a_0 = \sum_{j=1}^{m} \varphi_j \quad \text{and} \quad a_k = \sum_{j=1}^{m} \frac{-\varphi_i (z_i - z_C)^k}{k}.$$

The vector of coefficients $\{a_k, 0 \le k \le p\}$ is called the multipole expansion.

Local expansion. Suppose the m source densities $\{\varphi_j\}$ located at $\{z_j\}$, with $|z_j - z_C| > R$, then for any $|z - z_C| < r$, the induced potential $u(z)$ can be approximated by:

$$u(z) = \sum_{k=0}^{p} c_k (z - z_C)^k + O(\frac{r^p}{R^p}) \tag{3.2}$$

where $\{c_k, 0 \le k \le p\}$ satisfies

$$c_0 = \sum_{j=1}^{m} \varphi_j \log(z_C - z_j) \quad \text{and} \quad c_l = \sum_{j=1}^{m} \frac{-\varphi_j}{l \cdot (z_j - z_C)^l}.$$

The vector of coefficients $\{c_k, 0 \le k \le p\}$ is called the local expansion.

In both expansions, p is usually a small constant determining from the desired accuracy of the result.

FMM employs the above representations in a recursive way. The computational domain, a box large enough to contain all source and target points, is hierarchically partitioned into a tree structure (a quadtree in 2D or an octtree in 3D). Each node of the tree corresponds to geometric box (square or cube). The tree is constructed so that the leaves contain no more than a prespecified number of points. For each box, the potential induced by its source densities is represented using a multipole expansion, while the potential induced by the sources from non-adjacent boxes is encoded in a local expansion. For a prescribed relative error ϵ, the number of expansion terms p is chosen to be $|\log_c \epsilon|$ where c is $(4-\sqrt{2})/\sqrt{2}$ in 2D and $(4-\sqrt{3})/\sqrt{3}$ in 3D.

Not only these expansions (multipole and local) can be used for efficient evaluation, but translations between these expansions are also available which make an $O(N)$ algorithm possible. In particular, the following types of translations are used:

M2M: The *multipole to multipole* translation transforms the multipole expansions of a box's children to its own multipole expansion.

M2L: The *multipole to local* translation transforms the multipole expansion of a box to the local expansion of another non-adjacent box.

L2L: Finally, *the local to local* translation of the local expansion of a box's parent to its own local expansion.

M2M translation. Suppose z_C is the center of a box and z_M is the center of its parent. Suppose further $\{a_k\}$ is the multipole expansion at z_C, then the multipole expansion at z_M can be written as:

$$u(z) = b_0 \log(z - z_M) + \sum_{l=1}^{p} \frac{b_l}{(z - z_M)^l} + O(\epsilon),$$

where $\{b_k, 0 \le k \le p\}$ satisfies

$$b_0 = a_0 \quad \text{and} \quad b_l = -\frac{a_0(z_C - z_M)^l}{l} + \sum_{k=1}^{l} a_k (z_C - z_M)^{l-k} \binom{l-1}{k-1}.$$

M2L translation. Suppose z_M and z_L are the centers of two non-adjacent boxes on the same level, $\{b_k\}$ is multipole expansion at z_M. Then the local exp-anion at z_L transformed from $\{b_k\}$ is:

$$u(z) = \sum_{l=0}^{p} c_l (z - z_L)^l + O(\epsilon),$$

where $\{c_k, 0 \leq k \leq p\}$ satisfies

$$c_0 = b_0 \log(z_L - z_M) + \sum_{k=1}^{p} \frac{b_k}{(z_M - z_L)^k}(-1)^k$$

$$c_l = -\frac{b_0}{l \cdot (z_M - z_L)^l} + \frac{1}{(z_M - z_L)^l} \sum_{k=1}^{p} \frac{b_k}{(z_M - z_L)^k} \binom{l+k-1}{k-1} (-1)^k.$$

L2L translation. Suppose z_T is the center of a box and z_L the center of its parent. Suppose further $\{c_l\}$ is the local expansion at z_L, then the local expansion at z_T can be written as

$$u(z) = \sum_{l=0}^{p} d_l (z - z_T)^l + O(\epsilon),$$

where $\{d_k, 0 \leq k \leq p\}$ satisfies

$$d_l = \sum_{k=l}^{p} c_k \binom{k}{l} (z_T - z_L)^{(k-l)}.$$

Using the tree structure, FMM consists of two basic steps. During the first step, the *upward pass*, the tree is traversed in postorder (i.e., the children of a box are visited before the box itself) to compute the multipole expansion for each box. At the leaves, the multipole expansions are built following Equation (3.1) (this procedure is also called the *source to multipole* (S2M) translation). At each non-leaf node, the multipole expansion is shifted from its children using the M2M translation. In the second step, the *downwards pass*, the tree is traversed in a preorder (i.e., the children of a box are visited after the box itself) to compute the local expansion. For each box B, the local expansion is the sum of two parts: first, the local-to-local transformation collects the local expansion of B's parent (the result condenses the contributions from the sources in all the boxes which are not adjacent to B's *parent*), and second, the multipole-to-local transformation collects the multipole expansions of the boxes which are the children of the neighbors of B's parent but are not adjacent to B (these boxes compose the *interaction list* of B). The sum of these two parts encodes all the contribution from the sources in the boxes which are not adjacent to B *itself*. At the end, for each box, the far interaction, which is evaluated using the local expansion at this box (this step is called the *local to target* (L2T) translation), is combined with the near interaction evaluated by iterating over all the source points in the neighborhood of the target box to obtain the potential (see Figure 3.1).

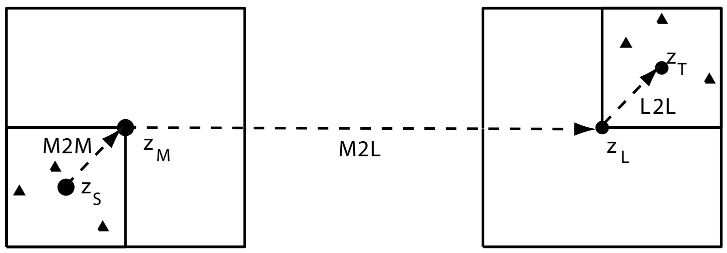


Figure 3.1: The multipole expansion at z_S encodes the influence from the source densities (marked with "+") to the far field. The local expansion at z_T encodes the influence from the far field to the target points (marked with "$\vartriangle$"). M2M translation transforms between the multipole expansions of the boxes in adjacent levels (z_S to z_M), M2L translation transforms multipole expansion of a box to the local expansion of non-adjacent boxes (z_M to z_L), and finally L2L translation transforms between local expansions between adjacent levels (z_L to z_T).

Instead of Laurent series, in three dimensions the far field is represented by spherical harmonics. There are several implementation details (mostly for the M2L transformation) that are required for efficient implementation (especially in 3D), but we do not mention them here. Overall, however, the organization of the computation is the same as the two dimension case. For the derivation of the expansions and a detailed discussion on error bounds and implementation details see [20] and [35].

3.3 The New Algorithm

Our algorithm is designed to generalize FMM to second-order constant coefficient non-oscillatory elliptic partial differential equations. Examples of such systems are given in Appendix A, where we also list the corresponding fundamental solution kernels. Such kernels satisfy the underlying PDE everywhere but the singularity location (pole), and are smooth away from the singularity. All problems under consideration admit a unique solution for the properly posed interior/exterior Dirichlet problems. Smoothness and uniqueness are the basic properties that we use to develop our FMM approximation.

Our algorithm has the same structure with the original FMM method. The differences are how the densities are represented efficiently and how the M2M, M2L, and L2L transformations are computed. We first describe these representations and transformations, then state the complete algorithm and conclude with a discussion on efficient implementation. Below we summarize the notation we use in the description of the method; these notations are defined in Section 3.3.1.

3.3.1 Density Translations

Given a set of N points, we define the computational domain to be a box large enough to contain all points. We construct a hierarchical tree (a quadtree in 2D and an octtree in 3D) so that each leaf of the tree contains no more than s points where s is a prescribed number. We assume that some points are labeled as sources y_i and other points as targets x_i. The source densities φ_i at the source locations y_i, $i = 1 \ldots N$ are given, and we want to evaluate the potential $\{u_i\}$ at the target locations $\{x_i\}$.

We refer to the tree nodes (squares in 2D and cubes in 3D) as boxes. For a spatial

B	a box in the computation tree
$\mathcal{N}^B$	the near range of the box B in $\mathbb{R}^d$
$\mathcal{F}^B$	the far range of the box B in $\mathbb{R}^d$
I_s^B	the set of of indices of source points or densities in B
I_t^B	the set of indices of target points or potentials in B
$y^{B,u}$	the upward equivalent surface of B
$\varphi^{B,u}$	the upward equivalent density of B
$x^{B,u}$	the upward check surface of B
$u^{B,u}$	the upward check potential of B
$y^{B,d}$	the downward equivalent surface of B
$\varphi^{B,d}$	the downward equivalent density of B
$x^{B,d}$	the downward check surface of B
$u^{B,d}$	the downward check potential of B
p	the degree of discretization for equivalent densities
s	the maximum number of source (or target) points allowed in a leaf box
N	the total number of source and target points
R	the depth of the computation tree
M	the total number of boxes in the computation tree

Table 3.1: FMM related notations.

region R, we use I_s^R and I_t^R to denote the index sets of the source and target points in R. Most commonly, R is a box of the computational tree.

If B is a box centered at c and has side length $2r$, then the box centered at c with side length $6r$ is called the *near range* of B and is denoted by $\mathcal{N}^B$. $\mathbb{R}^d \setminus \mathcal{N}^B$ is called the *far range* and is denoted by $\mathcal{F}^B$. Note that in our definition, B is a part of $\mathcal{N}^B$.

Equivalent densities and check potentials. We represent the potential in $\mathcal{F}^B$ from the source densities $\{\varphi_i, i \in I_s^B\}$ in B as the potential from a density distribution $\varphi^{B,u}$ supported at prescribed locations $y^{B,u}$ in $\mathcal{N}^B$ (Figure 3.2). We call $\varphi^{B,u}$ the *upward* *equivalent density* and $y^{B,u}$ the *upward equivalent surface* of box B.

Results from potential theory put two restrictions on the positions of $y^{B,u}$ (see [47], chapter 6). First, to guarantee the smoothness of the potential produced by $\varphi^{B,u}$, its support $y^{B,u}$ should not overlap with $\mathcal{F}^B$. Second, to guarantee that $\varphi^{B,u}$ is able to represent the potential produced by any source distribution in B, $y^{B,u}$ needs to enclose B. Therefore, in order to ensure the existence of $\varphi^{B,u}$, $y^{B,u}$ is required to lie between B and the boundary of $\mathcal{F}^B$. We use a circle in 2D and a sphere or cube in 3D for reasons that will be explained later.

The potentials induced by the source densities and the upward equivalent density satisfy the underlying second order linear elliptic PDE. As the solution of an exterior Dirichlet problem for such PDE is unique, these two potentials are guaranteed to be equal in all of $\mathcal{F}^B$ if they coincide at the boundary of $\mathcal{F}^B$, or any surface between $\mathcal{F}^B$ and $y^{B,u}$. We call such an intermediate surface the *upward check surface* and denote it by $x^{B,u}$. We call the potential computed on this surface the *upward check potential* and denote it by $u^{B,u}$. These surfaces are also chosen to be circles in 2D, and spheres or cubes in 3D. The equality of potentials on the upward check surface can be written as follows:

$$\int_{y^{B,u}} G(x,y)\varphi^{B,u}\,\mathrm{d}y = \sum_{i\in I_s^B} G(x,y_i)\varphi_i = u^{B,u}, \text{ for any } x \in x^{B,u}. \tag{3.3}$$

Similarly, we represent the potential in B from the source densities in $\mathcal{F}^B$ as the potential induced by a density distribution $\varphi^{B,d}$ defined at prescribed location $y^{B,d}$ in $\mathcal{N}^B$ (Figure 3.2). We call $\varphi^{B,d}$ *downward equivalent density* and $y^{B,d}$ *downward equivalent surface*. To ensure the existence of $\varphi^{B,d}$, $y^{B,d}$ needs to be located between $\mathcal{F}^B$ and B. As the solution of the interior Dirichlet problem for the PDE we consider is also unique, we need to match the potentials only on a surface $x^{B,d}$ between B and

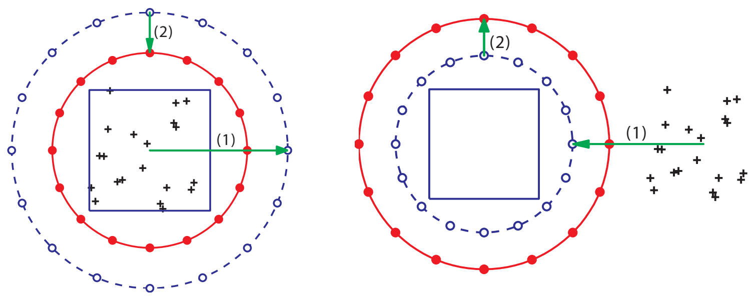


Figure 3.2: The equivalent/check surfaces in 2D. *Left*: Given the potential generated by the source densities inside a box, located at the points marked with "$+$", we represent it by using the upward equivalent density located at the upward equivalent surface. The equivalent surface is shown as the solid circle enclosing the box. The upward check potentials induced by the sources and the upward equivalent density are matched at the upward check surface (the dashed circle). *Right*: To represent the potential in the box generated by the source in the far range, we use the downward equivalent density located at the downward equivalent surface. The downward equivalent potentials induced by both sources are matched at the upward check surface. In both plots, the discretization points of the equivalent and check surfaces are equally spaced and marked with "●" and "○" respectively. For both upward or downward steps, the computation of the equivalent density includes two steps shown by arrows in each plot: (1) the evaluation of the check potential using the original source, and (2) the inversion of the integral equation to obtain the equivalent density.

$y^{B,d}$. We call the surface $x^{B,d}$ *downward check surface*, and the matched potential $u^{B,d}$ the *downward check potential*.

We usually choose both $y^{B,d}$ and $x^{B,d}$ to be circles in 2D and spheres or cubes in 3D. The potential $y^{B,d}$ satisfies the following equation for any $x \in x^{B,d}$:

$$\int_{y^{B,d}} G(x,y)\varphi^{B,d}\,\mathrm{d}y = \sum_{i \in I_s^{\mathcal{F}B}} G(x,y_i)\varphi_i = u^{B,d}. \tag{3.4}$$

The integral equations (3.3) and (3.4) are the first-kind Fredholm equations. Inverting such equations for a general right-hand side is an ill-conditioned problem since it is an ill-posed infinite dimensional problem. However, the right-hand sides have a special form that guarantees the existence of the solution of the integral equation. To solve these equations numerically in a stable way, we use a regularization scheme, as discussed in Section 3.3.2.

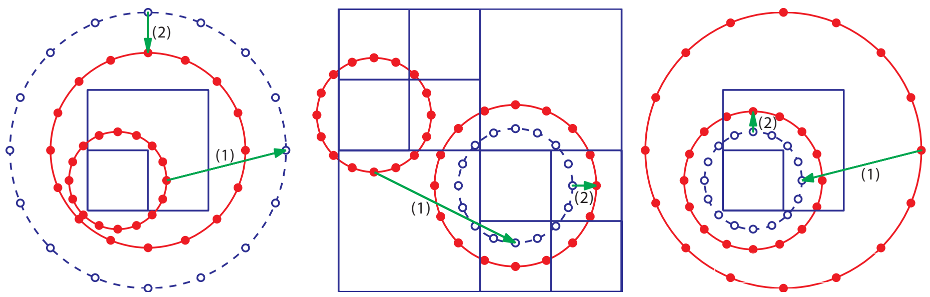


Figure 3.3: Three translations in 2D. *Left*: M2M translation. To compute the upward equivalent density of the large square, we evaluate the (upward check) potential at the dashed circle using its child box's upward equivalent density at the small solid circle (this operation is marked with arrow (1)), and invert the integral equation to get its upward equivalent density at the large solid circle (marked with arrow (2)). *Middle*: M2L translation transforms the upward equivalent density of the left box (surrounded by one circle) to the downward equivalent density of the right box (surrounded by two circles). We first evaluate the downward check potential at the dashed circle using the upward equivalent density (located at the small solid circle) (marked with (1)), and then invert the equation to obtain the downward equivalent density at the downward equivalent surface — the large solid circle (marked with (2)). *Right*: L2L translation transforms the downward equivalent density of the large box to its child — the the small box.
In all three figures, the discretization points for the equivalent surface are marked with "•" and the ones for check surface are marked with "○".

M2M translation. For every leaf box B in the tree, the computation of the upward equivalent density $\varphi^{B,u}$ from the source densities uses equation (3.3). The procedure of M2M translation is similar (Figure 3.3). To translate the upward equivalent density from a box A to its parent box B, we solve the following equation for $\varphi^{B,u}$:

$$\textbf{M2M:} \int_{y^{B,u}} G(x,y)\varphi^{B,u}(y)\,\mathrm{d}y = \int_{y^{A,u}} G(x,y)\varphi^{A,u}(y)\,\mathrm{d}y, \quad \text{for all } x \in x^{B,u}. \tag{3.5}$$

To ensure the existence of $\varphi^{B,u}$ for B, $y^{B,u}$ must enclose $y^{A,u}$ for any of its children A.

M2L translation. Once the upward equivalent density has been computed for each box, M2L translation computes the downward equivalent density (Figure 3.3). Suppose A is a box in $\mathcal{F}^B$. The M2L translation is similar to (3.4), and we solve the following equation to find $\varphi^{B,d}$:

$$\textbf{M2L:} \int_{y^{B,d}} G(x,y)\varphi^{B,d}(y)\,\mathrm{d}y = \int_{y^{A,u}} G(x,y)\varphi^{A,u}(y)\,\mathrm{d}y, \quad \text{for all } x \in x^{B,d}. \tag{3.6}$$

To ensure the existence of $\varphi^{B,d}$, $y^{B,d}$ must be disjoint from $y^{A,u}$ for all A in $\mathcal{F}^B$.

L2L translation. The L2L translation computes the downward equivalent density of a box B at level i from that of its parent A at level $i-1$ (Figure 3.3). The procedure is again similar to equation (3.4). The potential $\varphi^{B,d}$ satisfies

$$\textbf{L2L:} \int_{y^{B,d}} G(x,y)\varphi^{B,d}(y)\,\mathrm{d}y = \int_{y^{A,d}} G(x,y)\varphi^{A,d}(y)\,\mathrm{d}y, \quad \text{for all } x \in x^{B,d}. \tag{3.7}$$

To ensure the existence of $\varphi^{B,d}$, $y^{B,d}$ must lie in $y^{A,d}$.

Equations (3.5), (3.6) and (3.7) corresponding to M2M, M2L and L2L translations are all ill-conditioned for an arbitrary right-hand side. However, similar to (3.3) and (3.4), the right hand sides in our case are sufficiently smooth to guarantee the existence and stability of the solution of the integral equation.

Summary. We have described two density representations and three translations used to convert between these densities. The two equivalent densities correspond to the multipole and local expansions in FMM, while the three translations replace the three transformations in FMM.

In order to guarantee the existence of the equivalent densities the equivalent and check surfaces have to satisfy certain restrictions. We summarize them as follows: for each box B

- $y^{B,u}$ and $x^{B,u}$ lie between B and $\mathcal{F}^B$; $x^{B,u}$ encloses $y^{B,u}$;
- $y^{B,d}$ and $x^{B,d}$ lie between B and $\mathcal{F}^B$; $y^{B,d}$ encloses $x^{B,d}$;
- $y^{B,u}$ encloses $y^{A,u}$ for any descendant box A,
- $y^{B,u}$ is disjoint from $y^{A,d}$ for all A in $\mathcal{F}^B$,
- $y^{B,d}$ lies inside $y^{A,d}$, where A is B's parent.

3.3.2 Discretization

Regularization. Equations (3.3), (3.5), (3.6), and (3.7) need to be discretized. Each one of them consists of two steps. First, we need to evaluate the check potential at box B using the equivalent density from box A. This step is discretized using a simple numerical quadrature. Second, we need to compute the equivalent density at B from the check potential computed in the previous step. This requires the numerical solution of a first-kind Fredholm equation. We denote this equation as

$$K\varphi = u$$

where φ is the equivalent density of B, u is the check potential of B and K evaluates u from the kernel and φ. We solve this equation using Tikhonov regularization [47]:

$$\varphi = (\alpha I + K^* K)^{-1} K^* u.$$

This becomes a second-kind Fredholm integral equation, and in our implementation we solve it using the Nyström method (Galerkin or collocation methods could be used).

Surface geometry and discretization. The above two steps need to discretize the equivalent surfaces and check surfaces. In 2D we choose circular equivalent and check surfaces. We use the trapezoidal rule to integrate the check potential and to discretize the integral equations; in this manner we obtain super-algebraic convergence. In 3D this is no longer possible: to the best of our knowledge, there are no simple quadrature rules for functions defined on spheres that converge super-algebraically. Instead, we use cubes as the equivalent and check surfaces (Figure 3.4 and 3.5), and construct quadratures of fixed order on the faces of the cubes. In Section 3.3.4 we explain how this approach facilitates fast M2L translations, and in Section 3.5 we show that the accuracy in 3D is not too different from the 2D case.

2D case. For a box B centered at c with side length $2r$, all related surfaces are circles centered at c. The upward equivalent surface $y^{B,u}$ has radius $(\sqrt{2}+d)r$ where d a fixed number satisfying $0 \leq d \leq \frac{4-\sqrt{2}}{3}$. The upward check surface $x^{B,u}$ has radius $(4-\sqrt{2}-2d)r$. The downward equivalent surface $y^{B,d}$ has radius $(4-\sqrt{2}-2d)r$. Finally, the downward check surface $x^{B,d}$ has radius $(\sqrt{2}+d)r$ (Figure 3.2 and 3.3). Note that our choice of the surfaces satisfies all restrictions at the end of Section 3.3.1. All circles are discretized with p equally spaced points with trapezoidal rule. The accuracy of our method is determined by the choice of p. This simple rule is known to have super-algebraic convergence for smooth functions. d is chosen to be quite small (equal to 0.1 in our implementation). By doing so, the equivalent surface and check surfaces involved in each translations are well-separated and the kernel used in the check potential integration step is very smooth. Therefore the trapezoidal rule gives good accuracy in the integration of check potential.

Remark 3.1. We could have chosen the upward/downward check surface to be identical with the upward/downward equivalent surface. However, in this case the integral equation would have a kernel-dependent form and we would need more complex quadrature rules that can be used to integrate singular kernels.

3D case. For a box B centered at c with side length $2r$, all the related surfaces are the boundaries of cubes centered at c. The upward equivalent surface $y^{B,u}$ is the boundary of a box with halfwidth $(1+d)r$ where $0 \leq d \leq \frac{2}{3}$. The upward check surface $x^{B,u}$ is the boundary of a box with halfwidth $(3-2d)r$. The downward equivalent surface $y^{B,d}$ is the same as $x^{B,u}$. Finally, the downward check surface $x^{B,d}$ is the same as $y^{B,u}$ (Figure 3.4 and 3.5). These surfaces satisfy the restrictions at the end of Section 3.3.1. For every surface, the quadrature points are distributed evenly on six faces, and on every face, the points are distributed on an evenly spaced 2D Cartesian grid. Under this distribution, the quadrature points at the corner of the box are shared by three faces, and those at the edge of the box are shared by two faces. We can also view these quadrature points as the boundary nodes of a 3D regular Cartesian grid. Similarly to the 2D case, we use p to denote the total number of quadrature points on the surface of the box. Note that in 3D analytic FMM p is used to denote the order of the multipole/local expansion, therefore, p^2 is the actual number of coefficients used in the expansion. The quadrature weights are chosen in a way such that on every face the quadrature rule integrates low order 2D polynomials exactly. In our experiments, good quadrature results are observed since all the kernels are smooth away from the singularity. The parameter d is chosen to be quite small (again equal to 0.1 in our implementation) due to the reason stated in the 2D case.

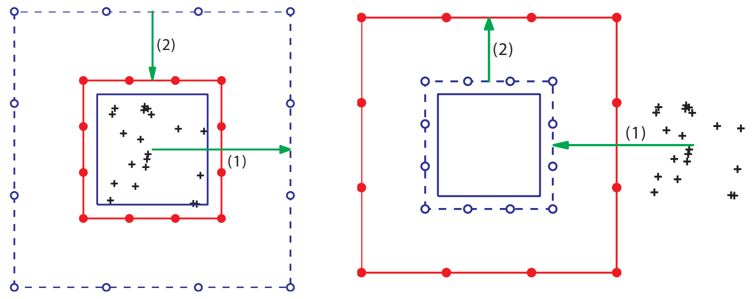


Figure 3.4: The cross sections of the equivalent/check surfaces in 3D. *Left*: the upward equivalent density. *Right*: the downward equivalent density. In both plots, the innermost square is the source box. The equivalent and check surfaces are both discretized using the boundary nodes of a regular Cartesian grid. The nodes for the equivalent surfaces are marked with "●" and those for the check surfaces with "○".

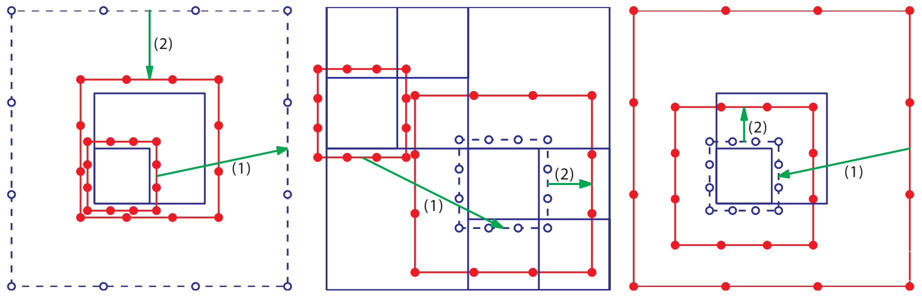


Figure 3.5: Three translations in 3D. *Left*: M2M translation. *Middle*: M2L translation. *Right*: L2L translation. 3D translations are similar to 2D. There are two differences: (1) equivalent/check surfaces are now cubes and (2) discretization points are the boundary nodes of a regular Cartesian grid. Note that for M2L translation the discretization points of upward equivalent surface and downward check surface are from the same Cartesian grid, therefore it can be sped up with FFT (interior nodes are padded with zero density).

Summary. Each one of the discretized M2M, M2L and L2L translations involves a potential evaluation and a solution of an integral equation. However, by choosing the quadrature points fixed relative to the box, both the evaluation and the solving depend only the level and the relative positions of the boxes involved in these translations. We can precompute and store these operators for each level and each relative position. Therefore, each translation invokes two matrix multiplications.

3.3.3 The Complete Algorithm

In this section we describe our algorithm in detail. First we give some definitions related to the algorithm. Our definitions closely follow Greengard [32].

Definitions. The neighbors of a box are adjacent boxes in the same level. For uniform distributions of particles, a uniformly refined grid is used. In this case the *neighbor list* L_N^B of a box B is the set of all neighbors of B and B itself. For a box away from the boundaries, the neighbor list contains 9 boxes in 2D or 27 boxes in 3D. These boxes are all contained in $\mathcal{N}^B$.

The *interaction list* L_I^B is the set of children of the neighbors of B's parent which are not B's neighbors. Again, ignoring the boundary effects, this list contains 27 boxes in 2D and 189 boxes in 3D. These boxes are all contained in $\mathcal{F}^B$.

If the particle distribution is uniform, a regular grid can be used; however, we are primarily interested in non-uniform particle distributions. In this case an adaptively refined grid is needed. The grid is recursively refined until the number of points in each leaf box is less than a fixed number s. Following the adaptive FMM algorithm, we give the following definitions (Figure 3.6).

For a leaf box B, the *U list* L_U^B contains B itself and the leaf boxes which are adjacent to B. For a non-leaf box, the U list is empty.

The V *list* L_V^B is the set of the children of the neighbors of the parent of B which are not adjacent to B.

If B is a leaf box, the W *list* L_W^B consists of all the descendants of B's neighbors whose parents are adjacent to B, but who are not adjacent to B themselves. For a non-leaf box, the W list is empty.

The X *list* L_X^B consists of all boxes A such that $B \in L_W^C$.

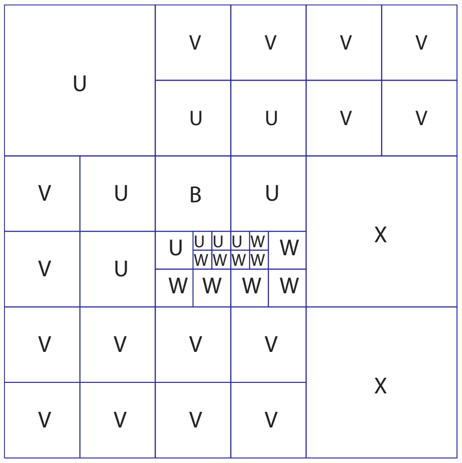


Figure 3.6: Lists L_U^B, L_V^B, L_W^B and L_X^B of box B.

For a leaf box B, L_U^B is similar to L_N^B in the uniform case, and L_V^B is similar to L_I^B. There is also a conjugate relation on these four lists. Suppose that A and B are two boxes.

- If A is in L_U^B, then B is in L_U^A.
- If A is in L_V^B, then B is in L_V^A.
- If A is in L_W^B, then B is in L_X^A.
- If A is in L_X^B, then B is in L_W^A.

For a box B, the U,V,W and X lists contain all boxes whose contribution needs to be processed by B itself. The contribution from more distant boxes are considered by B's ancestors. For a box U in L_U^B, a direct computation of the interaction of U's source points with B's target points is necessary since U and B are adjacent. For a box V in L_V^B, we compute the interaction from V to B using M2L translation since two boxes are well-separated. For a box W in L_W^B, we can evaluate the potential directly at B's target points using the upwards equivalent density of W, as B is in the far range of W. Finally, for a box X in L_X^B, since B is still in the near range of X, we represent the potential from X to B by first evaluating the potential at the downwards check surface at B and then invert it to the downwards equivalent density $\varphi^{B,d}$. The pseudocode is given in Figure 3.7.

```
STEP 1 TREE CONSTRUCTION
  for each box $B$ in preorder traversal of the tree do
    subdivide $B$ if $B$ has more than $s$ points in it
  end for
  for each box $B$ in preorder traversal of the tree do
    construct $L_U^B$, $L_V^B$, $L_W^B$ and $L_X^B$ for $B$
  end for

STEP 2 UPWARDS PASS
  for each leaf box $B$ in postorder traversal of the tree do
    evaluate $u^{B,u}$ at $x^{B,u}$ using $\{\varphi_i, i \in I_s^B\}$
    solve for $\varphi^{B,u}$ at $y^{B,u}$ that matches $u^{B,u}$ at $x^{B,u}$ (Equation (3.3))
  end for
  for each non-leaf box $B$ in postorder traversal of the tree do
    add to $u^{B,u}$ at $x^{B,u}$ the contribution from $\varphi^{C,u}$ for each child $C$ of $B$
    solve for $\varphi^{B,u}$ at $y^{B,u}$ that matches $u^{B,u}$ at $x^{B,u}$ (Equation (3.5))
  end for

STEP 3 DOWNWARDS PASS
  for each non-root box $B$ in preorder traversal of the tree do
    add to $u^{B,d}$ at $x^{B,d}$ the contribution from $\varphi^{V,u}$ for each box $V$ in $L_V^B$
    add to $u^{B,d}$ at $x^{B,d}$ the contribution from $\{\varphi_i, i \in I_s^X\}$ for each box $X$ in $L_X^B$
    add to $u^{B,d}$ at $x^{B,d}$ the contribution from $\varphi^{P,d}$, where $P$ is the parent of $B$
    solve for $\varphi^{B,d}$ at $y^{B,d}$ that matches $u^{B,d}$ at $x^{B,d}$ (Equation (3.6) and (3.7))
  end for
  for each leaf box $B$ in preorder traversal of the tree do
    add to $\{u_i, i \in I_t^B\}$ the contribution from $\varphi^{B,d}$
    add to $\{u_i, i \in I_t^B\}$ the contribution from $\{\varphi_i, i \in I_s^U\}$ for each box $U$ in $L_U^B$
    add to $\{u_i, i \in I_t^B\}$ the contribution from $\varphi^{W,u}$ for each box $W$ in $L_W^B$
  end for
```

Figure 3.7: Kernel independent FMM algorithm, adaptive case.

3.3.4 Implementation Issues

In the previous section we described the overall structure of the algorithms with some implementation details omitted for clarity. These details, however, are very important for an efficient implementation of any FMM method. The most important issues are the efficient acceleration of the M2L computation, and the overall memory management.

Another aspect of our discussion is the distinction between the setup phase and the fast summation phase. Many times the particle distributions come from discretization of integral equations; then, given a fixed spatial particle distribution, the summation is carried many times (i.e., the matrix vector multiplication within an iterative solver such as GMRES). Many issues that we discuss here are related to efficient multiple evaluations.

Acceleration techniques. In our complexity analysis, we consider only the uniform particle distribution and uniform grids. While analysis of adaptive refinement is possible it requires assumptions on particle distribution. We refer the reader to [55]. The most expensive part of our algorithm are the M2L translations: the evaluation of the contribution to $u^{B,d}$ of a target box B from $\varphi^{A,u}$ where A is a source box in the interaction list of B.

We denote the size of the interaction list by I. For a single box, the complexity of the M2L translation is $O(I \cdot p^2)$. The M2M and L2L translations are applied only once for each box and their contribution to the overall algorithm is not as important. Thus, the M2L part needs to be efficiently implemented since it is one of the two most expensive parts of the algorithm. (The other bottleneck is the computation of particle-to-particle dense interactions).

SVD-based acceleration (2D). In 2D, we use an SVD-based acceleration technique. We first assemble the matrix M of the interaction from $y^{A,u}$ to $x^{B,d}$. We observe that M is numerically low rank. The number of the significant singular values of M is small compared to the dimension of M, and the rest of the singular values are less than the accuracy required by the pairwise interaction evaluation. Suppose $USV^T = M$ is the SVD of M. We can store only the columns of U and V which correspond to the dominant singular values of S and discard the rest. This approach gives us an efficient representation of M. In 3D this approach does not yield satisfactory results. Although M2L operators are low rank, in practice the cutoff number of equivalent density points in which the compression is effective, is very large. For this reason an FFT-based approach is preferable.

FFT-based acceleration (3D). Suppose box A is in the interaction list of box B. As mentioned in Section 3.3.2, $y^{A,u}$ is chosen to be the boundary of A, and the integration points are the nodes of a Cartesian grid which are on the boundary of of A. The same is true for $x^{B,d}$. Therefore, by assigning zero density to the grid points in the interior of B we can view the evaluation of the potential $u^{B,d}$ from the density $\varphi^{A,u}$ as a 3D convolution. This convolution can be evaluated efficiently by FFT. Since we use 3D convolutions, there are $O(p^{3/2})$ instead of p densities and targets in each M2L translation. For each box, we carry out the FFT and inverse FFT only once, to obtain an $O(p^{3/2}\log(p))$ complexity. The convolution (pointwise vector multiplication) is applied I times for each box, with $O(I \cdot p^{3/2})$ complexity.

Several acceleration schemes for the M2L translation of the analytic FMM have been proposed in the past. In [25], a 2D FFT based scheme is used to transform the multipole coefficients to the local coefficients. This scheme gives a $O(p\log(p))$

complexity for each M2L translation. In [36] and [20] *exponential representation*, an intermediate representation between multipole and local expansions is introduced. Based on this new representation, a *diagonal transformation* is used to transfer between exponential expansions efficiently. This technique cuts down the complexity to $O(I \cdot p)$. An essential step of the translation to exponential representation is the computation of some nontrivial kernel-dependent quadrature weights. While both of these two schemes give asymptotically superior complexity than the $O(I \cdot p^{3/2})$ complexity of our FFT based acceleration technique, our FFT based technique only involves potential evaluations and thus is kernel independent.

Storage compression. Since the M2M, M2L and L2L translations are used repeatedly, we precompute and store the matrices of these operators. Three storage compression techniques are used to reduce the memory usage.

Homogeneity. Many kernels in the problems we are considering are homogeneous: if we scale the distance between the source point and the target point by a factor α, the potential at the target is amplified by a factor α^k, where k is a constant. For example, the 3D Laplace single layer kernel, $S(x, y) = \frac{1}{4\pi}\frac{1}{r}$, has this property. Since the integration points of the equivalent densities of a box are fixed relative to the box, the translation operators between different levels of the computation tree only differ by a constant, usually a power of 2. Hence, instead of storing the matrices for each level, we store only the matrices for a single level. Modified kernels, like modified Laplace, modified Stokes and modified Navier equations, do not have this property.

Symmetry. In 2D the integration points are equally spaced on a circle; in 3D the integration points of the equivalent densities are chosen to be the nodes of a regular Cartesian grid. In both cases they are symmetric with respect to the x, y and z axes. For example, if we flip the positive x direction to be the negative x direction, the positions of the set of the integration points do not change, even though two integration points might swap their positions. Consider the M2M translation: Suppose B is the parent box of two different boxes C_1 and C_2 and we need to evaluate the potential $u^{B,u}$ at $x^{B,u}$, the contribution from $\varphi^{C_1,u}$ at $y^{C_1,u}$ and from $\varphi^{C_2,u}$ at $y^{C_2,u}$. Further suppose we already have the matrix of the operator from $y^{C_1,u}$ to $x^{B,u}$. In order to evaluate the contribution from $\varphi^{C_2,u}$ at $x^{B,u}$, we first perform a change of coordinates to move $y^{C_2,u}$ to $y^{C_1,u}$, and then evaluate the contribution using the operator from $y^{C_1,u}$ to $x^{B,u}$. We then perform another change of coordinates to move $y^{C_1,u}$ back to $y^{C_2,u}$. The same techniques can be carried out for M2L and L2L translations.

The above procedure is only correct in the case of a scalar density and a scalar potential. In the cases with vector or tensor densities and potentials, the change of coordinates not only affects the support of the density or potential, but it also modifies their values. Therefore, a rescaling step is necessary after each change of coordinates. A general translation using symmetry involves five steps: (a) forward change of coordinates, (b) rescaling of density, (c) translation using stored matrix, (d) rescaling of potential, and (e) backward change of coordinates. This technique works for all the kernels listed in the appendix, and gives us a compression factor of eight in 3D and four in 2D.

Lazy computation. In the case of non-uniform density distribution, the depth of the computation tree can be quite large. However, not all the M2L translations are actually needed in the computation. Therefore, in our algorithm, the matrix representation of a M2L translation is only computed where it is actually needed by some box. This *lazy computation* strategy results in significant savings on memory usage in non-uniform density distributions, and modified kernels.

3.3.5 Complexity

The analysis of the adaptive algorithm is essentially the same, but more involved and requires assumptions about the particle distribution. For simplicity, we give the complexities of our method and FMM in [20] for 3D uniform particle distribution. The number of boxes M is approximately N/s. We use p to denote the number of coefficients.

Step	Our method	FMM
S2M translation	$O(Np + Mp^2)$	$O(Np)$
M2M translation	$O(Mp^2)$	$O(Mp^{3/2})$
M2L translation	$O(Mp^{3/2}\log p + 189Mp^{3/2})$	$O(20Mp^{3/2} + 189Mp)$
L2L translation	$O(Mp^2)$	$O(Mp^{3/2})$
L2T translation	$O(Np)$	$O(Np)$
Near Interaction	$O(27Ns)$	$O(27Ns)$

Table 3.2: Complexity comparison of our method and analytic FMM.

The hidden constants in the complexity estimates are approximately the same for all translations; 189 is the number of the M2L boxes and 27 is the number of boxes in the near interaction. In practice, s is of the same order as p. Therefore, the S2M and L2T steps of both methods are of the same order $O(Np)$. Our M2L translation is also of the same order as that of [20]. The M2M and L2L steps have higher complexity in our method, due to the fact that no acceleration techniques are applied in these two steps. However, in all experiments in Section 3.5, we observe that this does not slow down our method significantly since these steps are applied once for each box.

3.4 Error Analysis

Given the direct interaction operator G between the sources in a box B at level l and targets in a well-separated target box A at level m, we examine the error related to the FMM approximation. First, we show that our FMM acceleration can be viewed as a factorization of G, provided that all integrations are carried out exactly. Then we present analysis of the discretization error behavior for homogeneous kernels from scale invariant PDE in 2D case. The scale invariance means that the PDE only involves the second order derivatives of the potential variable, such PDEs includes Laplace, Stokes and Navier equations.

Numerical results indicate that the method works well in 3D and for inhomogeneous kernels; we leave derivation of rigorous error bounds in these cases as future work.

It is important to point out that here we prove an error bound of the FMM approximation of the interaction operator G. This error is a relative error in the sense that the absolute error for the computed potential is bounded by the product of the relative error with the magnitude of the exact potential.

3.4.1 FMM Factorization

FMM can be viewed as a factorization of the operator G. Suppose the M2L translation operator is applied at level k when the interaction between A and B is evaluated. Let $B = B_l, B_{l-1}, \dots, B_k$ be the sequence of ancestor boxes of B up to level k, and $A = A_m, A_{m-1}, \dots, A_k$ the sequence of ancestor boxes of A. For our purposes, it is convenient to consider a single sequence of boxes, $B_l, \dots B_k, A_k, \dots A_m$, of length $l + m - 2k + 2$; we denote this single sequence $\{C_i\}$, $i = 0 \dots n + 1$, where $n = l + m - 2k$. With each box C_i, we associate an equivalent surface y_i and a check surface x_i, with equivalent density φ_i defined on y_i and potential u_i defined on x_i. For boxes B_i upward surfaces are used, and for boxes A_i downward surfaces are used.

We introduce sequences of operators K_i and E_i mapping densities defined on equivalence surfaces to potentials defined on check surfaces. These operators correspond to left and right-hand sides of (3.5), (3.6) and (3.7). We use an auxiliary operator $\mathcal{K}[Y \to X] : C(Y) \to C(X)$, where Y and X are regions in 2D or 3D (typically surfaces or boxes). The operator $\mathcal{K}$ is defined by

$$(\mathcal{K}[Y \to X]f)(x) = \int_Y G(x,y)f(y)\,\mathrm{d}y \quad \text{for } x \in X.$$

Then

$$K_i = \mathcal{K}[y_i \to x_i], \quad E_i = \mathcal{K}[y_i \to x_{i+1}], \quad L_i = E_i K_i^+. \tag{3.8}$$

where $K_i^+ = (K_i^* K_i)^{-1} K_i^*$ is the pseudo-inverse of K_i.

Finally, let $D = \mathcal{K}[y^{A,u} \to A]$, the operator evaluating the density on the upward equivalent surface of $A = C_{n+1}$ at an arbitrary point inside A. Using these operators, evaluation of the potential u_A at the target box due to the sources in B using our hierarchical decomposition can be written in the following form:

$$u^A_{hier} = DK^+_{n+1}E_n \dots E_0 K^+_0 u^{B,u}. \tag{3.9}$$

As illustrated in Figure 3.8,the first sequence part of the sequence of operators corresponds to the upward traversal of the tree, with the M2M translation defined by (3.5) applied on each step. It is followed by the M2L translation (3.6) and the downward traversal with the L2L translation (3.7) applied on each step. Since the kernels are homogeneous, the operators K_i and E_i are level-independent of C_i up to an identical scale factor, and the composition $L_i = E_i K^+_i$ is level-independent as these factors cancel. For such kernels, we rescale E_i and K_i to make them completely level-independent.

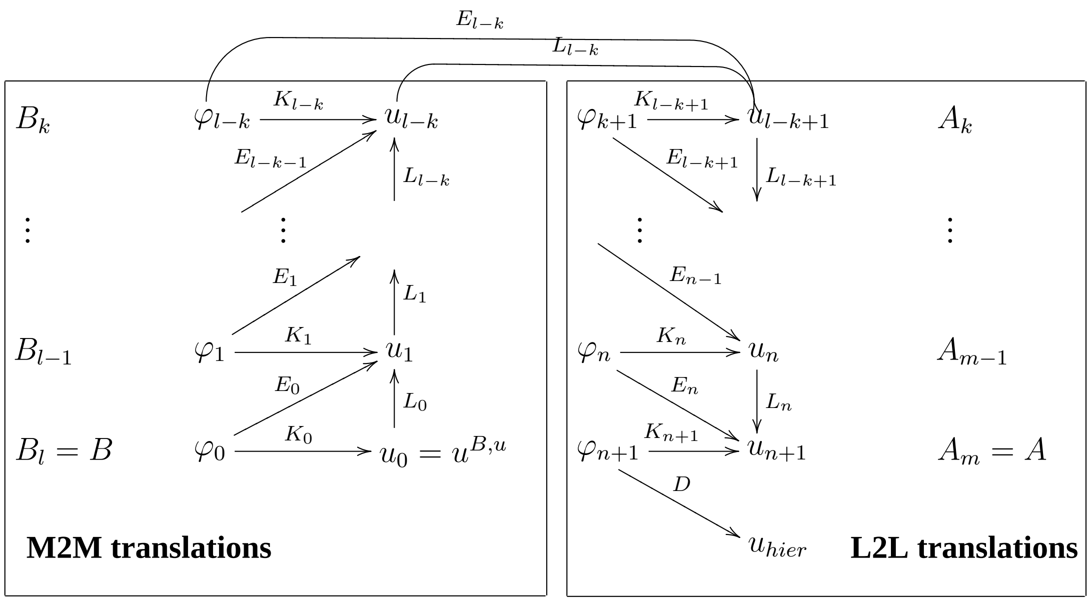


Figure 3.8: Operators used in the error analysis.

In comparison, direct evaluation yields

$$u_{direct} = \sum_{i \in I_s^B} G(x, y_i)\varphi_i.$$

Expression (3.9) can be viewed as a sequence of transformations of densities, starting with $\varphi_0 = \varphi^{B,u}$ to $\varphi_{n+1} = \varphi^{A,d}$, defined on the sequence of upward and downward equivalent surfaces. Let $\{D_i\}$ be the sequence of nested open domains with boundaries x_i: $Ext(x^{B_l,u}) \supset \ldots \supset Ext(x^{B_k,u}) \supset Int(x^{A_k,d}) \supset \ldots \supset Int(x^{A_m,d}) \supset A$ (for the upward traversal, we use exterior domains, for the downward traversal, interior). Similarly we define $\{F_i\}$ to be the sequence of the nested open domains with boundary y_i: $B \subset Int(y^{B_l,u}) \subset \ldots \subset Int(y^{B_k,u}) \subset Ext(y^{A_k,d}) \subset \ldots \subset Ext(y^{A_m,d})$.

It is sufficient to show that the potential u_i^{vol} in D_i induced by φ_i, $u_i^{vol} = \mathcal{K}[y_i \to D_i]\varphi_i$, is equal to u_{i+1}^{vol} in $D_{i+1} \subset D_i$, and that the potential induced by the first density φ_0 is the same as u_{direct} in D_0, the exterior of $x^{B,u}$. Equivalence of u_{hier}^A and u_{direct} in the interior of A follows by induction.

The key is the observation that in the interior of D_i, u_i^{vol} satisfies the elliptic PDE for which the kernel $G(x, y)$ satisfies the underlying elliptic PDE. Therefore, we can regard it as the solution of the Dirichlet problem with boundary conditions $u_i^{vol}|_{x_i} = u_i$. The Dirichlet problem is exterior for upward check surfaces x_i and interior for downward surfaces x_i. In either case, from the uniqueness of the solution of the Dirichlet problem, it follows that the potential is defined uniquely by its boundary values. The density φ_{i+1} is computed from φ_i using $K_{i+1}\varphi_{i+1} = E_i\varphi_i$, i.e., the potentials induced by these densities on x_{i+1} are required to coincide. It follows that the potentials coincide in all of D_{i+1}. Similarly, φ_0 is computed using the condition that the induced potential coincides with $p^{B,u}$ i.e., u_{direct} evaluated at $x^{B,u} = x_0$;

therefore, u_0 coincides with u_{direct} in D_0.

3.4.2 Discretization Error

We present a qualitative error analysis in 2D, determining the dependence of the error on the tree depth l and the discretization error ϵ introduced at a single translation step. In 2D, the equivalent surfaces and check surfaces are chosen to be circles. Our analysis is carried out in the Sobolev spaces on a unit circle $H^t[0, 2\pi]$ for $t \geq 1$, which we denote H^t. We use $\|\cdot\|$ to denote the H^t norm. Since the kernel is C^∞ everywhere away from the singularity, u^A is in H^t for any t. Although the error is more naturally measured in L_2, H^t is a more convenient choice for analysis of our method, as the Nyström method for integral equations is norm-convergent in H^t for $t \geq 1$ in 2D. Note that this approach also yields an upper bound for the L_2 error, although this bound is likely to be too conservative.

We also define S_i, a subspace of H^t, with

$$S_i = \{\mathcal{K}[F_i \cup y_i \to x_i](u), u \in H^t\}. \tag{3.10}$$

Since the potential produced by the density in F_i can be represented by the one produced by the density on y_i, we can also write S_i to be $\{\mathcal{K}[y_i \to x_i](u), u \in H^t\}$.

To simplify the exposition, in our error analysis we omit the last step DK^+_{n+1} which introduces an additional fixed error due to solution of $K_{n+1}\phi_{n+1} = u_n$. Expression (3.9) with the last step excluded can be written as

$$u_n = L_n L_{n-1} \dots L_0 u_0. \tag{3.11}$$

We use notation $L^{(j:i)}$ for the composition $L_j L_{j-1} \dots L_i$ for $j \geq i$; we also abbreviate $L^{(j:0)}$ as $L^{(j)}$. We define $L^{(j:i)}$ to be the identity for $j < i$.

We use the following four auxiliary results in in our error analysis. The proofs of the first two lemmas can be found in the Appendix.

Lemma 3.1. *$E_i : H^t \to S_{i+1}$, $K_i : H^t \to S_i$ and $L_i : S_i \to S_{i+1}$ are all compact in the H^t norm.*

Lemma 3.2. *The H^t norm of any operator $L^{(j:i)} = L_j L_{j-1} \dots L_i : S_i \to S_{j+1}$ is uniformly bounded independently of i and j.*

Lemma 3.3. *Suppose P_n is a sequence of bounded symmetric operators from H^t to H^t with $P_n \to I$ pointwisely, and D is a compact operator also from H^t to H^t. Then sequences $P_n D$ and DP_n are norm convergent to D.*

Proof. Approximate D by a finite dimensional operator. □

Lemma 3.4. *In 2D, the Nyström method with trapezoidal rule is H^t norm convergent for second-kind Fredholm integral equations with smooth kernels.*

Proof. See Chapter 12 of [47]. □

The proofs of Lemma 3.1 and Lemma 3.2 are given at the end of this subsection.

As mentioned in Section 3.3.2, we use Tikhonov regularization to invert K_i. We introduce the regularized operator $\bar{L}_i$ as

$$\bar{L}_i = E_i(\alpha_i I + K_i^* K_i)^{-1} K_i^*,$$

and its Nyström discretization by

$$\tilde{L}_i = \tilde{E}_i(\alpha_i I + \tilde{K}_i^* \tilde{K}_i)^{-1} \tilde{K}_i^*.$$

$\tilde{K}_i$ is the discretization of K_i defined by $\tilde{K}_i f(x) = \sum_{r=1}^{p_i} w_i^r G(x, y_i^r) f(y_i^r)$ for $x \in x_i$, where u_i is the number of quadrature points and w_i^r and y_i^r are quadrature weights and discretization points respectively, $\tilde{E}_i$ is defined in the same way. It is important to notice that $\tilde{K}_i$ is from H^t to S_i since the quadrature points $\{y_i^r\}$ stay on y_i. Similarly, $\tilde{E}_i$ is an operator from H^t to S_{i+1}. Therefore, both $\bar{L}_i$ and $\tilde{L}_i$ are well-defined operators from S_i to S_{i+1}.

It can be shown that closure of S_i in H^t is the orthogonal complement of a finite number of functions. These functions span the null space of K. Therefore, L_i can be extended to be defined over the whole H^t by using continuity and assigning L_i to be zero operator on these finite number of functions. The norm of the extension of L_i is bounded by the H^t norm of L_i on S_i. The compactness of L_i is also preserved. Similarly, the same argument applies to $L^{(j;i)}$, $\bar{L}_i$ and $\tilde{L}_i$. All of them can be defined over H^t. The goal of our analysis is to estimate the H^t norm of $\tilde{L}^{(n)} - L^{(n)} = \tilde{L}_n\tilde{L}_{n-1}\dots\tilde{L}_0 - L_nL_{n-1}\dots L_0$.

Proof of Lemma 3.1

Proof. First, we prove the compactness of K_i and E_i. Since y_i and x_i are disjoint, the kernel G in K_i is C^∞ in both variables. Thus, K_i, as a convolution operator with C^∞ kernel, is compact in H^t norm. E_i is also compact in H^t norm since y_i is disjoint from x_{i+1}.

Now, we prove that L_i is compact in H^t norm (see Figure 3.9 for the domains involved). Suppose $u_i \in S_i$ on x_i, we can find $\varphi_i \in H^t$ on y_i such that $\varphi_i = K_i^+ u_i$.

Since $\mathcal{K}[y_i \to x_i](\varphi_i) = K_i\varphi_i = K_iK_i^+u_i = u_i$, $\mathcal{K}[y_i \to D_i](\varphi_i)$ is the solution of boundary value problem on domain D_i with boundary condition u_i. On the other hand, $u_{i+1} = E_i(\varphi_i) = \mathcal{K}[y_i \to x_{i+1}]$ is the solution of this problem on x_{i+1}. Hence, $L_i(u_i) = E_iK_i^+(u_i) = E_i(\varphi_i) = u_{i+1}$ is equivalent to the Poisson formula which evaluates the potential at x_{i+1} from the potential at x_i. The kernel in Poisson formula, which corresponds to the fundamental solution of the PDE with domain D_i, is C^∞

smooth since x_i and x_{i+1} are disjoint. This means that the Poisson formula represents a compact operator in H^t norm for any t. Therefore, L_i is a compact operator in H^t norm. $\square$

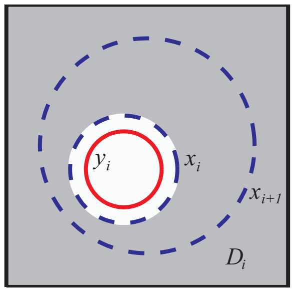


Figure 3.9: The domains used in the proof of Lemma 3.1 where L_i corresponds to a M2M translation. The grayed region is D_i.

To clarify the idea behind the proof, we give the analytic form of the M2M translation operator for a simplified case for the single layer potential for the 2D Laplacian. The main reason for the compactness is the inclusion of x_i in x_{i+1}.

We assume that the three surfaces y_i, x_i and x_{i+1} are concentric circles such that their radii ρ_i^e, ρ_i^c and ρ_{i+1}^c satisfy the condition $0 \le \rho_i^e \le \rho_i^c \le \rho_{i+1}^c$.

Standard logarithm expansion and simple algebraic manipulations yield

$$\log|x-y| = \log|x| + \sum_{k=-\infty, k\neq 0}^{\infty} \frac{(-1)^k}{|k|} \left(\frac{|y|}{|x|}\right)^{|k|} e^{ik\theta_x} e^{-ik\theta_y},$$

where θ_x and θ_y are the polar coordinate angles of the position vectors x and y respectively. If we assume that this kernel acts on the space of continuous periodic functions in $[0, 2\pi]$ with zero mean and we can drop the $\log|x|$ term. As the trigonometric functions are orthogonal on $L^2(0, 2\pi)$, the above expression is a diagonalization of the single layer operator. As the eigenvalues are all positive, they coincide with singular values.

First, we solve $K_i\varphi_i = u_i$. In this case, since $|x| = \rho_i^c \geq |y| = \rho_i^e$, the singular values decay exponentially, so the problem $K_i\varphi_i = u_i$ is ill-posed: small perturbations on the high frequency components of u^c get exponentially amplified. However, since u_i is the potential induced by the densities in the interior of y_i, φ_i is a well-defined function with the following relationship on Fourier coefficients:

$$\hat{\varphi}_i(k) = (-1)^k|k|\left(\frac{\rho_i^c}{\rho_i^e}\right)^{|k|}\hat{u}_i(k)$$

Second, we evaluate u_{i+1} with $E_i\varphi_i$. The Fourier coefficients of u_{i+1} are given by

$$\hat{u}_{i+1}(k) = \frac{(-1)^k}{|k|}\left(\frac{\rho_i^e}{\rho_{i+1}^c}\right)^{|k|}\hat{\varphi}_i(k) = \left(\frac{\rho_i^c}{\rho_{i+1}^c}\right)^{|k|}\hat{u}_i(k)$$

This expression actually gives the singular value decomposition of L_i using the trigonometric basis on the circle, where $\left(\frac{\rho_i^c}{\rho_{i+1}^c}\right)^{|k|}$ are the singular values of L_i. The singular values decay exponentially to zero since $\rho_i^c < \rho_{i+1}^c$, therefore L_i a compact operator (with analytic kernel).

Proof of Lemma 3.2

Proof. A product $L^{(k)}$, with k terms each one being an L operator, represents a sequence of M2M translations followed by a M2L translation and followed by a sequence L2L translations. To prove the lemma, we only need to show the existence of uniform bounds for the cases where $L^{(k)}$ corresponds to a sequence of M2M translations or a sequence of L2L translations. Here we prove the latter case. The proof for the other case is the same.

Suppose $L^{(k)}$ transforms $u^{A,d}$ at $x^{A,d}$ of box A into $u^{B,d}$ at $x^{B,d}$ of box B. Since $L^{(k)}$ only involves L2L translations, B is contained in A and it is k level deeper in the computation tree. Suppose A has halfwidth r, from Section 3.3.2, we know $x^{A,d}$ has radius $(\sqrt{2}+d)r$ and $x^{B,d}$ for any box B is contained in a circle which is concentric to $x^{A,d}$ and has radius $(\sqrt{2}+\frac{d}{2})r$. Hence, $x^{B,d}$ is always away from $x^{A,d}$ by a distance $\frac{d}{2}r$, which is independent of k.

As we pointed out in the proof of Lemma 3.1, the transformation $L^{(k)}$ can be viewed in a different way: it is equivalent to the Poisson formula which evaluates the potential at $x^{B,d}$ from the potential $u^{A,d}$ at $x^{A,d}$. The H^t norm of the Poisson formula grows to infinity only when $x^{B,d}$ and $x^{A,d}$ approach to each other. In our case, since $x^{B,d}$ and $x^{A,d}$ are separated by a distance $\frac{d}{2}r$ which is independent of k, the norm of $L^{(k)}$ is bounded from above uniformly. $\square$

3.4.3 Single Step Error

Our first step is to estimate the error $\tilde{L}_i - L_i$ for a single translation step. We split the error into two parts: $\bar{L}_i - L_i$ and $\bar{L}_i - \tilde{L}_i$.

We take H^t as a Hilbert space with the standard scalar product defined by $(f,g) = \sum_{i=0}^{t}\int_0^{2\pi} D^i f\, D^i g$. Since K_i is a compact operator in H^t (Lemma 3.1), for any $f \in H^t$ we can expand $K_i f$ as

$$K_i f = \sum_{r=0}^{\infty} \sigma_i^r (f, v_i^r) u_i^r$$

were $\{u_i^r\}$ and $\{v_i^r\}$ are orthonormal bases in H^t and σ_i^r are singular values of K_i. In operator form, this decomposition can be written as $U_i S_i V_i$, where $V_i : H^t \to l_2$ is defined by the map from f to the sequence $\{(f, v_i^r)\}$, $U_i : l_2 \to H^t$ maps a sequence of coefficients $\{a^r\}$ to $\sum_r a^r u_i^r$, and $S_i : l_2 \to l_2$ is a diagonal operator with entries σ_i^r. Clearly, $U_i U_i^* = I$ and $V_i V_i^* = I$ because the bases $\{u_i^r\}$ and $\{v_i^r\}$ are orthonormal.

Then

$$\begin{aligned}\bar{L}_i &= E_i(\alpha_i I + K_i^* K_i)^{-1} K_i^* \\ &= E_i K_i^+ K_i (\alpha_i I + K_i^* K_i)^{-1} K_i^* \\ &= L_i U_i S_i^2 (\alpha_i I + S_i^2)^{-1} U_i^*.\end{aligned}$$

As α_i approaches 0, $U_i S_i^2(\alpha_i I + S_i^2)^{-1} U_i^*$ approaches I pointwisely. Since L_i is compact in H^t norm, $\bar{L}_i \to L_i$ in H^t norm as $\alpha_i \to 0$ (Lemma 3.3 applied to the extensions of L_i and $\bar{L}_i$ to H^t). Hence, for any fixed ϵ, we can choose a fixed α_i such that $\|L_i - \bar{L}_i\| \leq \frac{\epsilon}{2}$.

Since Nyström's method is norm convergent for second kind Fredholm integral equations in H^t (Lemma 3.4), as p_i increases, $(\alpha_i I + \tilde{K}_i^* \tilde{K}_i)^{-1}$ approaches $(\alpha_i I + K_i^* K_i)^{-1}$ in H^t norm. Therefore, for any fixed ϵ we can find p_i such that $\|\bar{L}_i - \tilde{L}_i\| \leq \frac{\epsilon}{2}$.

Combining the above estimates we get the following theorem:

Theorem 3.1.

$$\|\tilde{L}_i - L_i\| \leq \epsilon \tag{3.12}$$

by choosing α_i and p_i based on ϵ.

Since the kernel is homogeneous and related to a scale invariant PDE, S_i and L_i depend only on the relative positions of the boxes C_i and C_{i+1}. Therefore, there are only finite number of operators L_i that can appear in the above analysis: 4 from each of the M2M and L2L translations and $7^2 - 3^2 = 40$ from the M2L translations. As we stated before, E_i and K_i can also be chosen to be level independent. Similarly, there are only a finite number of E_i and K_i operators as well. Therefore, we can choose α and p uniformly so that the estimate (3.12) applies for any L_i, K_i and E_i.

3.4.4 Total Discretization Error

Using a single step norm estimate of $\tilde{L} - L$, we can estimate $\|\tilde{L}^{(n)} - L^{(n)}\|$ using Lemma 3.2. We use a constant C to denote the uniform bound for $L^{(i:j)}$ for all $0 \le j \le i \le n$. Then for any i,

$$\|L^{(i)} - \tilde{L}^{(i)}\| = \Big\|\sum_{j=0}^{i} L^{(i:j+1)}(L_j - \tilde{L}_j)\tilde{L}^{(j-1)}\Big\| \le C\epsilon(1 + \sum_{j=0}^{i-1} \|\tilde{L}^{(j)}\|).$$

This expression gives us a recurrence relationship on the norm of $\|\tilde{L}^{(i)}\|$:

$$\|\tilde{L}^{(i)}\| \le C + C\epsilon\big(1 + \sum_{j=0}^{i-1} \|\tilde{L}^{(j)}\|\big)$$

Assuming $C \ge \epsilon$, from the recurrence we obtain

$$\|\tilde{L}^{(j)}\| \le 2C(1 + C\epsilon)^j$$

and thus

$$\|\tilde{L}^{(n)} - L^{(n)}\| \le C\epsilon(1 + \sum_{j=0}^{n-1} 2C(1 + C\epsilon)^j) = C(2((1 + C\epsilon)^n - 1) + \epsilon). \quad (3.13)$$

Although this estimate has an exponential dependence on n, it is only an upper bound and, in our experience, quite pessimistic. Moreover, our numerical experiments show that the uniform bound C is a small constant both in 2D and 3D for various kernels. Further, in actual calculations n is likely to be less than 40, and ϵ at least of order 10^{-4}. Therefore, in practice $(1 + C\epsilon)^n - 1$ behaves as $Cn\epsilon$. Finally, we have the following theorem:

Theorem 3.2.

$$\|\tilde{L}^{(n)} - L^{(n)}\| = O(n\epsilon).$$

Remark 3.2. Unlike our method, in the original analytic FMM method, there is no error associated with M2M, M2L and L2L transformations. The only error introduced in the analytic FMM are the S2M and M2T operators.

Remark 3.3. The basic parameters in our approximation are the regularization parameter α and the number of quadrature points p. In general, the regularization parameter α is chosen to filter out the noise or error in the data. In our experiments we choose α to be a constant factor of the desired accuracy of the FMM approximation (ϵ) and then we choose the correct number of quadrature points by trial-and-error. The latter is very inexpensive because is independent of the size of the problem, and thus can be estimate quickly with a small test case.

Remark 3.4. The error associated with an approximate integral evaluation

$$u - \tilde{u} = \int G(x,y)\varphi - \sum_i w_i G(x,y_i)\varphi_i,$$

is the quadrature error. In 2D we use the trapezoidal rule on the circle which is super-algebraically convergent. This enables us to approximate the operator L with $\tilde{L}$ with a small number of quadrature points. However, to our knowledge, in 3D there is no simple integration rule on the sphere that will result in similar high order accuracy; standard polynomial accuracy algorithms must be used. This is an important difference with the analytic FMM, which guarantees exponential convergence (on the number of multipole terms) for the far field approximation. Nonetheless, in our numerical experiments we did not observe noticeable differences between the 2D and 3D version.

3.5 Numerical Results

In this section we present numerical results for our method. First, we examine the accuracy of the equivalent density approximation. Second, we present results on the overall accuracy of the method.

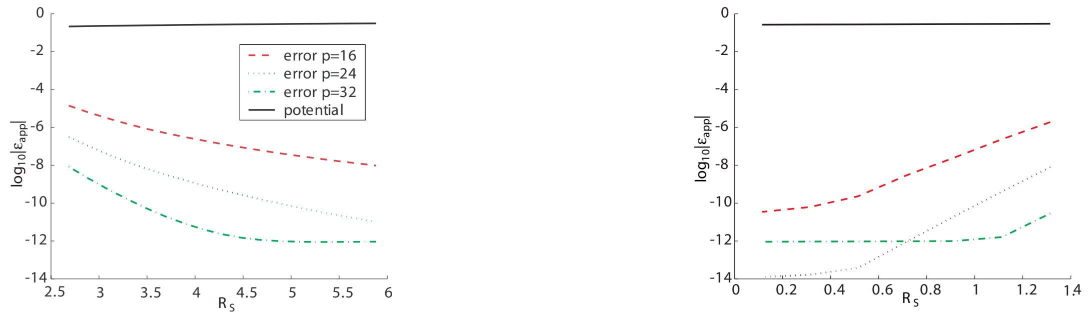
3.5.1 Accuracy on the Equivalent Density Approximation

In this section we present results that indicate that our equivalent density approximations give good accuracy in both two and three dimensions.

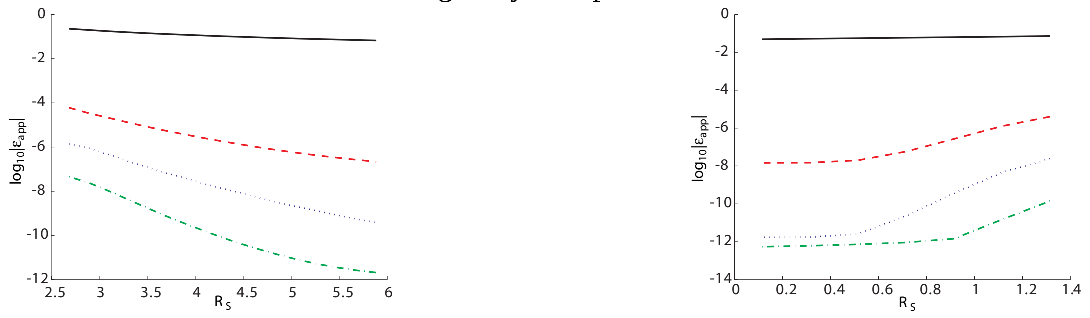
For two and three dimensions we show the results of three kernels: the Laplace single layer kernel, the modified Stokes double layer kernel and the Navier single layer kernel (Figure 3.10 and 3.11). For each kernel, the left plot is the accuracy of the upward equivalent density approximation, and the right one is the accuracy of the downward equivalent density approximation. For the upward equivalent density, we give the error for points in the exterior of the source box in the region corresponding to the interaction list of the box. For the downward equivalent density we give the error in the interior of the box. In all plots, the side length of the box is 2; we calculate the error by taking the maximum norm over a sphere centered at the center of the box. The abscissa of a plot is the radius of the sphere, and the ordinate is the logarithm of the error.

2D case. Figure 3.10 shows the error of the equivalent density approximation for the 2D Laplace single layer kernel, the 2D modified Stokes double layer kernel and the 2D Navier single layer kernel. In all three cases, the source density is located close to a corner of the box. The regularization parameter α is chosen to be 10^{-12} in all plots. Although not reported here, we have generated similar plots for all kernels given in Appendix A. All results exhibit similar accuracy. In some plots for 2D case, the 32-point error curve has larger error than the 24-point error curve. This is related to the regularization: we use 10^{-12} for α when solving the inverse problem and this complicates direct comparisons as we increase p. We do not have a strict analytic error bound like the analytic FMM algorithm for the Laplace equation. However, Figure 3.10 shows that our scheme gives comparable accuracy.

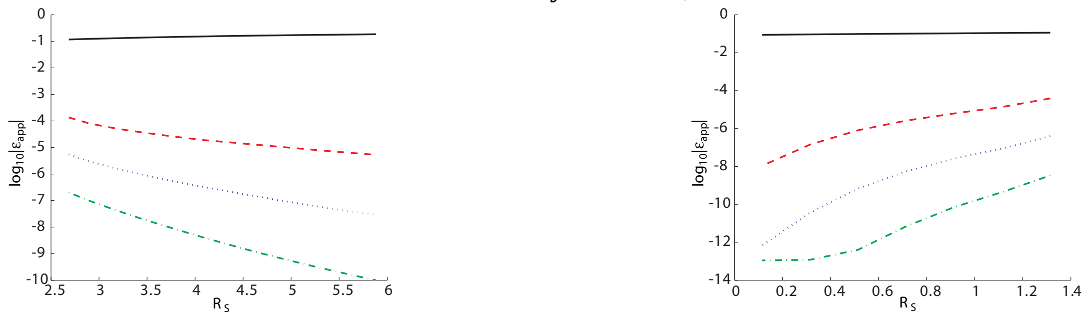
3D case. Figure 3.11 shows the equivalent density approximation errors for the 3D Laplace single layer kernel, the 3D modified Stokes double layer kernel and the 3D Navier single layer kernel. In each case, the source density is again placed close to one corner of the cube. The regularization parameter α used in these plots is 10^{-9}.



Single-layer Laplacian.

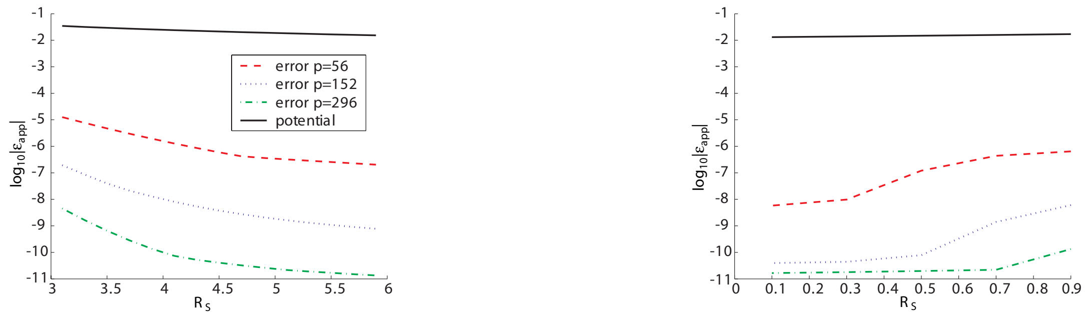


Modified double-layer Stokes, $\lambda = 1$.

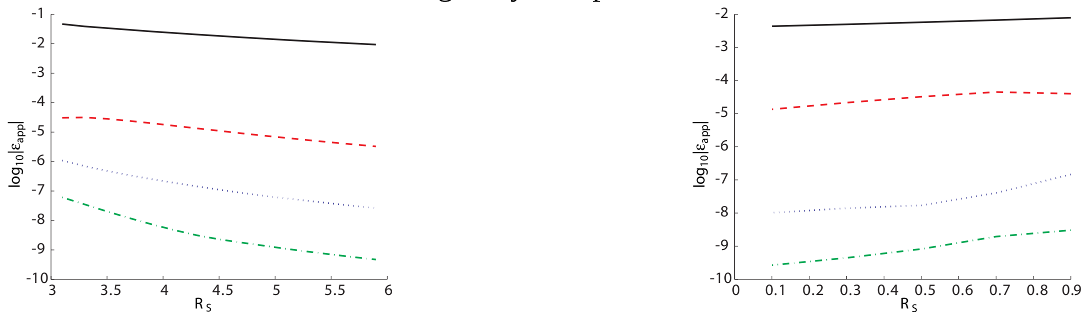


Single-layer Navier.

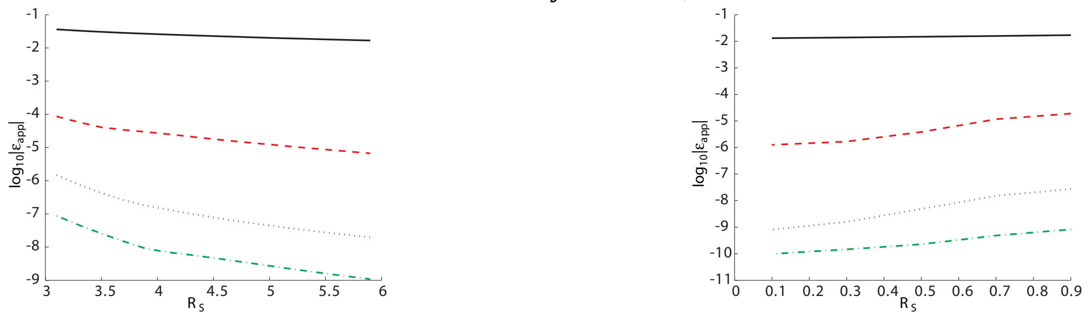
Figure 3.10: Results of the equivalent density approximation in 2D. *Left*: the error of the upward equivalent density approximation. *Right*: the error of the downward equivalent density approximation. The abscissa of the plots is the radius of the sphere R_s, and the ordinate is the logarithm of the error ϵ_{app}. The solid curve is the maximum norm of the potential. The remaining three curves show the maximum norm error for 16-, 24- and 32-point approximation of the equivalent densities. For modified Stokes, we tested λ from 1e-3 to 1e+3 and obtained similar error plots. For λ greater than 1e+3, far field interaction is negligible.



Single-layer Laplacian.



Modified double-layer Stokes, $\lambda = 1$.



Single-layer Navier.

Figure 3.11: Results of the equivalent density approximation in 3D. *Left*: the error of the upward equivalent density approximation. *Right*: the error of the downward equivalent density approximation. The abscissa of the plots is the radius of the sphere R_s, and the ordinate is the logarithm of the error ϵ_{app}. For each plot, the solid curve shows the maximum norm of the potential. The rest three plots show the maximum norm error where the equivalent density is approximated with 56, 152 and 296 points. These numbers correspond to discretization points that are the boundary nodes of volume Cartesian grids of size $4 \times 4 \times 4$, $6 \times 6 \times 6$, $8 \times 8 \times 8$ (per box).

3.5.2 Overall Approximation Error

In this section we give wall-clock time and memory requirements for several kernels. All experiments were performed on a Sun Ultra 80 workstation with a 450 MHz CPU. In 3D case, the FFTW package is used for FFT computation. Our code has been implemented in C++.

In our experiments we assume that the source points and the target points coincide. We use three sets of density distributions in the cube with range $[-1, 1]$ in each dimension. The first set is a distribution on a sphere, which is typically non-uniform. The second set is a uniform distribution of density in a cube. The last set has densities only at the box corners. The objective of this set of points is to check the stability of multiple M2M and L2L transformations of our method. For all density distributions the densities are chosen randomly from $[0, 1)$. The three data sets for the 3D case are shown in Figure 3.12.

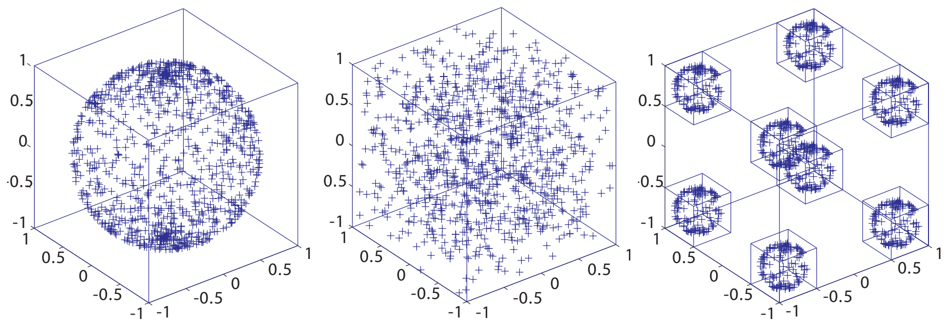


Figure 3.12: Three data sets in 3D: *Left*: densities distributed on the unit sphere, *Middle*: densities distributed uniform in the unit cube, *Right*: densities distributed at the eight corners of the unit cube.

We organize the table in a way similar to [20]. The columns of every table represent the following quantities.

N : the number of points used in computation (we use the same number of source and target points).

R : the number of levels of the computation tree.

M : the number of boxes in the computation tree.

p : the number of discretization points used in the equivalent density approximations. In 2D examples, we use 16, 24, and 32 points. In 3D examples, we choose the discretization points to be the boundary nodes of volume Cartesian grids of size $4 \times 4 \times 4$, $6 \times 6 \times 6$, $8 \times 8 \times 8$. These numbers correspond to 56, 152 and 296 points respectively.

s : the maximum number of points allowed in a leaf box of the computation tree.

S : the memory used to store M2M, M2L, and L2L translations.

T_{fmm} : the running time of our algorithm.

T_{dir} : the running time of the direct evaluation. For each table, only the number in the first line is actually tested; all other numbers are obtained by extrapolation. The error is computed in relative 2-norm. We randomly select k points $x_1, x_2, \cdots, x_k$, evaluate the potential u_i using our algorithm and the potential $\tilde{u}_i$ using direct evaluation at these k points. The error is estimated using the formula from [20]:

$$E = \left(\frac{\sum_{i=1}^{k} |u_i - \tilde{u}_i|^2}{\sum_{i=1}^{k} |\tilde{u}_i|^2} \right)^{1/2},$$

where k is chosen to be 40 in all experiments.

Below, we report the results on the first two data sets (non-uniform and uniform distribution) for five different kernels:

- 2D Laplace single layer kernel (Table 3.3),
- 3D Laplace single layer kernel (Table 3.4),
- 3D Modified Laplace single layer kernel (Table 3.5),
- 3D Modified Stokes double layer kernel (Table 3.6),
- 3D Navier single layer kernel (Table 3.7).

Our results from 2D are quite satisfactory since we can compute interactions between 2 million particles in 6 digits of accuracy in around 90 seconds, as we can see in Table 3.3. We discuss relative performance of our method in greater detail in the 3D case since this is more difficult to implement efficiently. We compare with results from two papers: the single-layer 3D Laplacian results of Cheng, Greengard, and Rokhlin [20] and modified single-layer 3D Laplacian results of Greengard and Huang [33].

In the first paper the authors use a 167 MHz Sun workstation and in the second a 440 MHz Sun platform. As mentioned before we are using a 450 MHz Sun workstation. The metric we use for the purposes of comparison is the total number of CPU cycles in millions per grid point. We compute this number as

$$\eta = \frac{T_{fmm} \times \mathrm{CPU}}{N}.$$

where η_a and η are the numbers of cycles per particle for the analytic FMM and and for our algorithm respectively. This is a only rough estimate that does not take into account the difference in chip architecture (e.g., memory bus clock), different floating point precision of the calculations (most calculations in the first paper were performed in single precision, all our results are in double precision), and different input densities.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
32768	10	2989	16	40	1.52e+00	1.53e+00	1.71e+02	2.80e-06
131072	12	11857	16	40	1.91e+00	5.85e+00	2.74e+03	1.24e-06
524288	14	47241	16	40	2.30e+00	2.36e+01	4.39e+04	1.51e-06
2097152	16	190601	16	40	2.69e+00	9.32e+01	7.02e+05	2.80e-06
32768	9	1597	24	60	2.97e+00	1.92e+00	1.71e+02	2.68e-08
131072	12	6505	24	60	3.94e+00	7.47e+00	2.74e+03	2.84e-08
524288	14	26073	24	60	5.10e+00	2.97e+01	4.39e+04	3.36e-08
2097152	16	104129	24	60	5.98e+00	1.24e+02	7.02e+05	2.24e-08
32768	9	1493	32	80	5.28e+00	2.23e+00	1.71e+02	1.89e-10
131072	11	5953	32	80	6.84e+00	1.03e+01	2.74e+03	1.77e-10
524288	13	23825	32	80	8.41e+00	4.04e+01	4.39e+04	7.05e-10
2097152	15	95425	32	80	9.97e+00	1.49e+02	7.02e+05	6.03e-10

The particles are uniformly distributed on the perimeter of a circle.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
32768	8	2837	16	40	1.14e+00	1.45e+00	1.71e+02	5.72e-07
131072	10	12245	16	40	1.53e+00	5.26e+00	2.74e+03	3.71e-07
524288	12	47829	16	40	1.92e+00	2.16e+01	4.39e+04	4.46e-07
2097152	14	189717	16	40	2.31e+00	8.89e+01	7.02e+05	5.24e-07
32768	7	1557	24	60	2.13e+00	1.78e+00	1.71e+02	2.05e-09
131072	9	5909	24	60	3.01e+00	7.21e+00	2.74e+03	2.50e-09
524288	11	25557	24	60	3.88e+00	2.75e+01	4.39e+04	1.64e-09
2097152	14	104085	24	60	4.85e+00	1.07e+02	7.02e+05	1.48e-09
32768	7	1557	32	80	3.78e+00	2.12e+00	1.71e+02	2.83e-11
131072	9	5269	32	80	5.34e+00	8.81e+00	2.74e+03	2.87e-11
524288	11	23893	32	80	6.91e+00	3.54e+01	4.39e+04	2.17e-11
2097152	13	95253	32	80	8.47e+00	1.34e+02	7.02e+05	6.50e-11

The particles are uniformly distributed inside a cube.

Table 3.3: Performance for particles interacting via the single-layer Laplacian in 2D.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
24576	6	1377	56	60	1.72e+00	5.72e+00	9.74e+01	2.12e-05
98304	7	5049	56	60	1.72e+00	2.38e+01	1.56e+03	3.21e-05
393216	8	19065	56	60	1.72e+00	9.51e+01	2.49e+04	6.08e-05
1572864	9	76185	56	60	1.72e+00	3.82e+02	3.99e+05	6.03e-05
24576	5	585	152	150	5.90e+00	1.16e+01	9.74e+01	3.34e-07
98304	6	2289	152	150	5.90e+00	4.76e+01	1.56e+03	5.86e-08
393216	7	11193	152	150	5.90e+00	2.18e+02	2.49e+04	2.45e-07
1572864	9	44145	152	150	5.90e+00	8.35e+02	3.99e+05	3.08e-07
24576	4	273	296	250	1.47e+01	1.81e+01	9.74e+01	1.59e-09
98304	6	1449	296	250	1.47e+01	8.15e+01	1.56e+03	1.40e-09
393216	7	5073	296	250	1.47e+01	3.41e+02	2.49e+04	1.10e-09
1572864	8	19161	296	250	1.47e+01	1.38e+03	3.99e+05	2.81e-09

The particles are distributed on the surface of a sphere.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
24576	4	585	56	60	1.72e+00	6.40e+00	9.74e+01	6.64e-06
98304	5	3657	56	60	1.72e+00	3.11e+01	1.56e+03	1.27e-05
393216	7	28233	56	60	1.72e+00	1.30e+02	2.49e+04	5.00e-05
1572864	8	88137	56	60	1.72e+00	4.08e+02	3.99e+05	5.84e-05
24576	4	585	152	150	5.90e+00	1.60e+01	9.74e+01	1.54e-08
98304	5	3657	152	150	5.90e+00	9.28e+01	1.56e+03	4.70e-08
393216	6	14409	152	150	5.90e+00	3.18e+02	2.49e+04	1.10e-07
1572864	7	37449	152	150	5.90e+00	8.47e+02	3.99e+05	2.13e-07
24576	4	585	296	250	1.47e+01	3.65e+01	9.74e+01	5.25e-10
98304	4	585	296	250	1.47e+01	1.11e+02	1.56e+03	4.57e-10
393216	5	3657	296	250	1.47e+01	4.31e+02	2.49e+04	6.85e-10
1572864	6	17481	296	250	1.47e+01	1.46e+03	3.99e+05	1.46e-09

The particles are uniformly distributed inside a cube.

Table 3.4: Performance for particles interacting via the single layer Laplacian in 3D.

First, we compare Table 3.4 with Tables IV, V, and VI of [20]. For the three digit accuracy (Table IV) the average η_a is 0.07 for single precision. Our method achieves an η equal to 0.11 (in double digit accuracy), approximately a factor of 1.5 slower. Similar conclusions hold for the 6-digit accuracy results(Table V), for which the analytic FMM achieves $\eta_a = 0.15$ in single precision, whereas our method achieves $\eta = 0.23$ in double precision. For the modified single layer Laplacian we compare the 6-digit accuracy entries (Table I, [33]), with Table 3.5 (uniform distribution in a cube). In this case $\eta_a = 0.3$ and $\eta = 0.4$, which is slightly better than 1.5; the actual difference in performance is even less, since we achieving about one additional digit of accuracy (average error 7×10^{-7} for the analytic FMM compared an average of 7×10^{-8} in our case).

Another reason our method is slower might be related to the dense interactions. In order to save storage we do not precompute them, and we have found that this slows down our method by a factor of 2 to 4. The most time consuming part is computing the $1/\sqrt{(r \cdot r)}$ term, which we have found impossible to optimize either with lookup tables or with special vector routines available from most vendors. For large problems that require several summations for the same particle partitions further running time improvements can be achieved by precomputing and storing all dense interactions. The memory requirements in this case can be substantial.

In conclusion, it appears that our method compares reasonably well with the analytic FMM by being a factor of 1.5 or less slower. Extending our code from the Laplacian to the modified Laplacian was very easy as we simply implemented a different kernel evaluation. Inspecting the results for the other kernels, we can confirm the $O(N)$ complexity of our method and the convergence to the exact sum as we increase the number of quadrature points.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
6144	5	441	56	60	4.55e+00	1.97e+00	1.15e+01	3.55e-05
24576	6	1377	56	60	6.27e+00	8.24e+00	1.83e+02	7.71e-05
98304	7	5049	56	60	8.29e+00	3.33e+01	2.94e+03	3.11e-05
393216	8	19065	56	60	1.00e+01	1.28e+02	4.70e+04	8.22e-05
6144	4	225	152	150	1.08e+01	4.38e+00	1.15e+01	2.48e-07
24576	5	585	152	150	1.57e+01	1.99e+01	1.83e+02	9.55e-08
98304	6	2289	152	150	2.26e+01	7.58e+01	2.94e+03	3.18e-07
393216	7	11193	152	150	2.85e+01	3.39e+02	4.70e+04	3.63e-07
6144	3	57	296	250	1.18e+01	6.90e+00	1.15e+01	2.50e-09
24576	4	273	296	250	2.64e+01	3.00e+01	1.83e+02	1.88e-09
98304	6	1449	296	250	5.30e+01	1.23e+02	2.94e+03	1.96e-09
393216	7	5073	296	250	6.99e+01	5.35e+02	4.70e+04	3.71e-09

The particles are distributed on the surface of a sphere.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
6144	4	585	56	60	3.35e+00	3.72e+00	1.15e+01	5.28e-06
24576	4	585	56	60	3.35e+00	1.06e+01	1.83e+02	2.29e-05
98304	5	3657	56	60	5.07e+00	4.25e+01	2.94e+03	3.98e-05
393216	7	28233	56	60	8.14e+00	1.64e+02	4.70e+04	4.88e-05
6144	3	73	152	150	5.38e+00	4.09e+00	1.15e+01	2.10e-08
24576	4	585	152	150	1.13e+01	2.11e+01	1.83e+02	9.86e-08
98304	5	3657	152	150	1.72e+01	1.08e+02	2.94e+03	7.23e-08
393216	6	14409	152	150	2.31e+01	4.14e+02	4.70e+04	4.57e-08
6144	3	73	296	250	1.29e+01	5.87e+00	1.15e+01	7.15e-10
24576	4	585	296	250	2.75e+01	4.39e+01	1.83e+02	6.02e-10
98304	4	585	296	250	2.75e+01	1.98e+02	2.94e+03	4.28e-10
393216	5	3657	296	250	4.22e+01	6.65e+02	4.70e+04	8.24e-10

The particles are uniformly distributed in a cube.

Table 3.5: Performance of our method for particles interacting via the modified single layer Laplacian in 3D.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
6144	5	441	56	60	8.18e+01	2.65e+01	1.04e+02	9.56e-04
24576	6	1377	56	60	1.13e+02	1.02e+02	1.66e+03	1.45e-03
98304	7	5049	56	60	1.49e+02	3.91e+02	2.66e+04	1.47e-03
6144	4	225	152	150	2.00e+02	7.59e+01	1.04e+02	5.66e-06
24576	5	585	152	150	2.92e+02	2.39e+02	1.66e+03	6.90e-06
98304	6	2289	152	150	4.20e+02	1.01e+03	2.66e+04	1.06e-05
6144	3	57	296	250	2.16e+02	6.44e+01	1.04e+02	8.77e-08
24576	4	273	296	250	4.89e+02	3.59e+02	1.66e+03	1.67e-07
98304	6	1449	296	250	9.87e+02	1.69e+03	2.66e+04	1.88e-07

The particles are distributed on the surface of a sphere.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
6144	4	585	56	60	6.03e+01	6.97e+01	1.04e+02	5.32e-04
24576	4	585	56	60	6.03e+01	1.23e+02	1.66e+03	5.01e-04
98304	5	3657	56	60	9.13e+01	6.09e+02	2.66e+04	7.00e-04
6144	3	73	152	150	9.87e+01	4.35e+01	1.04e+02	1.77e-06
24576	4	585	152	150	2.09e+02	3.57e+02	1.66e+03	2.96e-06
98304	5	3657	152	150	3.19e+02	2.04e+03	2.66e+04	9.32e-06
6144	3	73	296	250	2.36e+02	7.63e+01	1.04e+02	3.71e-08
24576	4	585	296	250	5.09e+02	8.28e+02	1.66e+03	8.02e-08
98304	4	585	296	250	5.09e+02	2.01e+03	2.66e+04	9.88e-08

The particles are uniformly distributed in a cube.

Table 3.6: Performance of our method for particles interacting via the modified double layer Stokes kernel in 3D.

In all experiments, we store only the linear operators for M2M, M2L and L2L translations, since these operators are applied repetitively in a single pairwise interaction evaluation. The dense interactions between adjacent boxes are not stored. The storage number reported in all tables considers only the memory used by M2M, M2L and L2L operators, while the storage used to store the densities and potentials (which

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
6144	5	441	56	60	1.55e+01	1.29e+01	5.91e+01	8.54e-05
24576	6	1377	56	60	1.55e+01	4.93e+01	9.46e+02	6.71e-05
98304	7	5049	56	60	1.55e+01	1.98e+02	1.51e+04	6.32e-05
6144	4	225	152	150	5.50e+01	3.29e+01	5.91e+01	1.07e-06
24576	5	585	152	150	5.50e+01	1.10e+02	9.46e+02	1.66e-06
98304	6	2289	152	150	5.50e+01	4.59e+02	1.51e+04	1.02e-06
6144	3	57	296	250	1.08e+02	3.28e+01	5.91e+01	7.30e-09
24576	4	273	296	250	1.36e+02	1.82e+02	9.46e+02	8.51e-09
98304	6	1449	296	250	1.36e+02	8.51e+02	1.51e+04	8.73e-09

The particles are distributed on the surface of a sphere.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
6144	4	585	56	60	1.55e+01	3.41e+01	5.91e+01	3.70e-05
24576	4	585	56	60	1.55e+01	6.65e+01	9.46e+02	4.82e-05
98304	5	3657	56	60	1.55e+01	3.13e+02	1.51e+04	6.68e-05
6144	3	73	152	150	4.94e+01	2.19e+01	5.91e+01	1.81e-07
24576	4	585	152	150	5.50e+01	1.62e+02	9.46e+02	3.50e-07
98304	5	3657	152	150	5.50e+01	9.48e+02	1.51e+04	4.86e-07
6144	3	73	296	250	1.18e+02	3.78e+01	5.91e+01	2.56e-09
24576	4	585	296	250	1.36e+02	4.22e+02	9.46e+02	3.58e-09
98304	4	585	296	250	1.36e+02	1.00e+03	1.51e+04	4.39e-09

The particles are uniformly distributed in a cube.

Table 3.7: Performance of our method for particles interacting via the single layer Navier kernel in 3D.

scales linearly with respect to the number of points and boxes) is not included. This explains why for the results of homogeneous kernels (Tables 3.4 and 3.7), the storage numbers remain small and do not increase with the number of points and the number of levels.

Stability of multiple M2M and L2L translations. Here we test the stability of the M2M and L2L translations of our algorithm using the last data set which only has density distribution at the corners of the cube. Table 3.8 shows the result on this data set with 2D Laplace single layer kernel. Table 3.9 reports the errors with 3D Laplace single layer kernel.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
524288	18	47449	16	40	2.17e+00	2.17e+01	4.39e+04	4.46e-06
524288	18	26041	24	60	4.54e+00	2.63e+01	4.39e+04	1.20e-08
524288	17	23833	32	80	7.91e+00	3.50e+01	4.39e+04	1.04e-10

Table 3.8: Performance of our method for the 2D single layer Laplacian. In this experiment the particles are distributed over the boundaries of four circles. These circles are quite small compared the size of the (square) computational domain, and located near to the four corners of the domain. In this way the tree is “forced” to have many levels (up to 18). We use this experiment to test the numerical stability of our M2M and L2L translations.

N	R	M	p	s	S (Mb)	T_{fmm} (s)	T_{dir} (s)	Error
196608	12	11057	56	60	1.72e+00	4.58e+01	6.23e+03	1.75e-05
196608	11	4721	152	150	5.90e+00	1.04e+02	6.23e+03	1.20e-07
196608	10	2225	296	250	1.47e+01	1.50e+02	6.23e+03	1.53e-09

Table 3.9: Performance of our method for the 3D single layer Laplacian. In this experiment the particles are distributed over the boundaries of eight spheres. These spheres are quite small compared the size of the (cubic) computational domain, and located near to the eight corners of the box.

3.6 Summary

In this chapter, we have described a new kernel-independent fast multipole method, which generalizes FMM to a broad class elliptic kernels while attaining an algorithmic complexity (including constants) which is on par with the analytic FMM. Here we summarize the main features of our algorithm.

- Our algorithm has the same structure as the original adaptive FMM method.
- We have demonstrated that the method performs well for single and double layers, the Laplacian, the modified Laplacian, the Stokes, the modified Stokes, and the Navier kernels in two and three dimensions. By providing just a kernel evaluation routine our method is immediately applicable, as long as the kernel is associated with a non-oscillatory second-order elliptic PDEs.
- Comparisons of the running times between our method and the best known FMM implementations, and for same accuracy levels, indicate that our approach was successful in efficiently extending FMM to other kernels.
- To our knowledge, our results are the first fast summation computations for the modified Stokes and Navier operators.
- Our method is also directly applicable for derivatives of the kernels we have presented here. Indeed, we have tested our method on the hypersingular kernels resulting from differentiating the double layer Stokes and Navier equations.
- The M2L translations in our method are suboptimal. In 3D, the analytic exponential translations require $O(p)$, whereas our method requires $O(p^{3/2})$ where

p is the number of coefficients used in the approximation (the number of moments in the analytic FMM, and the number of discretization points in our method).

- Our method does not have a level independent error estimate that comes with the original FMM algorithm for Laplacian kernel. However, the error analysis in Section 3.4 shows that in practice the error can increase with the depth at most in a linear fashion.

Chapter 4

Parallel Implementation of Kernel Independent FMM

For large scale problems, even the iterative solvers with fast multipole method acceleration can still be quite slow on one computer. This brings up the request for parallel implementation of these solvers. This chapter describes a message passing based parallel implementation of the kernel independent FMM algorithm described in Chapter 3. In the Nyström solver for boundary integral equation, the particle positions and densities are associated to the discretization of integral equations, and at each time step the interaction computation (matrix vector multiplication within a Krylov method) is carried out multiple times. Therefore, our parallel implementation is designed to achieve maximum efficiency in the multiplication phase.

4.1 Introduction

To parallelize an FMM algorithm on a distributed memory platform, one faces with two major challenges. First, the data, including the particles and the octtree data structure) needs to be stored in a distributed way, since one processor often does not have enough memory to contain the whole data set. Therefore, we need efficient schemes to partition the particles over processors and to build a distributed octtree based on this distributed particle set. In our implementation, we use the local essential tree [90] to store the data structure.

Second, the algorithm needs to meet certain communication and synchronization requirements: (1) *Tree-related communication* is required to maintain a consistent global tree. (2) *Synchronization* at upward and downward pass since there is data dependence between the equivalent densities of a parent and its child, and these two boxes can belong to different processors. (3) *Communication related to M2L translation* is necessary since one box needs the upward equivalent density or source information of another box owned by a different processor. In our implementation we have logically separated the computation and communication. During the upward and downward passes, a processor performs its own computation ignoring the existence of other processors. Between them, a single step combines the upward equivalent density computed by all processors and takes care of the communication. The advantage of this approach is that no synchronization is required at the computation passes. A disadvantage is the redundant computation at the nodes which are close to the root of the global computation tree. However since the number of these nodes is small, this has negligible influence on the overall computation.

Related work on parallel tree-codes. The first successful distributed-memory parallel implementations for non-uniform particle distributions were obtained for the Barnes-Hut algorithm by Warren and Salmon [90]. Key ideas in this paper were the local essential trees (LETs), which provide a framework for parallelization of Barnes-Hut algorithm and can be extended to the FMM. The hashed octtree data structures were first introduced in [91] along with space-filling curves used for partitioning and load balancing, and further increased efficiency and scalability of tree-codes. A similar approach for shared memory machines, and one of the first scalable FMM implementations, is found in [80], in which a cost-zones partitioning is used with orthogonal recursive bisection. A comparison between FMM algorithms, hybrids, and the Barnes-Hut method can be found in [10]. The main conclusion is that for higher accuracies, FMM is the fastest method. Another nice comparison between different platforms and algorithms can be found in Hu and Johnsson [43], in which the authors report results on up to 100 million particles on uniform particle distributions on a CM-5.

Recent papers on distributed-memory implementations include FMM for electromagnetics [40]; Helmholtz-type problems using optimal M2L translations [59]; and molecular dynamics FMM implementations that scale to 24 millions of particles on thousands of processors [56, 57]. Efficient data-structures and discussions on the theory of partitioning and complexity can be found in [77] and [85].

Other approaches for particle interactions include particle-mesh algorithms like those used in NAMD-2 [65] which employs FFTs for Ewald summation on regular grids. Such approaches could be extended to more general kernels, but they are restricted to approximately uniform particle distributions. Parallel Stokes solvers were presented in [64], but without FMM or Barnes-Hut acceleration.

4.2 The Parallel Algorithm

4.2.1 Data Partitioning and Tree Generation

Our partitioning scheme is fairly straightforward. We take advantage of the fact that our input is a set of surface patches on which the particles are generated. We first gather all input surface patches on a single processor, and assign to each patch a weight which in the simplest case is equal to the number of particles in that patch. Second, we partition the clusters into groups with equal weights and assign each group to one processor. To do this we use Morton curve partitioning. Alternatively, we could use Morton curve partitioning directly on the particles but we have found the first approach faster. No additional load balancing information is used besides the number of particles. Work estimates from a previous time step could be used to obtain more balanced partitioning.

An essential part in the FMM algorithm is the generation of the octtree. However, since our applications require tens to hundreds of interaction computations, we have adopted a simple but suboptimal tree construction algorithm. An important idea in parallel tree-based algorithms is the *Local Essential Tree* (LET) [90], which is the global tree subset that a processor needs to evaluate the interaction on particles it owns. In an adaptive FMM algorithm, in order to calculate the interaction at a box B, we need the information from the boxes in the following four lists ([20],[32]): (1) U list L_U^B which contains B itself and the leaf boxes which are adjacent to B if B is leaf, and it is empty when B is non-leaf; (2) V list L_V^B which contains the children of the neighbors of B's parent, which are not adjacent to B; (3) W list L_W^B which contains all the descendants of B's neighbors whose parents are adjacent to B but who are not adjacent to B themselves if B is leaf, and it is empty if B is a non-leaf;

and (4) X list L_X^B, which contains all boxes A such that $B \in L_W^A$. Therefore, for a certain processor P, its LET first contains the boxes which contains points belonging to P and second the boxes in the U, V, W, and X lists of these boxes. For a box B of the first kind, we say P *contributes* to B, or equivalently, P is a *contributor* of B. If B is of the second kind, we say P *uses* B or P is a *user* of B.

In the tree generation, besides LET, we maintain a compact representation of the global tree by using an array in every processor, which we call the global tree array. Each entry in the global tree array corresponds to a box in the global tree, and this array is ordered according to a level-by-level traversal of the tree. The only information stored is the global number of particles in the box and the indices of its children boxes in the array. This representation only contains topological structure of the tree. In practice, for a 200M points data set with s chosen to be 60, the size of the array is less than 16M. Our algorithm constructs the local tree and this array structure level by level. All processors begin at level 0 with the same box which is large enough to contain the global particle set. At every level l, each processor puts its local number of points in boxes at level l as well as into its local copy of the global tree array. Then, an MPI_Allreduce is used over all local copies of the global tree array to sum up the local number of points for each box at level l. After this collective communication, each local array contains the global number of points in each box in level l. By comparing each box's global number of points with s (the maximum number of points allowed in each leaf box), each processor can decide whether a box in level l should be further subdivided. Based on this decision, a processor can construct the $l+1$ level of its local tree and the array representation of the $l+1$ level of the global tree. After the construction of the local tree, the computation of the local FMM lists is straightforward by using the global tree represented in the array.

4.2.2 Interaction Calculation

Before the interaction calculation, we first partition the global tree array, so that for each box B the owner processor coordinates the communication related to B. If only one processor contributes to B, then it is the owner of B. If multiple processors contribute to B, then it can be owned by any processor, and the owner is chosen to balance the communication load. This can be done as follows. For each processor P, first we use the global tree array to decide the boxes for which P is the only contributor, and mark them as "taken". Second, we use MPI_Allreduce to combine the information, so that every processor P knows all boxes already taken by some processor. Third, every processor P uses the same sequential algorithm to assign unmarked boxes to processors in order to balance communication load. In the end, all processors have the owner information for any box in the global computation tree.

The interaction calculation part is logically separated into three stages. The first stage is a computation step which performs the upward computation. Each processor P builds the upward equivalent densities for the LET nodes to which it contributes (ignoring the existence of the other processors).

The second stage has two components. First, for each leaf box, we need to collect its source positions and source density (also known as ghost information) from its contributors and make them available for its users. The *gather/scatter* procedure for doing this is given in Figure 4.1. Second, for each box (leaf or non-leaf), we need to sum up the upward equivalent densities produced by its contributors and make it available for its users. The procedure for this is similar with two modifications: (1) we iterate over all boxes in the LET instead of just the leaf boxes, and (2) the owner of a box *sums up* the received upward equivalent densities to obtain the global upward equivalent densities for that box.

```
STEP 1 GATHER
  for each box $B$ in the LET do
    if $P$ contributes to $B$ then
      $P$ sends its local source position/density information of $B$ to the owner of $B$
    end if
    if $P$ owns $B$ then
      $P$ receives the local information of $B$ from all the contributors of $B$
      $P$ combines them into the global position/density information of $B$
    end if
  end for
STEP 2 SCATTER
  for each box $B$ in the LET do
    if $P$ owns $B$ then
      $P$ sends the global position/density information of $B$ to all users of $B$
    end if
    if $P$ uses $B$ then
      $P$ receives the global position/density information of $B$ from the owner of $B$
    end if
  end for
```

Figure 4.1: Gather/Scatter procedure.

The third stage performs the downward computation. Here, for each LET node B, to which it contributes, P transforms the source density or upward equivalent density of the boxes in U, V, W and X lists into local equivalent density or target potential at node B (ignoring the existence of the other processors again). In our implementation the upwards traversal is overlapped with the ghost communication; and the equivalent densities communication is overlapped with the dense and X-list computations.

4.3 Scalability Results

We present fixed-size and isogranular scalability analysis. For fixed-size scalability analysis, we increase the number of processors for a fixed problem size. This analysis exposes the grain size (i.e., the number of particles per processor) for which we can expect reasonable speedups. For isogranular scalability analysis we keep the grain size fixed and we increase the number of processors (and thus the problem size). Such analysis reveals communication problems related to the size and frequency of the messages as well as global reductions and problems with algorithmic scalability. In our case, however, we expect good algorithmic scalability since FMM is an $O(N)$ algorithm under reasonable assumptions on the particle distribution [55].

Before we describe our numerical experiments, we emphasize two main conclusions from our work on the sequential performance of our method. First, the most expensive parts of the FMM algorithm are the M2L interactions and the dense interactions, both in the downward traversal of the tree, especially in three dimensions. Second, our method achieves algorithmic speed-ups which are on par with the fastest known implementations of the FMM for the Laplacian [20] and modified Laplacian kernels [33].

The problem setup is the following: The input is a set of surfaces, which we then sample to get the particle positions. We build the hierarchical tree structure and then we perform several interaction calculations. In this article we always report results for a single interaction calculation, averaged over several iterations.

We assume the sets of source and target points to be identical. We use two sets of density distributions in the cube with range $[-1, 1]$ in each dimension. The first set is produced by sampling 512 spheres centered at an $8 \times 8 \times 8$ Cartesian grid in the unit cube. For relatively low sampling rates, up to 10 million particles, we obtain a uniform particle distribution. For higher sampling rates the distribution per processor becomes non-uniform since the sampling over a single sphere is non-uniform. Our second particle set is a non-uniform distribution of particles clustered at the eight corners of the unit cube. In all density distributions, the densities are chosen randomly from $[0, 1]$, and the relative error in all experiments is 10^{-5}.

Our algorithm has been implemented in C++. We used the fast exponential, square root and reciprocal libraries in the CXML routines, FFTW [26] for the M2L translations, and PETSc [4] for profiling and for its Krylov iterative solvers. All our tests were performed on the Pittsburgh Supercomputing Center's TCS-1 terascale computing HP Alphaserver Cluster comprising of 750 SMP ES45 nodes. Each node is equipped with four Alpha EV-68 processors at 1 GHz and 4 Gbytes of memory. The peak performance is approximately 6 Tflops/s, and the peak performance for the top-500 LINPACK benchmark is approximately 4 Tflops/s. The nodes are connected by the Quadrics interconnect which delivers over 500 MB/s of message-passing bandwidth per node and has a bisection bandwidth of 187 GB/s. In all our tests we have used 4 processors per node.

Description of results We report wall-clock timings, Gflops/s rates and parallel efficiency measurements for several problem sizes and kernels. Fixed size scalability results for 3.2M particles are reported in Table 4.1, in which we provide timings for the interaction calculation and for the tree construction, including communication. We also report aggregate Gflops/s rates. In these examples we have used 3.2 million particles. We report wall-clock time in seconds and flop rates for a single interaction computation. We also report timings for the tree construction and communication phases. Here (*P*) is the number of processors; (*Total*) is the total time (averaged across processors) of the interaction phase; (*Ratio*) is the difference between the maximum and minimum time across processors, which is an indication of load imbalance; *Comm* is the average time spent in MPI communication; (*Up*) is the average time spent in the upward traversal of the tree; (*Down*) is the average time spent in the downward traversal of the tree; (*Avg*) is the average Gflops/s during the interaction calculation; (*Peak*) is the peak Gflops/s; and (*Gen/Comm*) is the time spent in the tree construction phase. Overall, we can observe that we obtain excellent scalability up to 512 processors. Communication costs however, become significant once we hit 512 processors or more. In these cases we use less than 6,000 particles per processor, i.e., too fine a grain size.

In Figure 4.2, we report the aggregate CPU cycles (across processors) per point for the interaction calculation. These numbers are computed as $\frac{P \times C \times T(P)}{N}$, where P is the number of processors; C is the clock rate which in our case is 1GHz; $T(P)$ is the wall-clock time on P processors; and N is the number of particles. This metric is used to measure parallel scalability and can be used to compare among different machines, clock-rates and problems sizes. However, it hides architecture-dependent characteristics like cache performance and memory bus speed. There are five stages: (*Up*) is the upward traversal of the tree used to built equivalent densities; (*Comm*) is the communication of the ghost points and equivalent densities; (*DownU*) is the dense interaction calculations for L_U lists; (*DownV*) is the M2L translations for L_V lists; (*DownX*) and (*DownW*) are the calculations for L_X and L_W lists associated with the adaptive algorithm. In the right column, (*Avg*) is the the average, across processors, Mflops rate; (*Peak*) is the peak rate; and the (*Max/Min*) indicate the Maximum/minimum average rates across processors—an indication of load imbalance.

The work efficiency is computed using the timings in Table 4.1. The Mflops/s efficiency is computed based on the rates given in the figure. As we can see, up to 512 processors the efficiency is quite good: there is only a small increase in the total work per particle.

We also compute a floating point efficiency as an index of efficient utilization of the machine. The work efficiency is $\frac{T(1)}{T(P)P}$ and the flop-rate efficiency is computed as $f(P)/f(1)$, where $f(P)$ is the flop-rate per processor on P-processors.

Laplacian kernel

	Interaction computation							Tree
	Time (sec)					GFlops/s		Time (sec)
P	*Total*	*Ratio*	*Comm*	*Up*	*Down*	*Avg*	*Peak*	*Gen/Comm*
1	392.75	1.00	0.00	58.41	334.34	0.28	0.31	13.97
4	103.67	1.00	0.87	14.63	88.30	1.06	1.25	3.81
8	51.33	1.00	0.54	7.42	43.48	2.15	2.46	2.27
16	25.49	1.10	0.55	3.69	21.99	4.30	4.96	1.47
64	6.74	1.10	0.33	0.96	6.10	16.53	19.00	0.68
256	1.67	1.20	0.15	0.24	1.55	64.53	77.49	0.51
512	1.10	1.20	0.40	0.12	0.74	104.89	154.69	0.87
1024	1.13	1.20	0.81	0.07	0.45	108.67	258.55	1.09

Modified Laplacian kernel

	Interaction computation							Tree
	Time (sec)					GFlops/s		Time (sec)
P	*Total*	*Ratio*	*Comm*	*Up*	*Down*	*Avg*	*Peak*	*Gen/Comm*
1	478.66	1.00	0.00	75.09	403.57	0.31	0.36	13.88
4	120.43	1.00	2.48	19.24	99.41	1.22	1.40	3.74
8	59.58	1.00	0.62	9.50	49.57	2.49	2.83	2.19
16	30.32	1.00	0.46	4.78	25.82	4.84	5.62	1.35
64	7.48	1.10	0.23	1.21	6.56	19.16	22.19	0.55
256	2.08	1.30	0.22	0.32	1.96	69.77	83.16	0.59
512	1.24	1.20	0.32	0.16	1.01	125.29	166.35	0.55
1024	1.25	1.20	0.85	0.10	0.51	127.67	261.21	1.25

Stokes kernel, non-uniform particle distribution

	Interaction computation							Tree
	Time (sec)					GFlops/s		Time (sec)
P	*Total*	*Ratio*	*Comm*	*Up*	*Down*	*Avg*	*Peak*	*Gen/Comm*
1	1171.92	1.00	0.00	146.28	1025.64	0.37	0.56	22.95
4	332.69	1.00	19.17	41.63	284.49	1.29	1.97	6.05
8	155.07	1.00	3.51	19.97	135.53	2.76	4.10	2.94
16	78.02	1.00	1.51	10.02	70.39	5.47	8.28	1.51
64	21.11	1.20	0.75	2.70	19.95	20.47	31.71	0.80
256	5.92	1.50	0.53	0.74	6.18	72.77	122.11	0.81
512	3.29	1.70	0.48	0.40	3.59	130.60	237.08	0.73
1024	2.35	1.80	0.82	0.22	2.18	191.96	446.76	0.96

Table 4.1: Fixed size scalability results.

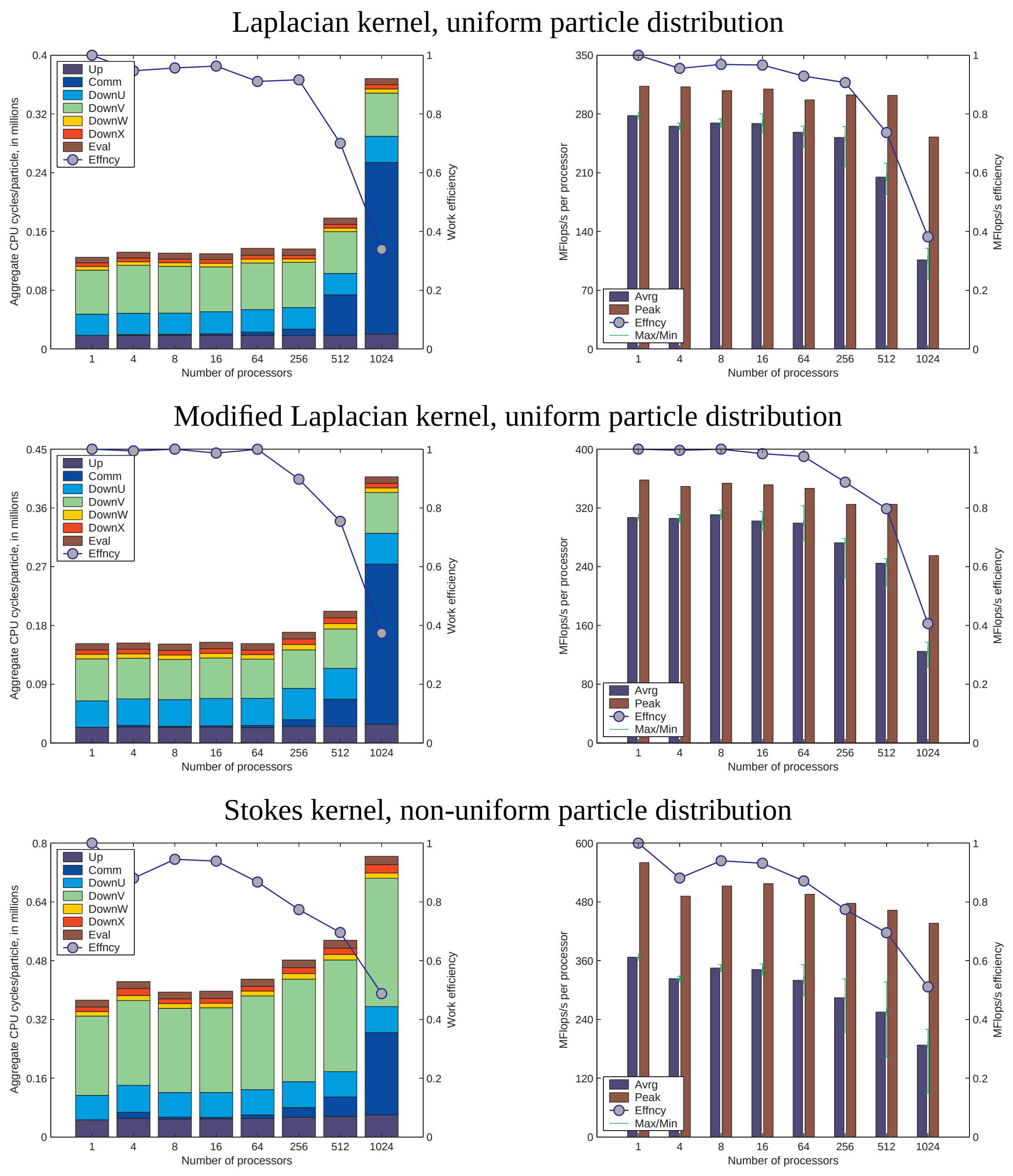


Figure 4.2: Fixed-size scalability results for different kernels. Here 3.2 million particles were used. The left column shows aggregate CPU cycles per particle; the right column shows Mflops/s/processor for different stages of the interaction calculation phase and the flop-rate efficiency.

In Table 4.2 and Figure 4.3, we report isogranular scalability results for 200 thousand particles per processor and for the Laplace and Stokes kernels.

In these examples we have used 200,000 particles per processor. We report the wall-clock time and Gflops/s for the interaction computation, and timings for the tree construction and communication phases. *P*, *Total*, *Ratio*, *Comm*, *Up*, *Down*, *Avg*, *Peak* and *Gen/Comm* have the same meaning as in Table 4.1. For the larger problem we have a total 400 million particles, which for the Stokes case corresponds to 1.2 billion unknowns. In these experiments we do not report work efficiency because the algorithm behavior slightly changes as we increase the problem size and at first sight it appears that we obtain superlinear speedups. The particles are sampled for 512 spheres regularly arranged in a cube. For small numbers of particles we have uniform distributions, but for the very large problems the problem locally is non-uniform. As a result the number of M2L interactions drops and since this is the most costly part of the computation it appears that the work efficiency improves. Overall, we observe that the running time is slightly decreasing. This is due to algorithmic changes; the M2L translations work (*Down*) is decreasing. We observe very good scalability, i.e., low communication costs during the interaction calculation phase; the increase for 2048 processors, in the non-uniform distribution case is due to the load imbalance.

As in Figure 4.2 we report total cycles per particle and flop-rates for the whole interaction calculation and its different phases in Figure 4.3. The Mflops/s efficiency is computed based on the rates given in the figure. We are not reporting work efficiency because the algorithmic behavior of the problem changes with increasing problem size, due to the increase in non-uniformity of the distribution at finer scales. In this case the number of M2L translations decreases, resulting in a overall work decrease. In addition, M2L computations run at about 300 Mflops/s, while all other parts run at about 400+ Mflops/s. This produces apparent superlinear efficiencies, especially in the work-intensive Stokes case. Our implementation exhibits excellent scalability results on thousands of processors: As we can observe from the plots for the Laplacian kernel we maintain an 80% efficiency on up to 2048 processors. In the non-uniform distribution (last row) we observe a rather significant decrease of the efficiency down to 65%. This is due to load imbalance, and is something that we are currently working to improve. Finally notice that that for the largest problem the peak performance was 1Tflops/s.

Laplacian kernel, uniform particle distribution

	Interaction computation							Tree
	Time (sec)					GFlops/s		Time (sec)
P	*Total*	*Ratio*	*Comm*	*Up*	*Down*	*Avg*	*Peak*	*Gen/Comm*
1	30.56	1.00	0.00	2.87	27.69	0.27	0.28	0.49
4	28.59	1.00	0.23	3.13	25.29	1.03	1.16	0.70
16	25.97	1.10	1.06	3.77	22.80	4.17	4.84	1.42
64	21.76	1.10	1.38	3.73	18.26	17.04	17.78	3.02
256	22.06	1.10	1.65	3.48	19.54	64.36	73.89	13.48
1024	22.22	1.10	3.09	3.84	18.39	247.78	262.64	71.26
2048	23.54	1.20	0.96	4.05	21.34	488.07	568.89	964.47

Stokes kernel, uniform particle distribution

	Interaction computation							Tree
	Time (sec)					GFlops/s		Time (sec)
P	*Total*	*Ratio*	*Comm*	*Up*	*Down*	*Avg*	*Peak*	*Gen/Comm*
1	133.26	1.00	0.00	7.38	125.88	0.30	0.49	0.49
4	166.79	1.00	0.63	8.62	157.86	1.03	2.06	0.64
16	146.72	1.10	2.00	12.34	139.38	4.41	8.46	1.59
64	106.00	1.10	2.13	10.74	99.06	18.86	33.21	3.54
256	88.06	1.10	2.43	10.59	81.94	81.57	127.91	14.16
1024	79.34	1.10	2.10	10.59	72.53	338.56	494.97	69.42
2048	76.28	1.10	2.06	10.58	69.36	669.99	986.45	936.78

Stokes kernel, non-uniform particle distribution

	Interaction computation							Tree
	Time (sec)					GFlops/s		Time (sec)
P	*Total*	*Ratio*	*Comm*	*Up*	*Down*	*Avg*	*Peak*	*Gen/Comm*
1	82.68	1.00	0.00	10.39	72.29	0.34	0.54	1.17
4	80.43	1.00	0.72	9.88	70.01	1.39	2.14	1.37
16	78.03	1.10	1.30	9.75	70.75	5.46	8.51	1.47
64	84.16	1.20	4.02	10.67	79.55	20.05	31.40	2.52
256	86.24	1.50	8.97	11.17	92.44	71.49	119.76	8.21
1024	92.60	2.00	4.20	12.55	114.93	248.16	426.40	69.34
2048	108.64	2.50	12.32	17.72	139.36	453.44	752.03	988.24

Table 4.2: Isogranular scalability results.

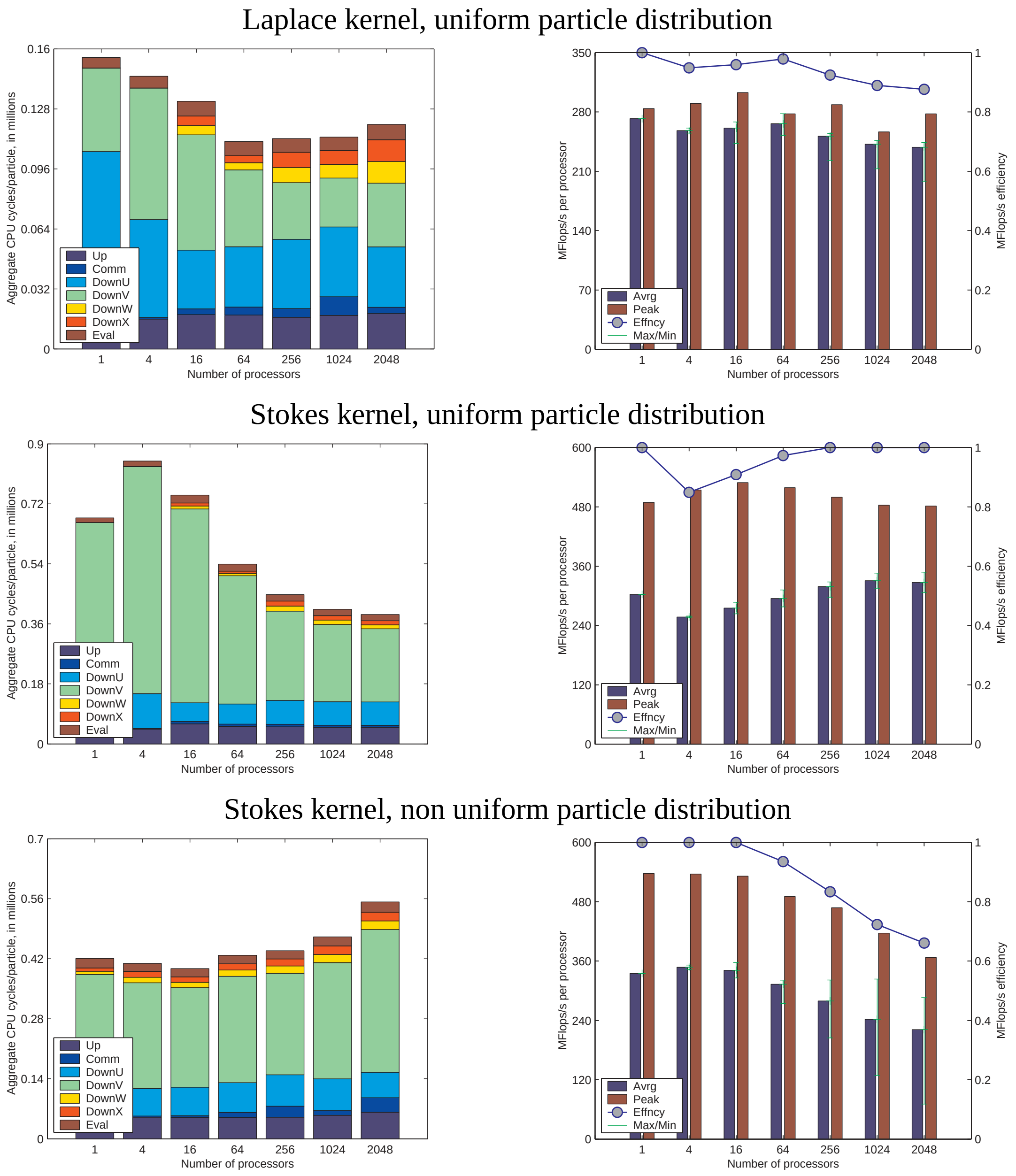


Figure 4.3: Isogranular scalability results for Laplacian and the Stokes kernels. These charts show the aggregate CPU cycles per particle and Mflops/s/processor for the different stages of the interaction calculation.

Finally in Table 4.3 we report results from our largest runs on 3000 processors. In this set of runs the geometry is the 512 spheres and we solve problems for the Laplace equation with 100K and 230K particles per processor and for the Stokes equations also with 230K particles per processor. For all the other experiments we have used rough 60 particles per box, while in this experiment we use 120 particles per box to slightly reduce the costs of tree construction.

	Interaction computation							Tree
	Time (sec)					GFlops/s		Time (sec)
unknowns	*Total*	*Ratio*	*Comm*	*Up*	*Down*	*Avg*	*Peak*	*Gen/Comm*
0.300 B	7.63	1.5	1.03	2.43	5.69	696.8	802.2	837.4
0.690 B	21.59	2.2	3.23	4.13	15.29	789.3	972.1	1101
2.070 B	65.97	1.8	3.06	9.87	62.10	1134	1587	1077

Table 4.3: 3000 processor runs. In these examples we have solved three problems 100K and 230K particles per CPU for for the Laplace and Stokes equations all for the 512 spheres input. For the larger problem we have a total 700 million particles, which for the Stokes case corresponds to 2.1 billion unknowns. Notice that for the interaction calculation we have sustained 1.13 Tflops/s which translates to 25% efficiency compared to the sustained performance for the LINPACK benchmark on the TCS-1.

Discussion Examination of the performance numbers leads to the following observations: (1) The code uses about 160 thousand CPU cycles per particle for five digits of accuracy for the Laplacian kernel and about 200 thousand and 800 thousand cycles for the modified Laplacian and Stokes respectively. (2) For the fixed problem size (3.4 million particles) we obtain 80% efficiency for up to 256 processors and then the communication costs start increasing. (3) In the isogranular scalability good efficiency is maintained up to 2048 processors with peak performance of 1 Tflops/s and sustained performance of 0.7 Tflops/s. (4) The communication costs during the interaction computation scale very well. (5) The tree construction and communication does not scale beyond 1024 processors. (6) Load imbalance for highly non-uniform distributions is significant. (7) In our largest runs we have obtained 1.13 Tflops/s sustained performance and 1.6 Tflops/s peak performance for 2.1 billion unknowns.

It is apparent that we get better performance for the Stokes kernel. The reason is that the scalar kernels like the Laplacian and modified Laplacian have less work per particle and less communication than the Stokes kernels. We have observed an increase in the flop-rate for the Stokes kernel albeit the higher communication costs.

We should note that our implementation of the tree construction and load balancing is not optimal; our focus was on efficient implementation of the interaction computation, which we apply several times before we update the particle positions. Tree construction and load balancing are well isolated parts in our code and known techniques can be used to improve their efficiency. We plan to incorporate more efficient algorithms in the near future. In particular, we plan to use workload information from previous time steps for load balancing. In addition we are currently changing our level-to-level tree construction in order to obtain a completely scalable algorithm.

4.4 Summary

In this chapter, we have presented a MPI-based scalable and platform-independent parallel implementation of our kernel independent fast multipole method.

Our parallel implementation has several important features. First, our MPI-based parallel implementation logically separates computation and communication to avoid synchronization in upward and downward pass, and to exploit maximal computation and communication overlapping. Second, we verified that the method scales up to 3000 processors and achieves very good per processor sustained performance (up to 480 Mflops/s). As a result, we were able to reach 1.13 Tflops/s sustained performance for a Stokes flow problem with 2.1 billion unknowns.

Chapter 5

Nyström Integration

5.1 Introduction

In this chapter, we describe the Nyström discretization method for the numerical solution of the integral equation

$$\frac{1}{2}\varphi(x) + (D\varphi)(x) = f(x).$$

As we mentioned earlier, this equation is solved using a Krylov space iterative linear solver, such as GMRES. The essential step of this solver is the evaluation of $(D\varphi)(x)$, which is required to be accurate and efficient. In the following sections, we first present the discretization scheme and quadrature rule of our Nyström solver. Then, we prove its error bounds and give the complete algorithm for the evaluation of $(D\varphi)$. We also describe how to extend this algorithm to evaluate other related physical quantities such as pressure p and stress s. Then, we present an accurate and efficient way to evaluate velocity, pressure and stress at any point *inside* the domain Ω. Finally, we comment on how to extend the algorithms to other equations, including the Laplace's equation, the Navier's equation and their modified version.

In this chapter, the boundary Γ of the computation domain is represented with the surface representation scheme introduced in Chapter 2. The basic ideas are motivated by the work of Bruno and Kunyansky [16].

5.2 Discretization and Singular Integral Evaluation for Velocity

By using the surface representation scheme in Chapter 2, the boundary Γ is covered by several patches P_i for $i = 1, \cdots, K$, each smoothly parameterized over a chart C_i (Figure 5.1).

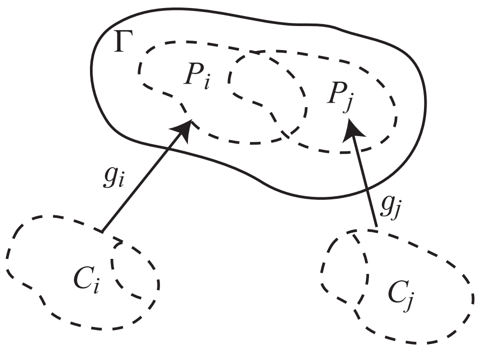


Figure 5.1: Parameterization of boundary.

Two sets of functions are essential to the discretization scheme described in this chapter:

- $g_i : C_i \to \mathbf{R}^3$ which parameterizes the domain boundary using charts in 2D.
- $w_i : \Gamma \to \mathbf{R}$ which provides a partition of unity (POU) of the boundary Γ and the support of w_i is in P_i. Following the convention of [16], we call w_i the

fixed partition of unity, since it does not depend on the point x at which the integration is evaluated.

We first rewrite the integral $(D\varphi)(x)$ in terms of the domain C_i:

$$(D\varphi)(x) = \sum_{k=1}^{K} \int_{C_i} D(x, g_k(c_k)) w_k(g_k(c_k)) \varphi(g_k(c_k)) J_k(c_k) \, \mathrm{d}c_k,$$

where J_k denote the Jacobian of map g_k. We define

$$\psi_k(c_k) = w_k(g_k(c_k)) \varphi(g_k(c_k)) J_k(c_k)$$

which is a function vanishing at the boundary of C_i since w_k does so. Then, we have

$$(D\varphi)(x) = \sum_{k=1}^{K} \int_{C_k} D(x, g_k(c_k)) \psi_k(c_k) \, \mathrm{d}c_k. \tag{5.1}$$

We discretize each domain C_k with a Cartesian grid with uniform spacing h. The set of grid points inside C_k is denoted by $\{c_{k,i}\}$. The union of all grid points $\bigcup_{k=1}^{K} \{c_{k,i}\}$ is the Nyström points used in the numerical approximation of the integral $D\varphi(x)$. By $\varphi_{k,i}$ we denote the value of function φ at points $c_{k,i}$, $\psi_{k,i}$ the value of ψ, and $x_{k,i}$ the position $g_k(c_{k,i})$. The goal of the integral solver is to find the values $\varphi_{k,i}$. At each step of the GMRES solver, we need to approximate $(D\varphi)(x_{k,i})$ efficiently and accurately using the values $\varphi_{k,i}$.

We now consider the integrals in (5.1) one by one. By dropping the index k, the integral over a single chart is of the following form

$$\int_{C} D(x, g(c)) \psi(c) \, \mathrm{d}c. \tag{5.2}$$

Non-singular part. If $x \notin g(C)$, then $D(x, g(c))$ is non-singular for any point $c \in C$, and the integrand $D(x, g(c))\psi(c)$ and its derivatives vanish at the boundary of C.

Such a function can be regarded as a periodic function on a rectangular domain which contains C. Therefore, the trapezoidal rule with weights h^2 at points c_i gives super-algebraic convergence if the function ψ (and equivalently φ) is infinitely smooth. If ψ is a C^M function, the trapezoidal rule gives an error of order $O(h^M)$.

Singular part. If $x = g(c')$ for some $c' \in C$, we introduce a C^∞ function $\eta_{c'}$ defined by

$$\eta_{c'}(c) = \theta(\frac{|c-c'|}{\sqrt{h}})$$

where $\theta : [0, \infty) \to [0, 1]$ is a non-increasing C^∞ function satisfying $\theta(r) = 1$ for $r \leq \frac{1}{4}$ and $\theta(r) = 0$ for $r \geq 1$ and $| \cdot |$ is the Euclidean distance in 2D. By definition, $\eta_{c'}$ is a radial function with support in a disk of radius $\sqrt{h}$ centered at c'. We call $\eta_{c'}$ a *floating partition of unity* (following [16]), since its support depends on the position of x. Using the function $\eta_{c'}$, we rewrite (5.2) into two parts:

$$\begin{aligned} \int_C D(g(c'), g(c))\psi(c)\,\mathrm{d}c &= \int_C D(g(c'), g(c))(1-\eta_{c'}(c))\psi(c)\,\mathrm{d}c \qquad (5.3) \\ &+ \int_C D(g(c'), g(c))\eta_{c'}(c)\psi(c)\,\mathrm{d}c. \qquad (5.4) \end{aligned}$$

(5.3) is not singular since $\eta_{c'}(c)$ vanishes in the neighbors of c', thus we integrate it using the trapezoidal rule as we did before for the non-singular part. To integrate (5.4), we transform the domain into a polar coordinate system centered at c'. Let $q = c - c' = (\rho\cos(\theta), \rho\sin(\theta))$, the second integral is equivalent to:

$$\int_0^\pi \mathrm{d}\theta \int_{-\sqrt{h}}^{\sqrt{h}} D(g(c'), g(c(\rho,\theta)))\theta(\frac{|\rho|}{\sqrt{h}})\psi(c(\rho,\theta))\rho\,\mathrm{d}\rho$$

(notice here we allow ρ to be negative in order to make the integrand a periodic function). We choose the Cartesian grid in polar coordinates

$$\left((a-\frac{1}{2})\cdot h, b\cdot 2\pi/\lceil\frac{2\pi}{\sqrt{h}}\rceil\right)$$

as the quadrature points, where a and b range over the integers where these points are in the integration domain (Figure 5.2). The number of quadrature points is of order $O(\frac{1}{h})$ and the grid is fully symmetric in the polar coordinates.

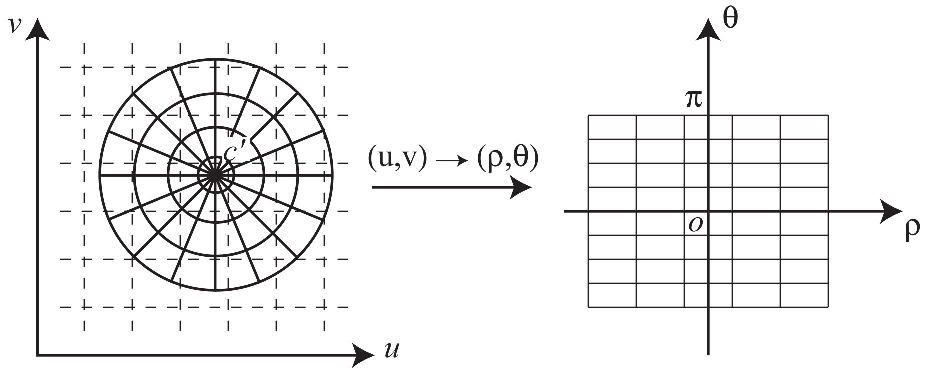


Figure 5.2: Integration points in Cartesian and polar coordinates.

We will show in the next section that the above algorithm gives a high-order quadrature rule for the integral (5.2) for $x \in g(C)$. However, the quadrature points for (5.4) are on a Cartesian grid in polar coordinates, instead of the quadrature points x_i in the Cartesian coordinates. This necessitates an efficient and accurate interpolation procedure to obtain the values of ψ at the polar-coordinate quadrature points from its values at the Cartesian quadrature points. The method we use, which is similar to the one in [16], has two steps: the preprocessing step and the evaluation step.

In the preprocessing step, we perform the following procedure for each chart C_k. Since ψ_k is a periodic function on a rectangular domain which contains $x_{k,i}$ as its Cartesian grid, we first use a 2D fast Fourier transform (FFT) to calculate the Fourier coefficients of ψ_k from the values $\psi_{k,i}$ at $x_{k,i}$. We then use these Fourier coefficients to approximate the value of ψ_k on a new grid, which is much finer than the original grid again by means of an FFT. The spacing of the new grid is chosen to be 8 times smaller than the spacing of the old grid. If the function ψ_k is infinitely smooth, these approximation values are super-algebraically close to the true value of ψ_k.

The evaluation step is as follows. Given a point x, for each patch C_k such that $x \in g(C_k)$, we need to interpolate the values of ψ_k at a polar-coordinate grid centered at the preimage of x in C_k. For each point on this polar-coordinate grid, we collect the values of ψ_k on the 4×4 subgrid (of the refined grid), which contains the point in the center cell, and use Lagrange interpolation to approximate the value of ψ_k.

This interpolation procedure is highly efficient due to the periodicity of ψ which enables us to use the fast Fourier transform. In practice, we find 8-fold refinement gives approximation error of order 10^{-8}, which is much smaller compared to the other error introduced in the overall algorithm, thus we can safely neglect the error from this interpolation procedure.

Compared with the discretization scheme in [16], our algorithm differs in two major ways.

- The support of the floating partition of unity $\eta_{c'}$ is of size $\sqrt{h}$.
- The integration grid for the singularity is fully symmetric in the polar coordinates. As a result, 2D fast Fourier transform is used in the interpolation procedure.

The first change enables us to give strict error bounds for the integration error. The second change allows us to integrate integrals with a higher order of singularity.

5.3 Error Analysis

In this section, we derive the error bounds for the procedure described in Section 5.2 for the integration of

$$\int_C D(x, g(c))\psi(c)\,\mathrm{d}c$$

when $x = g(c')$ for some $c' \in C$. We assume that the interpolation procedure explained in the previous section introduces no error, and the functions φ is C^M continuous for some $M \geq 2$. Our plan is to show that the integration schemes for both (5.3) and (5.4) are high-order accurate.

Lemma 5.1. *Suppose $D \subset \mathbf{R}^2$ and functions $a \in C^M(D)$ and $b \in C^M(D)$ satisfy the following conditions.*

$$\|a^{(m)}\|_\infty \leq C_1(m)\left(\frac{1}{\sqrt{h}}\right)^{m+s} \qquad m = 0, \cdots, M$$

and

$$\|b^{(m)}\|_\infty \leq C_2(m)\left(\frac{1}{\sqrt{h}}\right)^{m+t} \qquad m = 0, \cdots, M$$

where $s \geq 0$ and $t \geq 0$, then

$$\|(ab)^{(m)}\|_\infty \leq C_3(m)\left(\frac{1}{\sqrt{h}}\right)^{m+s+t} \qquad m = 0, \cdots, M.$$

Here C_1, C_2 and C_3 are all constants depending on m.

Proof. Use Leibniz rule on $(ab)^m$ for $m = 1, \cdots, M$. $\square$

Lemma 5.2. *Assume D is a bounded open set in $\mathbf{R}^2$ and $f : D \to \mathbf{R}$ can be written into the following decomposition.*

$$f(q) = \frac{1}{|q|}H(\frac{q}{|q|})S(q) \quad q \in \mathbf{R}^2$$

where H is a homogeneous function, and S is a C^M function that vanishes on the boundary of D. Then the error of the trapezoidal rule with spacing h on function

$$\int_D f(q)(1-\theta(\frac{|q|}{\sqrt{h}}))\,\mathrm{d}q$$

is $O(h^{\frac{M-1}{2}})$, where η_0 is the floating partition of unity at the origin.

Proof. Since $1-\theta(\frac{|\cdot|}{\sqrt{h}})$ is equal to zero in the neighborhood of the origin and S vanishes at the boundary of D, we obtain the following estimate for the error e of the trapezoidal rule:

$$e \leq C_0(M)h^M \|(f(1-\theta(\frac{|\cdot|}{\sqrt{h}}))^{(M)}\|_\infty.$$

Now we estimate $\|(f(1-\theta(\frac{|\cdot|}{\sqrt{h}}))^{(M)}\|_\infty$. If $|q| \leq \frac{\sqrt{h}}{4}$, $(f(1-\theta(\frac{|\cdot|}{\sqrt{h}}))^{(M)}(q) = 0$ since $\theta(\frac{|q|}{\sqrt{h}}) = 1$.

Now suppose $|q| \geq \frac{\sqrt{h}}{4}$. We have the following estimates:

$$|f^{(m)}(q)| \leq C_1(m)(\frac{4}{\sqrt{h}})^{m+1}, \quad m = 1, \cdots, M,$$

from f's decomposition, and

$$|(1-\theta(\frac{|\cdot|}{\sqrt{h}}))^{(m)}(q)| \leq C_2(m)(\frac{1}{\sqrt{h}})^m \quad m = 1, \cdots, M$$

since θ is a fixed function. Using Lemma 5.1, we have

$$|(f(1-\theta(\frac{|\cdot|}{\sqrt{h}})))^{(m)}(q)| \leq C_3(m)(\frac{1}{\sqrt{h}})^{m+1} \quad m = 1, \cdots, M.$$

Take $m = M$, we have

$$e \leq C_0(M)h^M C_3(M)(\frac{1}{\sqrt{h}})^{M+1} = C_0(M)C_3(M)h^{\frac{M-1}{2}} = O(h^{\frac{M-1}{2}}).$$

□

Lemma 5.3. *The trapezoidal rule on (5.3) with spacing h gives $O(h^{\frac{M-1}{2}})$ accuracy if φ is C^M continuous.*

Proof. Using the definition of $D(x,y)$, we can write

$$D(x,y) = \frac{1}{|r|}H(\frac{r}{|r|})$$

where H is a homogeneous function. Suppose $x = g(c')$ and $y = g(c)$ for c and c' in the chart C. By letting $q = c - c'$ and $q = |q|$, we can write

$$D(g(c'), g(c)) = \frac{1}{|q|}H'(\frac{q}{|q|})T'(q)$$

where H' is a homogeneous function on chart C and T' is an infinity smooth function. Since φ is C^M continuous, ψ is C^M continuous and vanishing on the boundary of C. Therefore $D(g(c'), g(c))\psi(c)$ can be rewritten as:

$$\frac{1}{|q|}H'(\frac{q}{|q|})S'(q)$$

where $S' = T'\psi$ is C^M smooth and vanishes at the boundary C. Using Lemma 5.2, we proved that the trapezoidal rule is $O(h^{\frac{M-1}{2}})$ accurate for

$$\int_C D(x, g(c))(1 - \eta_{c'}(c))\psi(c)\,\mathrm{d}c.$$

□

Lemma 5.4. *Suppose $f : \mathbf{R} \to \mathbf{R}$ is a C^M function. Then the trapezoidal rule with spacing h has $O(h^{\frac{M}{2}})$ accuracy on the integral:*

$$\int_{-\sqrt{h}}^{\sqrt{h}} f(\rho)\theta(\frac{|\rho|}{\sqrt{h}})\,\mathrm{d}\rho$$

Proof. The proof is similar to the one for Lemma 5.2. The trapezoidal rule gives error

$$C_0 h^M \|\theta(\frac{|\cdot|}{\sqrt{h}})\|_\infty.$$

We can bound the magnitude of the Mth derivative of $\theta(\frac{|\cdot|}{\sqrt{h}})$ by $h^{\frac{M}{2}}$. Therefore, the error is of order $O(h^{\frac{M}{2}})$. □

Lemma 5.5. *The polar-coordinate trapezoidal rule on (5.3) gives $O(h^{\frac{M}{2}})$ accuracy if φ is C^M continuous.*

Proof. First we make two observations: (1) the function $D(x, g(c(\rho, \theta)))\rho$ is an infinitely smooth function in the polar coordinates parameterization (even at the neighborhood of the origin), and (2) the function $\psi(c(\rho, \theta))$ is C^M continuous since φ is continuous.

Let us define

$$B(\theta) = \int_{-\sqrt{h}}^{\sqrt{h}} D(x, g(c(\rho, \theta)))\theta(\frac{|\rho|}{\sqrt{h}})\psi(c(\rho, \theta))\rho \,\mathrm{d}\rho.$$

Since $D(x, g(c(\rho, \theta)))\psi(c(\rho, \theta))\rho$ is a smooth function, from Lemma 5.4 we know that the trapezoidal rule with spacing h on the variable ρ can integrate the function $B(\theta)$ for any fixed theta with $O(h^{\frac{M}{2}})$ accuracy.

We notice that $B(\theta)$ itself is also a periodic function defined on $\theta \in [0, \pi]$. Using the trapezoidal rule with spacing $\sqrt{h}$ gives accuracy $h^{\frac{M}{2}}$ on the following integral

$$\int_0^\pi B(\theta)\,\mathrm{d}\theta.$$

The values of $B(\theta)$ used for integration are not exact. There is an $O(h^{\frac{M}{2}})$ error between the actual value of $B(\theta)$ and the approximation value computed from the trapezoidal rule integration in the ρ direction. Nevertheless, since all the quadrature weights for the trapezoidal rule are positive, the error on the values of $B(\theta)$ can introduce errors at most of the same order. Thus, the overall error is again $O(h^{\frac{M}{2}})$.

$\square$

Now we can state the overall error estimate of our quadrature algorithm.

Theorem 5.1. *Our quadrature rule for the integral $(Df)(x)$, evaluated at any x on the boundary Γ gives $O(h^{\frac{M-1}{2}})$ accuracy if φ is C^M continuous.*

Proof. Combine Lemma 5.3 and Lemma 5.5. $\square$

In the following, we prove an additional result which is necessary in estimating the singular integral evaluation for other kernels.

Lemma 5.6. *Suppose that $f : \mathbf{R} \to \mathbf{R}$ can be written in the following form:*

$$f(\rho) = \frac{\rho}{|\rho|^2} S(\rho),$$

where S is a C^M function. Then the trapezoidal rule with quadrature points at $(k - \frac{1}{2})h$ gives $O(h^{\frac{M-1}{2}})$ accuracy on the integral

$$\int_{-\sqrt{h}}^{\sqrt{h}} f(\rho)\theta(\frac{|\rho|}{\sqrt{h}})\,\mathrm{d}\rho$$

which is understood in the Cauchy sense.

Proof. We use the singularity subtraction to reorganize the integral:

$$\begin{aligned}\int_{-\sqrt{h}}^{\sqrt{h}} \frac{\rho}{|\rho|^2} S(\rho)\theta(\frac{|\rho|}{\sqrt{h}})\,\mathrm{d}\rho &= \int_{-\sqrt{h}}^{\sqrt{h}} \frac{\rho}{|\rho|^2}(S(\rho) - S(0))\theta(\frac{|\rho|}{\sqrt{h}})\,\mathrm{d}\rho + \\ &\quad \left(\int_{-\sqrt{h}}^{\sqrt{h}} \frac{\rho}{|\rho|^2}\theta(\frac{|\rho|}{\sqrt{h}})\,\mathrm{d}\rho\right) S(0)\end{aligned}$$

The integrand of the first integral is a actually a C^{M-1} (due to the singularity subtraction) and periodic function. The trapezoidal rule with quadrature points $(k - \frac{1}{2})h$

gives $O(h^{\frac{M-1}{2}})$ accuracy by using Lemma 5.4. The second integral is a Cauchy integral with principal value zero, and the trapezoidal rule with $(k-\frac{1}{2})h$ as quadrature points produces zero as well, since the quadrature points, the function $\theta(\frac{|\cdot|}{\sqrt{h}})$ and the integration domain are all symmetric around the origin.

Therefore, the trapezoidal rule with symmetric quadrature points $(k-\frac{1}{2})h$ gives the overall error $O(h^{\frac{M-1}{2}})$ on the principal value of the integral. $\square$

Theorem 5.2. *Suppose a kernel K can be written into the following form*

$$K(x,y)=\frac{r}{|r|^3}H(\frac{r}{|r|})S(r),$$

where $r=x-y$, H is a homogeneous function and S is an infinitely smooth function. Then the quadrature rule described for $D\varphi$ gives $O(h^{\frac{M}{2}-1})$ accuracy on $(K\varphi)(x)$ at any x on Γ if φ is C^M smooth.

Proof. We separate the integral into two parts using a floating POU at the preimage of x.

$$\int_C D(x,g(c))(1-\eta_{c'}(c))\psi(c)\,\mathrm{d}c+\int_C D(x,g(c))\eta_{c'}(c)\psi(c)\,\mathrm{d}c.$$

Following the argument of Lemma 5.3, we can show that our quadrature method for the first integral gives the order $O(h^{\frac{M}{2}-1})$ error. Using Lemma 5.6 and following the argument of Lemma 5.5, we can show our quadrature method gives $O(h^{\frac{M-1}{2}})$ error for the second integral. Therefore the overall error is of order $O(h^{\frac{M}{2}-1})$. $\square$

All the error estimates proved in this section are under the condition φ is C^M continuous for $M\geq 2$. In the case where φ is a C^∞ function, following the same arguments, it is straightforward to show that our quadrature rule achieves super-algebraic convergence.

5.4 Efficient Implementation

In this section, we show that the quadrature algorithm described can be implemented in a highly efficient way without compromising the accuracy proved in the previous section.

For each x on Γ, we have

$$(Df)(x) = \sum_{k=1}^{K} \int_{C_k} D(x, g_k(c_k))\psi_k(c_k)\,\mathrm{d}c_k,$$

which can be written as the sum of three parts.

$$\sum_{k:x\notin g_k(C_k)} \int_{C_k} D(x, g_k(c_k))\psi_k(c_k)\,\mathrm{d}c_k \tag{5.5}$$

$$\sum_{k:x\in g_k(C_k)} \int_{C_k} D(x, g_k(c_k))(1-\eta_{c'_k}(c_k))\psi_k(c_k)\,\mathrm{d}c_k \tag{5.6}$$

$$\sum_{k:x\in g_k(C_k)} \int_{C_k} D(x, g_k(c_k))\eta_{c'_k}(c_k)\psi_k(c_k)\,\mathrm{d}c_k \tag{5.7}$$

where c'_k is the preimage of x in the chart C_k under the map g_k. (5.7) is evaluated using the trapezoidal rule in local polar coordinates at c'_k. (5.5) and (5.6) are the non-singular parts of the integral. They are approximated by the trapezoidal rule on the quadrature points $c_{k,i}$ in the forms:

$$\sum_{k:x\notin g_k(C_k)} \sum_i D(x, g_k(c_{k,i}))\psi_{k,i}h^2$$

and

$$\sum_{k:x\in g_k(C_k)} \sum_i D(x, g_k(c_{k,i}))(1-\eta_{c'_k}(c_{k,i})\psi_{k,i}h^2.$$

Summing them up, we have

$$\sum_k \sum_i D(x, g_k(c_{k,i}))\psi_{k,i}h^2 - \sum_{k:x\in g_k(C_k)} \sum_i D(x, g_k(c_{k,i}))\eta_{c'_k}(c_{k,i})\psi_{k,i}h^2.$$

In the first summation, the sources $\psi_{k,i}h^2$ are independent of the evaluation point x. In this case, we can use the kernel independent fast multipole method described in Chapter 3 to evaluate the value for all quadrature points x efficiently without compromising the accuracy. For every point x, the second summation only involves the points $c_{k,i}$ where $\eta_{c'_k}$ is positive, which is of a small number since the support of $\eta_{c'_k}$ is localized into a disk with radius $\sqrt{h}$. The algorithm is given in Figure 5.3.

```
Calculate $\psi_{k,i} = w_{k,i}\varphi_{k,i}J_k(c_{k,i})$ for each $x_{k,i}$.
Evaluate $u_{l,j} = \sum_{k,i} D(x_{l,j}, x_{k,i})\psi_{k,i}h^2$ using the kernel independent FMM.
for each $x_{l,j}$ do
    for each $C_k$ such that $x_{l,j} \in g_k(C_k)$ do
        Calculate $\sum_i D(x_{l,j}, g_k(c_{k,i}))\eta_{c'_k}(c_{k,i})\psi_{k,i}h^2$ and subtract the resulting value
        from $u_{l,j}$.
    end for
end for
Preprocessing the grids $\psi_{k,i}$ for high-order interpolation.
for each $x_{l,j}$ do
    for each $C_k$ such that $x_{l,j} \in g_k(C_k)$ do
        Integrate $\int_{C_k} D(x, g_k(c_k))\eta_{c'_k}(c_k)\psi_k(c_k)\,\mathrm{d}c_k$, using the trapezoidal rule in po-
        lar coordinates and adding the resulting value to $u_{l,j}$.
    end for
end for
```

Figure 5.3: Singular integral for velocity.

We denote by N the total number of quadrature points $x_{k,i}$. In general, we have $N \approx \frac{K}{h^2}$, where K is the total number of patches covering the boundary Γ. Since K remains constant as we refine h, we have $N = O(\frac{1}{h^2})$. The computation cost of different stages of the algorithm is listed as follows:

1. The kernel independent FMM algorithm has complexity $O(N)$.

2. In the first double **for** loop, for each $x_{l,j}$ and each C_k, since the support of floating POU is of size h, the number of quadrature points at which the evaluation is required is $O(\sqrt{N})$. Therefore, the overall complexity $O(N^{3/2})$.

3. The preprocessing procedure has complexity $O(N \log N)$ due to the efficiency of fast Fourier transform.

4. The second double **for** loop is also $O(N^{3/2})$ since for each the $x_{l,j}$ and each C_k the number of polar-coordinate quadrature points is of order $O(\frac{1}{h}) = O(\sqrt{N})$.

Summing up over all stages of the algorithm, we have shown that the complexity of our quadrature algorithm is $O(N^{3/2})$.

In practice, we notice several important points. First, the constant in the complexity analysis of the kernel independent FMM algorithm is large, despite the fact that the algorithm itself is $O(N)$. Second, the dominant complexity from the correction steps (subtraction and addition) is highly related to the radius of the floating POU. We have chosen the radius to be $\sqrt{h}$, but by using a POU which shrinks faster (e.g. $h^{3/4}$), we can lower the complexity of the algorithm at the cost of decreasing the accuracy. On the contrary, by using a POU which shrinks slower (e.g. $h^{1/4}$), we can lower the error bound but the complexity of the algorithm will increase. In practice, we observe that the FMM computation step is dominant for large values of h while the correction steps are dominant for small values of h.

We denote this quadrature algorithm by an operator Y^D, associated with kernel D. To summarize,

- Y^D is efficient: it has complexity $O(N^{3/2})$.
- Y^D is accurate: $(Y^D\varphi - D\varphi)(x)$ is of order $\max(h^{\frac{M-1}{2}}, \epsilon)$ for x on Γ, where ϵ is the error tolerance used in the kernel independent FMM step.

5.5 Singular Integration for Pressure and Stress

The previous sections described the numerical quadrature algorithm Y^D for the high-order and efficient evaluation of $D\varphi(x)$ for $x \in \Gamma$. With Y^D available, we use iterative linear algebraic solvers (such as GMRES) to solve for the double layer density φ in

$$\frac{1}{2}\varphi + D\varphi = f.$$

In this section, we describe the algorithms to evaluate the limit pressure p and stress s on the boundary Γ from the solution φ. These quantities are of great importance for practical applications of the Stokes equations. We first present the integral formulations for p and s on Γ in terms of the double layer density φ. Similar to the velocity field u, each formulation involves a term which represents the jump of the pressure or stress across the interface, and a singular integral (with higher order singularity). We then derive the jumps and present an algorithm to evaluate the jumps with high-order accuracy. Finally, we present the algorithms to evaluate the related singular integrals using the operator Y.

Let φ be the double layer density on Γ. For $x \in \Omega$, the double layer representation for pressure $p(x)$ is

$$p(x) = (K\varphi)(x) = \int_\Gamma \frac{\mu}{2\pi}\left(\frac{n}{|r|^3} - 3\frac{(r \cdot n(y))r}{|r|^5}\right)\varphi(y)\,\mathrm{d}s(y),$$

where $r = x - y$ and K is used to denote both the kernel and the integral operator. The double layer representation for stress $s(x)$ is

$$s(x) = (T\varphi)(x) = \int_\Gamma T(x,y)\varphi(y)\,\mathrm{d}s(y),$$

where T is quite complicated and given in Appendix B. It is important to notice that both K and T have singularities of order $\frac{1}{|r|^3}$.

In order to derive the formula for x on the boundary Γ, we use the following fact from the potential theory of the Stokes equation: if $\varphi \equiv c$ is a constant, then the velocity field u in Ω generated by φ is again a constant, and correspondingly, the pressure field p and the stress field s are both zero [47, 69, 70]. Suppose $x' \in \Omega$ approaches to the boundary point $x \in \Gamma$, then the following is also true:

$$\begin{aligned} p(x') &= \int_\Gamma K(x',y)(\varphi(y)-\varphi(x))\,\mathrm{d}s(y), \\ s(x') &= \int_\Gamma T(x',y)(\varphi(y)-\varphi(x))\,\mathrm{d}s(y). \end{aligned}$$

However, these two integrals have the same singularity type as the integral formulation for the velocity field u. If x' would be on the boundary Γ, these integrals should be interpreted in the Cauchy sense. We know that for this kind of kernels, the interior limits of $p(x')$ (denoted by $p(x)$) and $s(x')$ (denoted by $s(x)$) have the following integral form:

$$p(x) = \frac{1}{2}[[p]](x) + \int_\Gamma K(x,y)(\varphi(y)-\varphi(x))\,\mathrm{d}s(y), \tag{5.8}$$

$$s(x) = \frac{1}{2}[[s]](x) + \int_\Gamma T(x,y)(\varphi(y)-\varphi(x))\,\mathrm{d}s(y). \tag{5.9}$$

where $[[p]]$ is the difference between the interior limit and the exterior limit of p at x, and $[[s]]$ the difference of s. Notice, both integrals are interpreted in the *Cauchy* sense, and the integrands depend on the evaluation point. We sometimes also write

$$\begin{aligned} p(x) &= \frac{1}{2}[[p]](x) + \int_\Gamma K(x,y)\varphi(y)\,\mathrm{d}s(y), \\ s(x) &= \frac{1}{2}[[s]](x) + \int_\Gamma T(x,y)\varphi(y)\,\mathrm{d}s(y), \end{aligned}$$

where the integral is understood in the *Hadamard* sense [39].

We derive the jump $[[p]](x)$ and $[[s]](x)$ in terms of the double layer density φ.

We are equipped with three conditions:

$$[[u]] = \varphi, \tag{5.10}$$

$$[[(-pI + \mu(\nabla u + \nabla u^t))n]] = 0 \quad \text{Lyaponov-Tauber condition}, \tag{5.11}$$

$$[[\operatorname{div} u]] = 0. \tag{5.12}$$

We choose a local orthonormal frame α, β in the tangent plane at x. Taking the derivative with respect to α and β in (5.10), we get

$$[[\nabla u]]\alpha = \varphi_\alpha \quad [[\nabla u]]\beta = \varphi_\beta.$$

This can be extended into the following six equalities:

$$\begin{aligned} \alpha^t[[\nabla u]]\alpha &= \alpha^t\varphi_\alpha \quad \alpha^t[[\nabla u]]\beta = \alpha^t\varphi_\beta, \\ \beta^t[[\nabla u]]\alpha &= \beta^t\varphi_\alpha \quad \beta^t[[\nabla u]]\beta = \beta^t\varphi_\beta, \\ n^t[[\nabla u]]\alpha &= n^t\varphi_\alpha \quad n^t[[\nabla u]]\beta = n^t\varphi_\beta. \end{aligned}$$

Multiplying (5.11) with α and β, we get two equalities:

$$\alpha^t([[\nabla u]] + [[\nabla u]]^t)n = 0 \quad \beta^t([[\nabla u]] + [[\nabla u]]^t)n = 0.$$

Since $\operatorname{div} u = tr(\nabla u)$ and $tr(A) = tr(ABB^{-1}) = tr(B^{-1}AB)$, we have

$$tr((\alpha, \beta, n)^t[[\nabla u]](\alpha, \beta, n)) = 0.$$

This is equivalent to

$$\alpha^t[[\nabla u]]\alpha + \beta^t[[\nabla u]]\beta + n^t[[\nabla u]]n = 0.$$

From these nine equalities, we get immediately:

$$[[\nabla u]] = (\alpha, \beta, n) \begin{pmatrix} \alpha^t\varphi_\alpha & \alpha^t\varphi_\beta & -n^t\varphi_\alpha \\ \beta^t\varphi_\alpha & \beta^t\varphi_\beta & -n^t\varphi_\beta \\ n^t\varphi_\alpha & n^t\varphi_\beta & -\alpha^t\varphi_\alpha - \beta^t\varphi_\beta \end{pmatrix} (\alpha, \beta, n)^t.$$

The jumps for p and s are

$$[[p]] = -2\mu(\alpha^t\varphi_\alpha + \beta^t\varphi_\beta)$$

and

$$[[s]] = \mu(\alpha,\beta,n)\begin{pmatrix} 4\alpha^t\varphi_\alpha + 2\beta^t\varphi_\beta & \alpha^t\varphi_\beta + \beta^t\varphi_\alpha & 0 \\ \alpha^t\varphi_\beta + \beta^t\varphi_\alpha & 2\alpha^t\varphi_\alpha + 4\beta^t\varphi_\beta & 0 \\ 0 & 0 & 0 \end{pmatrix}(\alpha,\beta,n)^t.$$

To evaluate the jumps $[[p]](x)$ and $[[s]](x)$, we need to evaluate φ_α for a unit direction α in the tangent plane at an arbitrary point x from the values of φ at the quadrature points $x_{k,i}$. One method is the following. First, we pick a patch C_k such that $x = g_k(c'_k)$ for some $c'_k \in C_k$ and calculate the direction q such that $\nabla g_k(c'_k)q = \alpha$. Then, we interpolate the gradient of φ in the direction of q using the values of φ at the Nyström points $c_{k,i}$ in chart C_k. However, since φ is not a periodic function on C_k, this interpolation procedure in general can only be local. Moreover, the interpolation accuracy is limited when c'_k — the point at which we interpolate — is close to the boundary of the domain C_k since the interpolation now becomes an "extrapolation" procedure.

To achieve high-order accuracy for the φ_α, we use the partition of unity. We write

$$\varphi_\alpha(x) = \sum_{k: x\in g_k(C_k)} (w_k\varphi)_\alpha(x).$$

For each k, since $w_k\varphi$ is now a periodic function in domain C_k. Therefore, on each C_k we can use an interpolation procedure similar to the one we developed for interpolating ψ to approximate $(w_k\varphi)_\alpha$. In our implementation, a preprocessing step is performed for each function φ to build an eight-fold refined grid for interpolation. At the evaluation stage, we use Lagrange interpolation again to approximate the value of $(w_k\varphi)_\alpha$.

We now explain the algorithm for the evaluation of the singular integrals in the integration formulation of $p(x)$ and $s(x)$. Here we present the pressure evaluation as an example, and the case for the stress is the same since they have the same singularity. First we observe that in the integral

$$\int_{\Gamma} K(x,y)(\varphi(y)-\varphi(x))\,\mathrm{d}s(y),$$

the integrand $K(x,y)(\varphi(y)-\varphi(x))$ has the form

$$\frac{r}{|r|^3}H(\frac{r}{|r|})S(r),$$

where H is a homogeneous function, and S is a C^M function. Theorem 5.2 guarantees that, by using our numerical integration operator Y^K on the double layer density $\varphi-\varphi(x)$, the result at point x is $O(h^{\frac{M}{2}-1})$ accurate. Notice that $\varphi-\varphi(x)$ depends on the target point x, and the result from using Y^K on $\varphi-\varphi(x)$ is only valid for the pressure at the point x. However, evaluating the operator Y^K once for each x is clearly too expensive and not acceptable. The algorithm we propose uses the linearity of Y^K to evaluate $(K\varphi)(x)$ at all points x much more efficiently.

We write

$$\begin{aligned} Y^K(\varphi-\varphi(x)) &= Y^K\varphi - Y^K\varphi(x) \\ &= Y^K\varphi - Y^K(e^1,e^2,e^3)\varphi(x) \\ &= Y^K\varphi - (Y^K e^1, Y^K e^2, Y^K e^3)\varphi(x), \end{aligned}$$

where e^1, e^2 and e^3 are the constant vector functions defined on Γ with value $(1,0,0)^t$, $(0,1,0)^t$ and $(0,0,1)^t$ respectively. The most important point here is: although $K\varphi(x)$ is not defined for a general function φ (it is only defined for a smooth function φ which vanishes at x), $Y^K\varphi(x)$ is defined for all points x as a *numerical* integration operator. The complete algorithm to integrate $K\varphi(x)$ for a set of points x on Γ is given in Figure 5.4.

```
Evaluate $g^d = Y^K e^d$ for $d = 1, 2, 3$.
Evaluate $p = Y^K \varphi$.
**for** each evaluation point $x$ **do**
  $p(x) \leftarrow p(x) - (g^1(x), g^2(x), g^3(x))\varphi(x)$.
**end for**
```

Figure 5.4: Singular integration for the pressure.

Since the operator Y^K has complexity $O(N^{3/2})$, it is obvious that the current algorithm also has complexity $O(N^{3/2})$. For a fixed combination of evaluation points and the Nyström discretization points, the first step of the algorithm only needs to be done once, and can be reused for different double layer density function φ. This algorithm depends on the idea of cancellation: both the values $(Y^K\varphi)$ and $Y^K\varphi(x)$ at point x are "wrong"; however, their difference gives the correct value. This indicates that the algorithm may potentially suffer from floating point errors due to cancellation. In our numerical experiments which all use double precision floating point, we have not observed the degradation of the accuracy.

The algorithm for the stress s is exactly the same, since the integral formulation of s has the same order of singularity as the one of p. To summarize, the algorithm in Figure 5.4 has $O(N^{3/2})$ complexity and gives $O(h^{\frac{M}{2}-1})$ accuracy for the pressure and stress if the double layer potential φ is C^M continuous. If φ is infinitely smooth, the algorithm described is super-algebraic convergent.

5.6 Evaluation of Nearly Singular Integrals

In this section, we present the algorithm to evaluate the velocity u, pressure p and stress ψ at an arbitrary point x in the domain. We explain the algorithm in terms of the velocity field u, and comment on the algorithms for p and s at the end of this section. For $x \in \Omega$, the integral formulation of u is

$$u(x) = (D\varphi)(x) = \int_{\Gamma} D(x, y)\varphi(y)\,\mathrm{d}y.$$

The integrand is not singular since $D(x, y)$ is not singular when $x \in \Omega$. If x is of a constant distance away from Γ, then we can bound the derivatives of the integrand. Using the trapezoidal rule on each chart $C_k, k = 1, \cdots, K$ with quadrature points $c_{k,i}$ gives high-order accuracy. The difficult part is when x approaches the boundary Γ, in which case $D(x, y)$ can become nearly singular and oscillatory, and there are no *a priori* bounds for the derivatives of $D(x, y)$ as x approaches Γ, and thus the error bounds on the trapezoidal rule do not apply anymore.

The main task of the algorithm described in this section is to evaluate $(D\varphi)(x)$ at any point $x \in \Omega$ with *uniform* high-order accuracy and with high efficiency, given the values of a smooth function φ at the Nyström quadrature points $x_{k,i}$ on Γ.

The idea of our algorithm is to partition Ω into different regions and use different schemes for integral evaluation for each region. Given the discretization spacing h, we partition the domain Ω into three regions:

- $\Omega_0 = \{x \in \Omega | dist(x, \Gamma) \in (\sqrt{h}, \infty)\}$,
- $\Omega_1 = \{x \in \Omega | dist(x, \Gamma) \in (h, \sqrt{h}]\}$,
- $\Omega_2 = \{x \in \Omega | dist(x, \Gamma) \in (0, h]\}$,

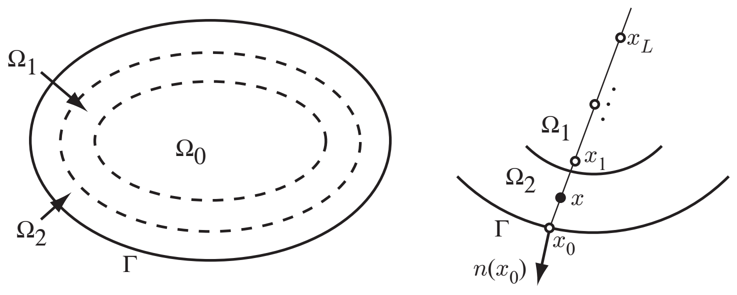


Figure 5.5: Evaluation of nearly singular integrals. Left: different regions based on their distance to the boundary. Right: evaluation procedure for $x \in \Omega_2$.

(see Figure 5.5).

The algorithm works as follows. If $x \in \Omega_0$, we use the trapezoidal on the Nyström points $x_{k,i}$ with weights $\psi_{k,i}$ (which is computed by scaling $\varphi_{k,i}$ using fixed POU and the Jacobian) to evaluate $D\varphi(x)$.

For the points $x \in \Omega_1$, in each chart C_k we resample the function ψ_k on a refined Cartesian grid with spacing $h^{3/2}$ using the values $\psi_{k,i}$ at the old grid $x_{k,i}$ (which has grid size h). Since both grids are Cartesian and ψ_k is periodic function on C_k, the resampling can be done by first computing the discrete Fourier transform of ψ_k, then padding the higher frequency with zeros, and finally using inverse Fourier transform to compute the values on the refined grid. Both the Fourier transform and the inverse Fourier transform can be done using FFT. With ψ_k's values on the refined grid available, we use the trapezoidal rule on the new grid to evaluate $D\varphi(x)$.

For each point $x \in \Omega_2$, we first find a point $x_0 \in \Gamma$ such that

$$\left| \frac{x - x_0}{|x - x_0|} \cdot n(x_0) \right| \geq \alpha,$$

where α is less than but close to 1. This basically states that $x - x_0$ is almost orthogonal to the tangent plane at x_0. We can achieve this by using a Newton-type nonlinear solver to maximize the quantity in the equation above. In practice, the nonlinear solver finds such a point x_0 in a small number of steps. We then define points $x_l, l = 1, \cdots, L$ by

$$x_l = x_0 + l \cdot \frac{x - x_0}{|x - x_0|} \beta h$$

where β is a constant which satisfies $\alpha \cdot \beta \geq 1$. Since α is close to 1, we can choose β to be close to 1 as well. We then use the singular integral algorithm described in Section 5.2 to evaluate $\frac{1}{2}\varphi(x_0) + D\varphi(x_0)$ which is the limit of $D\varphi$ at x_0. The points $\{x_l, l = 1, \cdots L\}$ are now in Ω_1 since $\alpha \cdot \beta > 1$, and we evaluate them using the trapezoidal rule described above. Finally, we use these values to perform a one dimensional Lagrange interpolation to obtain the value of $D\varphi$ at x. In practice, we choose L to be equal to 3 (see Figure 5.5).

We now prove that the procedure described gives high-order convergence.

Lemma 5.7. *For $x \in \Omega_0$, the procedure described gives $O(h^{\frac{M}{2}-1})$ error if φ is C^M continuous.*

Proof. For a fixed x, we can write

$$g(y) = D(x, y)\varphi(y) = \frac{1}{|r|^2} S(r)$$

where S is a C^M function. Since x is at least $\sqrt{h}$ away from Γ, we have the following estimates for the Mth derivatives of g:

$$\|g^{(M)}\|_\infty \leq C_0 \frac{1}{|r|^{M+2}} \leq C_0 \frac{1}{h^{M/2+1}}$$

where C_0 is a constant. The error from the trapezoidal rule is then bounded by

$$\|g^{(M)}\|_\infty \cdot h^M = O(h^{\frac{M}{2}-1}).$$

$\square$

Lemma 5.8. *For $x \in \Omega_1$, the procedure described gives $O(h^{\frac{M}{2}-2})$ error if φ is C^M continuous.*

Proof. For a fixed x, we write again

$$g(y) = D(x,y)\varphi(y) = \frac{1}{|r|^2} S(r)$$

where S is a C^M function. Since x is at least $\sqrt{h}$ away from Γ, we have the following estimates for the Mth derivatives of g:

$$\|g^{(M)}\|_\infty \leq C_0 \frac{1}{|r|^{M+2}} \leq C_0 \frac{1}{h^{M+2}}$$

where C_0 is a constant. The values of ψ (which are computed from φ) on the refined Cartesian grid themselves have at most $O(h^M)$ error since the FFT based refining scheme achieves the maximum accuracy possible. The error from the trapezoidal rule on the refined grid with spacing $h^{3/2}$ can be bounded by

$$\|g^{(M)}\|_\infty \cdot h^{3M/2} = O(h^{\frac{M}{2}-2}).$$

□

Lemma 5.9. *For $x \in \Omega_2$, the procedure described gives $O(h^{min(M/2-2,L)})$ error if φ is C^M continuous.*

Proof. The limit velocity at x_0, $\frac{1}{2}\varphi(x_0) + D\varphi(x_0)$, is $O(h^{\frac{M}{2}-1})$ accurate from Theorem 5.1. From the previous lemma, we know $(D\varphi)(x_l), l = 1, \cdots, L$ are $O(h^{\frac{M}{2}-2})$ accurate. The error introduced by the Lagrange interpolation procedure is of order $O(h^L)$. Therefore, the overall error is at most $O(h^{min(M/2-2,L)})$. □

Combining these lemmas, we are ready to state the following theorem:

Theorem 5.3. *For any $x \in \Omega$, the procedure described gives $O(h^{min(M/2-2,L)})$ error if φ is C^M continuous.*

This procedure can be implemented efficiently. Given a set of $X \subset \Omega$, the algorithm to evaluate the velocities at all x in X is given in Figure 5.6. The complexity of this algorithm is $O(N^{3/2})$.

```
Partition the points in $X$ into three sets $X_0$, $X_1$ and $X_2$ depending on which region ($\Omega_0$, $\Omega_1$ or $\Omega_2$) they belong to.
Evaluate $D\varphi(x)$ for points in $X_0$ using trapezoidal rule on grid points $x_{k,i}$ with FMM acceleration.
Use FFT to interpolate $\psi$ onto a refined grid with spacing $h^{3/2}$.
Evaluate $D\varphi(x)$ for points in $X_1$ using trapezoidal rule on the refined grid with again FMM acceleration.
For every point $x \in X_2$, find its correspondent $x_0$ and $x_l, l = 1, \cdots, L$,
Evaluate $\frac{1}{2}\varphi + D\varphi$ at all points $x_0$ using the algorithm in Section 5.2, again accelerated by FMM algorithm
Evaluate $D\varphi$ at all points $x_l$ using the refined grid with FMM acceleration.
Use Lagrange interpolate to calculate the velocity at $x$s in $X_2$.
```

Figure 5.6: Nearly singular integration for velocity.

The algorithm to evaluate the pressure p and the stress s at arbitrary point x in Ω is similar to the presented algorithm for u. The only difference is that since p and s has a high-order singularity, the error bound for the velocity of points in regions Ω_1 and Ω_2 has a larger estimate, and it is observed in practice. To improve the accuracy, we can apply the idea of singularity subtraction here: for a point x in Ω_1 or Ω_0 which is close to Γ, we find its nearest point x_0 on the boundary. Then instead of evaluating the pressure using

$$\int_\Gamma K(x,y)\varphi(y)\,\mathrm{d}s(y),$$

we use

$$\int_\Gamma K(x,y)(\varphi(y)-\varphi(x_0))\,\mathrm{d}s(y).$$

The integrand in the later is much smoother. Efficient implementation for the second integral follow the ideas in Section 5.5.

5.7 Singular and Nearly Singular Integral Evaluation for Other Equations

In this section, we briefly describe how to adapt the algorithms described in this chapter to other equations, including the Laplace's equation, Navier's equation, and their modified versions. We discuss first the case of singular integral evaluation, then the case of nearly singular integral evaluation.

For the Laplace equation, we can use the operator Y to evaluate the double layer formulation for the potential, since its kernel has the same singularity property as the double layer kernel for the velocity of the Stokes equations. For the potential gradient, the algorithm described in Section 5.5 can be used. The only difference is the jump, which is given in Appendix B.

For the Navier equation, the double layer representation for displacement is an integral of the Cauchy type. As stated in Theorem 5.2, the numerical integration operator Y can still be used. The double kernel for the stress has the same order of singularity as the one for the Stokes equations, and thus the same algorithm can be used. The jump formula are given in Appendix B.

For the modified Stokes equations, since the kernel for the velocity field u has the same singularity behavior as the one for the Stokes equations, Y can be used without any modification. The algorithm for evaluating the pressure p and stress s is different, as we explain as follows using p as an example. We denote by K^M the modified pressure kernel. It can be decomposed into

$$K^M = K^S + K^D,$$

where K^S is the pressure kernel for the Stokes equation, and their difference K^D has no singularity. We now have for $x \in \Omega$,

$$p(x) = (K^M\varphi)(x) = \int_\Gamma K^S(x,y)\varphi(y)\,\mathrm{d}s(y) + \int_\Gamma K^D(x,y)\varphi(y)\,\mathrm{d}s(y),$$

The first integral is equivalent to

$$\int_\Gamma K^S(x,y)(\varphi(y) - \varphi(x))\,\mathrm{d}s(y).$$

Following the same argument in Section 5.5, then we have the following representation for the pressure limit at a boundary point $x \in \Gamma$:

$$p(x) = \frac{1}{2}[[p]](x) + \int_\Gamma K^S(x,y)(\varphi(y) - \varphi(x))\,\mathrm{d}s(y) + \int_\Gamma K^D(x,y)(\varphi(y))\,\mathrm{d}s(y).$$

We use the operator Y to integrate both integrals numerically, though for the second integral a simpler procedure would suffice. The approximation to both integrals takes the following form:

$$\begin{array}{cl} & (Y^{K^S}(\varphi - \varphi(x)))(x) + (Y^{K^D}\varphi)(x) \\ = & (Y^{K^S}\varphi)(x) - (Y^{K^S}\varphi(x))(x) + (Y^{K^D}\varphi)(x) \\ = & ((Y^{K^S} + Y^{K^D})\varphi)(x) - (Y^{K^S}\varphi(x))(x) \\ = & (Y^{K^M}\varphi)(x) - (Y^{K^S}\varphi(x))(x). \end{array}$$

Here we use the fact that Y is a numerical integration operator defined for any functions φ as opposed to a singular integral operator which is only defined on specific kinds of functions. The algorithm to perform the above integration is similar to the one in Figure 5.4 with the only difference that the correction functions g^d are computed using the kernels from the Stokes equation. The jumps for p and s are the same as the ones for the Stokes equations.

The algorithms for the modified Laplace equation and the modified Navier equation are modified in the same way as we described for the modified Stokes equations.

Finally, we point out that, with the available algorithms for evaluating singular integrals, the algorithms for evaluating the nearly singular integrals for the modified equations are identical to the ones for their non-modified versions.

5.8 Summary

In this chapter, we presented several algorithms to integrate the singular integrals coming from the boundary integral formulation. The singular integral was partitioned into two parts: the non-adjacent part and the adjacent part. The smooth non-adjacent part was integrated using the trapezoidal rule. The singular adjacent part was integrated in polar coordinate to remove or decrease the singularity. Our algorithm has high-order efficiency, low complexity, works on the kernels coming from a range of equations without any modification, and has proved error bounds. We also proposed efficient and accurate algorithms to handle the nearly singular integrals which emerges from the evaluation of the boundary integral anywhere in the computation domain.

Chapter 6

Results and Applications

6.1 Results

In this section, we present the numerical results of our 3D boundary integral solver in several examples. All the algorithms described in Chapter 5 have been implemented in C++. The timings cited are the wall-clock time and all the experiments are performed on a Pentium II 650MHz machine.

In every example, we use an exact solution to specify the velocity on the domain boundary which is represented using the high-order surface representation. We then perform several runs with several increasing discretizations.

For a fixed discretization, we carry out the following steps:

- Solve for the double layer density φ on Γ from the specified velocity u. We choose the error tolerance of the FMM algorithm to be 10e-6.
- Select a set of N points on the boundary, where N is the number of discretization used in solving φ, and compute the velocity u and pressure p on these

points. This step is used to test the accuracy of our algorithms for the singular integration.

- Select another set of N points in the domain but close to the boundary, and evaluate u and p on these points. This step is used to test the accuracy of our algorithms for integrating nearly singular integrals.

For both the singular and nearly singular integral evaluation, we use the following formulas to test the accuracy of our algorithm:

$$\epsilon_u = \left(\frac{\sum_{i=1}^{N} |u_i - \tilde{u_i}|^2}{\sum_{i=1}^{N} |\tilde{u_i}|^2} \right)^{1/2},$$

$$\epsilon_p = \left(\frac{\sum_{i=1}^{N} |p_i - \tilde{p_i}|^2}{\sum_{i=1}^{N} |\tilde{p_i}|^2} \right)^{1/2}$$

where u_i and p_i are the velocity and pressure computed by our algorithms and $\tilde{u_i}$ and $\tilde{p_i}$ are those of the exact solution.

For each example, we organize the table into three parts corresponding to the three steps of the tests. In the first part which reports the computation time for solving φ, the columns of the table represent the following quantities:

- $Discretization$: the number of charts and the size of Cartesian grid used for discretization in each chart.

- $iters$: the number of iterations used in the GMRES solver for φ.

- $T_{total}/iter$: the time used to evaluate $D\varphi$ in each iteration.

- $T_{fmm}/iter$: the time spent on the FMM computation for the non-adjacent integration in each iteration.

- $T_{mod}/iter$: the time spent on integrating the local singular integral in each iteration, including subtracting part of the FMM computation, and adding the results by integration in polar coordinates.

The second part reports the time and error of evaluating u and p on the boundary Γ. The columns besides $Discretization$ are:

- T_u: the time used for evaluating velocity.

- ϵ_u: the error of the evaluated velocity.

- T_p: the time used for evaluating pressure.

- ϵ_p: the error of the evaluated pressure.

The last part of the table reports the time and error of evaluating u and p in the domain. The columns have the same meaning as the ones in the second part of the table.

The associated figure consists of three plots. The first plot reports the relationship between the time spent on each GMRES iteration and the number of discretization point N. In the second and third plots, we show the errors of our algorithms for different discretizations, where $S \approx O(\frac{1}{h})$ is the number of discretization points in each direction of the 2D Cartesian grid used for discretizing every chart.

The boundaries of the domains used in the examples are shown in Figure 6.1. Every boundary is contained in the cube with range $[-1, 1]$ in every dimension.

Example 1. This simple example is an interior problem with the first surface in Figure 6.1 as the domain boundary. The exact solution used is

$$u = (y - y^2, 0, 0)^t \quad \text{and} \quad p = -2\mu x.$$

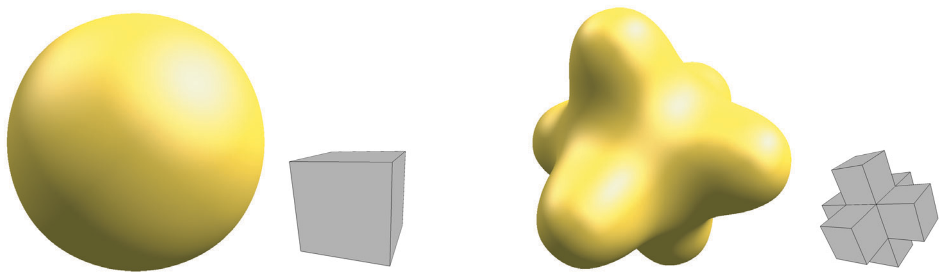


Figure 6.1: Domains.

The results are shown in Table 6.1 and Figure 6.2.

Example 2. We use the second surface in Figure 6.1 as the domain boundary. This is an interior problem and the exact solution is

$$u = (y^3z, z^3x, x^3y)^t \quad \text{and} \quad p = 6\mu xyz$$

The results are shown in Table 6.2 and Figure 6.3.

Example 3. In this example, the domain is a unbound region with boundary as the second surface in Figure 6.1. The exact solution for this exterior problem is a unit Stokeslet centered at the point $(0.1, 0.1, 0.1)^t$. The results are given in Table 6.3 and Figure 6.4.

$Discretization$	$iters$	$T_{total}/iter$	$T_{fmm}/iter$	$T_{mod}/iter$
$8 \times 12 \times 12$	8	4.70e+00	3.59e+00	1.04e+00
$8 \times 24 \times 24$	8	2.25e+01	1.43e+01	7.88e+00
$8 \times 48 \times 48$	7	1.31e+02	6.17e+01	6.54e+01
$8 \times 96 \times 96$	7	7.67e+02	2.28e+02	5.36e+02

$Discretization$	T_u	ϵ_u	T_p	ϵ_p
$8 \times 12 \times 12$	4.68e+00	8.90e-04	1.52e+00	1.84e-01
$8 \times 24 \times 24$	2.19e+01	1.15e-04	7.76e+00	5.64e-02
$8 \times 48 \times 48$	1.28e+02	1.51e-05	3.98e+01	1.21e-02
$8 \times 96 \times 96$	7.90e+02	1.74e-06	2.46e+02	3.14e-03

$Discretization$	T_u	ϵ_u	T_p	ϵ_p
$8 \times 12 \times 12$	7.94e+00	1.41e-03	2.84e+00	2.34e-01
$8 \times 24 \times 24$	6.31e+01	1.62e-04	2.14e+01	6.14e-02
$8 \times 48 \times 48$	5.12e+02	2.03e-05	1.84e+02	1.51e-02
$8 \times 96 \times 96$	4.36e+03	2.41e-06	1.44e+03	4.45e-03

Table 6.1: Results of Example 1.

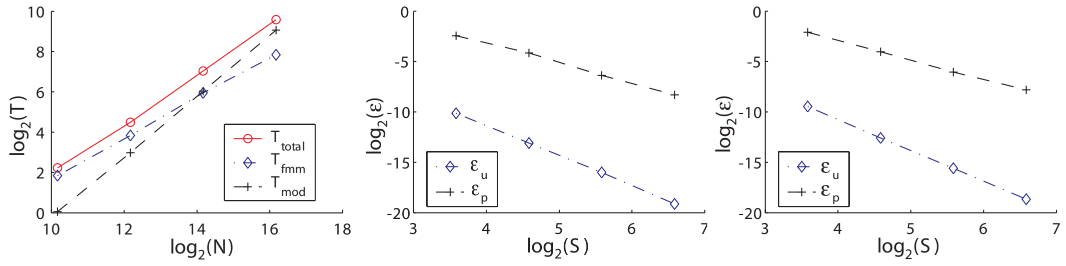


Figure 6.2: Results of Example 1.

As shown by the results, the time T_{fmm} spent on FMM in each iteration of the GMRES solver increases linearly with N — the number of discretization points, while the time T_{mod} spent on the local integrator for the singularity increases as $O(N^{3/2})$. This matches with the complexity analysis of our algorithms. In our

$Discretization$	$iters$	$T_{total}/iter$	$T_{fmm}/iter$	$T_{mod}/iter$
$32 \times 12 \times 12$	15	2.29e+01	1.44e+01	8.16e+00
$32 \times 24 \times 24$	15	1.27e+02	5.71e+01	6.65e+01
$32 \times 48 \times 48$	14	7.71e+02	2.37e+02	5.17e+02

$Discretization$	T_u	ϵ_u	T_p	ϵ_p
$32 \times 12 \times 12$	2.19e+01	3.19e-03	7.35e+00	1.82e-01
$32 \times 24 \times 24$	1.23e+02	4.07e-04	4.24e+01	4.64e-02
$32 \times 48 \times 48$	7.63e+02	4.32e-05	2.62e+02	6.65e-03

$Discretization$	T_u	ϵ_u	T_p	ϵ_p
$32 \times 12 \times 12$	3.44e+01	4.05e-03	1.13e+01	2.59e-01
$32 \times 24 \times 24$	2.67e+02	5.24e-04	9.04e+01	6.98e-02
$32 \times 48 \times 48$	2.12e+03	6.76e-05	7.15e+02	8.39e-03

Table 6.2: Results of Example 2.

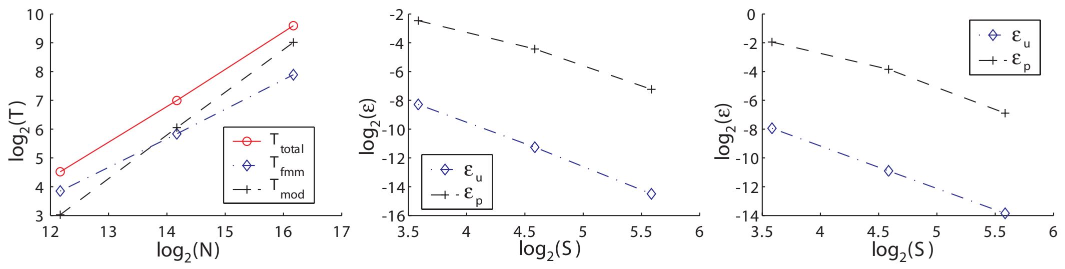


Figure 6.3: Results of Example 2.

examples, for coarse discretizations, the FMM part takes more time than the local integrator, while for finer discretizations, the situation is reversed. We note that the point where this transition happens depends on various parameters used in our algorithms, most notably, the tolerance of the FMM algorithm and the support of the floating partition of unity (POU).

The time used on checking u and p on the boundary Γ is proportional to the time used by every iteration of the GMRES solver. The error plots show that convergence

$Discretization$	$iters$	$T_{total}/iter$	$T_{fmm}/iter$	$T_{mod}/iter$
$32 \times 12 \times 12$	17	2.46e+01	1.47e+01	8.35e+00
$32 \times 24 \times 24$	17	1.31e+02	6.10e+01	6.39e+01
$32 \times 48 \times 48$	16	7.44e+02	2.37e+02	4.91e+02

$Discretization$	T_u	ϵ_u	T_p	ϵ_p
$32 \times 12 \times 12$	2.35e+01	1.70e-03	7.44e+00	1.73e-01
$32 \times 24 \times 24$	1.19e+02	2.38e-04	3.99e+01	4.39e-02
$32 \times 48 \times 48$	7.87e+02	2.91e-05	2.54e+02	6.08e-03

$Discretization$	T_u	ϵ_u	T_p	ϵ_p
$32 \times 12 \times 12$	3.40e+01	2.87e-03	1.12e+01	2.51e-01
$32 \times 24 \times 24$	2.73e+02	3.37e-04	9.08e+01	7.08e-02
$32 \times 48 \times 48$	2.02e+03	4.29e-05	7.19e+02	7.84e-03

Table 6.3: Results of Example 3.

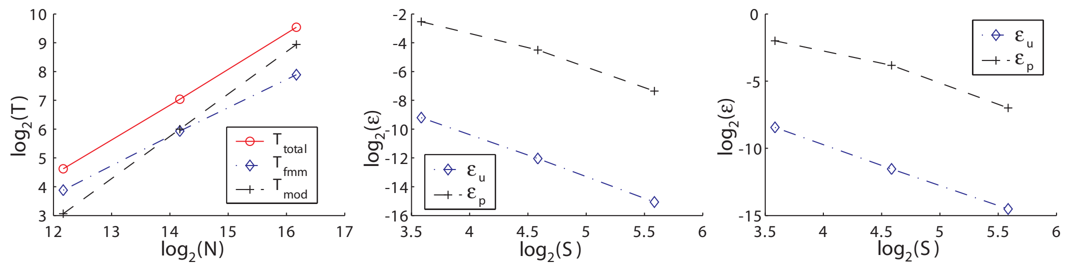


Figure 6.4: Results of Example 3.

rate of u is roughly h^3, while the convergence rate of p is about h^2. This demonstrates the high-order behavior of our algorithms for singular integrals. We notice that for the coarsest discretization with 12 points per dimension, the error for p is relatively large, usually of order 10e-1. The reason is that, in such case, the number of discretization points is too low for the interpolation procedure for the jumps of the pressure to approximate even the derivatives of the fixed partition of unity, which is used in our surface representation. One of the possible solutions is to use a smoother partition of unity.

The time used by the checks of u and p in the domain Ω grows like $O(N^{3/2})$. Most of the computation time is spent on the evaluation at points in regions Ω_1 and Ω_2, where an FMM evaluation on a finer discretization with spacing $O(h^{3/2})$ is required.

6.2 Applications

In this section, we presents several applications which use the our boundary integral solver.

Embedded boundary integral solver. This is a method for the solution of the Stokes equations with distributed force. The equations are given by:

$$\begin{aligned} -\mu\Delta u + \nabla p &= b \quad \text{in} \quad \Omega, \\ \operatorname{div} u &= 0 \quad \text{in} \quad \Gamma, \\ u &= f \quad \text{on} \quad \Gamma, \end{aligned} \tag{6.1}$$

where b is a known forcing term.

Following [52], we split the solution of the problem into several steps as follows. We first embed Ω in a domain Ω' which can be discretized easily, typically a rectangle (Figure 6.5). By linearity we decompose (6.1) into two problems: one problem that has an inhomogeneous body force and zero boundary conditions for Ω'; the other on Ω has no body force, but nontrivial boundary conditions:

$$\begin{aligned} -\mu\Delta u_1 + \nabla p_1 &= b \quad \text{in} \quad \Omega', \\ \operatorname{div} u_1 &= 0 \quad \text{in} \quad \Gamma', \\ u_1 &= 0 \quad \text{on} \quad \Gamma', \end{aligned} \tag{6.2}$$

where Γ' is the boundary of Ω', and

$$\begin{aligned} -\mu\Delta u_2 + \nabla p_2 &= 0 \quad \text{in} \quad \Omega, \\ \operatorname{div} u_2 &= 0 \quad \text{in} \quad \Gamma, \\ u_2 &= f - u_1 \quad \text{on} \quad \Gamma. \end{aligned} \tag{6.3}$$

For (6.2), we discretize the domain Ω' by a regular grid and use the Q1-Q1 finite element method to solve the equation. For (6.3), we use the boundary integral solver described in this thesis. The solution of the original problem (6.1) is $u = u_1 + u_2$ and $p = p_1 + p_2$.

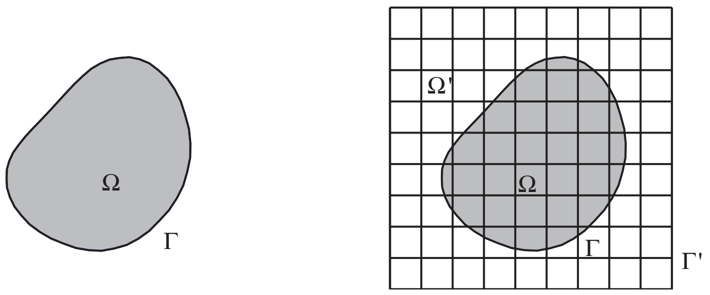


Figure 6.5: Domains in the embedded boundary integral solver. Left: the origin domain Ω with boundary Γ. Right: Ω is embedded into a rectangular domain Ω' with boundary Γ'.

Interaction between Stokes fluid and rigid body objects. The simulation of the motion and dynamics of a rigid object immersed in a viscous fluid is important to various applications in biomedical engineering (Figure 6.6). We use the *linear* integral formulation of the Stokes equation to model the viscous fluid. The motion of the objects are modeled by the rigid body dynamics. The interface conditions are the balance of the force and the continuity of the interface velocity. We use semi-implicit time discretization: at each time step, we first update the position of the rigid objects using the old velocity, then we solve a coupled nonlinear system (including the boundary integral equation for the Stokesian fluid, the rigid body equations and the coupling conditions) to compute the interface quantities (including the velocity and force).

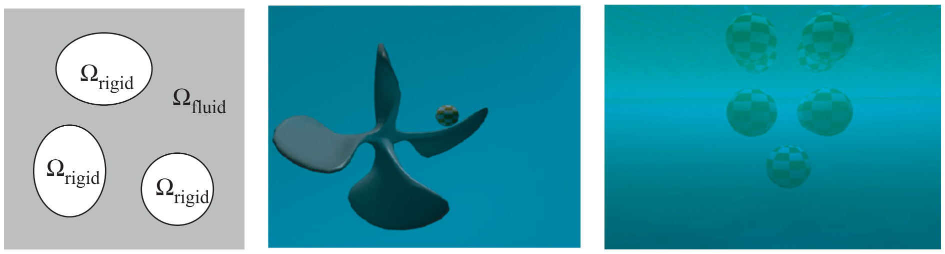


Figure 6.6: Interaction between viscous fluid and rigid body objects. Left: fluid and object domains. Middle and Right: frames from the simulations of fluid and rigid object interaction.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we have presented an efficient and high-order boundary integral solver for the Stokes Equations in complex 3D geometries. Our solver uses indirect boundary integral formulation and discretizes the equation using the Nyström method to achieve high-order accuracy. Although it is developed for the Stokes equations, we have shown that it can be used for other elliptic PDEs with minimal modification. The three major components of our solver are:

C^∞ **Surface Representation.** We have presented a simple method to model C^∞ surfaces. We first use polynomials as bases to construct local geometry on each C^∞ smooth chart, then blend them together to obtain the global geometry using a C^∞ partition of unity. The generated surfaces are C^∞-continuous with explicit nonsingular C^∞ parameterizations, have high-order flexibility at control vertices, depend linearly on control points, have fixed-size local support for basis functions, and demonstrate good visual quality.

Kernel Independent Fast Multipole Method and its Parallel Implementation. We have developed a kernel independent fast multipole algorithm, which generalizes FMM to a wide range of kernels. We use “equivalent densities” to replace the multipole and local expansions, and develop translation procedures which only use the kernel evaluation. In the 3D case, the translation procedure is implemented using FFT to achieve maximal efficiency. Our algorithm is adaptive and has the same structure as the original FMM algorithm. It exhibits comparable efficiency and accuracy results with the best published FMM implementation and has a proved error bound. We also provide an MPI-based parallel implementation for distributed memory architectures. Our implementation achieves computation and communication balance and demonstrates good scalability results.

High-order efficient Nyström Integrator. We have presented algorithms to integrate with high-order accuracy the singular integrals coming from the boundary integral formulation. The singular integral is partitioned into two parts: the adjacent part and the non-adjacent part. The non-adjacent part is smooth and we integrate it using the trapezoidal rule. The adjacent part is singular and we integrate it in polar coordinates to remove or decrease the singularity. Our algorithm has low complexity, works on the kernels coming from a range of equations without any modification, and has proved error bounds. We also proposed efficient and accurate algorithms to handle the nearly singular integrals which emerge from the evaluation of the boundary integral anywhere in the computation domain.

7.2 Future Work

High-order Surface Representation. Our construction can be improved in a variety of ways, as most of the components were identified empirically: in particular, to get good behavior for higher order derivatives, one needs a better partition of unity. As some alternatives mentioned in Section 2.3.6, it is quite possible that there are better charts, fewer or lower degree polynomials to be used for geometric functions, or entirely different geometric functions which can yield better results.

Another direction is to generalize our construction to model manifolds with boundaries, crease edges or even corners. Although most of these extension are straightforward, they are the first step to extend our boundary integral solver to domains with non-smooth boundaries.

For most of the current applications, the domain boundary is represented with either a piecewise linear triangle mesh or a densely sampled point cloud, even though the boundary itself is high-order or even infinitely smooth. A common example is the sphere, which is C^∞ smooth but usually represented by a triangle mesh in most of the finite element or boundary element simulations. An interesting but challenging problem is to transform these triangle meshes or densely sampled point clouds into high-order representations like our scheme.

Kernel Independent Fast Multipole Method and its Parallel Implementation. In our kernel independent FMM algorithm, we have focused on second order constant coefficient PDEs with non-oscillatory solutions. However, our method is not restricted to such systems. It should be straightforward to generalize it to higher order systems like the biharmonic equation. In such cases, the Dirichlet problem involves first and second derivatives of the underlying field. We can either differentiate the kernel to obtain the derivatives or use a set of two check-point surfaces. We plan to explore this approach in the future.

Another class of problems is related to second order PDEs with oscillatory solutions or Helmholtz-type problems. For low frequencies we have performed preliminary tests (on the M2M and L2L transformations) that indicate that our method works as is. An implementation for this class of problems is under way.

Although our parallel implementation exhibits good scalability results, two problems in our implementation are the tree construction algorithm and the inefficient load balancing algorithm which create problems for more than 1024 processors. We are currently working on adaptations of our algorithm for applications such as molecular dynamics, where efficient tree construction is much more important.

High-order efficient Nyström Integrator. Our Nyström integrator successfully integrates the singular and nearly singular integrals from the boundary integral formulation with high efficiency and high-order accuracy. However, our choices on some parameters of the algorithm (e.g., the radius of the floating POU) are not optimal. One can study how to choose these parameters depending on the efficiency and accuracy required for the real life applications.

In our algorithm, we discretize each chart of the boundary using an evenly spaced Cartesian grid. In most of the problems, however, in order to be more efficient, one needs to take an adaptive approach: to discretize the boundary based on the complexity of the geometry and the behavior of the boundary condition. One approach is based on the idea of the partition of unity. Instead of using one POU function at each chart, one can use several POU functions which cover regions with different geometric complexity. The chart is then sampled with several overlapping Cartesian grids with different spacings, one for each POU. The spacings are chosen based on the behavior of the geometry and boundary function at the support of each POU.

The idea behind our Nyström integrator can be also readily extended to integrate singular and hyper-singular volume integrals.

Boundary Integral Solver. The boundary integrals of the second order have theoretically good spectral properties which make iterative methods the standard approach to solve these equations. However, when we deal with domains with high curvature and/or complexity topology, the condition number can grow quickly to a point that plain iterative solvers can still be quite slow. Our Nyström solver can be adapted into the existing two-grid and multi-grid preconditioners simply by discretizing the charts using coarser Cartesian grids.

The solver proposed in this thesis does not handle boundaries with crease edges or corners. To handle these sharp features, one method is to use the charts which are degenerate at these features. All components of our Nyström integrator remain the same except we need to increase the radius of the floating POU around these features in order to achieve high-order accuracy. However, the complexity of this method will increase substantially. A challenging problem is to develop integrators which are both efficient and accurate for boundaries with these sharp features.

We have mentioned the approach of using the Galerkin method with wavelet decomposition to solve boundary integral equations of the second kind. Our surface representation makes the wavelet design easy in this case: we simply define wavelets of periodic domain in the chart first and then lift them onto the surface. One interesting direction is to develop a Galerkin method solver based on this kind of wavelets and compare its efficiency and accuracy with the Nyström solver described in this thesis.

Appendix A

Kernels Tested with FMM

In this appendix, we give a summary of the kernels tested with our kernel independent fast multipole method. In the following formulas, y is the location of the singularity, x is the location the evaluation point, n is the surface normal direction at y, $r = x - y$. We use S to stand for the single layer kernel and D for the double layer kernel.

Laplace Equation.

$$-\Delta u = 0,$$

$$S(x,y) = \begin{cases} \frac{1}{2\pi} \ln \frac{1}{|r|} & \text{(2D)} \\ \frac{1}{4\pi} \frac{1}{|r|} & \text{(3D)} \end{cases} \quad D(x,y) = \begin{cases} -\frac{1}{2\pi} \frac{1}{|r|^2}(r \cdot n) & \text{(2D)} \\ -\frac{1}{4\pi} \frac{1}{|r|^3}(r \cdot n) & \text{(3D)} \end{cases}.$$

Modified Laplace Equation.

$$\alpha u - \Delta u = 0,$$

$$S(x,y) = \begin{cases} \frac{1}{2\pi} k_0(\lambda|r|) & \text{(2D)} \\ \frac{1}{4\pi} \frac{1}{|r|} e^{-\lambda|r|} & \text{(3D)} \end{cases} \quad D(x,y) = \begin{cases} -\frac{\lambda}{2\pi} \frac{k_1(\lambda|r|)}{|r|}(r \cdot n) & \text{(2D)} \\ -\frac{e^{-\lambda|r|}}{4\pi} \left(\frac{1}{|r|^3} + \frac{\lambda}{|r|^2} \right)(r \cdot n) & \text{(3D)} \end{cases},$$

where $\lambda = \sqrt{\alpha}$.

Stokes Equation (Incompressible creeping flows).

$$-\mu\Delta u + \nabla p = 0, \;\; \operatorname{div} u = 0$$

$$S(x,y) = \begin{cases} \frac{1}{4\pi\mu}(\ln\frac{1}{|r|}I + \frac{r\otimes r}{|r|^2}) & \text{(2D)} \\ \frac{1}{8\pi\mu}(\frac{1}{|r|}I + \frac{r\otimes r}{|r|^3}) & \text{(3D)} \end{cases} \quad D(x,y) = \begin{cases} -\frac{1}{\pi}\frac{r\otimes r}{|r|^4}(r\cdot n) & \text{(2D)} \\ -\frac{6}{8\pi}\frac{r\otimes r}{|r|^5}(r\cdot n) & \text{(3D)} \end{cases}.$$

Modified Stokes Equation (Unsteady incompressible creeping flows).

$$\alpha u - \mu\Delta u + \nabla p = 0, \;\; \operatorname{div} u = 0$$

$$S(x,y) = \frac{1}{\mu}\left(G(|r|)I + H(|r|)(r\otimes r)\right),$$

$$D(x,y) = A(|r|)\left((r\cdot n)I + n\otimes r\right) + B(|r|)(r\otimes n) + C(|r|)(r\cdot n)(r\otimes r),$$

where

$$G(s) = -f''(s) - (d-2)\frac{f'(s)}{s},$$

$$H(s) = \frac{f''(s)}{s^2} - \frac{f'(s)}{s^3},$$

$$A(s) = -\frac{f'''(s)}{s} - (d-3)\frac{f''(s)}{s^2} + (d-3)\frac{f'(s)}{s^3},$$

$$B(s) = -p + 2\frac{f''(s)}{s^2} - 2\frac{f'(s)}{s^3},$$

$$C(s) = 2\frac{f'''(s)}{s^3} - 6\frac{f''(s)}{s^4} + 6\frac{f'(s)}{s^5},$$

and

$$f(s) = \begin{cases} \frac{1}{2\pi\lambda^2}(\ln(\frac{1}{s}) - k_0(\lambda s)) & \text{(2D)} \\ \frac{1}{4\pi\lambda^2}(\frac{1}{s} - \frac{1}{s}e^{-\lambda s}) & \text{(3D)} \end{cases}, \quad p(s) = \begin{cases} \frac{1}{2\pi}\frac{1}{s^2} & \text{(2D)} \\ \frac{1}{4\pi}\frac{1}{s^4} & \text{(3D)} \end{cases}, \quad \lambda = \sqrt{\frac{\alpha}{\mu}}.$$

Navier Equation (Elastostatics).

$$-\mu\Delta u - \frac{\mu}{1-2\nu}\nabla\cdot\,\mathrm{div}\,u = 0$$

$$S(x,y) = \begin{cases} \frac{1}{\mu}\left(\frac{3-4\nu}{8\pi(1-\nu)}\log(\frac{1}{|r|}) + \frac{1}{8\pi(1-\nu)}\frac{(r\otimes r)}{|r|^2}\right) & \text{(2D)} \\ \frac{1}{\mu}\left(\frac{3-4\nu}{16\pi(1-\nu)}\frac{1}{|r|} + \frac{1}{16\pi(1-\nu)}\frac{(r\otimes r)}{|r|^3}\right) & \text{(3D)} \end{cases}$$

$$D(x,y) = \begin{cases} \frac{1-2\nu}{4\pi(1-\nu)}\left(-\frac{((r\cdot n)I+n\otimes r)}{|r|^2} + \frac{(r\otimes n)}{|r|^2} - \frac{2}{1-2\nu}\frac{(r\cdot n)(r\otimes r)}{|r|^4}\right) & \text{(2D)} \\ \frac{1-2\nu}{8\pi(1-\nu)}\left(-\frac{((r\cdot n)I+n\otimes r)}{|r|^3} + \frac{(r\otimes n)}{|r|^3} - \frac{3}{1-2\nu}\frac{(r\cdot n)(r\otimes r)}{|r|^5}\right) & \text{(3D)} \end{cases}.$$

Modified Navier Equation (Elastodynamics).

$$\alpha u - \mu\Delta u - \frac{\mu}{1-2\nu}\nabla\cdot\,\mathrm{div}\,u = 0$$

$$S(x,y) = \frac{1}{\mu}\left(G(|r|)I + H(|r|)(r\otimes r)\right),$$

$$D(x,y) = A(|r|)\left((r\cdot n)I + n\otimes r\right) + B(|r|)(r\otimes n) + C(|r|)(r\cdot n)(r\otimes r),$$

where

$$G(s) = \eta^2 f - f''(s) + (\beta+1-d)\frac{f'(s)}{s},$$

$$H(s) = \beta\frac{f''(s)}{s^2} - \beta\frac{f'(s)}{s^3},$$

$$A(s) = -\frac{1}{s}f'''(s) + \frac{2\beta+1-d}{s^2}f''(s) + (\frac{\eta^2}{s} - \frac{2\beta+1-d}{s^3})f'(s),$$

$$\begin{aligned} B(s) \quad &= \quad \frac{\gamma(\beta-1)}{s}f'''(s) + \frac{2\beta+\gamma(\beta-1)(d-1)}{s^2}f''(s) + \\ &\quad (\frac{\gamma\eta^2}{s} - \frac{2\beta+\gamma(\beta-1)(d-1)}{s^3})f'(s), \end{aligned}$$

$$C(s) = \frac{2\beta}{s^3}f'''(s) - \frac{6\beta}{s^4}f''(s) + \frac{6\beta}{s^5}f'(s),$$

and

$$f(s) = \begin{cases} \frac{1}{2\pi(\lambda^2-\eta^2)}(k_0(\eta s) - k_0(\lambda s)) & \text{(2D)} \\ \frac{1}{4\pi(\lambda^2-\eta^2)}(\frac{1}{s}e^{-\eta s} - \frac{1}{s}e^{-\lambda s}) & \text{(3D)} \end{cases},$$

$$\lambda = \sqrt{\frac{\alpha}{\mu}}, \quad \eta = \sqrt{\frac{1-2\nu}{2(1-\nu)} \cdot \frac{\alpha}{\mu}}, \quad \beta = \frac{1}{2(1-\nu)}, \quad \gamma = \frac{2\nu}{1-2\nu}.$$

Appendix B

Jumps and Integrals

In this appendix, we give a summary of the boundary integral formulas for several elliptic PDEs and their associated jumps across the boundary. For the jumps, we choose orthonormal directions α and β in the tangent plane at point x.

Laplace Equation. Potential:

$$u(x) = \frac{1}{2}[[u]](x) + \int_\Gamma D(x,y)\varphi(y)\,\mathrm{d}s(y)$$

$$[[u]] = \varphi$$

$$D(x,y) = -\frac{1}{4\pi}\frac{(r \cdot n(y))}{|r|^3}$$

Gradient of potential:

$$\nabla u = \frac{1}{2}[[\nabla u]] + \int_\Gamma T(x,y)\varphi(y)\,\mathrm{d}s(y)$$

$$[[\nabla u]] = (\varphi_\alpha, \varphi_\beta, 0)(\alpha, \beta, n)^t$$

$$T(x,y) = -\frac{1}{4\pi}\left(\frac{n(y)}{|r|^3} - 3\frac{(r\cdot, n)r}{|r|^5}\right)$$

Stokes Equations. Velocity:

$$u(x) = \frac{1}{2}[[u]](x) + \int_\Gamma D(x,y)\varphi(y)\,\mathrm{d}s(y)$$

$$[[u]] = \varphi$$

$$D(x,y) = -\frac{6}{8\pi}\frac{r \otimes r}{|r|^5}(r \cdot n(y))$$

Pressure:

$$p(x) = \frac{1}{2}[[p]](x) + \int_\Gamma K(x,y)\varphi(y)\,\mathrm{d}s(y)$$

$$[[p]] = -2\nu(\alpha^t\varphi_\alpha + \beta^t\varphi_\beta)$$

$$K(x,y) = \frac{\nu}{2\pi}\left(\frac{n(y)}{|r|^3} - 3\frac{(r \cdot n(y))r}{|r|^5}\right)$$

Stress:

$$s(x) = \frac{1}{2}[[s]](x) + \int_\Gamma T(x,y)\varphi(y)\,\mathrm{d}s(y)$$

$$[[s]] = \mu(\alpha,\beta,n)\begin{pmatrix} 4\alpha^t\varphi_\alpha + 2\beta^t\varphi_\beta & \alpha^t\varphi_\beta + \beta^t\varphi_\alpha & 0 \\ \alpha^t\varphi_\beta + \beta^t\varphi_\alpha & 2\alpha^t\varphi_\alpha + 4\beta^t\varphi_\beta & 0 \\ 0 & 0 & 0 \end{pmatrix}(\alpha,\beta,n)^t$$

$$\begin{aligned} T(x,y)\varphi(y) \quad = \quad & -\frac{6\nu}{8\pi}\left\{\frac{(r \otimes \varphi(y) + \varphi(y) \otimes r)(r \cdot n(y))}{r^5} + \right. \\ & \frac{(r \otimes n(y) + n(y) \otimes r)(r \cdot \varphi(y))}{r^5} + \\ & \left. \frac{(n(y) \cdot \varphi(y))}{r^3}I - 5\frac{(r \cdot \varphi(y))(r \cdot n(y))r \otimes r}{r^7}\right\} \end{aligned}$$

Navier Equation. Displacement:

$$u(x) = \frac{1}{2}[[u]](x) + \int_\Gamma D(x,y)\varphi(y)\,\mathrm{d}s(y)$$

$$[[u]] = \varphi$$

$$D(x,y) = \frac{1-2\nu}{8\pi(1-\nu)}\left\{-\frac{((r\cdot n(y))I + n\otimes r)}{|r|^3} + \frac{(r\otimes n(y))}{|r|^3} - \frac{3}{1-2\nu}\frac{(r\cdot n(y))(r\otimes r)}{|r|^5}\right\}$$

Stress:

$$s(x) = \frac{1}{2}[[s]](x) + \int_\Gamma T(x,y)\varphi(y)\,\mathrm{d}s(y)$$

$$[[s]] = \mu(\alpha,\beta,n)\begin{pmatrix} \frac{4}{2-2\nu}\alpha^t\varphi_\alpha + \frac{4\nu}{2-2\nu}\beta^t\varphi_\beta & \alpha^t\varphi_\beta + \beta^t\varphi_\alpha & 0 \\ \alpha^t\varphi_\beta + \beta^t\varphi_\alpha & \frac{4\nu}{2-2\nu}\alpha^t\varphi_\alpha + \frac{4}{2-2\nu}\beta^t\varphi_\beta & 0 \\ 0 & 0 & 0 \end{pmatrix}(\alpha,\beta,n)^t$$

$$\begin{aligned} T(x,y)\varphi(y) &= \frac{\mu(1-2\nu)}{8\pi(1-2\nu)}\left\{-\frac{n(y)\otimes\varphi(y) + \varphi(y)\otimes n(y)}{|r|^3}\right. \\ &+\frac{-6\nu}{1-2\nu}(r\cdot\varphi(y))\frac{r\otimes n(y) + n(y)\otimes r}{|r|^5} \\ &+\frac{-6\nu}{1-2\nu}(r\cdot n(y))\frac{r\otimes\varphi(y) + \varphi(y)\otimes r}{|r|^5} \\ &+(\frac{2-8\nu}{1-2\nu}\frac{(r\cdot n(y))}{|r|^3} - \frac{6}{1-2\nu}\frac{(r\cdot n(y))(r\cdot\varphi(y))}{|r|^5})I \\ &\left.+ (-6\frac{n(y)\cdot\varphi(y)}{|r|^5} + \frac{30}{1-2\nu}\frac{(r\cdot n(y))(r\cdot\varphi(y))}{|r|^5}(r\otimes r)\right\}. \end{aligned}$$

Bibliography

[1] C. R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific and Statistical Computing*, 13(4):923–947, 1992.

[2] K. E. Atkinson. *The numerical solution of integral equations of the second kind*, volume 4 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 1997.

[3] C. T. H. Baker. *The numerical treatment of integral equations*. Clarendon Press, Oxford, 1977. Monographs on Numerical Analysis.

[4] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, L. C. McInnes, and B. F. Smith. PETSc home page. http://www.mcs.anl.gov/petsc, 2001.

[5] C. L. Berman. Grid-multipole calculations. *SIAM Journal on Scientific Computing*, 16(5):1082–1091, 1995.

[6] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms. I. *Comm. Pure Appl. Math.*, 44(2):141–183, 1991.

[7] G. Biros, L. Ying, and D. Zorin. The embedded boundary integral method for the unsteady incompressible Navier-Stokes equations. Tech-

nical Report TR2003-838, Courant Institute, New York University, 2002. http://www.cs.nyu.edu/csweb/Research/TechReports/TR2003-838/TR2003-838.pdf.

[8] G. Biros, L. Ying, and D. Zorin. A fast solver for the Stokes equations with distributed forces in complex geometries. *J. Comput. Phys.*, 193(1):317–348, 2004.

[9] R. L. Bishop and R. J. Crittenden. *Geometry of manifolds*. AMS Chelsea Publishing, Providence, RI, 2001. Reprint of the 1964 original.

[10] G. Blelloch and G. Narlikar. A practical comparison of n-body algorithms. In *Parallel Algorithms*, Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1997.

[11] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz. AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems. *Radio Sci.*, 31:1225, 1996.

[12] H. Bohl and U. Reif. Degenerate Bézier patches with continuous curvature. *Comput. Aided Geom. Design*, 14(8):749–761, 1997.

[13] W. M. Boothby. *An introduction to differentiable manifolds and Riemannian geometry*, volume 120 of *Pure and Applied Mathematics*. Academic Press Inc., Orlando, FL, second edition, 1986.

[14] C. Brebbia, J. Telles, and L. Wrobel. *Boundary element techniques*. Springer-Verlag,, Berlin, 1984.

[15] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[16] O. P. Bruno and L. A. Kunyansky. A fast, high-order algorithm for the solution of surface scattering problems: Basic implementation, tests, and applications. *Journal of Computational Physics*, 169:80–110, 2001.

[17] X.-C. Chai and O. B. Widlund. Domain decomposition algorithms for indefinite elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, 13(1):243–258, 1992.

[18] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta Numerica*, 1994.

[19] G. Chen and J. Zhou. *Boundary element methods*. Computational Mathematics and Applications. Academic Press Ltd., London, 1992.

[20] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155:468–498, 1999.

[21] D. Colton and R. Kress. *Inverse Acoustic and Electromagnetic Scattering Theory, 2nd Edition*. Applied Mathematical Sciences. Springer, 1998.

[22] W. Dahmen and R. Schneider. Composite wavelet bases for operator equations. *Math. Comp.*, 68(228):1533–1567, 1999.

[23] M. P. do Carmo. *Differential geometry of curves and surfaces*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1976. Translated from the Portuguese.

[24] T. Duchamp, A. Certain, A. DeRose, and W. Stuetzle. Hierarchical computation of pl harmonic embeddings. Technical report, University of Washington, 1997.

[25] W. D. Elliott and J. A. Board. Fast fourier transform accelerated fast multipole algorithm. *SIAM Journal on Scientific Computing*, 17(2):398–415, 1996.

[26] M. Frigo and S. G. Johnson. FFTW home page. http://www.fftw.org, 2000.

[27] Y. Fu et al. A fast solution for three-dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering*, 42:1215–1229, 1998.

[28] Y. Fu and G. J. Rodin. Fast solution method for three-dimensional Stokesian many-particle problems. *Communications in Numerical Methods in Engineering*, 16:145–149, 2000.

[29] O. Ghattas and X. Li. A variational finite element method for stationary nonlinear fluid-solid interaction. *J. Comput. Phys.*, 121(2):347–356, 1995.

[30] Z. Gimbutas and F. Rokhlin. A generalized fast mulipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing*, 24(3):796–817, 2002.

[31] J. Gómez and H. Power. A multipole direct and indirect BEM for 2D cavity flow at low Reynolds number. *Engineering Analysis with Boundary Elements*, 19:17–31, 1997.

[32] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, Cambridge, MA, 1988.

[33] L. Greengard and J. Huang. A new version of the fast multipole method for screened Coulomb interactions in three dimensions. *Journal of Computational Physics*, 180:642–658, 2002.

[34] L. Greengard, M. C. Kropinski, and A. Mayo. Integral equation methods for Stokes flow and isotropic Elasticity in the plane. *Journal of Computational Physics*, 125:403–414, 1996.

[35] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[36] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, pages 229–269, 1997.

[37] J. A. Gregory and J. M. Hahn. A C^2 polygonal surface patch. *Comput. Aided Geom. Design*, 6(1):69–75, 1989.

[38] C. M. Grimm and J. F. Hughes. Modeling surfaces of arbitrary topology using manifolds. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 359–368, Aug. 1995.

[39] W. Hackbusch. *Integral equations*, volume 120 of *International Series of Numerical Mathematics*. Birkhäuser Verlag, Basel, 1995. Theory and numerical treatment, Translated and revised by the author from the 1989 German original.

[40] B. Hariharan, S. Aluru, and B. Shanker. A scalable parallel fast multipole method for analysis of scattering from perfect electrically conducting surfaces.

In *Proceedings of Supercomputing*, The SCxy Conference series, Baltimore, Maryland, November 2002. ACM/IEEE.

[41] T. Hermann. G^2 interpolation of free form curve networks by biquintic Gregory patches. *Comput. Aided Geom. Design*, 13(9):873–893, 1996. In memory of John Gregory.

[42] T. Y. Hou, J. S. Lowengrub, and M. J. Shelley. Removing the stiffness from interfacial flows with surface tension. *J. Comput. Phys.*, 114(2):312–338, 1994.

[43] Y. Hu and S. L. Johnsson. A data-parallel implementation of $o(n)$ hierarchical N-body methods. In *Proceedings of Supercomputing*, The SCxy Conference series, Pittsburgh, Pennsylvania, November 1996. ACM/IEEE.

[44] S. Kapur and D. E. Long. IES_3: Efficient electrostatic and electromagnetic simulation. *IEEE Computational Science and Engineering*, 5(4):60–67, 1998.

[45] S. Kapur and J. Zhao. A fast method of moments solver for efficient parameter extraction of MCMs. In *Design Automation Conference*, pages 141–146, 1997.

[46] S. Kim and S. J. Karrila. *Microhydrodynamics: Principles and Selected Applications*. Butterworth-Heinemann, 1991.

[47] R. Kress. *Linear Integral Equations*. Applied Mathematical Sciences. Springer, 1999.

[48] X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshes in 3D. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms,* Washington, DC, 2001. ACM.

[49] C. T. Loop and T. D. DeRose. A multisided generalization of bézier surfaces. *ACM Trans. Graph.*, 8(3):204–234, 1989.

[50] J. Makino. Yet another fast multipole method without multipoles–pseudoparticle multipole method. *Journal of Computational Physics*, 151:910–920, 1999.

[51] A. Mammoli and M. Ingber. Parallel multipole BEM simulation of two-dimensional suspension flows. *Engineering Analysis with Boundary Elements*, 24:65–73, 2000.

[52] A. Mayo. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM Journal on Numerical Analysis*, 21(2):285–299, 1984.

[53] S. G. Mikhlin. *Integral equations and their applications to certain problems in mechanics, mathematical physics and technology*. Second revised edition. Translated from the Russian by A. H. Armstrong. A Pergamon Press Book. The Macmillan Co., New York, 1964.

[54] S. G. Mikhlin and S. Prössdorf. *Singular integral operators*. Springer-Verlag, Berlin, 1986. Translated from the German by Albrecht Böttcher and Reinhard Lehmann.

[55] K. Nabors, F. Korsmeyer, F. Leighton, and J. K. White. Preconditioned, adaptive, multipole-accelerated interative methods for three-dimensional first-kind integral equations of potential theory. *SIAM Journal on Scientific and Statistical Computing*, 15:713–735, 1994.

[56] A. Nakano. Parallel multilevel preconditioned conjugate-gradient approach to variable-charge molecular dynamics. *Computer Physics Communications*, 104:59–69, 1997.

[57] A. Nakano et al. Scalable atomistic simulation algorithms for materials research. In *Proceedings of Supercomputing*, The SCxy Conference series, Denver, Colorado, November 2001. ACM/IEEE.

[58] J. C. Navau and N. P. Garcia. Modeling surfaces from meshes of arbitrary topology. *Comput. Aided Geom. Design*, 17(7):643–671, 2000.

[59] J. J. Ottusch, M. A. Stalzer, J. L. Visher, and S. M. Wandzura. Scalable electromagnetic scattering calculations on the SGI Origin 2000. In *Proceedings of Supercomputing*, The SCxy Conference series, Portland, Oregon, November 1999. ACM/IEEE.

[60] J. Peters. Curvature continuous spline surfaces over irregular meshes. *Comput. Aided Geom. Design*, 13(2):101–131, 1996.

[61] J. Peters. Patching Catmull-Clark meshes. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 255–258, July 2000.

[62] J. Peters. C^2 free-form surfaces of degree $(3,5)$. *Comput. Aided Geom. Design*, 19(2):113–126, 2002.

[63] N. Phan-Thien and S. Kim. *Microstructures in elastic media*. The Clarendon Press Oxford University Press, New York, 1994. Principles and computational methods.

[64] N. Phan-Thien, K. Y. Lee, and D. Tullock. Large scale simulation of suspensions with PVM. In *Proceedings of SC97*, The SCxy Conference series, San Jose, CA, November 1997. ACM/IEEE.

[65] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kalé. Namd: Biomolecular simulation on thousands of processors. In *Proceedings of Supercomputing*, The SCxy Conference series, Baltimore, Maryland, November 2002. ACM/IEEE.

[66] J. R. Phillips and J. K. White. A precorrected-FFT method for electorstatic analysis of complicated 3-D structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1059–1072, 1997.

[67] V. Popov and H. Power. An O(N) taylor sereis multipole boundary element method for three-dimensional elasticity problems. *Engineering Analysis with Boundary Elements*, 25:7–18, 2001.

[68] H. Power. The completed double layer integral equation method for two-dimensional Stokes flow. *IMA Journal of Applied Mathematics*, 51:123–145, 1993.

[69] H. Power and L. Wrobel. *Boundary Integral Methods in Fluid Mechanics*. Computational Mechanics Publications, 1995.

[70] C. Pozrikidis. *Boundary Integral and Singularity Methods for Linearized Viscous Flow*. Cambridge University Press, 1992.

[71] H. Prautzsch. Freeform splines. *Comput. Aided Geom. Design*, 14(3):201–206, 1997.

[72] U. Reif. TURBS—topologically unrestricted rational B-splines. *Constr. Approx.*, 14(1):57–77, 1998.

[73] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60:187–207, 1983.

[74] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional nonconvex polyedra. *Discrete and Computational Geometry*, 7(3), 1992.

[75] L. L. Schumaker. *Spline functions: basic theory*. Robert E. Krieger Publishing Co. Inc., Malabar, FL, 1993. Correlated reprint of the 1981 original.

[76] H.-P. Seidel. Polar forms and triangular B-Spline surfaces. In D.-Z. Du and F. Hwang, editors, *Euclidean Geometry and Computers, 2nd Edition*, pages 235–286. World Scientific Publishing Co., 1994.

[77] F. E. Sevilgen and S. Aluru. A unifying data structure for hierarchical methods. In *Proceedings of Supercomputing*, The SCxy Conference series, Portland, Oregon, November 1999. ACM/IEEE.

[78] M. Shelley, T. Hou, and J. Lowengrub. Boundary integral methods for multicomponent fluids and multiphase materials. *Journal of Computational Physics*, 160:302–363, 2001.

[79] J. R. Shewchuk. Sweep algorithms for constructing higher-dimensional constrained Delaunay triangulations. In *Proceedings of the sixteenth annual symposium on Computational geometry*, Kowloon, Hong Kong, 2000. ACM.

[80] J. P. Singh, C. Holt, J. L. Hennessy, and A. Gupta. A parallel adaptive fast multipole method. In *Proceedings of Supercomputing*, The SCxy Conference series, Portland, Oregon, November 1993. ACM/IEEE.

[81] I. H. Sloan. Error analysis of boundary integral methods. In *Acta numerica, 1992*, Acta Numer., pages 287–339. Cambridge Univ. Press, Cambridge, 1992.

[82] B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain decomposition*. Cambridge University Press, Cambridge, 1996. Parallel multilevel methods for elliptic partial differential equations.

[83] J. Song, C. Lu, W. Chew, and S. Lee. Fast Illinois solver code (fisc). *IEEE Antennas Propag. Mag.*, 40:27, 1998.

[84] J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 395–404, Orlando, Florida, July 1998. ACM SIGGRAPH / Addison Wesley. ISBN 0-89791-999-8.

[85] S.-H. Teng. Provably good partitioning and load balancing algorithms for parallel adaptive N-body simulation. *SIAM Journal on Scientific Computing*, 19(2), 1998.

[86] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press Inc., San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.

[87] G. Wahba. *Spline models for observational data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990.

[88] T. J. Wang. A C^2-quintic spline interpolation scheme on triangulation. *Comput. Aided Geom. Design*, 9(5):379–386, 1992.

[89] J. Warren and H. Weimer. *Subdivision Methods for Geometric Design*. Morgan Kaufmann, 2001.

[90] M. S. Warren and J. K. Salmon. Astrophysical N-body simulations using hierarchical tree data structures. In *Proceedings of Supercomputing*, The SCxy Conference series, Minneapolis, Minnesota, November 1992. ACM/IEEE.

[91] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree N-body algorithm. In *Proceedings of Supercomputing*, The SCxy Conference series, Portland, Oregon, November 1993. ACM/IEEE.

[92] W. L. Wendland. On some mathematical aspects of boundary element methods for elliptic problems. In *The mathematics of finite elements and applications, V (Uxbridge, 1984)*, pages 193–227. Academic Press, London, 1985.

[93] N. Yarvin and V. Rokhlin. Generalized gaussian quadratures and singular value decompositions of integral operators. *SIAM Journal on Scientific Computing*, 20(2):699–718, 1998.

[94] N. Yarvin and V. Rokhlin. An improved fast multipole algorithm for potential fields on the line. *SIAM Journal on Numerical Analysis,* pages 629–666, 1999.

[95] K. Yoshida, N. Nishimura, and S. Kobayashi. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *International Journal for Numerical Methods in Engineering*, 50(3):525–547, 2001.

[96] D. Zorin, P. Schröder, A. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animation. SIGGRAPH 2001 Course Notes, 2001.