Summer 8-4-2021

A Fast Method For Computing Volume Potentials In The Galerkin Boundary Element Method In 3D Geometries

Sasan Mohyaddin
Southern Methodist University, smohyaddin@smu.edu

A FAST METHOD FOR COMPUTING VOLUME POTENTIALS IN THE GALERKIN BOUNDARY ELEMENT METHOD IN $3D$ GEOMETRIES

Approved by:

Dr. Johannes Tausch
Department of Mathematics

Dr. Choon Lee
Department of Electrical and Computer Engineering

Dr. Weihua Geng
Department of Mathematics

Dr. Benno Rumpf
Department of Mathematics

A FAST METHOD FOR COMPUTING VOLUME POTENTIALS IN THE GALERKIN BOUNDARY ELEMENT METHOD IN $3D$ GEOMETRIES

A Dissertation Presented to the Graduate Faculty of the

Dedman College of Humanities and Sciences

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Philosophy

with a

Major in Computational and Applied Mathematics

by

Sasan Adil Mohyaddin

B.S., University of Mosul, Iraq
M.S., University of North Texas

August 4, 2021

ACKNOWLEDGMENTS

Primarily, I would like to thank my advisor, Dr.Johannes Tausch for guiding and supporting me over the years. You have set an example of excellence as a researcher, mentor, instructor, and role model. Thank you so much for all the time you have spent with me. I learned a lot from your advice and suggestion. That makes my dream come true.

Also, I would like to thank my thesis committee members for all of their guidance through this process; your discussion, ideas, and feedback have been absolutely invaluable. I am very grateful for all you have done in this committee. Dr. Geng, thank you for being such an inspiring, kind, and positive professor who was there always to support and mentor me. Dr. Rumpf, I don't know how to thank you for helping me to organize my resume and paper work when I was looking for a job, and so many thanks for nonlinear dynamical system. Dr. Lee thank you so much for your time to serve in my committee I am very grateful.

I would also like to thank the Department of Computational and Applied Mathematics of Southern Methodist University in the United State of America and all of my great professors who taught me mathematical simulations, scientific computing for real physical phenomena, algorithm design, ..etc. The first day I came here, it was like I dived in a deep sea, but you all helped and taught me how to swim and survive. Thank you so much for your financial support so that I could complete this work. I will never forget my office M217.

Also, I would like to thank Head of Department Dr. Aceves who supported and encourage me for the whole program duration, he was always there to listen to student challenge and provide great solution during the hard time. Thank you for your support. Of course, I would like to thank Prof. Adriana Aceves with whom I was think I am a professor not only a TA. Thank you for your recommendation and teaching instruction.

Also, I would like to thank the great United State of America where my kids were born and grew up. When I came to this country a couple years ago from a different world, I didn't feel home sick because I felt I am living with my family. But I am sure I will face a hard home sick when I go back home again, and I will miss you every day ”USA”. Thank you for this magnificent democracy.

Finally, I would like to thank financial support that granted from National Science Foundation NSF so that I could be able to focus on my research. Thank you so much for the grant $DMS - 1720431$.

The numerical calculations were performed in the Center for Research Computing at Southern Methodist University SMU. I would like to thank ManeFrame II for providing this height performance computing technology to support this work.

Mohyaddin , Sasan Adil

B.S., University of Mosul, Iraq
M.S., University of North Texas

A Fast Method for Computing Volume Potentials
in the Galerkin Boundary Element Method
in $3D$ Geometries

Advisor: Professor Johannes Tausch
Doctor of Philosophy degree conferred August 4, 2021
Dissertation completed July 22, 2021

We discuss how the Fast Multipole Method (FMM) applied to a boundary concentrated mesh can be used to evaluate volume potentials that arise in the boundary element method. If h is the meshwidth near the boundary, then the algorithm can compute the potential in nearly $\mathcal{O}(h^{-2})$ operations while maintaining an $\mathcal{O}(h^p)$, p is order of expansion, convergence of the error. The effectiveness of the algorithms are demonstrated by computing boundary integral equations of the Poisson equation $-\Delta\phi = f$ in a domain $\Omega \subset \mathbf{R}^3$ with Dirichlet boundary conditions $\phi = g$ on $\Gamma = \partial\Omega$. Our approach is based on the representation of the solution as a combination of the single layer potential and the Newton potential, specifically

$$\phi(x) = \int_\Gamma G(x,y)q(y)\,ds(y) + \int_\Omega G(x,y)f(y)dy,$$

where $G(x,y)$ is the Green's function, and $q(y)$ is surface density. For the discretization of the domain we consider a boundary concentrated subdivision of Ω into tetrahedra. That is, the tetrahedra on the boundary have diameters of size $\mathcal{O}(h)$ and the size of the tetrahedra increases linearly with the distance to the boundary. In this setting there are $\mathcal{O}(h^{-2})$ tetrahedra in Ω and $\mathcal{O}(h^{-2})$ faces that make up the boundary. To our knowledge, this is the first time that an algorithm with $\mathcal{O}(h^{-2})$ has been thoroughly studied in the literature. In the Galerkin method the function q is approximated by piecewise polynomial functions subject to a triangulation of Γ, in our case this triangulation consists of the boundary faces of the

tetrahedra. This results in a linear system $A_h q_h = g_h - b_h$, where

$$A_h(i,j) = \int_\Gamma \int_\Gamma G(x,y)\varphi_i(x)\varphi_j(y)\, ds(y)ds(x),$$

and

$$b_h(i) = \int_\Gamma \int_\Omega G(x,y)\varphi_i(x)f(y)\, dyds(x), \quad g_h(i) = \int_\Gamma \varphi_i(x)g(x)ds(x)\,.$$

Here, φ_i is a basis function of the finite element space on the boundary, and f is a given volume function. When solving the linear system we have to compute matrix vector products with A_h and b_h. For the matrix A_h, standard fast methods, such as the Fast Multipole Method, are available to compute the matrix vector multiplication in nearly optimal $\mathcal{O}(h^{-2})$ operations. The new contribution of this work is to derive a nearly optimal algorithm for constructing the right hand side in nearly $\mathcal{O}(h^{-2})$ operations. Thus, solutions of the Poisson's equation can also be obtained in nearly $\mathcal{O}(h^{-2})$ operations. Here "nearly" means that the complexity estimate may contain additional logarithmic factors of $\mathcal{O}(h^{-1})$.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

To my wife and kido, you have made me stronger, better and more fulfilled than I could have ever imagined. I love you to the moon and back;your encouragement and support of me goes beyond what words can adequately express.

To my father you will live with me and never forget you.....

Chapter 1

Introduction

1.1. General Overview

The Poisson equation is one of the mathematical models that plays an important role in the mathematical simulation for of several physical phenomena for instance, particle - particle interactions inside periodic system, potential field, mass density distributions, and electrostatic field etc. [11].

Finding fast and accurate method for solving the model problem sometime is challenging in particular in complicated three-dimensional geometries. Therefore, accurate, efficient, rigorous, and fast methods for computing the solution of such models are in high demands. Not only the solution, but also its gradient and curl have physical importance [1, 37, 29]. Recently, several fast and accurate methods have been proposed and demonstrated for computing the solution of Poisson's equation in complex and possibly multi-connected geometries, such as, Fast Multipole Method (FMM) after translating the model into second kind boundary integral equation (SKIE) [40], and an Adaptive Fast Multipole Accelerated Poisson solver [1] by decomposing the model into Laplace equation and particular model that does not satisfy given boundary condition of the original model. Also, one of the most powerful method is integral equation techniques which can be used to compute the solution of the model inside complex geometries, and it can lead to accurate result even though the generated dense matrix needs memory allocation which is considered to be one of the disadvantage of the method [38]. However, evaluating Poisson's problem of multi-connected region or domain with singular interfaces(complex geometries) is more challenge, and it is a critical task because of the isolated singularities located on the interface which needs special treatment to maintenance stability and accuracy of the method used [34]. In fact, there are several numerical techniques developed to handle interface problems, for instance, the Immersed Interface Method(IIM) and Matched Interface and Boundary Method(MIB) which are based derivative, whereas the others are integral equation based scheme used to overcome a complex geometries[34, 42, 43]. For instance, the integral scheme used in [1] handled the Poisson's equation by decomposing it into two model problems which are Laplace equation with the Poisson's original data and non-homogeneous that not satisfies the proper boundary conditions in general. In this case, there are a lot of fast and accurate methods available can be used to deal with Laplace equation and compute particular solution (ϕ^{NH}) [21, 19, 10]. In [10, 15] a grid adaptive and FMM method is used to compute the solution for the Poisson's problem on simple geometries, while on the other hand in [16, 25, 26] the Poisson's equation is solved on complex geometries using FMM - accelerated quadrature, but it based on uniform grids. However, T. Askham and A.J. Cerfon in[1] proposed an adaptive FMM - accelerated Poisson solver for 2D complex geometries Ω which generated by embedding the domain Ω in a square domain Ω_B, with the help of potential theoretic, the solution of Poisson decomposed into particular solution which computed on the artificial square domain mentioned above following [14] and solution of Laplace equation.Also, its worth mentioning here that the volume integral in [1] computed by extending (extrapolate f) the source function f such that be smooth beyond the domain Ω using global function extrapolation in the same manner of[34] with different context; however, this may not guarantee smoothness near the original domain Ω. The purpose of our work is to propose a new version of fast Galerkin boundary integral equation method for the potential equation in three dimensional $3D$ geometries. The main improvements in this work over the aforementioned papers are:

– Instead of using a subdivision of the domain into cubes, we use a much more flexible subdivision of Ω into a tetrahedral boundary concentrated subdivision. This eliminates the need to extend the function f beyond Ω. This is a great improvement since it is difficult in general to construct this extension in a stable way.

– We show both analytically and numerically that the asymptotic cost of constructing a volume potential is the same as the cost for a surface potential. Only with this feature the BEM approach for solving PDEs with inhomogeneous right hand sides is competitive with the more standard BEM discretization.

The details about how volume potentials arise in boundary integral equations are presented in the next sections. Also, we will review some existing approaches that used to compute the potentials and comparing them with our proposed method.

1.2. Potential Theoretic Approaches to Poisson's Equation

Boundary integral reformulation of inhomogeneous PDEs give rise boundary integral equations where the right hand side involves a volume integral. To illustrate this, consider the Poisson equation with Dirichlet boundary conditions in a bounded domain $\Omega \subset \mathbf{R}^3$ with boundary surface $\Gamma = \partial\Omega$. For $f \in H^{-1}(\Omega)$ and $g \in H^{\frac{1}{2}}(\Gamma)$ the problem is well posed in $\phi \in H^1(\Omega)$. For more details we refer to the homographs [35, 31]

$$\begin{aligned} -\Delta\phi &= f, \qquad \text{in } \Omega, \\ \phi &= g, \qquad \text{on } \Gamma. \end{aligned} \tag{1.1}$$

A well known approach is to represent the solution ϕ as a combination of a boundary and a volume integral

$$\phi(\mathbf{x}) = \tilde{\mathcal{V}}q(\mathbf{x}) + \tilde{\mathcal{N}}f(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{1.2}$$

where q is an unknown surface density and $\mathcal{V}$ and $\mathcal{N}$ are the single-layer and the Newton potential, defined by

$$\begin{aligned} \tilde{\mathcal{V}}q(\mathbf{x}) &= \int_\Gamma G(\mathbf{x},\mathbf{y})q(\mathbf{y})\, ds_{\mathbf{y}}\,, \\ \tilde{\mathcal{N}}f(\mathbf{x}) &= \int_\Omega G(\mathbf{x},\mathbf{y})f(\mathbf{y})\, d\mathbf{y}. \end{aligned}$$

The kernel $G(\cdot,\cdot)$ is the free space Green's function of the PDE, which in the case of the Poisson equation is

$$G(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi}\frac{1}{|\mathbf{x}-\mathbf{y}|}.$$

As customary, a tilde indicates that the operators are evaluated in the domain, whereas the operator without the tilde indicates the evaluation on the boundary. It follows from the well known jump relations of layer and volume potentials that $\mathcal{V}q = \gamma_0 \tilde{\mathcal{V}} q$ and $\mathcal{N}f = \gamma_0 \tilde{\mathcal{N}} f$, where γ_0 denotes the boundary trace. Taking the trace in (1.2) then leads to a boundary integral equation on Γ where the right hand side involves a volume integral over Ω

$$\mathcal{V}q(\mathbf{x}) = g(\mathbf{x}) - \mathcal{N}f(\mathbf{x}), \qquad \mathbf{x} \in \Gamma. \tag{1.3}$$

This equation is well posed and $q \in H^{-\frac{1}{2}}(\Gamma)$. An alternative to the indirect formulation (1.3) is the direct formulation which is based on the Green's representation formula

$$\mathcal{V}\frac{\partial u}{\partial n} = \left(\frac{1}{2} + \mathcal{K}\right) g + \mathcal{N}f, \qquad \text{on } \Gamma, \tag{1.4}$$

where $\mathcal{K}$ is the double layer potential

$$\mathcal{K}g(\mathbf{x}) = \int_\Gamma \frac{\partial}{\partial n_{\mathbf{y}}} G(\mathbf{x}, \mathbf{y}) g(\mathbf{y}) \, ds_{\mathbf{y}}.$$

and also involves evaluating the Newton potential, see [35, 31]

1.3. Complexity of Homogeneous PDE

If the PDE (1.1) is homogeneous (i.e., $f = 0$), then the Newton potential disappears in (1.3) and (1.4), and the integral equation only involves quantities on the boundary surface. If we use a standard approximation scheme, e.g., the Galerkin method where q is approximated by a piecewise polynomial function on a triangulation of Γ then we obtain a linear system with $\mathcal{O}(h^{-2})$ unknowns.Here h is the meshsize. For BEM discretizations, the matrices are dense, but there is an abundance of methods to nevertheless compute a matrix-vector product in $\mathcal{O}(h^{-2})$ or nearly $\mathcal{O}(h^{-2})$ operations. Here and in the following 'nearly' means that the complexity estimate includes terms that grow logarithmically in h^{-1}. These methods include wavelets [41, 39], $\mathcal{H}$-matrices and the Fast multipole method, see, e.g.,[13, 6, 8, 40].

1.4. Complexity of Inhomogeneous PDE

For non-homogeneous equations the Newton potential appears and requires an efficient algorithm for its evaluation. This has generated considerable interest in the past [2]. The straight forward approach is to uniformly refine the domain Ω into $\mathcal{O}(h^{-3})$ tetrahedra of side length h and to evaluate the potential on $\mathcal{O}(h^{-2})$ boundary patches. Of course, the aforementioned fast evaluation methods can be adopted to accelerate the computations.

However, since $\mathcal{O}(h^{-3})$ sources have to be evaluated on $\mathcal{O}(h^{-2})$ targets the cost for this operation scales at least like $\mathcal{O}(h^{-3})$, hence this operation will asymptotically dominate the overall cost of the numerical solution of (1.4).

There are several papers that have appeared to solve the Poisson equation using volume potentials[9, 27, 1, 36, 17]. The commonly used approach is to embed Ω in a cube and extend the function f into the cube. The subdivision into cubes enables fast numerical schemes that rely on FMM accelerations. While the algorithms described in these papers differ in several aspects, they all rely on a uniform subdivision of the domain Ω in $\mathcal{O}(h^{-3})$ cubical cells, and therefore require $\mathcal{O}(h^{-3})$ time and storage.

1.5. Intro BC Refinement

In this work we will pursue a different approach, that is focused on reducing the complexity of representing the function f. To that end, we consider a boundary concentrated subdivision of Ω such a refinement has been previously used in finite element calculation [28, 5], but it is new in context of volume potential. In a boundary concentrated subdivision the domain Ω is subdivided into tetrahedra of variable site. Tetrahedra near the boundary have diameter proportional to h and diameter of interior tetrahedrons increased linearly with the distance from the boundary surface. Since there are larger tetrahedra in the interior than near the boundary, the number of tetrahedra is $\mathcal{O}(h^{-2})$. We will construct a new version of the FMM algorithm to approximately evaluate the potential of the non-uniform volume triangulation on the uniform boundary triangulation with $\mathcal{O}(h^{-2})$ cost. Here the involved constants depend on the desired accuracy which in turn is controlled by the expansion order of the approximation of the kernel by a Taylor (or multipole) expansion. By adjusting the order to the level of the hierarchical decomposition of Ω, we will show that the error can be bounded by $\mathcal{O}(h^{-p})$ by increasing the complexity by only logarithmically growing factors.

1.6. Application Where Volume Potential Arise

The Newton potential arise in many physical problems. We conclude this chapter by display some of these application in which the Newton potential appears, and it is crucial to have an efficient numerical method to compute such problems. We refer to the following applications:

1. The Lippmann-Schwinger Equation:

   The propagation of time harmonic acoustic waves is described by Helmholtz equation

$$\begin{gathered}\Delta u + \kappa^2 n u = 0, in \quad \mathbf{R^3}, \\ r(\frac{\partial u^s}{\partial r} - i\kappa u^s) \to 0, r = |x| \to \infty,\end{gathered} \tag{1.5}$$

   where n is refractive index and κ is number of waves. The equation (1.5) is equivalent to

$$\Delta u(\mathbf{x}) + \kappa^2 u(\mathbf{x}) = \kappa^2 a(\mathbf{x}) u(\mathbf{x}), \tag{1.6}$$

   when $a := 1 - n$ has compact support.The Green's function of the equation (1.5) is given by

$$G(\mathbf{x}, \mathbf{y}) = \frac{\exp(i\kappa\, |\mathbf{x} - \mathbf{y}|)}{4\pi\, |\mathbf{x} - \mathbf{y}|}.$$

   The solution u describes both the amplitude and the phase of the time harmonic wave. The total field is decomposed into reflected and income field such that

$$u = u^{inc} + u^{ref}.$$

   The income wave u^{inc} satisfies the free space Helmholtz equation (1.5) thus,

$$\Delta u^{ref} + \kappa^2 u^{ref} = \kappa^2 a(\mathbf{x}) u(\mathbf{x}), \quad \mathbf{x} \in \mathbf{R^3}, \tag{1.7}$$

   Using the Green's theorem and the Newton potential, it can be shown that the scattering problem (1.5) can be reformulated into the integral equation of the second kind

$$u(\mathbf{x}) = u^{inc}(\mathbf{x}) - \kappa^2 \int_\Omega G(\mathbf{x}, \mathbf{y}) a(\mathbf{y}) u(\mathbf{y}) d\mathbf{y}. \tag{1.8}$$

This is the Lippmann Schwinger Equation which involves a volume potential. In [22] is shown that this is well posed problem. Numerical methods to solve (1.8) are described in [20, 30]. Also, the numerical method presented in this dissertation provide an alternative approach to compute the volume potential, that has the potential for better scaling and applicability to handle a large class of contrasts $a(\mathbf{x})$.

2. Poisson-Boltzmann Equation:

   Describes the electrostatic potential of ions in a diffuse layer. This is important for modeling semiconductor devices [24] and biological macromolecules [33].Consider, for instance, a MOFSET transistor. The electric potential ϕ satisfies the Poisson's equation within the semiconductor

$$\Delta\phi = \frac{q}{\epsilon_s}(n - p - D), \tag{1.9}$$

   where q = initial electron charge, ϵ_s = the permittivity of silicon, n, p = the mobile electron and hole densities, respectively, and D = the concentration of ionized impurities. Thus, $n = n_i \exp(\frac{\phi}{V_T})$ and $p = n_i \exp(-\frac{\phi}{V_T})$ hence we arrive at the nonlinear the Poisson equation:

$$\Delta\phi - \frac{qn_i}{\epsilon_s}\sinh(\frac{\phi}{V_T}) = -\frac{q}{\epsilon_s}D. \tag{1.10}$$

   If the field strength are small, then the linearization is a sufficiently accurate approximation:

$$\Delta\phi - \frac{qn_i}{\epsilon_s V_T}\phi = -\frac{q}{\epsilon_s}D. \tag{1.11}$$

   This equation is supplemented with Dirichlet Conditions on the contact, and homogeneous Neumann conditions at the rest of the transistor. Again, this equation can be reformulated into IE involved the volume potentials. This proposed numerical method is optimal to compute the emerged potentials efficiency.

Chapter 2

The BEM Homogeneous and Inhomogeneous PDE

In chapter 1, we have introduced the Poisson equation. Also, we exhibited some numerical techniques that used to compute such kind Dirichlet boundary problems. Especially, we showed how the volume potential arise in the boundary integral equation after reformulating the PDE.

This chapter gives a more detailed description of boundary element method, and we will give a comprehensive basic terminology of elementary functional analysis and function spaces. Only in very few cases solutions of a PDE can be computed analytically, which is why it is important to develop and improve numerical techniques for such computing. Moreover, there is no generic numerical method that can be used to compute and analyze all of the problems. Therefore, we need to develop special method for doing specific tasks. For instance, homogeneous problems can be reformulated to boundary integral equations, combined with a fast solver they can achieve $\mathcal{O}(h^{-2})$ complexity. Furthermore, we have seen that for inhomogeneous equations the volume integrals lead to $\mathcal{O}(h^{-3})$ complexity. Thus the boundary integral approach has the same complexity as direct discretizations of the PDE.

The Boundary Element Method is a discretization method of boundary integral equations. The BEM uses elements and nodes on the boundary mesh only. Because of this reduction the method has advantages for solving engineering [4, 3, 7], physical, and natural problems. In addition, it is more accurate compared to the other numerical tools like FEM.

In this chapter we discuss two different ways to formulate a PDE into a boundary integral equations. The direct and indirect method, we then give some functional analysis background and discuss the Galerkin method, which is based on the variational formulated of the BIEs.

2.1. BEM Reformulations of Laplace and Poisson Equation

The boundary element method can be applied efficiently to homogeneous boundary value problems. However, if the BVP is inhomogeneous, the problem must be transformed into a homogeneous model by using the Newton potential. As we mentioned in the Chapter 1 the Newton potential $\tilde{\mathcal{N}}f$ can solve the equation (1.1). But, in general this potential will not satisfy the boundary conditions. Consequently, the Newton Potential is considered special solution for the Poisson equation with which we will decompose the equation (1.1) into two models; homogeneous and inhomogeneous. Then, the solution of the Poisson equation will be combination of solution of both models $\tilde{\mathcal{V}}q + \tilde{\mathcal{N}}f$. In general BEM classified into two formulas which are:

2.1.1. Direct BIE

The Direct BIE is based on Green's representation formula. It states that a solution of the Poisson equation (1.1) can be expressed explicitly in terms of surface potentials of the boundary data and volume potential of the inhomogeneity.

$$\phi(\mathbf{x}) = \int_\Gamma G(\mathbf{x},\mathbf{y})\frac{\partial\phi(\mathbf{y})}{\partial n_\mathbf{y}}ds_\mathbf{y} - \int_\Gamma \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial n_\mathbf{y}}\phi(\mathbf{y})ds_\mathbf{y} + \int_\Omega G(\mathbf{x},\mathbf{y})f(\mathbf{y})d\mathbf{y}, \quad \mathbf{x}\in\Omega. \tag{2.1}$$

Equation (2.1) can be written in the form of layer and volume potentials

$$\phi(\mathbf{x}) = \tilde{\mathcal{V}}\frac{\partial\phi}{\partial n} - \tilde{\mathcal{K}}\phi + \tilde{\mathcal{N}}f(\mathbf{x}), \qquad \mathbf{x}\in\Omega, \tag{2.2}$$

where $\tilde{\mathcal{V}}$,$\tilde{\mathcal{K}}$, and $\tilde{\mathcal{N}}$ are called the single layer, the double layer, and the Newton potential respectively. For $\mathbf{x}\in\Omega$, they are defined of the form

$$\begin{aligned}
\tilde{\mathcal{V}}q(\mathbf{x}) &= \int_\Gamma G(\mathbf{x},\mathbf{y})q(\mathbf{y})\,ds_\mathbf{y},\\
\tilde{\mathcal{K}}\phi(\mathbf{x}) &= \int_\Gamma \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial n_\mathbf{y}}\phi(\mathbf{y})\,ds_\mathbf{y},\\
\tilde{\mathcal{N}}f(\mathbf{x}) &= \int_\Omega G(\mathbf{x},\mathbf{y})f(\mathbf{y})d\mathbf{y},
\end{aligned}$$

the function $q(\mathbf{y})$ is surface density, and it is defined as $q(\mathbf{y}) = \frac{\partial\phi(\mathbf{y})}{\partial n_\mathbf{y}}$ and G is the Green's function:

$$G(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi}\frac{1}{|\mathbf{x}-\mathbf{y}|}. \tag{2.3}$$

Note that for $\mathbf{x} \in \Omega$ the distance of $\mathbf{x}$ and $\mathbf{y}$ is positive and therefore the integrands are smooth functions. If $\mathbf{x} \in \Gamma$, then we write $\mathcal{V}, \mathcal{K}, \mathcal{N}$. In this case the Green's function is of $\mathcal{O}(\frac{1}{r})$ and weakly singular at $r = 0$. For a smooth surface even the double layer is weakly singular because $(\mathbf{x} - \mathbf{y}) \cdot n_{\mathbf{y}} = \mathcal{O}(|\mathbf{x} - \mathbf{y}|^2)$ see [22].
The relation ship of the operators with a tilde to the ones without tilde follows from the well known jump relations of layer potentials and volume potentials. They are given by

$$\begin{aligned}
\lim_{\mathbf{x}\to\Gamma.} \tilde{\mathcal{V}}q(\mathbf{x}) &= \mathcal{V}q(\mathbf{x}), \\
\lim_{\mathbf{x}\to\Gamma.} \tilde{\mathcal{K}}\phi(\mathbf{x}) &= \pm\frac{1}{2}\phi(\mathbf{x}) + \mathcal{K}\phi(\mathbf{x}), \\
\lim_{\mathbf{x}\to\Gamma.} \tilde{\mathcal{N}}f(\mathbf{x}) &= \mathcal{N}f(\mathbf{x}),
\end{aligned}$$

see [22]. By taking traces in equation (2.2) following the boundary integral equation on Γ can be derived

$$\frac{1}{2}\phi(\mathbf{x}) = \mathcal{V}q(\mathbf{x}) - \mathcal{K}\phi(\mathbf{x}) + \mathcal{N}f(\mathbf{x}), \qquad \mathbf{x} \in \Gamma. \tag{2.4}$$

2.1.2. Indirect BIE

Without using the Green's identities, we can use the fundamental solution to construct the boundary integral equation. However, the constructed density function in this case does no relate to the boundary data of the solution. For that reason it is called indirect BIE. Also, one of the advantage of this formula its cost. In fact, it is less expensive than direct BIE since the double layer potential does not appear in the formulation. To that end, we start with a combined single layer potential and volume potential

$$\phi(\mathbf{x}) = \int_\Gamma G(\mathbf{x},\mathbf{y})q(\mathbf{y})ds_{\mathbf{y}} + \int_\Omega G(\mathbf{x},\mathbf{y})f(\mathbf{y})d\mathbf{y}, \quad \mathbf{x} \in \Omega, \tag{2.5}$$

which is equivalent to

$$\phi(\mathbf{x}) = \tilde{\mathcal{V}}q(\mathbf{x}) + \tilde{\mathcal{N}}f(\mathbf{x}), \qquad \forall\mathbf{x} \in \Omega, \tag{2.6}$$

where q is unknown density and $\mathcal{V}q(\mathbf{x})$ and $\mathcal{N}f(\mathbf{x})$ are single layer and Newton potentials, which were defined in the previous section. Because of the properties of layer potentials, we see that for $\mathbf{x} \in \Omega$

$$-\Delta\phi = \Delta\tilde{\mathcal{V}}q(\mathbf{x}) + \Delta\tilde{\mathcal{N}}f(\mathbf{x}) = 0 + f = f.$$

.

Applying the jump relations of the layer potentials in the integral equation (2.6) leads to a boundary integral equation on Γ

$$g(\mathbf{x}) = \mathcal{V}q(\mathbf{x}) + \mathcal{N}f(\mathbf{x}), \qquad \mathbf{x} \in \Gamma, \tag{2.7}$$

where $g(\mathbf{x})$ is the Dirichlet boundary condition. If the source function $f = 0$, the Newton potential will vanish. Therefore, $\mathcal{N}f(\mathbf{x}) = 0$ in the previous sections, and the equations (2.4) and (2.7) are direct and indirect formula for Laplace equation.

2.2. Well Posedness of IE in Proper Function Spaces

We present some fundamental spaces and function analysis that relevant to the BEM, and we need them later when we describe the proposed new Fast Algorithm. More details can be found in e.g., [31]

- *Continuous function spaces*

  Let $K \in \mathbf{N}_0$ and let $\Omega \subset \mathbf{R}^d$ be domain, the space $\mathbf{C}^K(\Omega)$ is called the K time differentiable continuous function on Ω, and $\partial^\alpha f$ can be extended as a continuous function on $\bar{\Omega}$ $\forall\ 0 \leq |\alpha| \leq K$. The space $\mathbf{C}^\infty(\bar{\Omega}) := \cap \mathbf{C}^k(\bar{\Omega}) \quad \forall K \in \mathbf{N_0}$ is the space of all infinitely differentiable functions.

- *Space of* $\mathbf{L}^2(\Omega)$

  Is a Hilbert space with inner product $\langle \phi, \psi \rangle_{0,\Omega} = \langle \phi, \psi \rangle_{\mathbf{L}^2(\Omega)} = \int_\Omega \phi(\mathbf{x})\bar{\psi(\mathbf{x})}d\mathbf{x}$

- *Sobolev Space*

  Let $\Omega \subset \mathbf{R}^d$ be a bounded domain. for all $l = 0, 1, 2, ...$ the Sobolev space $H^l(\Omega)$ is given by

  $$H^l(\Omega) := \{\phi \in \mathbf{L}^2(\Omega) : \textit{and} \quad \partial^\alpha \phi \in \mathbf{L}^2(\Omega) \quad \forall |\alpha| \leq l\}.$$

Remark: If the functions ϕ and ψ in $H^l(\Omega)$ then the inner product is defined as

$$\langle \phi, \psi \rangle_{\mathbf{H}^\ell(\Omega)} = \sum_{|\alpha| \le l} \langle \partial^\alpha \phi, \partial^\alpha \psi \rangle = \sum_{|\alpha| \le l} \int_\Omega \partial^\alpha \phi \partial^\alpha \psi dx.$$

For $\ell \ge 0$ the space $H^{-\ell}(\Omega)$ is the dual of $H^\ell(\Omega)$ with respect to the $L_2(\Omega)$ duality pairing. For boundary integral equations, we need Sobolev spaces of boundaries $\Gamma := \partial\Omega$ of domains. Also, to be Lipschitz domain, we need $l \le \mathbf{k}$ for space of $\mathbf{k}$ times differentiable and continuous.

For our purpose, we need $L^2(\Gamma)$ the space of square integrable functions on Γ which has the inner product

$$\langle \phi, \psi \rangle_{L^2(\Gamma)} = \int_\Gamma \phi(x)\psi(\ell) ds(\ell).$$

In addition, we need the space $H^{\frac{1}{2}}(\Gamma)$ which is the trace space of $H^1(\Omega)$ and the space $H^{-\frac{1}{2}}(\Gamma)$ which is the dual space of $H^{\frac{1}{2}}(\Gamma)$ with respect to the $L_2(\Gamma)$ duality pairing.

2.3. Variational Formulation and Galerkin Discretization

The Galerkin approximation method is one of the powerful discretization for the boundary integral equations (2.4) and (2.7) because Galerkin Method is based on variational forms of the integral equation. Find $\frac{\partial \phi}{\partial n} \in H^{\frac{1}{2}}$

$$\left\langle \phi, \mathcal{V} \frac{\partial \phi}{\partial n} \right\rangle = \int_\Gamma \int_\Gamma G(\mathbf{x}, \mathbf{y}) \phi(\mathbf{x}) \frac{\partial \phi(\mathbf{y})}{\partial n_\mathbf{y}} ds_\mathbf{x} ds_\mathbf{y}, \qquad \forall \phi \in H^{\frac{1}{2}},$$

and

$$\langle \phi, \mathcal{V} q \rangle = \int_\Gamma \int_\Gamma G(\mathbf{x}, \mathbf{y}) \phi(\mathbf{x}) q(\mathbf{y}) ds_\mathbf{x} ds_\mathbf{y}, \qquad \forall \phi \in H^{\frac{1}{2}}.$$

There it enables a complete error analysis in Sobolev spaces described above. Alternative approaches like collocation and Nyström methods can only be analyzed in more special settings [22]. The Galerkin method is considered the most stable approximation tools among all other available discretization methods [7]. We briefly describe the Galerkin discretization of these equations. It is well known that for $f \in H^{-1}(\Omega)$ and the Dirichlet condition $g \in H^{\frac{1}{2}}(\Omega)$ the integral equation is well posed in $q \in H^{-\frac{1}{2}}(\Omega)$. For simplicity of exposition we assume that $\Omega \subset \mathbb{R}^3$ is a polyhedron $\Omega \subset \mathbb{R}^3$ that is subdivided into a small number of coarsest level tetrahedra ω. In chapter 5 we will give more detail how the volume triangulation is obtained. Restricting this triangulation on the boundary surface results in a triangulation of Γ, where we assume that the triangulation is quasi-uniform with meshwidth h. The finite element space S_h consists of piecewise polynomial functions on each triangle γ of the boundary triangulation. The basis functions of S_h are denoted by ϕ_k and are the usual box-functions, hat-functions, and so on. The Galerkin discretization of (2.7) reads: find $q_h \in S_h$ such that

$$\langle \phi_k, \mathcal{V} q_h \rangle = \langle \phi_k, g \rangle - \langle \phi_k, \mathcal{N} f \rangle ,$$

where $\langle \cdot, \cdot \rangle$ is the L_2-inner product on Γ and k runs through all basis functions. Since q_h is a linear combination of all ϕ_k's this leads to a linear system

$$V \mathbf{q} = \mathbf{g} - \mathbf{b}, \tag{2.8}$$

where $\mathbf{q}$ is the vector of coefficients in the expansion of q_h in the basis, and the coefficients of $\mathbf{g}$, $\mathcal{V}$, and $\mathbf{b}$ are given by $g_k = \langle \phi_k, g \rangle$, $\mathcal{V}_{k\ell} = \langle \phi_k, \mathcal{V} \phi_\ell \rangle$, and $b_k = \langle \phi_k, \mathcal{N} f \rangle$ respectively. Also, the Galerkin discretization of the equation (2.4) is of the form

$$\langle \phi_k, \mathcal{V} q_h \rangle = \langle \phi_k, \psi \rangle - \langle \phi_k, \mathcal{N} f \rangle , \qquad \psi = (\frac{1}{2} + \mathcal{K}) g.$$

This leads to the same linear system (2.8) with different coefficients of g where $g_k = \langle \phi_k, \psi \rangle$.

Chapter 3

Numerical Quadrature for Singular Face-Face and Face-Tetrahedron Integrals

To construct the linear system (2.8) using direct or indirect BIEs, we must compute the layer and volume potentials after applying Galerkin discretization. the coefficients $\langle \phi_k, \mathcal{V}\phi_\ell \rangle$, and the volume potentials $\langle \phi_k, \mathcal{N}f \rangle$ must be computed.There are some challenges associated with the construction of the matrix V, and the vector b through the integrals (5.4) and (5.3). First, accurate quadrature need to be used efficiency. Second, since the matrix V is dense, the construction and memory allocation cost is $\mathcal{O}(h^{-4})$. Finally, the integrand function is not smooth when both $\mathbf{x}$ and $\mathbf{y}$ intersect on the boundary surface, and this intersection could be in one, two, or three vertices. The numerical quadrature of layer potentials is defined on a pair of reference panels $p_x \times p_y$, while the quadrature of volume potentials is defined on panel and tetrahedron $p_x \times \tau_y$. The integrand function of both the layer and volume are not smooth when a panel p_x intersect with a panel p_y or a tetrahedron τ_y. For the use of two panels there are transformations that map the integral over $p_x \times p_y$ to an integral over the four dimensional unit cube $[0,1]^4$, see [31]. These construction is reviewed in section 3.1. This will motivate our new transformation from $p_x \times \tau_y$ to $[0,1]^5$. This is discussed in section 3.2. In all cases, the transformed integrals are smooth and can be calculated by tensor-product Gauss quadrature.

3.1. Spatial Coordinates

For simplicity we combine layer potentials in one compact formula like introduced in [38].

$$\mathcal{K}f(\mathbf{x}) = \int_\Gamma \frac{\partial^\kappa}{\partial n_y^\kappa} \mathbf{G}(|\mathbf{x} - \mathbf{y}|) f(\mathbf{y}) d\Gamma_{\mathbf{y}}. \tag{3.1}$$

If $\kappa = 0$, $\mathcal{K}$ is single - layer potential, and it will be double layer potential if $\kappa = 1$. For the Galerkin discretization, the equation (3.1) is multiplied by hat or box functions ϕ_i and then integrated over the boundary Γ. Also, this discretization of the equation (3.1) introduces a dense matrix with entries

$$K_{i,j} = \langle \phi_i, \mathcal{K}\phi_j \rangle, \quad \forall \quad 1 \leq i, j \leq N.$$

To compute the dense matrix $\mathcal{K}$ entries, we must do integration over a reference pair of panels. In this case these panels could intersect in one, two, or there vertices. In another words, we may have one common vertex, one common edge, or identical face(three common edges or vertices)[31]. If this interaction happens between faces on the surface, we will have non-smooth integrand function. In this case, we need to employ or call spatial transformations to eliminate the singularities. Below we summarize this technique from Sauter [23, 31]. Later, we will modify this approach to remove the singularities when we have face - tetrahedron interactions.

3.1.1. Surface Mesh Setting

Consider the boundary surface mesh of a domain in $\mathbf{R}^3$ which is curved triangular of the surface Γ, if $\mathbf{P}$ represents as a panel on the surface, $\Gamma = \bigcup_i \mathbf{P}_i$ where each $\mathbf{P}_i$ is a parametric image of the unit triangle,

$$\sigma^{(2)} = \left\{ \hat{\mathbf{x}} \in \mathbb{R}^2 \,:\, 0 \leq \hat{x}_2 \leq \hat{x}_1 \leq 1 \right\},$$

i.e.,

$$\mathbf{P}_i = \left\{ \mathbf{P}_i(\hat{\mathbf{x}}),\ \hat{\mathbf{x}} \in \sigma^{(2)}, \forall i \right\}.$$

The vertices of surface triangulation elements are the parametric images of the points in 2$\mathbf{D}$ $(0,0)$, $(0,1)$, and $(1,1)$ and the parametric images of the line segments $(\xi, 0)$, $(1, \xi)$ and (ξ, ξ), $\xi \in [0,1]$ are the edges. In this work we always consider the parametric image of the origin $(0,0) \in \sigma^{(2)}$ is the intersection point between two panels $\mathbf{P}_i$ and $\mathbf{P}_j$ if they intersect in one point. Otherwise, the affine transformation must be applied to rotate the triangle(panel) such that the common vertex for both panels relocated at the origin[23]. Also, for the parameterization and triangulation we consider the following:

- The intersection of two different triangular panels $\mathbf{P}_i \cap \mathbf{P}_j$ is either empty(in this case we will have smooth integrand function, and we don't need singularities removal transformations), a common vertex or a common edge.
- If two panels intersect in a common vertex or edge, then they satisfy the cone condition. For more info about this condition, see [31, 23].
- If $\mathbf{P}_i$ intersects with $\mathbf{P}_j$ in a common edge, then $\mathbf{P}_i(\xi, 0) = \mathbf{P}_j(\xi, 0)$, $\xi \in [0, 1]$.

Consider the surface panel $\mathbf{P}_i$ with vertices $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{v}_2$, then a parametrization for the panel is defined as the following:

$$\mathbf{P}_i(\hat{\mathbf{x}}) = \mathbf{v}_0 + \mathbf{a}_1\hat{\mathbf{x}}_1 + \mathbf{a}_2\hat{\mathbf{x}}_2,$$

where $\mathbf{a}_1 = \mathbf{v}_1 - \mathbf{v}_0$ and $\mathbf{a}_2 = \mathbf{v}_2 - \mathbf{v}_1$ are edges of the panel $\mathbf{P}_i$ and $\hat{\mathbf{x}} \in \sigma^{(2)}$.

3.1.2. Panels or Patches Interaction

We can observe that if panels $\mathbf{P}_i$ intersects with $\mathbf{P}_j$, it could be a common vertex, a common edge, or self identical panel. In this cases, we can make the following observations about patches interactions:

- Self Panel, $\mathbf{P}_i = \mathbf{P}_j$ if both panels are parameterized as above then in this case we will have $\mathbf{P}_i(\hat{\mathbf{x}}) - \mathbf{P}_j(\hat{\mathbf{y}}) = \mathbf{a}_1(\hat{\mathbf{x}}_1 - \hat{\mathbf{y}}_1) + \mathbf{a}_2(\hat{\mathbf{x}}_2 - \hat{\mathbf{y}}_2)$
- One common edge, then the distance between the panels above will be $\mathbf{P}_i(\hat{\mathbf{x}}) - \mathbf{P}_j(\hat{\mathbf{y}}) = \mathbf{a}_1(\hat{\mathbf{x}}_1 - \hat{\mathbf{y}}_1) + \mathbf{a}_2(\hat{\mathbf{x}}_2) + \mathbf{a}_3(\hat{\mathbf{y}}_2) + \mathbf{v}_0$
- One common vertex, then $\mathbf{P}_i(\hat{\mathbf{x}}) - \mathbf{P}_j(\hat{\mathbf{y}}) = \mathbf{a}_1(\hat{\mathbf{x}}_1) + \mathbf{b}_1(\hat{\mathbf{y}}_1) + \mathbf{a}_2(\hat{\mathbf{x}}_2) + \mathbf{b}_2(\hat{\mathbf{y}}_2)$

3.1.3. The Transformations

To approximate the entries of the matrix $\mathcal{K}$, one needs to compute double numerical integration over surface panels, and the singular nodes must be removed from integrand function. The integrand functions will be non smooth when a pair of patches $\mathbf{P}_i$ and $\mathbf{P}_j$ intersect. This intersection, could be identical panels, panels with common edges, or panels with one common vertex. Otherwise, integrand functions is smooth if there is a positive distance between two panels(no common vertex). Therefore, an elegant transformations have been employed to treat the singularity, and the transformations classified as the following the details here are retrieved from [32, 23]:

3.1.3.1. Identical or Self Panels

since $\mathbf{P}_i(\hat{\mathbf{x}}) = \mathbf{P}_j(\hat{\mathbf{y}})$ are located on the same patch $\forall\, \hat{\mathbf{x}}, \hat{\mathbf{y}} \in \sigma^{(2)}$ in this case, the integrand is singular, and the following transformation can be employed:

$$\begin{aligned}
z_1 &= \hat{\mathbf{y}}_1 - \hat{\mathbf{x}}_1, & \hat{\mathbf{x}}_1 &= w_1,\\
z_2 &= \hat{\mathbf{y}}_2 - \hat{\mathbf{x}}_2, & \hat{\mathbf{x}}_2 &= w_2,\\
w_1 &= \hat{\mathbf{x}}_1, & \hat{\mathbf{y}}_1 &= z_1 + w_1,\\
w_2 &= \hat{\mathbf{x}}_2, & \hat{\mathbf{y}}_2 &= z_2 + w_2,
\end{aligned} \tag{3.2}$$

where

$$z = (z_1, z_2) \ \in \ \bigcup_{k=1}^{6} D_k.$$

See Figure 3.1 Buy using Duffy trick, each one of D_k $\forall k \in \{1, ..., 6\}$ can be mapped on the unit square. Also, the parameter $w = (w_1, w_2)$ depends on the variable z. Thus,

$$z = \xi R_k(\eta_1),\ z \in D_k, \quad (\xi, \eta_1) \in [0,1]^2,$$

$$w = v_k + (1-\xi)\begin{bmatrix} \eta_2 \\ \eta_2\eta_3 \end{bmatrix}, \quad (\eta_2, \eta_3) \in [0,1]^2.$$

Then, $\forall k$ the following subdomain can be obtained, and it's good to know that v_k is the

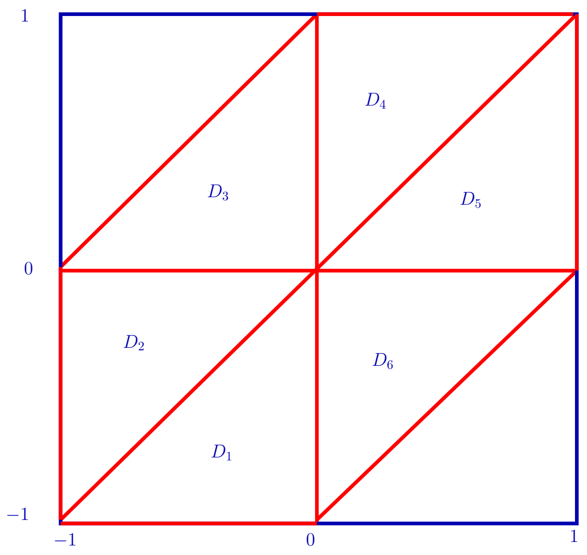


Figure 3.1: Domain of z - integration for the identical panels interactions during the numerical integration process for non-smooth functions.

lower left corner.

$$\begin{aligned}
R_1 &= \begin{bmatrix} -\eta_1 \\ -1 \end{bmatrix}, & v_1 &= \begin{bmatrix} \xi \\ \xi \end{bmatrix}, \\
R_2 &= \begin{bmatrix} -1 \\ -\eta_1 \end{bmatrix}, & v_2 &= \begin{bmatrix} \xi \\ \xi\eta_1 \end{bmatrix}, \\
R_3 &= \begin{bmatrix} -\eta_1 \\ 1-\eta_1 \end{bmatrix}, & v_3 &= \begin{bmatrix} \xi \\ 0 \end{bmatrix}, \\
R_4 &= \begin{bmatrix} 1-\eta_1 \\ -\eta_1 \end{bmatrix}, & v_4 &= \begin{bmatrix} \xi\eta_1 \\ \xi\eta_1 \end{bmatrix}, \\
R_5 &= \begin{bmatrix} 1 \\ \eta_1 \end{bmatrix}, & v_5 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\
R_6 &= \begin{bmatrix} \eta_1 \\ 1 \end{bmatrix}, & v_6 &= \begin{bmatrix} \xi(1-\eta_1) \\ 0 \end{bmatrix},
\end{aligned} \tag{3.3}$$

with Jacobian $J_k = \xi(1-\xi)^2\eta_2, \qquad \forall k \in \{1, ..., 6\}$.

3.1.3.2. Common Edge

The panels $\mathbf{P}_i$ and $\mathbf{P}_j$ have only common edges. For that, the transformations must be defined as the following in order to smooth the integrand function when both $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ on the same patches. where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}} \in \sigma^{(2)}$

$$\begin{aligned} z_1 &= \hat{\mathbf{y}}_1 - \hat{\mathbf{x}}_1, & \hat{\mathbf{x}}_1 &= w_1, \\ z_2 &= \hat{\mathbf{y}}_2, & \hat{\mathbf{x}}_2 &= z_3, \\ z_3 &= \hat{\mathbf{x}}_2, & \hat{\mathbf{y}}_1 &= z_1 + w_1, \\ w_1 &= \hat{\mathbf{x}}_1, & \hat{\mathbf{y}}_2 &= z_2, \end{aligned} \tag{3.4}$$

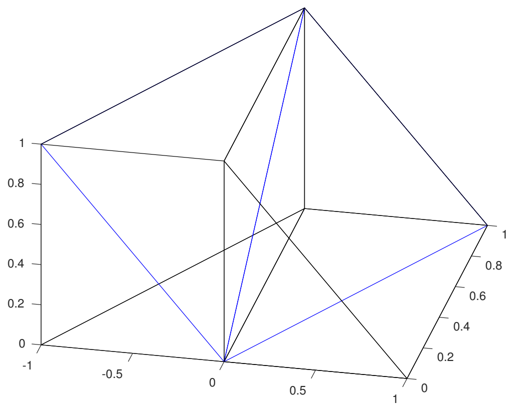


Figure 3.2: Domain of z - integration for the panels- panels interactions with common during the numerical integration process for non-smooth functions.

the variable z located inside the union of four polyhedra at the origin see Figure 3.2 , and w_1 must be in the intersection such that

$$w_1 \in [-z_1, 1 - z_1] \cap [0, 1],$$

$$z = (z_1, z_2, z_3) \ \in \ \bigcup_{k=1}^{4} D_k,$$

each of the element D_k in the given domain can be mapped into the $[0,1]^3$ by using Duffy trick such that:

$$z = \xi R_k(\eta_1, \eta_2), \quad z \in D_k, \quad (\xi, \eta_1, \eta_2) \in [0,1]^3,$$

where $R_k \quad \forall k$, w_1, and the Jcobian $J_k, \forall k \in \{1, .., 4\}$ of the transformations for each case defined as the following:

$$\begin{aligned}
R_1 &= \begin{bmatrix} -\eta_1 \\ 1-\eta_1 \\ \eta_2 \end{bmatrix}, & w_1 &= \xi + (1-\xi)\eta_3, & J_1 &= (1-\xi)\xi^2, \\
R_2 &= \begin{bmatrix} -\eta_1\eta_2 \\ (1-\eta_1)\eta_2 \\ 1 \end{bmatrix}, & w_1 &= \xi + (1-\xi)\eta_3, & J_2 &= (1-\xi)\xi^2\eta_2, \\
R_3 &= \begin{bmatrix} 1-\eta_1 \\ \eta_2 \\ \eta_1 \end{bmatrix}, & w_1 &= (1-\eta_1)\xi + (1-\xi)\eta_3, & J_3 &= (1-\xi)\xi^2, \\
R_4 &= \begin{bmatrix} \eta_1\eta_2 \\ 1 \\ (1-\eta_1)\eta_2 \end{bmatrix}, & w_1 &= (1-\eta_1\eta_2)\xi + (1-\xi)\eta_3, & J_4 &= (1-\xi)\xi^2\eta_2.
\end{aligned} \tag{3.5}$$

3.1.3.3. Common Vertex

when there is only one common vertex between the shape elements of two panels $\mathbf{P}_i$ and $\mathbf{P}_j$, there will be only two transformation to get rid from singularities. In this case simply the parametrization can be written as the following:

$$\begin{aligned}
z_1 &= \hat{\mathbf{x}}_1, \\
z_2 &= \hat{\mathbf{x}}_2, \\
z_3 &= \hat{\mathbf{y}}_2, \\
z_4 &= \hat{\mathbf{y}}_1,
\end{aligned} \tag{3.6}$$

where the variable z domain is union of the zone D_1 and D_2 thus,

$$z \in \bigcup_{k=1}^{2} D_k,$$

$$\begin{aligned} D_1 &= \left\{ z : \begin{array}{r} 0 \le z_2 \le z_1 \le 1 \\ 0 \le z_4 \le z_3 \le 1 \\ z_3 \le z_1 \end{array} \right\}, \\ D_2 &= \left\{ z : \begin{array}{r} 0 \le z_2 \le z_1 \le 1 \\ 0 \le z_4 \le z_3 \le 1 \\ z_1 \le z_3 \end{array} \right\}. \end{aligned} \tag{3.7}$$

Thus,

$$z = \xi R_k(\eta_1, \eta_2, \eta_3), \quad z \in D_k, \quad \forall \quad \xi, \eta_1, \eta_2 \in [0,1]^3$$

where

$$R_1 = \begin{bmatrix} 1 \\ \eta_1 \\ \eta_2 \\ \eta_2\eta_3 \end{bmatrix}, \qquad R_2 = \begin{bmatrix} \eta_2 \\ \eta_2\eta_3 \\ 1 \\ \eta_1 \end{bmatrix}, \tag{3.8}$$

with Jacobian for both is $J_k = \xi^2 \eta_2, \qquad \forall k \in \{1, 2\}$.

3.2. Computing Newton Potential

Consider the Galerkin discretization of the Newton or volume potentials that arises in the direct and indirect BIEs (2.4) and (2.7) $b_i = \langle \phi_i, \mathcal{N} f \rangle$ such that

$$b_{\gamma,i} = \int_\gamma \phi_i(\mathbf{x}) \int_\Omega G(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y} ds_{\mathbf{x}}. \tag{3.9}$$

To construct the vector b_i, we must compute integral through equation (3.9) in which the exterior integral is running over the boundary surface element, while the interior is domain integral. In this setting, the boundary concentrated subdivision of the Ω generates $\mathcal{O}(h^{-2})$

tetrahedra of domain reference elements and $\mathcal{O}(h^{-2})$ face that makes up the boundary surface reference elements. The integrand function in the is not smooth, and it has singularities when the surface panels intersect with the domain elements "tetrahedra". This singularities need special treatment so that it can be removed. For this purpose, we have improved Spatial Transformation to smooth the integrand function. Also, if a panel $\mathbf{p}$ on the boundary surface element Γ_P intersects with a tetrahedron τ in the Ω_τ in one corner, it is considered without loss of generality the corner vertex is parametric image of the origin $(0,0) \in \sigma^{(2)}$ ($\sigma^{(2)}$ defined in the chapter 2). If not, the affine transformations will be used to rotate the patches in the origin, and the same procedure can be follow for common edges between $\mathbf{p}$ and τ. Next, we will describe affine and spatial transformation, then both will be combined into the equation (3.9) in order to compute the volume potential.

3.2.1. Spatial Transformation and Singularities removal

Considered the boundary surface $\Gamma = \partial\Omega \subset \mathbf{R^3}$ subdivided into a triangulation, such that

$$\Gamma = \bigcup_{p=1}^{N} \Gamma_p,$$

where each element in the Γ_p is image of unit triangle i.e

$$\Gamma_p = \{X_p(\hat{\mathbf{x}}), \quad \hat{\mathbf{x}} \in \sigma^{(2)}\},$$

where $\sigma^{(2)}$ defined as the following:

$$\sigma^{(2)} = \{\hat{\mathbf{x}} \in \mathbf{R^2} : \quad 0 \le \hat{\mathbf{x}}_2 \le \hat{\mathbf{x}}_1 \le 1\},$$

the parametric images of the points $(0,0)$, $(0,1)$, and $(1,1)$ are the triangulation's vertices, while the line segments $(\xi,0)$, $(1,\xi)$, and (ξ,ξ) are the edges. while $\mathbf{y}$ can be parameterized as the following $\hat{\mathbf{y}} \in \sigma^{(3)}$ where

$$\Omega = \bigcup_{q=1}^{N} \tau_q,$$

each element in the τ_q is tetrahedron i.e

$$\tau_q = \{Y_q(\hat{\mathbf{y}}), \quad \hat{\mathbf{y}} \in \sigma^{(3)}\},$$

and $\sigma^{(3)}$ defined as the following:

$$\sigma^{(3)} = \{\hat{\mathbf{y}} \in \mathbf{R^3} : \quad 0 \le \hat{\mathbf{y}}_3 \le \hat{\mathbf{y}}_2 \le \hat{\mathbf{y}}_1 \le 1\}.$$

If a panel p intersects with a tetrahedron q, this intersection could be one vertex, one edge, or identical on the tetrahedron face.

3.2.1.1. Three Common Edges

The panels p intersects or identical on one faces of a tetrahedron q.This means both have the same edges, vertices, and the elements or particles $\hat{x} = \hat{y}$ in the integrand function. For $\hat{x} \in \sigma^{(2)}$, while $\hat{y} \in \sigma^{(3)}$.Since $\hat{x}$ and $\hat{y}$ are located on the same patch, the integrand is singular, and the following transformation has employed:

$$\begin{aligned}
\mathbf{z}_1 &= \hat{\mathbf{y}}_1 - \hat{\mathbf{x}}_1, & \hat{\mathbf{x}}_1 &= \mathbf{w}_1, \\
\mathbf{z}_2 &= \hat{\mathbf{y}}_2 - \hat{\mathbf{x}}_2, & \hat{\mathbf{x}}_2 &= \mathbf{w}_2, \\
\mathbf{w}_1 &= \hat{\mathbf{x}}_1, & \hat{\mathbf{y}}_1 &= \mathbf{z}_1 + \mathbf{w}_1, \\
\mathbf{w}_2 &= \hat{\mathbf{x}}_2, & \hat{\mathbf{y}}_2 &= \mathbf{z}_2 + \mathbf{w}_2 \ ; \ \hat{\mathbf{y}}_3 \le \hat{\mathbf{y}}_2,
\end{aligned} \tag{3.10}$$

where

$$\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2) \ \in \ \bigcup_{k=1}^{6} D_k.$$

See Figure 3.1. Buy using Duffy trick, each one of D_k $\forall k \in \{1, ..., 6\}$ can be mapped on the unit square. Also, the parameter $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2)$ depends on the variable $\mathbf{z}$. Thus,

$$\mathbf{z} = \xi R_k(\eta_1), \ \mathbf{z} \in D_k, \quad (\xi, \eta_1) \in [0,1]^2,$$

$$\mathbf{w} = \upsilon_k + (1-\xi) \begin{bmatrix} \eta_2 \\ \eta_2\eta_3 \end{bmatrix}, \quad (\eta_2, \eta_3) \in [0,1]^2.$$

Then, $\forall k$ the following subdomain can be obtained, and it's good to know that υ_k is the lower left corner.

$$\begin{aligned}
R_1 &= \begin{bmatrix} -\eta_1 \\ -1 \end{bmatrix}, & v_1 &= \begin{bmatrix} \xi \\ \xi \end{bmatrix}, \\
R_2 &= \begin{bmatrix} -1 \\ -\eta_1 \end{bmatrix}, & v_2 &= \begin{bmatrix} \xi \\ \xi\eta_1 \end{bmatrix}, \\
R_3 &= \begin{bmatrix} -\eta_1 \\ 1-\eta_1 \end{bmatrix}, & v_3 &= \begin{bmatrix} \xi \\ 0 \end{bmatrix}, \\
R_4 &= \begin{bmatrix} 1-\eta_1 \\ -\eta_1 \end{bmatrix}, & v_4 &= \begin{bmatrix} \xi\eta_1 \\ \xi\eta_1 \end{bmatrix}, \\
R_5 &= \begin{bmatrix} 1 \\ \eta_1 \end{bmatrix}, & v_5 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\
R_6 &= \begin{bmatrix} \eta_1 \\ 1 \end{bmatrix}, & v_6 &= \begin{bmatrix} \xi(1-\eta_1) \\ 0 \end{bmatrix},
\end{aligned} \tag{3.11}$$

with Jacobian $J_k = \xi(1-\xi)^2\eta_2 \qquad \forall k \in \{1, ..., 6\}$.

3.2.1.2. One Common Edge

The panels p and a face of a tetrahedron q have only one common edges. For that, the transformations must be defined as the following in order to smooth the integrand function when both $\hat{x}$ and $\hat{y}$ on the same patches. where $\hat{x} \in \sigma^{(2)} \quad and \quad \hat{y} \in \sigma^{(3)}$

$$\begin{aligned}
\mathbf{z}_1 &= \hat{\mathbf{y}}_1 - \hat{\mathbf{x}}_1, & \hat{\mathbf{x}}_1 &= \mathbf{w}_1, \\
\mathbf{z}_2 &= \hat{\mathbf{y}}_2, & \hat{\mathbf{x}}_2 &= \mathbf{z}_3, \\
\mathbf{z}_3 &= \hat{\mathbf{x}}_2, & \hat{\mathbf{y}}_1 &= \mathbf{z}_1 + \mathbf{w}_1, \\
\mathbf{w}_1 &= \hat{\mathbf{x}}_1, & \hat{\mathbf{y}}_2 &= \mathbf{z}_2, \ ; \ \hat{\mathbf{y}}_3 \leq \hat{\mathbf{y}}_2,
\end{aligned} \tag{3.12}$$

the variable $\mathbf{z}$ located inside the union of four polyhedron at the origin see Figure 3.2, and $\mathbf{w}_1$ must be in the intersection such that

$$\mathbf{w}_1 \in [-\mathbf{z}_1, 1 - \mathbf{z}_1] \cap [0, 1],$$

$$\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) \ \in \ \bigcup_{k=1}^{4} D_k,$$

each of the element D_k in the given domain can be mapped into the $[0,1]^3$ by using Duffy trick such that:

$$\mathbf{z} = \xi R_k(\eta_1, \eta_2), \quad \mathbf{z} \in D_k, \quad (\xi, \eta_1, \eta_2) \in [0,1]^3,$$

where $R_k \quad \forall k$, $\mathbf{w}_1$, and the Jcobian $J_k, \forall k \in \{1, .., 4\}$of the transformations for each case defined as the following:

$$\begin{aligned}
R_1 &= \begin{bmatrix} -\eta_1 \\ 1-\eta_1 \\ \eta_2 \end{bmatrix}, & \mathbf{w}_1 &= \xi + (1-\xi)\eta_3, & J_1 &= (1-\xi)\xi^2, \\
R_2 &= \begin{bmatrix} -\eta_1\eta_2 \\ (1-\eta_1)\eta_2 \\ 1 \end{bmatrix}, & \mathbf{w}_1 &= \xi + (1-\xi)\eta_3, & J_2 &= (1-\xi)\xi^2\eta_2, \\
R_3 &= \begin{bmatrix} 1-\eta_1 \\ \eta_2 \\ \eta_1 \end{bmatrix}, & \mathbf{w}_1 &= (1-\eta_1)\xi + (1-\xi)\eta_3, & J_3 &= (1-\xi)\xi^2, \\
R_4 &= \begin{bmatrix} \eta_1\eta_2 \\ 1 \\ (1-\eta_1)\eta_2 \end{bmatrix}, & \mathbf{w}_1 &= (1-\eta_1\eta_2)\xi + (1-\xi)\eta_3, & J_4 &= (1-\xi)\xi^2\eta_2.
\end{aligned} \tag{3.13}$$

3.2.1.3. One Common Vertex

In the case when there is only one common vertex between the shape elements $\hat{x}$ and $\hat{y}$, there will be only two transformation to get rid of singularities. In this setting simply the parametrization can be written as the following:

$$
\begin{aligned}
&\mathbf{z}_1 = \hat{\mathbf{x}}_1, \\
&\mathbf{z}_2 = \hat{\mathbf{x}}_2, \\
&\mathbf{z}_3 = \hat{\mathbf{y}}_2, \\
&\mathbf{z}_4 = \hat{\mathbf{y}}_1, \qquad\qquad \hat{\mathbf{y}}_3 \leq \hat{\mathbf{y}}_2,
\end{aligned}
\tag{3.14}
$$

where the variable $\mathbf{z}$ domain is union of the zone D_1 and D_2 thus,

$$\mathbf{z} \in \bigcup_{k=1}^{2} D_k,$$

$$
\begin{aligned}
D_1 &= \left\{ \mathbf{z} : \begin{array}{r} 0 \leq \mathbf{z}_2 \leq \mathbf{z}_1 \leq 1 \\ 0 \leq \mathbf{z}_4 \leq \mathbf{z}_3 \leq 1 \\ \mathbf{z}_3 \leq \mathbf{z}_1 \end{array} \right\}, \\
D_2 &= \left\{ \mathbf{z} : \begin{array}{r} 0 \leq \mathbf{z}_2 \leq \mathbf{z}_1 \leq 1 \\ 0 \leq \mathbf{z}_4 \leq \mathbf{z}_3 \leq 1 \\ \mathbf{z}_1 \leq \mathbf{z}_3 \end{array} \right\}.
\end{aligned}
\tag{3.15}
$$

Thus,

$$\mathbf{z} = \xi R_k(\eta_1, \eta_2, \eta_3), \quad \mathbf{z} \in D_k, \quad \forall \quad \xi, \eta_1, \eta_2 \in [0,1]^3,$$

where

$$
R_1 = \begin{bmatrix} 1 \\ \eta_1 \\ \eta_2 \\ \eta_2\eta_3 \end{bmatrix}, \qquad R_2 = \begin{bmatrix} \eta_2 \\ \eta_2\eta_3 \\ 1 \\ \eta_1 \end{bmatrix},
\tag{3.16}
$$

with Jacobian for both is $J_k = \xi^2 \eta_2 \qquad \forall k \in \{1, 2\}$.

3.3. Affine Transformations

If the integrand function is not smooth, there will be intersection between a surface panel and interior tetrahedra. This interaction could be one of the above cases. In this work and without loss of generality the common vertex if occurs is assumed to be parametric image of the origin $(0,0) \in \sigma^{(2)}$. Otherwise, we apply affine transformation to rotate the panels such that the singular point moves to the origin. Here in the work, since we have two different kinds of reference elements, which are surface panels and interior tetrahedra, we derive two affine transformations for this mission. They are affine panel transformation and affine tetrahedron transformation.

3.3.1. Affine Panel Transformation

Consider a unite triangle with vertices $\mathbf{v}_1 = (0,0), \mathbf{v}_2 = (1,0)$, and $\mathbf{v}_3 = (1,1)$ where the index of the vertex location on the panel is defined as $\{0,1,2\}$. This index represents that the vertex $\mathbf{v}_1$ is located at the origin. As it is mentioned above, if the vertex $\mathbf{v}_2$ intersects with one of the tetrahedra vertex, the vertex $\mathbf{v}_2$ must be swap with the vertex $\mathbf{v}_1$. in other word, it must be moved to the origin. The affine transformation for this process is defined as the following,

$$\top^{Affine}(\mathbf{X}) = \mathbf{A}\mathbf{X}^{Index} + \mathbf{b}_\tau, \tag{3.17}$$

where the matrix $\mathbf{A} \in \mathbf{R}^{3\times 2}$ and the vector $\mathbf{b}_\tau \in \mathbf{R}^{2\times 1}$ such that, $\mathbf{A} = (\mathbf{E}')^{-1}\mathbf{F}'$ where the matrices $\mathbf{E} \in \mathbf{R}^{3\times 3}$ and $\mathbf{F} \in \mathbf{R}^{2\times 3}$ are defined as the following,

$$\mathbf{E} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ 1 & 1 & 1 \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}, \tag{3.18}$$

written as tuples, there are six permutations of the index set of a panels $\{0,1,2\}, \{0,2,1\}, \{1,0,2\}, \{1,2,0\}, \{2,0,1\}$, and $\{2,1,0\}$. These are all possible ordering of these three - Indices of vertex location. There for we need six Affine transformations in order to be able to rotate singular points into the right position so that later Duffy Trick be able to remove it efficiency and get a smooth integrand. The vector $\mathbf{b}_\tau$ is made up from the last row of the matrix $\mathbf{A}$. In this case the matrix $\mathbf{A}$ is reduced into a square matrix. i.e $\mathbf{A} \in \mathbf{R}^{2\times 2}$

3.3.2. Affine Tetrahedron Transformation

Consider the vertices of a tetrahedron $\mathbf{w}_1 = (0,0,0), \mathbf{w}_2 = (1,0,0), \mathbf{w}_3 = (1,1,0)$, and $\mathbf{w}_4 = (1,1,1)$. In this order of the vertices the index of the location of each vertex is defined as $\{0,1,2,3\}$. To rotate any of the vertices defined above, we need to derive a special affine transformation. Since we have a set of four coordinates indices, there will be 24 affine transformations, and each of the can be called at specific time to eliminate the singularities efficiency to get a smooth integrand function. The Affine Transformation, is defined as the following,

$$\top^{Affine}(\mathbf{Y}) = \mathbf{B}\mathbf{Y}^{Index} + \mathbf{k}_\tau, \tag{3.19}$$

where the matrix $\mathbf{B} \in \mathbf{R}^{4\times 3}$ and the vector $\mathbf{k}_\tau \in \mathbf{R}^{3\times 1}$ such that, $\mathbf{B} = (\mathbf{E}')^{-1}\mathbf{F}'$ where the matrices $\mathbf{E} \in \mathbf{R}^{4\times 4}$ and $\mathbf{F} \in \mathbf{R}^{3\times 4}$ are defined as the following,

$$\mathbf{E} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3 & \mathbf{w}_4 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \qquad \mathbf{F} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3 & \mathbf{w}_4 \end{bmatrix}. \tag{3.20}$$

Then the matrix $\mathbf{B}$ will be reduced into square matrix after the vector $\mathbf{k}_\tau$ is made up by the last row of the matrix $\mathbf{B}$. Therefor, $\mathbf{B} \in \mathbf{R}^{3\times 3}$ in the affine transformation (5.15).

3.4. Combining Source and Target Transformation

We will parameterize the integral in the equation (3.9) in order to construct the vector $\mathbf{b}$. To turn to the type of integrals that to be computed in the implementation of Galerkin boundary element method, we rewrite the Newton potential as

$$b_{\gamma,i} = \int_\gamma \int_\Omega \Phi(\mathbf{x}, \mathbf{y}, \phi_i, f) d\mathbf{y} ds_{\mathbf{x}}, \tag{3.21}$$

where $\Phi(\mathbf{x}, \mathbf{y}, \phi_i, f) = \phi_i(\mathbf{x})G(\mathbf{x}, \mathbf{y})f(\mathbf{y})$. Since the point $\mathbf{x}$ on the surface and $\mathbf{y}$ inside the domain, $\mathbf{x}$ is parameterized as panel and $\mathbf{y}$ as tetrahedron. Therefore, $\mathbf{x}$ is $2D$, while $\mathbf{y}$ is in $3D$. This computation needs efficient, robust, and accurate $5D$ quadrature rule. Incorporate all shape functions and Jacobian of the both parameterization, we get the formula below

$$b_{\gamma,i} = \int_0^1 \int_0^{\hat{\mathbf{x}}_1} \int_0^1 \int_0^{\hat{\mathbf{y}}_1} \int_0^{\hat{\mathbf{y}}_2} \Phi(\hat{\mathbf{x}}, \hat{\mathbf{y}}) J(\hat{\mathbf{x}}) J(\hat{\mathbf{y}}) d\hat{\mathbf{y}}_3 d\hat{\mathbf{y}}_2 d\hat{\mathbf{y}}_1 d\hat{\mathbf{x}}_2 d\hat{\mathbf{x}}_1, \tag{3.22}$$

if this integrand function is smooth i.e $i \neq j$, then the integrand variable will be parameterized again and turned into function dependent on ζ and η such that

$$b_{\gamma,i} = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \Phi(\eta, \zeta) J(\hat{\mathbf{x}}) J(\hat{\mathbf{y}}) J(\eta) d\zeta d\eta_1 d\eta_2 d\eta_3 d\eta_4, \tag{3.23}$$

if the integrand function is not smooth, and there is intersection between $\mathbf{x}$ and $\mathbf{y}$ which could be one, two, or three vertices, the formula will be reformulated again by calling space and affine transformation. Then, parametrization again is required for $\mathbf{z}$ and $\mathbf{w}$. Therefore,

$$b_{\gamma,i} = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 \Phi(\mathbf{z}, \mathbf{w}) J(\hat{\mathbf{x}}) J(\hat{\mathbf{y}}) J(\eta) J(\mathbf{z}) d\zeta d\eta_1 d\eta_2 d\eta_3 d\eta_4. \tag{3.24}$$

This equation (3.24) is used in the implementation to compute the vector $\mathbf{b}$ efficiency based on the proposed new refinement scheme "Boundary Concentrated Subdivision" so that it can be treated and tuned with boundary element method.

Chapter 4

The Variable Order FMM For Surface Potentials

Although the boundary element method BEM a powerful numerical tools for scientific computing and mathematical simulation for physical and natural phenomena, it has limitation for analyzing large - scale models. This because of the dense matrix that generated after applying BEM to the model problems. It is clear that a dense matrix requires $\mathcal{O}(h^{-4})$ operations for computing the coefficients and solving the system by any direct method. The innovation of the fast multipole method FMM by Rokhlin and Greengard for studying simulations of particle interaction in artificial domains [12, 13] can also be used to reduce the work complexity of building, allocating, and solving the system generated by BEM. In this Chapter, we will review the FMM and variable order FMM [40] for surface layer potential because this helps reader to understand our proposed new version of FMM called boundary concentrated fast multipole method BCFMM for volume potentials. This will be described in detail in the following chapters.

4.1. The FMM for Surface Integrals

To facilitate the description of the classic FMM, consider the boundary element method system of equation (1.4)for the case that the source function vanishes. Recall that $A\mathbf{q} = \mathbf{b}$ where $A_{i,j} = \langle \phi_i, \mathcal{K}\phi_j \rangle$ are the matrix A entries. If h is meshwidth near the boundary $\partial\Omega$ of the domain Ω, the FMM can accelerate the computation of the matrix vector product to $\mathcal{O}(p^2h^{-2})$ operations and thus reduce the work complexity of solving this system if an iterative and preconditional solver is used. Here p is the order of the multipole expansion of the kernel. Also, it is important to mention here that in order to reduce the errors generated by domain mesh and multipole expansion, one needs to increase the order $\mathbf{p}$ and decrease the meshwidth h. This is considered to be expensive for large-scale model problems [13]. For more details about work complexity reductions by the FMM for surface integrals, we refer to [14, 40].

4.2. Hierarchical Subdivision of Surface

In order to understand our proposed domain subdivision scheme "Boundary Concentrated subdivision" of the $\Omega \subset \mathbf{R}^3$ which will be described in details in the next Chapter and hierarchical subdivision of volume in the Chapter 6, we briefly exhibit this surface splitting scheme on which the classic fast multipole method rely to compute far field particle-particle interactions. The presentation in this section is based on the papers [40] and [13]. In the first step of the process of refinement, the surface of the domain is embedded by a cube C_0, the index refers to top-level (level-zero). Then, the next generation of cubes is obtained by dividing the cube C_0 into 8 equal sized cubes, and the collection of the cubes at this level is called C_1. This procedure can be repeated l times until the finest level cubes contain no more than a predetermined number of panels. Generated C_l is the set of all nonempty cubes in level ℓ.

Suppose that the collection of the panels in a cube ν is denoted by S_ν then obviously we have $\Gamma = \bigcup\limits_{\nu \in C_\ell} S_\nu$ for any level $0 \leq \ell \leq L$.

The bounding box B_ν of S_ν is the smallest axiparallel box that contains S_ν. We choose a point x_ν on S_ν such that x_ν is a center of B_ν and satisfy

$$\rho_\nu = \max_{v:\text{vertex of } B_\nu} |v - x_\nu| \,.$$

We define the separation ratio of two different cubes in the same level as

$$\eta_{\nu,\nu'} = \frac{\rho_\nu + \rho_{\nu'}}{|x_\nu - x_{\nu'}|} \,. \quad \nu \neq \nu', \tag{4.1}$$

To describe the FMM, the following concepts will be needed.

- $\mathcal{K}(\nu)$ denotes the set of all nonempty children of a cube ν .
- $\pi(\nu)$ is the parent of a cube ν.

- The neighbors $\mathcal{N}(\nu)$ of cube $\nu \in C_l$ consist of the cubes that share at least one vertex with ν as well as the cubes $\nu' \in C_\ell$ that satisfy

$$\eta_{\nu,\nu'} \geq \eta. \tag{4.2}$$

  That is, neighbors are cubes with a large separation ratio. The parameter η is predetermined. The smaller η, the more cubes are considered neighbors.

- $\mathcal{I}(\nu)$ is called interaction list. This list consists of all cubes that they are not neighbor but whose parents are neighbors i.e

$$\mathcal{I}(\nu) := \{\nu' \in C_l : \pi(\nu') \in \mathcal{N}(\pi(\nu)) \text{ and } \nu' \notin \mathcal{N}(\nu)\} \; .$$

With this definition of neighbors and interaction lists, we can get the following group of elements:

$$\bigcup_{\nu' \in \mathcal{N}(\nu)} S_\nu \times S_{\nu'} = \bigcup_{\mu \in \mathcal{K}(\nu)} \left\{ \bigcup_{\mu' \in \mathcal{N}(\mu)} S_\mu \times S_{\mu'} \;\cup\; \bigcup_{\mu' \in \mathcal{I}(\mu)} S_\mu \times S_{\mu'} \right\}.$$

Applying this splitting repeatedly, we obtain the following decomposition:

$$\Gamma \times \Gamma = \bigcup_{\substack{\nu \in C_L \\ \nu' \in \mathcal{N}(\nu)}} S_\nu \times S_{\nu'} \;\cup \bigcup_{l=2}^{L} \bigcup_{\substack{\nu \in C_l \\ \nu' \in \mathcal{I}(\nu)}} S_\nu \times S_{\nu'}. \tag{4.3}$$

Hence the inner product of $\langle \phi, \mathcal{K} f \rangle$ can be computed as a combination of far field and near field, i.e :

$$\langle \phi, \mathcal{K} f \rangle = \sum_{\substack{\nu \in C_L \\ \nu' \in \mathcal{N}(\nu)}} \langle \phi_\nu, \mathcal{K} f_{\nu'} \rangle + \sum_{l=2}^{L} \sum_{\substack{\nu \in C_l \\ \nu' \in \mathcal{I}(\nu)}} \langle \phi_\nu, \mathcal{K} f_{\nu'} \rangle \,, \tag{4.4}$$

for $\phi, f \in L^2(S)$ such that

$$\phi_\nu(x) = \begin{cases} \phi(x), & x \in S_\nu, \\ 0, & \text{otherwise.} \end{cases}$$

4.3. Surface Fast Multipole Method

What is the key idea of the surface FMM? Equation (4.4), decompose the bilinear form into a combination of Far Field (FF) and Near Field (NF). We are employing direct summation to compute the (NF), while the fast multipole method is implemented to approximately compute the (FF) and save computational cost. The key idea is to expand the kernel into a truncated series expansion. This can be achieved by using the multipole expansion, kernel interpolation, or the Taylor series. Since the FF consists of interaction of well separated cubes, the kernel is smooth, and therefore these approximations converge rapidly. Specially, the cubes ν and $\nu\prime$ are well separated (i.e they are separated if they are satisfy special criteria (4.2)), then the potential $\phi_{\nu,\nu'}$ due to the panels in both clusters can be approximated by using a Taylor series expansion. as follows

$$\begin{aligned}
\phi_{\nu\nu'}(x) &= \int_{S_{\nu'}} \left(\frac{\partial}{\partial n'}\right)^{\kappa} G(|x-x'|)g(x')\,ds(x') \\
&\approx \sum_{|\alpha|\le p}\sum_{|\beta|\le p-|\alpha|} \frac{D^{\alpha+\beta}G(|r_{\nu\nu'}|)}{\alpha!\beta!}(x-x_\nu)^\alpha \int_{S'_\nu}(x_{\nu'}-x')^\beta g(x')\,ds(x') \\
&= \sum_{|\alpha|\le p} \lambda^{\alpha}_{\nu,\nu'}(x-x_\nu)^\alpha, \quad x\in S_\nu, \qquad (4.5)
\end{aligned}$$

where α,β are multiindices and $\kappa = 0$ for the single and $\kappa = 1$ for the double layer operator. Here the local expansion coefficient $\lambda^{\alpha}_{\nu,\nu'}$ is given by

$$\lambda^{\alpha}_{\nu,\nu'} = \sum_{|\beta|\le p-|\alpha|} \frac{D^{\alpha+\beta}G(|r_{\nu\nu'}|)}{\alpha!\beta!}(-1)^{|\beta|} m^{\beta}_{\nu'}(g), \quad |\alpha|\le p, \qquad (4.6)$$

where $m^{\beta}_{\nu'}(g)$ is a moment of the function g, which is given by

$$m^{\beta}_{\nu'}(g) = \int_{S_{\nu'}} \left(\frac{\partial}{\partial n_x}\right)^{\kappa}(x-x_{\nu'})^\beta g(x)\,ds(x), \quad |\beta|\le p, \qquad (4.7)$$

where $r_{\nu\nu'} = x_\nu - x_{\nu'}$ and the order of the expansion is p. The computation in the equation (3.4) is called MtL, and its matrix notation is $\lambda_\nu = T(\nu,\nu')m_{\nu'}$. However, the equation (3.6) used to compute MtM, and its matrix notation is $m_\nu = \sum_{\nu'} M(\nu,\nu')m_{\nu'}$.

$$\begin{aligned}
m^{\alpha}_{\nu}(g) &= \sum_{\nu'\in\mathcal{K}(\nu)} \int_{S_{\nu'}} \frac{\partial^\kappa}{\partial n_x^\kappa}(x_{\nu'}-x_\nu+x-x_{\nu'})^\alpha g(x)\,ds(x) \\
&= \sum_{\nu'\in\mathcal{K}(\nu)}\sum_{\beta\le\alpha} \binom{\alpha}{\beta}(x_{\nu'}-x_\nu)^{\alpha-\beta} m^{\beta}_{\nu'}(g), \qquad |\alpha|\le p. \qquad (4.8)
\end{aligned}$$

Also, the procedure of LtL can be computed by equation (3.10), and its matrix notation is $\lambda_{\nu'} = L(\nu', \nu)\lambda_\nu$

$$\begin{aligned}
\sum_{|\alpha|\leq p} \lambda_\nu^\alpha (x - x_\nu)^\alpha &= \sum_\alpha \lambda_\nu^\alpha (x - x_{\nu'} + x_{\nu'} - x_\nu)^\alpha && (4.9)\\
&= \sum_{|\alpha|\leq p} \lambda_\nu^\alpha \sum_{\beta\leq\alpha} \binom{\alpha}{\beta} (x_{\nu'} - x_{\nu'})^{\alpha-\beta} (x - x_{\nu'})^\beta && (4.10)\\
&= \sum_{|\beta|\leq p} \left(\sum_{\substack{\alpha\geq\beta \\ |\alpha|\leq p}} \binom{\alpha}{\beta} (x_{\nu'} - x_\nu)^{\alpha-\beta} \lambda_\nu^\alpha \right) (x - x_{\nu'})^\beta && (4.11)\\
&= \sum_{|\beta|\leq p} \lambda_{\nu'}^\beta (x - x_{\nu'})^\beta. && (4.12)
\end{aligned}$$

Algorithm 4.1 Classic Fast Multipole Method

```
1: for $\nu \in C_L$ do                                   ▷ Nearfield Calculation
2:     $\hat{\phi}_\nu = \sum_{\nu' \in \mathcal{N}(\nu)} K(\nu, \nu')\hat{f}_{\nu'}$
3: end for
4: for $\nu \in C_L$ do                                   ▷ Moment Calculation
5:     $m_\nu = Q(\nu)\hat{f}_\nu$
6: end for
7: for $l = L-1, \ldots, 2$ do                            ▷ Upward Pass
8:     for $\nu \in C_l$ do
9:         $m_\nu = \sum_{\nu' \in \mathcal{K}(\nu)} M(\nu, \nu')m_{\nu'}$
10:    end for
11: end for
12: △Interaction Phase:
13: for $l = L, \ldots, 2$ do                             ▷ Interaction Phase
14:     for $\nu \in C_l$ do
15:         $\lambda_\nu = \sum_{\nu' \in \mathcal{I}(\nu)} T(\nu, \nu')m_{\nu'}$
16:     end for
17: end for
18: for $l = 2, \ldots, L-1$ do                           ▷ Downward Pass
19:     for $\nu \in C_l$ do
20:         for $\nu' \in \mathcal{K}(\nu)$ do
21:             $\lambda_{\nu'} \mathrel{+}= L(\nu', \nu)\lambda_\nu$
22:         end for
23:     end for
24: end for
25: for $\nu \in C_L$ do                                   ▷ Evaluation Phase
26:     $\hat{\phi}_\nu \mathrel{+}= U(\nu)\lambda_\nu$
27: end for
```

4.4. The Variable Order FMM

The most dominant cost of the fast multipole method are the translations of the moments to far field coefficients which is known as MtL. In the classic version of FMM [12], the cost of this translation is $\mathcal{O}(\mathbf{p}^4)$, while the later version of FMM helped to reduce this translation cost into $\mathcal{O}(\mathbf{p}^2)$ [14]. A different approach was proposed in [40] to reduce the complexity of all translations by adjusting the expansion order to the level of refinement. In fact, the finest level of refinement started at low order of expansions, then the order incremented in each level. This helps to reduce the work complexity in the finest level where the most MtL translations need to be computed. On the other hand, only a few number of expensive translations in the coarse level need to be computed. Thus, computations at the coarser level do not contribute significantly to over all work cost. It was shown in [40] that in special situations the work complexity can be reduced into $\mathcal{O}(h^{-2})$ where h is meshwidth, while truncation error is $\mathcal{O}(h)$.
The variable order fast multipole method VFMM that implemented in [40] used a decreasing sequence of expansion order from $p_2, p_3, \ldots, p_L$. In the interaction level, low order expansion are used at the finest level $l = L$, then the order incremented in each upcoming level. In this case, order of expansion at level $l = 2$ is p_L, while it is p_2 at the level $l = L$. For the other translations (Upward Pass and Downward Pass) there are two alternatives:

1. Compute all moments in all level with the maximal order P_2. This ensure that all moments in all levels are exact.

2. Compute moments in level ℓ only up to order P_ℓ. Moments in the next coarser level are computed by MtM translations. Since $P_{\ell-1}$ is greater than P_ℓ, replace moments in level ℓ of order greater than P_ℓ by zero. This results in an approximation of the moment, which has been shown to be sufficiently accurate in [40].

4.5. Variable Order vs Fixed Order FMM

The FMM can accelerate the work complexity of well separated panel interactions. The dominant cost of the FMM are the interaction phase in which, the fixed order FMM reduces the work complexity into $\mathcal{O}(p^2)$, where p is expansion order, but the complexity of matrix vector multiplication is scale $\mathcal{O}(p^2h^{-2})$ with an improved constant factor [14]. To reduce the discretization and expansion errors, one can increase the order of approximation p and h approaches to zero. This leads to increase the work complexity faster than expected. Thus, the fixed order FMM is not always optimal. On the other hand, the use of the low order expansion in the finest level in the variable order FMM employs inexpensive translations in the finest level, where most translation must be computed. Thus, the variable order FMM can reduce the work complexity into $\mathcal{O}(h^{-2})$, and it is optimal option for accelerating a work complexity. For more details see [40].

Chapter 5

BCFMM For Volume Potentials

In this chapter, we present a new version of the fast multipole method FMM based on a boundary concentrated domain subdivision which is called BCFMM. Then, we will discuss how this new version applied to the boundary concentrated subdivision can be used to compute the Newton potential $\langle \phi_i, \mathcal{N} f \rangle$ that arise in the boundary element method (1.2) and (1.4). The tasks bellow need to be computed:

- **Task 1.** Given a function f, compute the potential $\tilde{\mathcal{N}} f(\mathbf{x})$ in the domain with sufficient accuracy using nearly $\mathcal{O}(h^{-2})$ operations.
- **Task 2.** Given a function f, compute the $b_{\gamma,i}$'s with sufficient accuracy using nearly $\mathcal{O}(h^{-2})$ operations.
- **Task 3.** Given a function q_h, compute the potential $\tilde{\mathcal{V}} q_h(\mathbf{x})$ in the domain with sufficient accuracy using nearly $\mathcal{O}(h^{-2})$ operations.

Note that in task 2 the Newton potential is integrated against a test function, whereas in tasks 1 and 3 the potential is evaluated in a pointwise sense on the boundary concentrated subdivision. An approximation of the function in Ω can then be obtained by interpolation. We remark that to compute the Dirichlet-to-Neumann map via (1.4) it suffices to employ task 1. Further, task 2 is required even for the homogeneous PDE, if the solution in the domain Ω is sought. These two tasks are closely related by the fact that they are adjoint.

5.1. Hierarchical Volume Splitting

We briefly describe the hierarchical volume decomposition that BCFMM uses to compute far field efficiently . Instead of using a cube as in classic FMM, the BCFMM begins with a

polyhedron $\Omega \subset \mathbb{R}^3$ that is subdivided into a small number of tetrahedra ω. These tetrahedra are the coarsest level, or level zero in the triangulation. The $\ell + 1$-st level tetrahedra are obtained by subdividing tetrahedra in the ℓ-st level into eight congruent tetrahedra.
The tetrahedra ω generated from this process of refinement is called children, and they are combined in one group which is denoted by $\mathcal{K}(\omega)$. If all tetrahedra in a level are subdivided, a uniform refinement results. In contrast, a boundary concentrated subdivision is obtained by refining only tetrahedra that are close to the boundary. A two dimensional situation is illustrated in Figure 5.1. Note that a boundary concentrated subdivision is not conforming, but this is not required for the Newton potential.

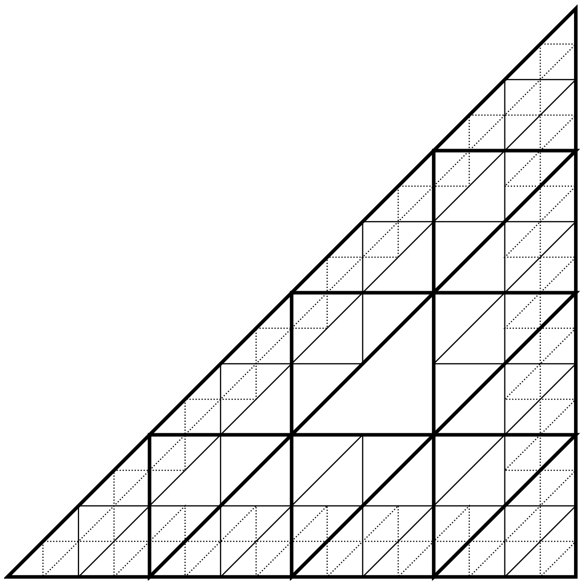


Figure 5.1: Illustration of a boundary concentrated subdivision of a triangle

To describe the process of a boundary concentrated subdivision in detail, we denote by C_ℓ the set of refined tetrahedra in the ℓ-th level. Further, for $\omega \in C_\ell$ the center of ω is denoted by $\mathbf{x}_\omega$ and the diameter is

$$\rho_\omega = \max_{\mathbf{v}:\text{vertex of } \omega} |\mathbf{v} - \mathbf{x}_\omega| \,. \tag{5.1}$$

We define the separation ratio of two different tetrahedra in the same level as

$$\eta(\omega, \omega') = \frac{\rho_{\omega'} + \rho_{\omega'}}{|x_\omega - x_{\omega'}|}\,, \quad \omega \neq \omega', \tag{5.2}$$

and set $\eta(\omega, \omega) = \infty$.

The neighbors of ω are the tetrahedra ω' in the same level for which the separation ratio is greater than a predetermined constant $\eta_0 < 1$. That is,

$$\mathcal{N}(\omega) = \{\omega' \in C_\ell \,:\, \eta(\omega, \omega') > \eta_0\}. \tag{5.3}$$

Further

$$B_\ell = \{\omega \in C_\ell \,:\, \omega \text{ has a face in } \Gamma\}, \tag{5.4}$$

denotes the set of all boundary tetrahedra. Here it is worth emphasizing that if ω only has one vertex in Γ it is not included in B_ℓ. Moreover,

$$M_\ell = \{\omega \in C_\ell \,:\, \mathcal{N}(\omega) \cap B_\ell \neq \emptyset\}, \tag{5.5}$$

denotes the tetrahedra that have a boundary tetrahedron among their neighbors. Thus, M_ℓ consists of tetrahedra that are close to the boundary relative to their size. We also write

$$U_\ell = C_\ell \setminus M_\ell \,. \tag{5.6}$$

In the boundary concentrated subdivision only the tetrahedra in M_ℓ are refined and included into the next level the tetrahedra U_ℓ are well separated from the boundary and therefore do not need to be defined. Thus, the next level list of tetrahedra is

$$C_{\ell+1} = \bigcup_{\omega \in M_\ell} \mathcal{K}(\omega). \tag{5.7}$$

Repeating this process L times we obtain the following subdivision of the domain

$$\bar{\Omega} = \bigcup_{\ell=0}^{L} \bigcup_{\omega \in U_\ell^*} \omega, \tag{5.8}$$

where $U_L^* = C_L$ and $U_\ell^* = U_\ell$ when $\ell < L$. In this decomposition, the interiors of all ω's are disjoint.

An important concept in the fast multipole method is the interaction list $\mathcal{I}(\omega)$, which consists of tetrahedra whose parents are neighbors, but who are not neighbors themselves. In level zero, we write $\mathcal{I}(\omega) = C_0 \setminus \mathcal{N}(\omega)$, which can be and often is an empty set.

Suppose now that for a level ℓ we have created all sets C_ℓ, B_ℓ and M_ℓ, and all neighbor and all interaction lists. Algorithm 5.1 generates the neighbor and interaction lists for the tetrahedra in $C_{\ell+1}$

Algorithm 5.1 Recursive generation of neighbor and interaction lists.

```
1: for $\omega \in M_\ell$ and $\omega' \in \mathcal{N}(\omega)$ do
2:     for $\tau \in \mathcal{K}(\omega)$ and $\tau' \in \mathcal{K}(\omega')$ do
3:         if $\eta(\omega, \omega') > \eta_0$ then
4:             add $\tau'$ to $\mathcal{N}(\tau)$
5:         else
6:             add $\tau'$ to $\mathcal{I}(\tau)$.
7:         end if
8:     end for
9: end for
```

5.2. Domain Decomposition of All Tasks

The key idea of the fast multipole method is a hierarchical splitting of the domain where sources and targets are located. To motivate the construction for a boundary concentrated subdivision we first review a uniform refinement. To that end, let D_ℓ denote all tetrahedra in the ℓ-th refinement level. We define neighbors and interaction lists of a tetrahedron exactly as before. Then by the definition of interaction lists,

$$\bigcup_{\omega \in D_\ell} \omega \times \mathcal{N}(\omega) = \bigcup_{\omega \in D_{\ell+1}} \omega \times \mathcal{N}(\omega) \ \cup \bigcup_{\omega \in D_{\ell+1}} \omega \times \mathcal{I}(\omega). \tag{5.9}$$

Applying this to the coarsest level,

$$\Omega \times \Omega = D_0 \times D_0 = \bigcup_{\omega \in D_0} \omega \times \mathcal{N}(\omega) \ \cup \bigcup_{\omega \in D_0} \omega \times \mathcal{I}(\omega) \tag{5.10}$$

$$= \bigcup_{\omega \in D_1} \omega \times \mathcal{N}(\omega) \ \cup \bigcup_{\omega \in D_1} \omega \times \mathcal{I}(\omega) \ \cup \bigcup_{\omega \in D_0} \omega \times \mathcal{I}(\omega), \tag{5.11}$$

and repeating for all finer levels gives

$$\Omega \times \Omega = \bigcup_{\omega \in D_L} \omega \times \mathcal{N}(\omega) \;\cup\; \bigcup_{\ell=0}^{L} \bigcup_{\omega \in D_\ell} \omega \times \mathcal{I}(\omega). \tag{5.12}$$

The first term is the nearfield, which is evaluated directly, and the second term consists of well separated sources and targets in all levels, that are approximated by fast evaluation methods. For the case that Ω is a unit cube this is exactly the decomposition was used in the original classic FMM papers [13], which explained in the Chapter 3.
As we mentioned earlier that we are pursuing different refinement scheme (BC) subdivision, based on that scheme the near and far field are generated for each tasks mentioned above. In fact, in the process of decomposition of each task only marked tetrahedra can be refined; see the following decomposition for each task:

5.2.1. Decomposition for Task 1

For the analogous decomposition for the boundary-concentrated subdivision, note first that $C_0 = D_0$ and $C_0 = M_0 \cup U_0$ thus we get for the coarsest level

$$\Omega \times \Omega = C_0 \times C_0 = \bigcup_{\omega \in M_0} \omega \times \mathcal{N}(\omega) \;\cup\; \bigcup_{\omega \in U_0} \omega \times \mathcal{N}(\omega) \;\cup\; \bigcup_{\omega \in C_0} \omega \times \mathcal{I}(\omega). \tag{5.13}$$

From the construction of interaction lists and the fact that $C_{\ell+1} = M_{\ell+1} \cup U_{\ell+1}$ we can conclude that similarly to (5.9) that

$$\bigcup_{\omega \in M_\ell} \omega \times \mathcal{N}(\omega) = \bigcup_{\omega \in M_{\ell+1}} \omega \times \mathcal{N}(\omega) \;\cup\; \bigcup_{\omega \in U_{\ell+1}} \omega \times \mathcal{N}(\omega) \;\cup\; \bigcup_{\omega \in C_{\ell+1}} \omega \times \mathcal{I}(\omega), \tag{5.14}$$

holds. Hence it follows by recursion through levels that

$$\Omega \times \Omega = \bigcup_{\ell=0}^{L} \bigcup_{\omega \in U_\ell^*} \omega \times \mathcal{N}(\omega) \;\cup\; \bigcup_{\ell=0}^{L} \bigcup_{\omega \in C_\ell} \omega \times \mathcal{I}(\omega). \tag{5.15}$$

The middle term does not appear in (5.12). It represents the nearfield terms in coarser levels that are distanced from the boundary. In the fast algorithm below, they will be treated by direct integration over level-ℓ tetrahedra.

5.2.2. Decomposition for Tasks 2 and 3

In task 1 the target domain is replaced by the boundary and in task 2 the source domain is replaced by the boundary. We denote the set of boundary faces of a tetrahedron by $\gamma(\omega)$, further

$$\mathcal{N}_\Gamma(\omega) \;=\; \bigcup_{\omega'\in\mathcal{N}_\Gamma(\omega)} \gamma(\omega') \quad \text{and} \quad \mathcal{I}_\Gamma(\omega) \;=\; \bigcup_{\omega'\in\mathcal{I}_\Gamma(\omega)} \gamma(\omega'). \tag{5.16}$$

By the definition of the set U_ℓ it follows that $\gamma(\omega) = \emptyset$ and $\mathcal{N}_\Gamma(\omega) = \emptyset$ when $\omega \in U_\ell$. Moreover, $\gamma(\omega)$ is non-empty if and only if $\gamma(\omega)$ is in B_ℓ. With this in mind the decompositions for task 1 and 2 can be easily obtained from (5.15) by restricting either the source or target domain to the boundary. Thus

$$\Gamma \times \Omega = \bigcup_{\omega\in B_L} \gamma(\omega) \times \mathcal{N}(\omega) \;\;\cup\;\; \bigcup_{\ell=0}^{L} \bigcup_{\omega\in B_\ell} \gamma(\omega) \times \mathcal{I}(\omega), \tag{5.17}$$

$$\Omega \times \Gamma = \bigcup_{\omega\in M_L} \omega \times \mathcal{N}_\Gamma(\omega) \;\;\cup\;\; \bigcup_{\ell=0}^{L} \bigcup_{\omega\in C_\ell} \omega \times \mathcal{I}_\Gamma(\omega)\,. \tag{5.18}$$

5.3. Potentional Evaluation Using Space Decompositions

The decomposition in (5.8) states that Ω can be decomposed into disjoint tetrahedra in $U^*_m, m = 0, \ldots, L$. For a given $\omega_m \in U^*_m$ there are unique tetrahedra in the coarser levels $\omega_\ell \in M_\ell$ that contain ω_m. From (5.15) it follows that

$$\tilde{\mathcal{N}} f(\mathbf{x}) = \sum_{\ell=0}^{L-1} \int_{\mathcal{N}(\omega_m)} G(\mathbf{x},\mathbf{y}) f(\mathbf{y})\, d\mathbf{y} + \sum_{\ell=0}^{m} \int_{\mathcal{I}(\omega_\ell)} G(\mathbf{x},\mathbf{y}) f(\mathbf{y})\, d\mathbf{y}, \quad \mathbf{x} \in \omega_m. \tag{5.19}$$

To obtain a discrete representation of the Newton potential, the potential will be evaluated on a finite number of interpolation nodes in ω_m.

As in the classical FMM for uniform refinements the fast evaluation is based on evaluating neighbor interactions using direct evaluations and an expansion of the kernel to evaluate interactions in the interaction lists. The main difference in the fast algorithm for the boundary concentrated subdivision is that the finest level in which calculations have to be performed depends on how far the evaluation point is removed from the boundary surface.

5.3.1. Translation Operation for Farfield

The evaluation of the farfield terms in the above sum can be accomplished by using the standard Moment-to-Local (MtL) expansion. We briefly recall the derivation for the case that the kernel is approximated by a truncated Taylor series expansion.

Suppose that $\omega' \in \mathcal{I}(\omega)$ then the Taylor expansion of G at centered at $(\mathbf{x}_\omega, \mathbf{x}_{\omega'})$ gives

$$\begin{aligned}
\phi_{\omega\omega'}(\mathbf{x}) &= \int_{\omega'} G(\mathbf{x}, \mathbf{y}) f(\mathbf{y})\, d\mathbf{y} \\
&\approx \sum_{|\alpha|\le p} \sum_{|\beta|\le p-|\alpha|} \frac{D^{\alpha+\beta} G(\mathbf{x}_\omega, \mathbf{x}_{\omega'})}{\alpha!\beta!} (\mathbf{x} - \mathbf{x}_\omega)^\alpha \int_{\omega'} (\mathbf{x}_{\omega'} - \mathbf{y})^\beta f(\mathbf{y})\, d\mathbf{y} \\
&= \sum_{|\alpha|\le p} \lambda^\alpha_{\omega,\omega'} (\mathbf{x} - \mathbf{x}_\omega)^\alpha, \quad \mathbf{x} \in \omega. \qquad (5.20)
\end{aligned}$$

In the formula above, the coefficients are given by

$$\lambda^\alpha_{\omega,\omega'} = \sum_{|\beta|\le p-|\alpha|} \frac{D^{\alpha+\beta} G(\mathbf{x}_\omega, \mathbf{x}_{\omega'})}{\alpha!\beta!} (-1)^{|\beta|} m^\beta_{\omega'}(f), \quad |\alpha| \le p, \qquad (5.21)$$

where $m^\beta_{\omega'}(f)$ is a moment of the function f, which is given by

$$m^\beta_{\omega'}(f) = \int_{\omega'} (\mathbf{y} - \mathbf{x}_{\omega'})^\beta f(\mathbf{y})\, d\mathbf{y}, \quad |\beta| \le p. \qquad (5.22)$$

The relationship between the expansion coefficients and moments is the *MtL* translation; in matrix notation this will be written as $\lambda_\omega = T(\omega, \omega') m_{\omega'}$.

The essence of the fast algorithm is to construct the moments $m_{\omega'}$, then translate moments to local expansion coefficients, and finally evaluate the potential using the series expansion. In order to recursively compute the moments of a parent tetrahedron from the moments of its children, it is necessary to translate the center. This operation is the *MtM* translation. Its coefficients are derived easily from the multivariate binomial formula

$$\begin{aligned} m_\omega^\alpha(f) &= \sum_{\omega' \in \mathcal{K}(\omega)} \int_{\omega'} (\mathbf{x}_{\omega'} - \mathbf{x}_\omega + \mathbf{y} - \mathbf{x}_{\omega'})^\alpha f(\mathbf{y})\, d\mathbf{y} \\ &= \sum_{\omega' \in \mathcal{K}(\omega)} \sum_{\beta \le \alpha} \binom{\alpha}{\beta} (\mathbf{x}_{\omega'} - \mathbf{x}_\omega)^{\alpha-\beta} m_{\omega'}^\beta(f), \qquad |\alpha| \le p. \end{aligned} \tag{5.23}$$

In matrix notation (5.23) is $m_\omega = \sum_{\omega'} M(\omega, \omega') m_{\omega'}$. The local expansion coefficients of a tetrahedron ω' are computed from the expansion coefficients of the tetrahedron's parent ω by translating the expansion center

$$\begin{aligned} \sum_{|\alpha| \le p} \lambda_\nu^\alpha (\mathbf{y} - \mathbf{x}_\omega)^\alpha &= \sum_\alpha \lambda_\omega^\alpha (\mathbf{y} - \mathbf{x}_{\omega'} + \mathbf{x}_{\omega'} - \mathbf{x}_\omega)^\alpha \\ &= \sum_{|\alpha| \le p} \lambda_\omega^\alpha \sum_{\beta \le \alpha} \binom{\alpha}{\beta} (\mathbf{x}_{\omega'} - \mathbf{x}_{\omega'})^{\alpha-\beta} (\mathbf{y} - \mathbf{x}_{\omega'})^\beta \\ &= \sum_{|\beta| \le p} \left(\sum_{\substack{\alpha \ge \beta \\ |\alpha| \le p}} \binom{\alpha}{\beta} (\mathbf{x}_{\omega'} - \mathbf{x}_\omega)^{\alpha-\beta} \lambda_\omega^\alpha \right) (\mathbf{y} - \mathbf{x}_{\omega'})^\beta \\ &= \sum_{|\beta| \le p} \lambda_{\omega'}^\beta (\mathbf{y} - \mathbf{x}_{\omega'})^\beta. \end{aligned} \tag{5.24}$$

In matrix notation we write $\lambda_{\omega'} = L(\omega', \omega)\lambda_\omega$.

5.3.2. Translation operators for Nearfield

For the evaluation of a nearfield interaction in the equation (6.19) we use a quadrature rule that adjusts the singularity of the kernel. For the case that $\omega = \omega'$ this can be accomplished with the Duffy transform. If the nodes and weights are $(\mathbf{y}_k, w_k)$, $k = 1, \ldots, n_q$ then

$$\int_{\omega'} G(\mathbf{x}, \mathbf{y}) f(\mathbf{y})\, d\mathbf{y} \approx \sum_{k=1}^{n_q} G(\mathbf{x}, \mathbf{y}_k) w_k f(\mathbf{y}_k), \tag{5.25}$$

which in vector notation is $u_\omega = K(\omega, \omega') f_{\omega'}$.

5.4. Algorithms

As in the regular classic FMM it is possible agglomerate moments of a node ω by translating the moments of the children's nodes. We denote the corresponding matrix by $M(\omega, \omega')$. Likewise, expansion coefficients can be re-centered from the parent's node to the children, the corresponding matrix is denoted by $L(\omega, \omega')$, The transformation of $\hat{f}_\omega$ to the moments $m_\omega(f)$ is linear and will be written in matrix form as $m_\omega = Q(\omega)\hat{f}_\omega$. Likewise, the transformation of expansion coefficients λ_ω to coefficients of the corresponding potential in the nodal basis is linear and is written as $\hat{\phi}_\omega = U(\omega)\lambda_\omega$. Note that the $Q(\omega)$ and $U(\omega)$ are only needed if $\omega \in U_\ell$ for some ℓ. The boundary concentrated FMM for task 3 is summarized in Algorithm 5.2. Note that the notation $a \mathrel{+}= b$ means that variable b must be added to the value of variable a.

Algorithm 5.2 BCFMM for Task 1

```
1:  for $\ell = 0 : L$ do                                    ▷ Nearfield Calculation.
2:      for $\omega \in U_\ell^*$ do
3:          $\phi_\omega = \sum_{\omega' \in \mathcal{N}(\omega)} K(\omega, \omega') \hat{f}_{\omega'}$
4:      end for
5:  end for
6:  for $\ell = 0 : L$ do                                    ▷ Moment Calculation.
7:      for $\omega \in U_\ell^*$ do
8:          $m_\omega = Q(\omega) \hat{f}_\omega$
9:      end for
10: end for
11: for $\ell = L - 1 : 0$ do                                ▷ Upward Pass.
12:     for $\omega \in M_\ell$ do
13:         $m_\omega = \sum_{\omega' \in \mathcal{K}(\omega)} M(\omega, \omega') m_{\omega'}$
14:     end for
15: end for
16: for $\ell = L : 0$ do                                    ▷ Interaction Phase.
17:     for $\omega \in C_\ell$ do
18:         $\lambda_\omega = \sum_{\omega' \in \mathcal{I}(\omega)} T(\omega, \omega') m_{\omega'}$
19:     end for
20: end for
21: for $\ell = 0 : L - 1$ do                                ▷ Downward Pass.
22:     for $\omega \in M_\ell$ and $\omega' \in \mathcal{K}(\omega)$ do
23:         $\lambda_{\omega'} \mathrel{+}= L(\omega', \omega) \lambda_\omega$
24:     end for
25: end for
26: for $\ell = 0 : L$ do                                    ▷ Evaluation Phase.
27:     for $\omega \in U_\ell^*$ do
28:         $\phi_\omega \mathrel{+}= U(\omega) \lambda_\omega$
29:     end for
30: end for
```

Algorithm 5.3 BCFMM for Task 2.

```
 1: for  $\omega \in B_L$ do                                                ▷ Nearfield Calculation.
 2:     $b_\omega = \sum_{\omega' \in \mathcal{N}(\omega)} K(\gamma(\omega), \omega') \hat{f}_{\omega'}$
 3: end for
 4:
 5: for  $\ell = 0 : L$ do                                              ▷ Moment Calculation.
 6:     for  $\omega \in U_\ell^*$ do
 7:         $m_\omega = Q(\omega) \hat{f}_\omega$
 8:     end for
 9: end for
10:
11: for  $\ell = L - 1 : 1$ do                                                   ▷ Upward Pass.
12:     for  $\omega \in M_\ell$ do
13:         $m_\omega = \sum_{\omega' \in \mathcal{K}(\omega)} M(\omega, \omega') m_{\omega'}$
14:     end for
15: end for
16: for  $\ell = L : 0$ do                                               ▷ Interaction Phase.
17:     for  $\omega \in B_\ell$ do
18:         $\lambda_\omega = \sum_{\omega' \in \mathcal{I}(\omega)} T(\omega, \omega') m_{\omega'}$
19:     end for
20: end for
21: for  $\ell = 0 : L - 1$ do                                                ▷ Downward Pass.
22:     for  $\omega \in B_\ell$ and $\omega' \in \mathcal{K}(\omega) \cap B_\ell$ do
23:         $\lambda_{\omega'} \mathrel{+}= L(\omega', \omega) \lambda_\omega$
24:     end for
25: end for
26: for $\omega \in B_L$ do                                                  ▷ Evaluation Phase.
27:     $b_\omega \mathrel{+}= U(\gamma(\omega)) \lambda_\omega$
28: end for
```

Algorithm 5.4 BCFMM for Task 3

```
1:  for $\omega \in U_L^*$ do                                          ▷ Nearfield Calculation.
2:      $\phi_\omega = \sum_{\gamma \in \mathcal{N}_\Gamma(\omega)} K(\omega, \gamma) q_\gamma$
3:  end for
4:
5:  for $\omega \in B_L$ do                                            ▷ Moment Calculation.
6:      $m_\omega = Q(\gamma(\omega)) q_{\gamma(\omega)}$
7:  end for
8:
9:  for $\ell = L - 1 : 0$ do                                          ▷ Upward Pass.
10:     for $\omega \in B_\ell$ do
11:         $m_\omega = \sum_{\omega' \in \mathcal{K}(\omega) \cap B_\ell} M(\omega, \omega') m_{\omega'}$
12:     end for
13: end for
14: for $\ell = L : 0$ do                                              ▷ Interaction Phase.
15:     for $\omega \in C_\ell$ do
16:         $\lambda_\omega = \sum_{\omega' \in \mathcal{I}(\omega)} T(\omega, \omega') m_{\omega'}$
17:     end for
18: end for
19: for $\ell = 0 : L - 1$ do                                          ▷ Downward Pass.
20:     for $\omega \in M_\ell$ and $\omega' \in \mathcal{K}(\omega)$ do
21:         $\lambda_{\omega'} \mathrel{+}= L(\omega', \omega) \lambda_\omega$
22:     end for
23: end for
24: for $\ell = 0 : L$ do                                              ▷ Evaluation Phase.
25:     for $\omega \in U_\ell^*$ do
26:         $\phi_\omega \mathrel{+}= U(\omega) \lambda_\omega$
27:     end for
28: end for
```

Chapter 6

Bondary Concentrated Domain Subdivision

We have given details about the procedure of FMM for the surface potentials. Also, it is important to note here that the efficiency of this method strongly relies on a uniform refinement scheme in each level. Whereas, the BCFMM for volume potentials depends on the boundary concentrated subdivision. In this chapter, we compare between both discretization scheme and analyze quadrature error of Newton potentials.

6.1. BC Subdivision vs. Uniform Refinement

The uniform refinement is a process of subdivision of all elements in the collection C_l(the set of all tetrahedron at level l). In another word, all elements are selected for refinement repeatedly. This will divide the domain Ω into $\mathcal{O}(h^{-3})$ tetrahedra of side length h. However, the BC scheme refines only a tetrahedron ω that in the collection M_ℓ see the set (5.5). In this non-uniform process, the diameter of the tetrahedra grows linearly with the distance from the boundary. Since there are larger tetrahedra in the interior than near the boundary the number tetrahedra is $\mathcal{O}(h^{-2})$. Figures 6.1 and 6.2 show a BC mesh of a cube split into tetrahedra. When the refinement level increases, the number of element generated by uniform refinement is much higher than the number of tetrahedra generated by BC subdivision. Figure 6.3 displays this reduction clearly. In addition, we display the data generated from the procedure of refinement of the cube in the Tables 6.1 and 6.2 . In these tables, $\#U_\ell$ is the number of leaves, and $\#C_\ell$ is the number of tetrahedra at level ℓ. We also display the number of edges, panels, and vertices are of each level.

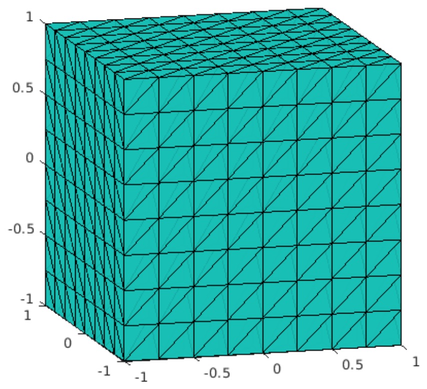


Figure 6.1: The boundary concentrated subdivision of the cube into tetrahedra and its faces that make up the surface triangulation

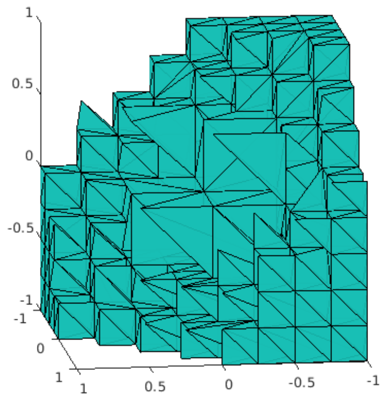


Figure 6.2: A visual view of the interior reference elements of the cube and the surface elements

Table 6.1: The number of elements of tetrahedra, faces, vertices, and edges was generated by the boundary concentrated subdivision for the unite cube.

Data Element Generated by BCR					
Level	**Vertices**	**Edges**	**Panels**	$\#C_\ell$	$\#U_\ell$
0	8	19	18	6	0
1	27	98	120	48	0
2	125	604	864	384	56
3	703	3862	5840	2680	932
4	3669	20670	31526	14524	5864
5	17195	97140	148174	68228	28604
6	75241	424568	646960	297632	126176
7	316863	1785362	2718132	1249632	531812
8	1307021	7355698	11191626	5142948	4449504

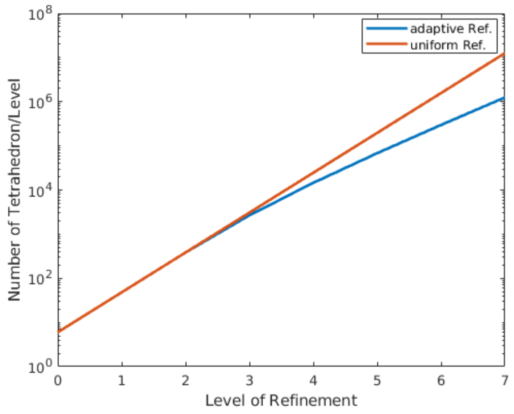


Figure 6.3: Rate of reduction of the reference element generated by BCR and compared to the uniform scheme

Table 6.2: The number of elements of tetrahedra, faces, vertices, and edges was generated by the uniform refinement scheme for the unite cube.

Data Element Generated by Uniform R.					
Level	**Vertices**	**Edges**	**Panels**	$\#C_\ell$	$\#U_\ell$
0	8	19	18	6	0
1	27	98	120	48	0
2	125	604	864	384	0
3	729	4184	6528	3072	0
4	4913	31024	50688	24576	0
5	35937	238688	399360	196608	0
6	274625	1872064	3170304	1572864	0
7	2146689	14827904	25264128	12582912	0
8	$\geq$64-Bit	$\geq$64-Bit	$\geq$64-Bit	$\geq$64-Bit	0

6.2. Accuracy and efficiency of Boundary Concentrated Subdivision

As a first test of the accuracy that can be obtained with the BC subdivision, we compute the Newton potential in one corner based on both uniform and BC subdivisions with same quadrature order for all tetrahedra. To that end, consider the point wise evaluation of the Newton potential $\mathcal{N}f$

$$\int_{\Omega} G(x,y)f(y)d(y) = \sum_{l=1}^{L} \sum_{\omega \in U_\ell} \int_{T} G(x,y)f(y)ds(y), \tag{6.1}$$

where L is total number of refinement levels, and the original domain Ω is a cube which divided into 6-Tetrahedra at Level C_0. Then, the second generation will be 48-tetrahedra which is level C_1 and so on. Now, the y coordinate must be parameterized with help of the Jacobian to relating infinitesimal areas in the artificial domain Ω before and after refinement process such that:

$$\int_{T} \Theta(y(t_1,t_2,t_3))|J|d\bar{t}, \tag{6.2}$$

where $\Theta(y) = G(x,y)f(y)$, and G is Green's function for potential equation with singularity. Now, Apply Duffy trick to handle the singularity of the integrand.

We observed that even though we do not refine the interior elements, the quadrature error generated by BC still goes down up to a certain point along with the quadrature error generated by uniform refinement. This illustrates the smoothness properties of the Newton potential for tetrahedra away from the boundary. Since in the BC subdivision the diameter of the tetrahedra grows linearly with the distance from the boundary surface, there are larger tetrahedra in the interior domain than near the boundary. Therefore, the quadrature error of the Newton potential at some point will stagnate if the BC subdivision continued. This is illustrated in Figure 6.4. To fix this issue, one must increase the quadrature order in the interior elements. For that reason we will use variable order expansion. In this case low order expansion will be used in the finest level of refinement, then this order is incremented in each level. By adjusting the order of the level of the domain decomposition Ω, we will show in the Chapter 8 that the error can be bounded by $\mathcal{O}(h^{-p})$. This could increase the work complexity by only logarithmically growing factors.

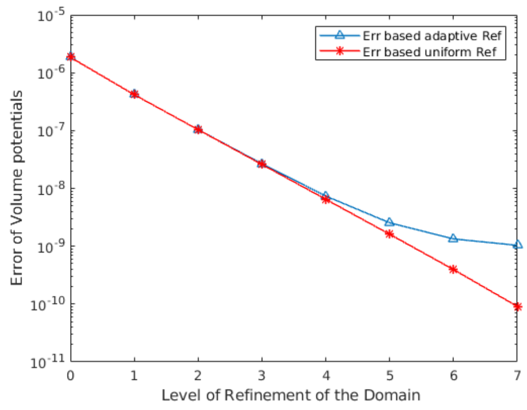


Figure 6.4: The quadrature error of the Newton potential based on boundary concentrated subdivision compared to the error based on uniform refinement when the evaluation points is located in one corner

Chapter 7

Error and Work Complexity Analysis

In this section we prove the main result of this dissertation. Namely, that the BCFMM can compute the volume potential to any desired accuracy in nearly optimal $\mathcal{O}(h^{-2L})$ operations. The Green's function in the BCFMM in the interaction list of well separated tetrahedra is approximated, while approximation can be achieved by various methods. We restrict ourselves to the truncated Taylor series. For more about estimates such expansion, we refer to [18]. In order to estimate the error for computing the Newton potential, we generalize the results in [40]. In this analysis, we will show the better and proper choice for the separation ration η and the expansion orders in the MtL translations. Also, we evaluate the flops counts for each step in the algorithm to determine overall complexity.

7.1. Error Estimate

Suppose that S_h is finite element space, and it consists of piecewise polynomial functions on each triangle γ which is one of the tetrahedron's face on the boundary. Also, let ϕ_h be function of S_h (h is mesh width) which makes up the boundary elements. Then, the bilinear form of the Newton potential can be written in the following form

$$\langle \phi_h, \mathcal{N}f \rangle = \int_\Gamma \int_\Omega G(\mathbf{x},\mathbf{y})\phi_h(\mathbf{x})f(\mathbf{y})d\mathbf{y}ds(\mathbf{x}). \tag{7.1}$$

By applying the hierarchical domain splitting, the domain is divided and decomposed as it is shown in chapter 5. Then we can write the above formula based on the domain subdivision.

$$\langle \phi_h, \mathcal{N}f \rangle = \sum_{\substack{\omega\in C_L \\ \omega'\in\mathcal{N}(\omega)}} \langle \phi_\omega, \mathcal{N}f_{\omega'} \rangle + \sum_{l=2}^{L} \sum_{\substack{\omega\in C_l \\ \omega'\in\mathcal{I}(\omega)}} \langle \phi_\omega, \mathcal{N}f_{\omega'} \rangle \ , \tag{7.2}$$

where the subscripts ω and ω' indicate restrictions of the function to the tetrahedron. The first sum is computed directly. The only error introduced by the BCFMM schemes from replacing the kernels in the second sum by truncated Taylor series. To that end, we have to estimate the remainders of the Taylor expansion for order p.

We consider a Green's function which depends on the distance of the source function and the field point that is $G(\mathbf{x}, \mathbf{y}) = G(|\mathbf{x} - \mathbf{y}|)$. Furthermore, we assume that $G(\cdot)$ is analytic except for the origin and that there is a constant C such that

$$|G(\tau)| \leq \frac{C}{|\tau|}, \quad 0 \neq \tau \in C. \tag{7.3}$$

The truncated Taylor series of the Green's function is given by

$$G(|\mathbf{r} + \mathbf{z}|) = \sum_{|\alpha| \leq p} \frac{D^\alpha G(|\mathbf{r}|)}{\alpha!} z^\alpha + R_p(\mathbf{r}, \mathbf{z}), \tag{7.4}$$

$$R_p(\mathbf{r}, \mathbf{z}) = G(\mathbf{x}, \mathbf{y}) - G(|\mathbf{r} + \mathbf{z}|) = G(\mathbf{x}, \mathbf{y}) - \sum_{|\alpha| \leq p} \frac{D^\alpha G(|\mathbf{r}|)}{\alpha!} z^\alpha, \tag{7.5}$$

where $\mathbf{r} = \mathbf{x}_\omega - \mathbf{x}_{\omega'}$ and $z = \mathbf{x} - \mathbf{y} - \mathbf{r}$. For separation ration $\eta < 1$ and for $\omega\prime \in \mathcal{I}(\omega)$, there is a constant C the remainder above is bounded by

$$|R_p(\mathbf{r}, \mathbf{z})| \leq c \frac{p}{|\mathbf{r}|} \left(\frac{|\mathbf{z}|}{|\mathbf{r}|} \right)^{p+1}, \tag{7.6}$$

where $\mathbf{r} = \mathbf{x}_\omega - \mathbf{x}_{\omega'}$ and $\mathbf{z} = \mathbf{x} - \mathbf{y} - \mathbf{r}$. For more details about proof of equation (7.6) see [40]. The bilinear form of Newton potential induced by FMM is defined as the following,

$$\langle \phi_h, \mathcal{N}_{\mathrm{F}} f \rangle = \sum_{\substack{\omega \in C_L \\ \omega' \in \mathcal{N}(\omega)}} \langle \phi_\omega, \mathcal{N} f_{\omega'} \rangle + \sum_{l=2}^{L} \sum_{\substack{\omega \in C_l \\ \omega' \in \mathcal{I}(\omega)}} \langle \phi_\omega, \mathcal{N}_{\omega,\omega'} f_{\omega'} \rangle \ . \tag{7.7}$$

There $\mathcal{N}_{\omega,\omega'}$ is the operator that results if the kernel G is replaced by the truncated series expansion. Comparing the bilinear forms of hierarchical domain splitting (7.2) with the above bilinear form (7.7) makes clear that the source of error comes from the MtL translation in the equation (6.21) . Thus for all $\phi_h \in S_h$ and source function f the error of the bilinear form is

$$\left\langle \phi_h, \left(\mathcal{N} - \mathcal{N}_{\mathrm{F}}\right) f_h \right\rangle = \sum_{l=2}^{L} \sum_{\substack{\omega \in C_l \\ \omega' \in \mathcal{I}(\omega)}} \left\langle \phi_\omega, \left(\mathcal{N} - \mathcal{N}_{\omega,\omega'}\right) f_{\omega'} \right\rangle . \tag{7.8}$$

Lemma 7.1 *Suppose that ω and $\omega' \in C_\ell$, and ω' is not neighbor of ω ($\omega' \notin \mathcal{N}(\omega)$) then*

$$\left|\left\langle \phi_\omega, \left(\mathcal{N} - \mathcal{N}_F\right) f_{\omega'} \right\rangle\right| \leq \frac{cp}{r_{\omega\omega'}} \eta^p_{\omega\omega'} 2^{-\ell} 2^{-\frac{3}{2}\ell} \|\phi\|_{L^2(\gamma(\omega'))} \|f\|_{L^2(\omega)} \tag{7.9}$$

Proof. For $\omega \in C_\ell$ and $\omega' \in \mathcal{I}(\omega)$

$$\eta_{\omega,\omega'} \geq \eta, \tag{7.10}$$

holds. Then the interaction in the equation (7.8) can be written in the integral formula as the following,

$$\begin{aligned} \langle \phi_\omega, (\mathcal{N} - \mathcal{N}_{\mathrm{F}}) f_{\omega'} \rangle &= \int_{\gamma(\omega')} \int_\omega \left(R_p(\mathbf{x}_\omega - \mathbf{x}_{\omega'}, \mathbf{x} - \mathbf{x}_\omega - \mathbf{y} - \mathbf{x}_{\omega'})\right) \phi(\mathbf{x}) f(\mathbf{y}) d\mathbf{y} ds(\mathbf{x}) && (7.11) \\ &\leq \frac{cp}{r_{\omega\omega'}} \eta^p \int_{\gamma(\omega')} |\phi(\mathbf{x})|\, ds(\mathbf{x}) \int_\omega |f(\mathbf{y})|\, d\mathbf{y} && (7.12) \\ &\leq \frac{cp}{r_{\omega\omega'}} \eta^p 2^{-\frac{5}{2}\ell} \|\phi\|_{L^2(\gamma(\omega'))} \|f\|_{L^2(\omega)}. && (7.13) \end{aligned}$$

The last step follows from the Cauchy Schwarz inequality and the fact that $|\gamma(\omega')| \leq C 2^{-2\ell}$ and $|\omega| \leq C 2^{-3\ell}$.

To obtain the error of the bilinear form, the contributions of all interactions must be added. Because of the factor $2^{-\frac{5}{2}l}$ in (7.9), the errors are smaller in the finer levels and it suffices to use a smaller expansion order. We set

$$p_l = p_L + \alpha(L - \ell), \tag{7.14}$$

where p_L and α are small numbers that will be optimized below.

Theorem 7.2 *For $f \in L_2(\Gamma)$, $\phi_h \in S_h$ and $\eta = 1/2$ the error of bilinear form induced by the FMM with orders given by (7.14) is bounded by*

$$\left|\left\langle \phi_h, \left(\mathcal{N} - \mathcal{N}_F\right) f_h \right\rangle\right| \leq c 2^{-p_L} \cdot 2^{-\min(\alpha, \frac{5}{2})L} \|\phi\|_{L^2(\Gamma)} \|f\|_{L^2(\Omega)} \tag{7.15}$$

Proof. Since interaction lists have a bounded number of elements, it follows that

$$\sum_{\substack{\omega\in C_\ell\\ \omega'\in\mathcal{I}(\omega)}} \|\phi_\omega\|\|f_{\omega'}\| \le \left(\sum_{\substack{\omega\in C_\ell\\ \omega'\in\mathcal{I}(\omega)}} \|\phi_\omega\|^2\right)^{\frac{1}{2}} \left(\sum_{\substack{\omega\in C_\ell\\ \omega'\in\mathcal{I}(\omega)}} \|f_\omega\|^2\right)^{\frac{1}{2}} \le c\|\phi\|_{L^2(\Gamma)}\|f\|_{L^2(\Omega)}. \tag{7.16}$$

We estimate (7.8) using (7.9)

$$\begin{aligned}
\left|\left\langle\phi_h, \left(\mathcal{N}-\mathcal{N}_{\mathrm{F}}\right)f_h\right\rangle\right| \;&\le\; c\sum_{\ell=2}^{L}\sum_{\substack{\omega\in\mathbf{C}_\ell\\ \omega'\in\mathcal{I}(\omega)}} p_\ell\eta_{\omega\omega'}^{p_\ell}2^{-\ell}2^{-\frac{3}{2}\ell}\|\phi\|_{L^2(\Gamma)}\|f\|_{L^2(\Omega)} && (7.17)\\
&=\; c2^{-p_L-\alpha L}\cdot\sum_{\ell=2}^{L}2^{(\alpha-\frac{5}{2})\ell}\|\phi\|_{L^2(\Gamma)}\|f\|_{L^2(\Omega)}, && (7.18)
\end{aligned}$$

if $\alpha < \frac{5}{2}$, then we will have

$$\sum_{\ell=2}^{L}2^{(\alpha-\frac{5}{2})\ell} \le 2, \tag{7.19}$$

if $\alpha > \frac{5}{2}$,

$$\sum_{\ell=2}^{L}2^{(\alpha-\frac{5}{2})\ell} \le 2\cdot 2^{(\alpha-\frac{5}{2})L}. \tag{7.20}$$

Hence it follows that

$$\left|\left\langle\phi_h, \left(\mathcal{N}-\mathcal{N}_{\mathrm{F}}\right)f_h\right\rangle\right| \le c2^{-p_L}\cdot 2^{-\min(\alpha,\frac{5}{2})L}\|\phi\|_{L^2(\Gamma)}\|f\|_{L^2(\Omega)}.$$

We can conclude that any convergence rate $2^{-L\mu}$ can be obtained by appropriate choices of α and p_L. For instance if $\mu \le \frac{5}{2}$, we simply set $\alpha = 3$. If $\mu \ge \frac{5}{2}$, we set $p_L = [\mu - \frac{5}{2}]\cdot L$ and $\alpha = 3$.

7.2. Complexity Estimate

We have seen in Chapter 5 that the number of tetrahedra generated by the BC subdivision at level ℓ is bounded by $c4^\ell$ i.e

$$\#C_l \le c4^\ell$$

Also, the number of flops in one MtL translation in the equation (5.21) is p_ℓ^6 where p_ℓ is the order of kernel expansion at level ℓ. Thus the overall all work complexity in the MtL translation is

$$N_{Inter} \leq \sum_{l=2}^{L} \#C_l p_l^6 \leq c \sum_{l=2}^{L} p_\ell^6 4^\ell \leq C 4^L \max p_\ell^6. \tag{7.21}$$

Since $h = 2^{-L}$ it follows that the complexity of the BCFMM is up to logarithmic factors, $\mathcal{O}(h^{-2})$.

Chapter 8

Implementation and Numerical Results

In order to verify the preceding analysis and test the variable order BCFMM for Galerkin discretizations of elliptic volume integral operators, we have implemented the method in the C programming language which combines all different modules presented in the previous chapters. To illustrate the reduction of CPU time (work complexity) and convergence of the variable order BCFMM, we evaluate the Poisson's equation in $\Omega \in \mathbb{R}^3$. The boundary concentrated subdivision is generated by refining two different domains a tetrahedron and a cube. The latter is given as a union of six tetrahedra. All computations in this work are in core on a single processor of Intel Xeon Phi 7230 known as Knights Landing or KNL with 385 GB of DDR4-2400 memory with 641.30GHz clock speed. To verify the method we consider the Green's equation

$$\frac{\phi(\mathbf{x})}{2} + \int_\Gamma G(\mathbf{x},\mathbf{y})q(\mathbf{y})\,ds_{\mathbf{y}} + \int_\Gamma \frac{\partial}{\partial n_{\mathbf{y}}} G(\mathbf{x},\mathbf{y})\phi(\mathbf{y})\,ds_{\mathbf{y}} = \int_\Omega G(\mathbf{x},\mathbf{y})f(\mathbf{y})d\mathbf{y}, \qquad \mathbf{x} \in \Gamma, \tag{8.1}$$

where $q(\mathbf{y}) = \frac{\partial \phi(\mathbf{y})}{\partial n_{\mathbf{y}}}$ and $G|\bullet|$ is the Green's function:

$$G(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi}\frac{1}{|\mathbf{x}-\mathbf{y}|}. \tag{8.2}$$

In this numerical tests, we have set

$$f(\mathbf{r}) = (6 - 4\,|\mathbf{r}|^2)\cdot \exp(-\,|\mathbf{r}|^2)$$
$$\phi(\mathbf{r}) = \exp(-\,|\mathbf{r}|^2) \qquad \mathbf{r} \in \mathbf{R^3},$$

which is a solution to the Poisson equation, and hence the surface potentials on the LHS of (8.1) must equal the volume potentials on the RHS of (8.1). For each numerical test, the domain Ω is the cube described in chapter 6. As discussed in the preceding chapters we have used the BCFMM to compute the volume potentials in the RHS of (8.1) and the surface FMM for the surface potentials on the LHS. Since these are two different numerical approximations, there will be a difference which we reported as the error.

8.1. Test Work Complexity of Slow Method

We first test the proposed boundary concentrated subdivision by computing the equation (8.1) where Newton potential appears in the right hand side. Specifically, we are solving for surface density q of the domain Ω where it is the cube described above. The number of tetrahedra at level $\ell + 1$ will be $6 \times 8^{\ell}$ with uniform scheme, and it is bounded by $c4^{\ell}$ with BC scheme. Also, the source function f is smooth exponential, and $\mathbf{x}$ is located on the surface we see that the error is going down even though we are ignoring large interior element in the next refinement process. Also, the running time is very high for this simple test, and this will be worst for large scale problems see Table 8.1 Note: SetupTime < 1 means the CPU time is less than one second because we are using time.h which only reports whole seconds.

Table 8.1: Results of Solving test problem (8.1) with BC subdivision and the slow method

Slow Method of Solving GBIE by BC Mesh					
Level	**SetupTime**	**Edges**	**Panels**	**Tetrahedra**	**Error**
0	< 1	19	18	6	2.31677
1	2	98	120	48	1.37725
2	16	604	864	384	0.575889
3	221	4184	6528	3072	0.157997
4	856	31024	50688	24576	0.064593

Also we see that the work complexity for computing the surface and volume integral in the equation (8.1) when the given domain is single polyhedron at level zero is high even for small scale problems see Table 8.2. Further, we note that the proposed domain subdivision boundary concentrated subdivision is accurate and efficient to compute surface integral on

Table 8.2: Work complexity and CPU time for computing surface and volume integral (slow method)

CPU Time for Volume and Surface Setup					
Level	**SetupTime**	**Surface**	**Volume**	**Tetrahedra**	**Error**
0	< 1	< 1	< 1	1	6.731
1	< 1	< 1	< 1	8	2.706
2	< 1	< 1	1	64	0.836
3	1	1	1	512	0.227
4	1	1	9	4096	0.058
5	11	11	178	32736	0.0125
ToTlevels: Tets=37417 ,Volume=2.666667					

the non-uniform volume triangulation. This means when the source function $f = 0$. The error goes down very fast. See Table 8.3.

8.2. Complexity and Convergence of Variable Order BCFMM

The proposed variable order BCFMM is applied to solve the test problem (8.1) which is a combination of volume and surface integrals with smooth source function f. We will test for the Ω tetrahedron and cube. This proposed BCFMM algorithm can compute the potential approximately of the generated non-uniform volume panels on uniform surface panels with CPU time and memory usage cost $\mathcal{O}(h^{-2})$ where h is diameter of tetrahedron with a distance from boundary in each mesh refinement. In other words, the time and memory usage is quadrupled each refinement step. See Table 8.4. The order of FMM translation in the Table 8.4 is constant for all translations which is set at $p = 2$ and the separation ratio is $\eta = \frac{1}{2}$.

The domain initially divided into 8 tetrahedra at the level one, then the algorithm of the proposed boundary concentrated subdivision moved tetrahedra near the boundary into

Table 8.3: Work complexity and CPU time for computing surface integral when source function vanished (slow method)

CPU Time for Setup Surface Integral				
Level	**SetupTime**	**Surface**	**Tetrahedra**	$L_2(\tau)$**-Error**
2	< 1	< 1	64	0.00381648
3	1	< 1	512	0.000645588
4	1	1	4096	0.000111469
5	12	110	32736	1.94652e-05
6	101	155	224736	3.41955e-06
ToTlevels: Tets=262153 leaves=229384				

Table 8.4: Work complexity and CPU time for computing surface and volume integrals via BCFMM with fixed order

CPU Time and Memory Usage				
Level	**SetupTime**	**Surface**	**Volume**	**Storage of Tets MB**
3	< 1	< 1	< 1	1.43e-01
4	2	< 1	1	1.14e+00
5	19	1	5	9.14e+00
6	156	3	25	6.40e+01
7	871	9	113	3.51e+02
ToTlevels: Tets=1436137 leaves=1256620				

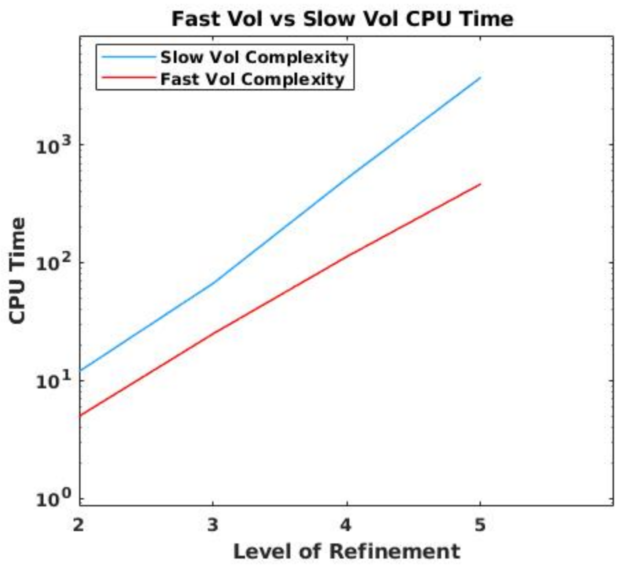


Figure 8.1: Work complexity of setup volume generated via BCFMM compared to the One generated by slow method

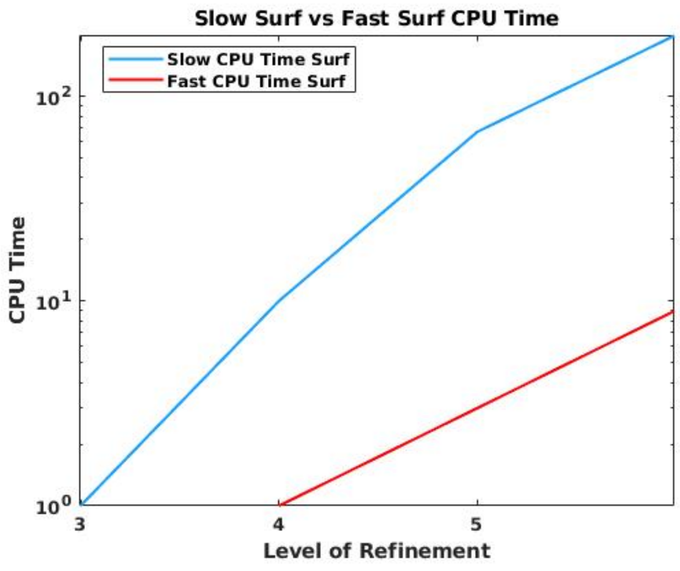


Figure 8.2: Work complexity of setup surface generated via BCFMM compared to the One generated by slow method

next process of refinement. In this case we must ignore elements that are away from the boundary. In the Table 8.5 we can observe this process of reduction.

Table 8.5: Number of elements that not in the marked list:Leaves and data statistics of BCFMM

Reduction of Element via BC Mesh/Level				
Level	**Volume**	**Leaves**	**Neighbor**	**Interaction**
3	0.00	0	244.22	251.27
4	0.002	4	435.04	1518.77
5	0.377	4644	562.93	2912.00
6	0.793	77088	597.91	3369.70
7	1.492	1173984	600.03	3369.28
Tot: leaves=1256620 nBrs=1378640193				

The neighbor and interaction lists in the Table 8.5 shows the average number of neighbor and interaction elements that attached to a tetrahedron per level. This non-uniform domain subdivision leads to generate non uniform tetrahedra with a different radius. Therefore, the interior elements are larger than the element that close to the boundary. In the Table 8.6 can see the change of radius per level of refinement, we can observe that through the columns labeled RMin and RMax. They display the maximal and minimal length of an edge for each level. We see from the table and the ration RMax/RMin remains constant. Therefore, there are no tetrahedra with a bad aspect ratio in the BC subdivision mesh.

Also, the table demonstrated that the number of tetrahedra is $\mathcal{O}(h^{-2})$ in the process of refinement. Due to the smooth properties of Newton Potential, the contribution of the interior tetrahedra do not need a high resolution as contribution near the boundary. Therefore, this BC subdivision will provide the same accuracy as the uniform mesh. However, since we use a fixed expansion order the error generated by the BC mesh will be grater by

Table 8.6: Data elements that generated by BC subdivision/ minimum and maximum radius of tetra per level with separation ratio $\eta = \frac{1}{2}$

Deta Element Generated by BC Mesh/Level				
Level	**RMin**	**RMax**	**Neighbor**	**Interaction**
0	1.73	1.73	1.00	0.00
1	0.86	1.06	8.00	0.00
2	0.43	0.53	61.93	2.07
3	0.21	0.26	244.22	251.27
4	0.10	0.13	435.04	1518.77
5	0.05	0.06	562.93	2912.01
6	0.02	0.03	597.91	3369.70
7	0.01	0.01	600.03	3369.28
Stge : Tetras = 2.29e+02: Nbrs=6.53e+03 MB				

logarithmical factor than the one generated by uniform refinement See Figure 6.4. As was described there, this ensure full convergence, while maintaining log-linear complexity in the number of tetrahedra. See the cost of VolTime for both variable and fixed order in the Table 8.7 and Figure 8.3.

Table 8.7: Comparing the work complexity generated by variable and fixed order of expansion in the BCFMM with separation ratio $\eta = \frac{1}{2}$

Variable and Fixed Order CPU Time						
Level	**FixOrder**	**VolTime**	**Ratio**	**VarOrder**	**VolTime**	**Ratio**
2	2	< 1	–	8	< 1	–
3	2	< 1	–	7	< 1	–
4	2	1	4	6	1	4
5	2	4	4	5	4	4
6	2	16	4.5	4	17	4.5
7	2	73	4.2	3	77	4.4
8	2	307	–	2	339	–
ToTlevels: Tets=1436137						

The error of computing equation 8.1 goes down fast and bounded by $\mathcal{O}(h^{-p})$ where p is the order of expansion. Also, the work complexity time and memory usage down to $\mathcal{O}(h^{-2})$ where h is diameter of a tetrahedron with a distance from boundary. In another words, They (memory usage and time of complexity) are quadrupled See Table 8.8.

Further, we compared the error generated by variable order boundary concentrated Fast Multipole Method with Fixed order BCFMM in all levels. As we mentioned earlier in this section, using variable order could increase the cost of complexity by logarithmical factor. However, since we store matrices $\mathcal{K}(\omega, \omega\prime)$ for the near field elements, and the matrices $Q(\omega)$ and $U(\omega)$ of the proposed algorithm, while all translation process MtM, MtL, and LtL are

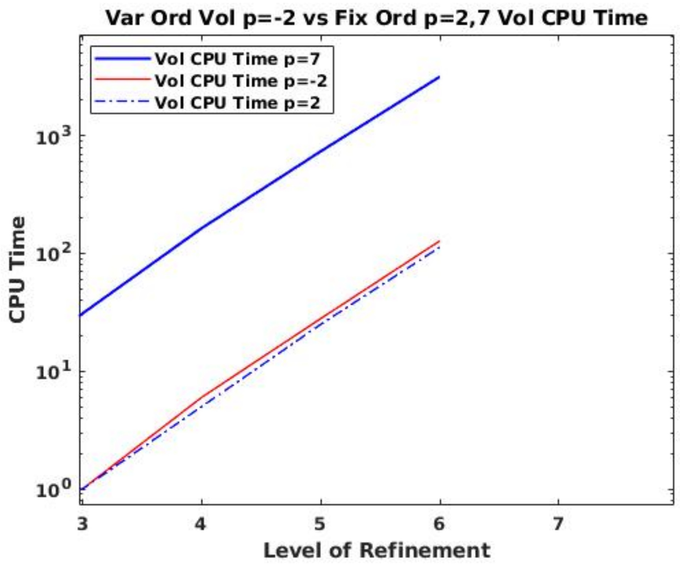


Figure 8.3: Comparing of the work complexity of the direct and the BCFMM method

Table 8.8: Error of computing the equation (8.1) with BCFMM where separation ratio is $\eta = \frac{1}{2}$ and the order of moment is $p = 2$ for all level of refinements.

The Error, CPU Time, and Memory Usage/MB					
Level	**Tetra**	**SetupTime**	**Volume**	**TetStorage/MB**	**Error**
2	64	0	< 1	1.78e-02	0.106291
3	512	0	< 1	1.43e-01	0.00718122
4	4096	2	1	1.14e+00	0.000435992
5	32736	19	4	9.14e+00	3.84577e-05
6	224736	156	16	6.40e+01	7.75485e-06
7	1173984	871	73	3.51e+02	1.91506e-06
8	5333472	2690	0307	1.65e+03	4.76851e-07
ToTlevels: tets=6769609, Separation Ratio=0.5, Mom Order=2					

computed on the fly, the significant increases in the work complexity due to the usage of variable order does not effect overall work complexity. But, this process of variable order is helpful to increase the work accuracy, and it can help to avoid the defection that may cause by the process of ignoring interior elements during the procedure of domain refinement via boundary concentrated subdivision. This can be observed from the error generated by both fixed and variable order see Table 8.9 and Figure 8.4. In the Figure 8.4 we compared error of variable order of expansion $p \in \{2 \dots 2 + L\}$ with error of fixed order $p = 2$ and $p = 7$. As a result, we observe that the error of the variable order is comparable to the error of $p = 7$; however, the cost of using $p = 7$ is very expensive than variable order Figure 8.3. This is another advantage of using lower variable order of expansion on fixed order.

Table 8.9: Comparing the work complexity and error generated by variable and fixed order of expansion in the BCFMM with separation ratio $\eta = \frac{1}{2}$

Variable and Fixed Order CPU Time						
Level	**FixOrder**	**VolTime**	**Error**	**VarOrder**	**VolTime**	**Error**
2	2	< 1	0.106291	8	< 1	0.106709
3	2	< 1	0.00718122	7	< 1	0.00718929
4	2	1	0.000435992	6	1	0.000438031
5	2	4	3.84577e-05	5	4	2.85974e-05
6	2	16	7.75485e-06	4	17	2.75146e-06
7	2	73	1.91506e-06	3	77	4.20676e-07
8	2	307	4.76851e-07	2	339	7.32864e-08
Number boundary faces=65536, boundary vertices=32770						

In the coarse level the BCFMM translations that need to be computed decrease exponentially. In this case higher order translation will not increase the work complexity because of the lower degree of freedom at these levels see Figure 8.5.

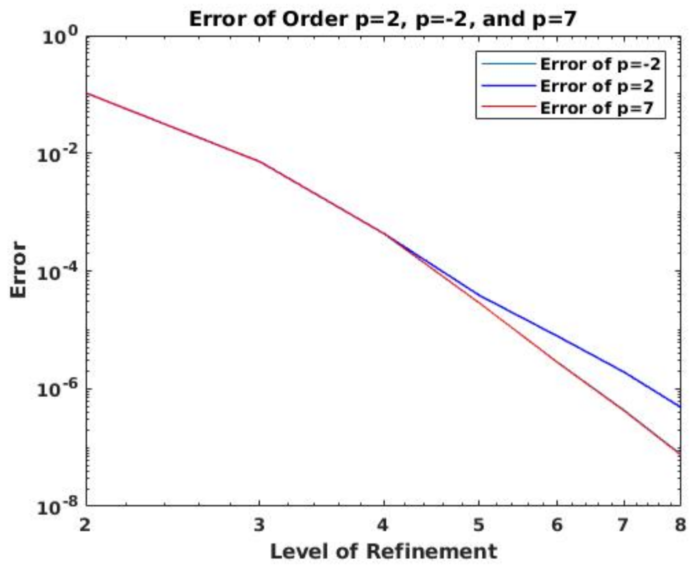


Figure 8.4: Comparing the error generated by variable order of expansion at the variable order with the one generated by fixed order at $p = 2$ and $p = 7$

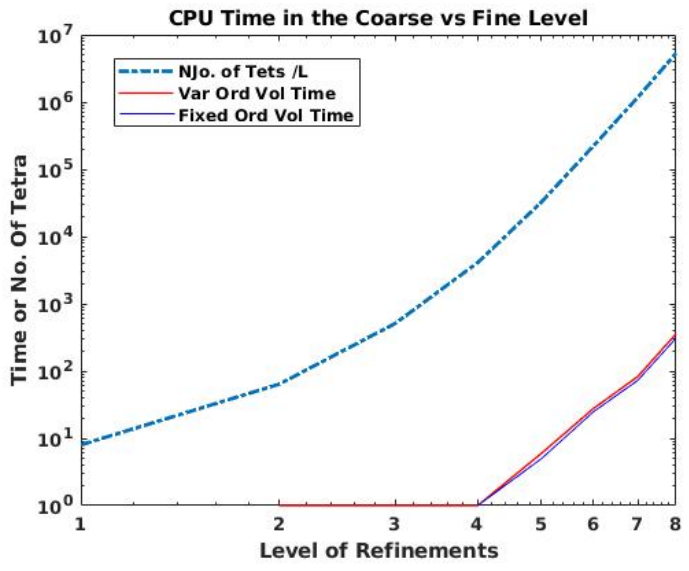
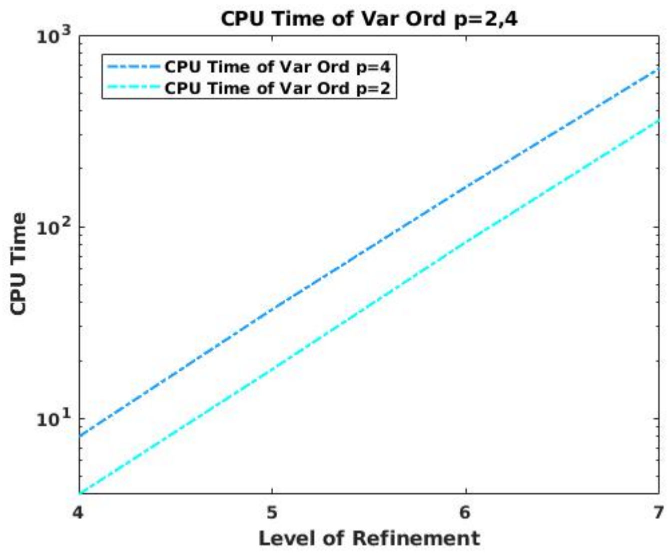


Figure 8.5: CPU time for computing volume in the coarse level decreases exponentially,it increases in finest L.

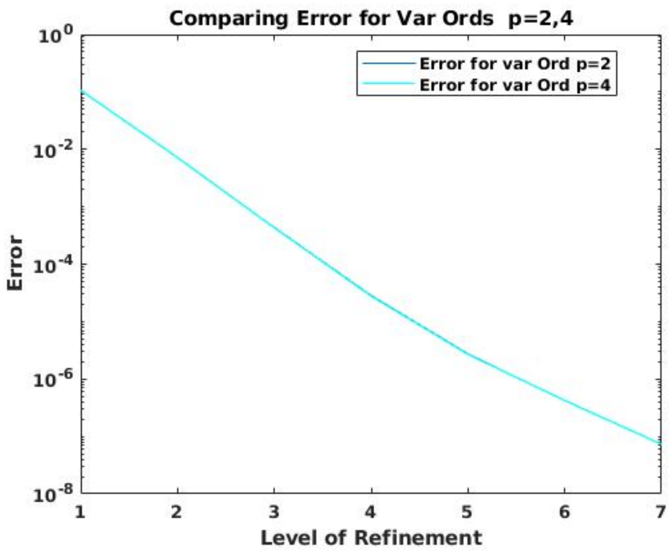
It is good to know which order of expansion should be used in the variable order BCFMM algorithm in the finest level. In fact, the results are shown in the Figures 8.6a, 8.6b, 8.7a, and 8.7b confirm that using high order of expansion $\mathcal{O}(Expansion) > 3$ in the finest level will duplicate the work complexity and increase the CPU time even though it gives the same accuracy as lower order. Therefore, it is recommended to start with low order of expansion in the finest level, such as $p = 2, 3$. This will give the same rate of Error as order $p > 3$. That is besides the point that the cost of work complexity at lower order $p = 2, 3$ is cheap and reasonable comparing to the fixed or higher variable order of expansion in the finest level during the translations processes MtM, MtL, and LtL. Also, to be more accurate in the Figures 8.6 and 8.7 if we compare the error and work complexity for all fixed and variable order of expansion, we can observe that using variable order at finest level of refinement at $p = 2$ is the best option due to the lower work complexity and and error of computing the volume potential that arise in the numerical test problem (8.1).

Also, it turns out that the variable order of expansion in this proposed version of BCFMM can lead to better accuracy even in computing surface integrals. In this case, if the source function $f = 0$ in the equation 8.1, the Newton potential does not have to be computed and only surface integrals remain. In the Figures 8.8a and 8.8b we tested both algorithms, fixed and variable order expansion, and we observed that the cost of computing the surface integrals using variable order of expansion $p = 2$ in the finest level, can provide better accuracy and smaller error than the fixed order besides the point that they are cost the same work complexity time.

In addition, we have tested this proposed BCFMM version to a different separation ratio η between a tetrahedron ω and its neighborhood $\omega\prime$. Particularly, we compared the error and work complexity time for computing the volume potential that arise in the integral equation (8.1), for both separation rations $\eta = \frac{1}{2}$ and $\eta = 0.8$. It turns out that the errors are comparable and the work complexity time of $\eta = 0.8$ is less than the complexity time of $\eta = \frac{1}{2}$ see Figure 8.9.

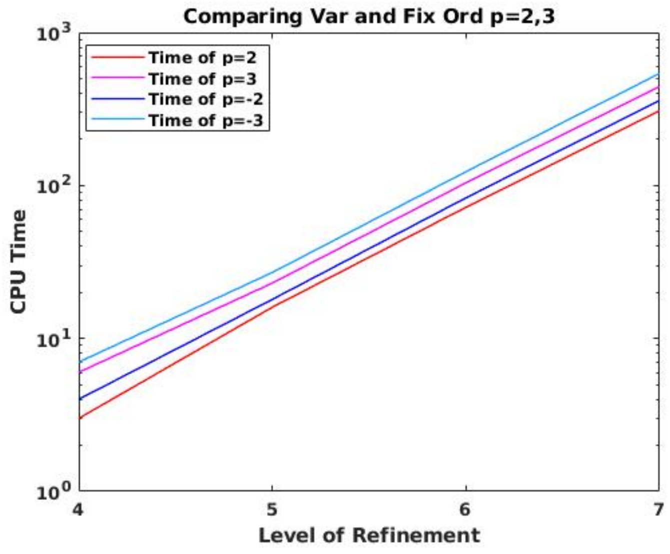


(a) Comparing CPU Times in the test problem 8.1 for two variable order $p = 2, 4$

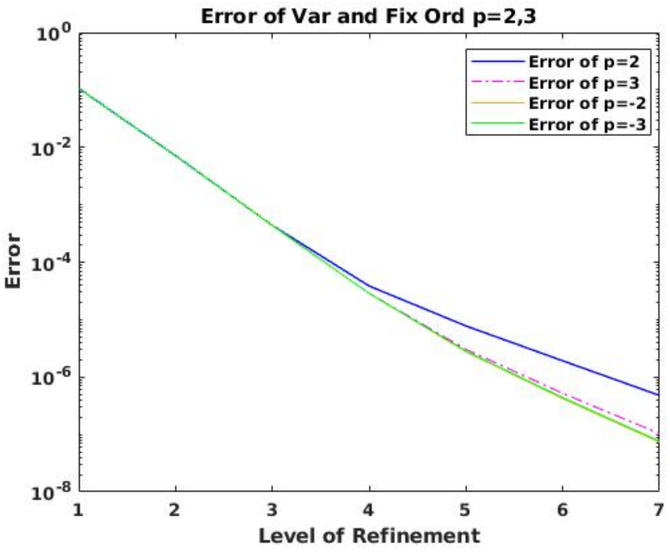


(b) Comparing error in the test problem 8.1 for two variable order $p = 2, 4$

Figure 8.6: Comparing error and work complexity for variable order of expansion $p = 2, 4$

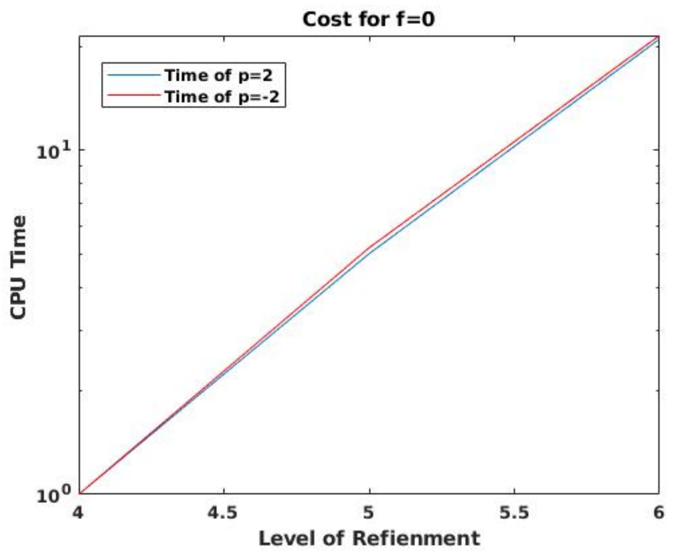


(a) Comparing CPU time for computing volume for variable and fixed order of moment expansion $p = 2, 3$.

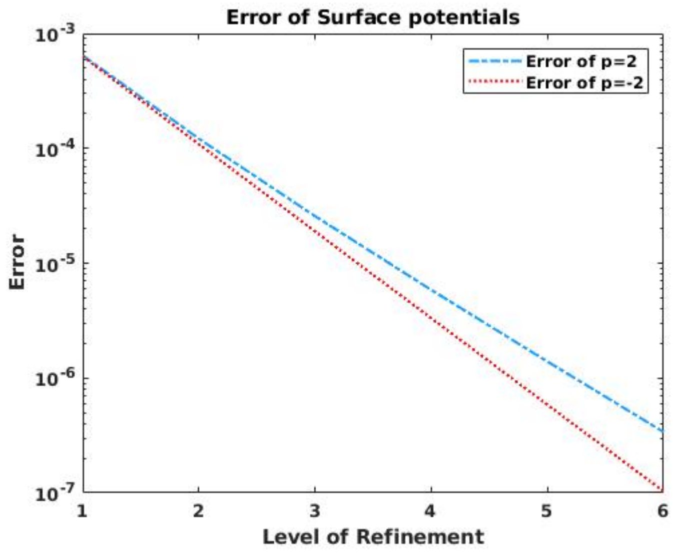


(b) Comparing error of the test problem 8.1 using variable and fixed order $p = 2, 3$

Figure 8.7: Comparing error and CPU time between variable and fixed order of expansion $p = 2, 3$

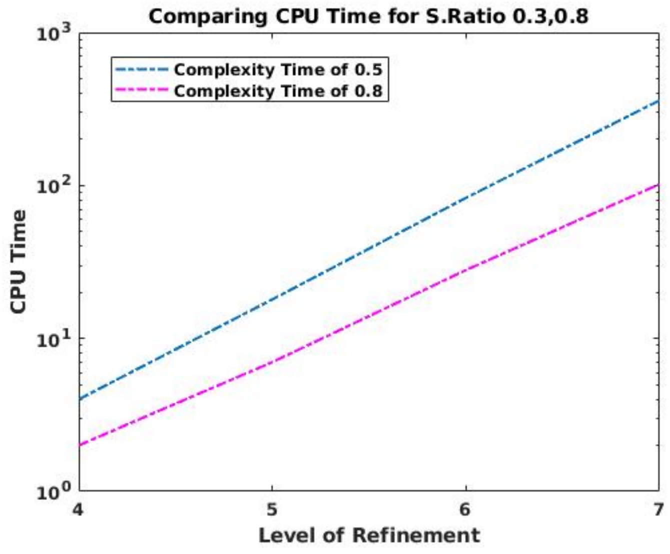


(a) Comparing CPU time of surface potentials of variable and fixed order of expansion $p = 2$

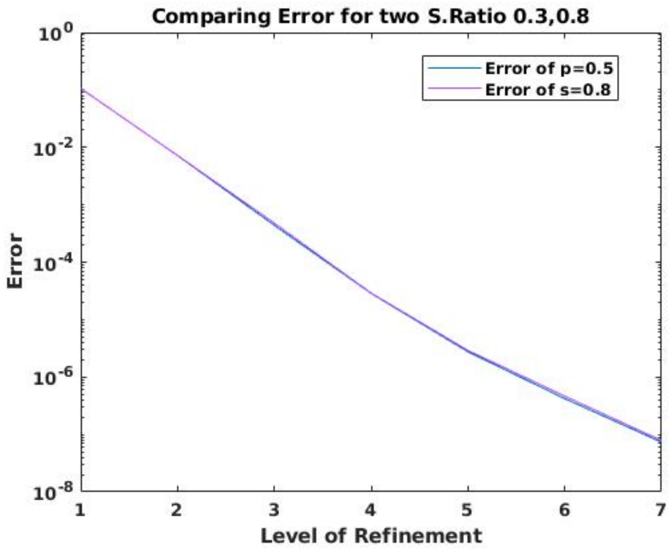


(b) Comparing error of surface potentials of variable and fixed order of expansion $p = 2$

Figure 8.8: Comparing error and CPU time between variable and fixed order of expansion $p = 2$ for $f = 0$



(a) Comparing CPU Time of two different $\eta = 0.5, 0.8$



(b) Comparing Error of the test problem (8.1) of two different $\eta = 0.5, 0.8$

Figure 8.9: Comparing error and CPU time for $\eta = 0.5, 0.8$

8.3. Conclusion

We have discussed and proposed a new algorithm of Fast Multipole Method, which called Boundary Concentrated FMM, BCFMM. This method applied effectively to compute the volume potential that arise in the boundary integral equations. Also, we found that this algorithm worked efficiency to reduce the overall work complexity to $\mathcal{O}(h^{-2})$. In fact, The CPU time and complexity scale linearly very well and the number of degree of freedom in the computational domain has been reduced. Also, we have demonstrated that this algorithm works even for separation ratios larger than expected. The reason for this is that the error estimates for the Taylor series are upper bounds, with much smaller actual error. The boundary concentrated subdivision plays a crucial role to align the volume potential with surface integral efficiency, and it guarantees to obtain high accuracy. In addition, despite the large elements that generated in the computational domain Ω, the variable order of FMM translations that adjusted to the level of the hierarchical domain decomposition of Ω optimizes the accuracy and the CPU time. Further, the variable order bounds the error by $\mathcal{O}(h^{-p})$, whereas the complexity increases by only logarithmically growing factors. Further, the capability of this algorithm can be improved to the vector equations such as the A-field due to currents. But, because of the BEM approaches, we have to restrict things to piecewise homogeneous materials. Moreover, it can be improved to study charge distribution inside complex geometries such as implicit solvent models. This improvement can be one of our future work that we have in mind. Finally, we note that the scope of applications we have in mind in scattering of time harmonic waves from an inhomogeneous medium and thermal equilibrium in semiconductor devices.

REFERENCES

[1] Askham, T., and Cerfon, A. An adaptive fast multipole accelerated Poisson solver for complex geometries. *J. Computational Physics 344* (2017), 1–22.

[2] Atkinson, K. The numerical evaluation of particular solutions for Poisson's equation. *IMA Journal of Numerical Analysis 5*, 3 (1985), 319–338.

[3] Banerjee, P. K. *The Boundary Element Method in Engineering*, 2nd ed. McGraw Hill, 1994.

[4] Banerjee, P. K., and R.Butterfield. *Boundary Element Methods in Engineering Science.* McGraw Hill, 1981.

[5] Beuchler, S., Hofer, K., Wachsmuth, D., and Wurst, J. Multilevel preconditioning for the boundary concentrated hp-FEM. *Comput. Optim. Appl. 62* (2015), 31–65.

[6] Börm, S., Grasedyck, L., and Hackbusch, W. Introduction to hierarchical matrices with applications. *Engrg. Anal. Boundary Elements* (2002), 405 – 422.

[7] Costabel, M. Boundary integral operators on Lipschitz domains: Elementary results. *SIAM J Math Anal 19*, 3 (1988).

[8] Dahmen, W., Harbrecht, H., and Schneider, R. Adaptive methods for boundary integral equations: complexity and convergence estimates. *Math. Comp 76*, 259 (2007), 1243–1274.

[9] Ethridge, F., and Greengard, L. A new fast-multipole accelerated Poisson solver in two dimensions. *SIAM J. Sci. Comput.* (2001), 741–760.

[10] Ethridge, F., and Greengrad, L. A new fast multipole accelerated Poisson solver in two dimensions. *SIAM J. on Sci. Comput. 23(3)*, 3 (2001), 741–760.

[11] Geng, W., and Krasny, R. A treecode accelerated boundary integral Poisson - Boltzmann solver for electrostatics of solvated biomolecules. *J. Computational Physics 247* (2013), 62–78.

[12] Greengard, L. *The Rapid Evaluation of Potential Fields in Particle Systems.* MIT Press, Cambridge, Massachusetts, 1988.

[13] Greengard, L., and Rokhlin, V. A fast algorithm for particle simulations. *J. Computational Physics 73*, 2 (1987), 325–348.

[14] Greengard, L., and Rokhlin, V. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica 6* (1997), 229–269.

[15] Greengrad, L.and Lee, J. A direct adptive Poisson solver of arbitrary order accuracy. *J. Computational Physics 23(3)* (1996), 415–424.

[16] Greengrad, L., Mayo, A., and Mckenney, A. A fast Poisson solver for complex geometries. *J. Computational Physics 118* (1995), 348–355.

[17] Guan, W., Jiang, Y., and Xu, Y. Computing the Newton potential in the boundary integral equation for the Dirichlet problem of the Poisson equation. *J. Intgral Eq. Appl 32*, 3 (2020), 293–324.

[18] Hackbusch, W., and Novak, Z. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math 54* (1989), 463–491.

[19] Helsing, J., and Ojala, R. On the evaluation of layer potential close to their sources. *J. Computational Physics 227*, 5 (2008), 2899–2921.

[20] Hohage, T. Fast numerical solution of the electromagnetic medium scattering problem and applications to the inverse problem. *Journal of Computational Physics 214* (05 2006), 224–238.

[21] Huang, J., and Greengard, L. A fast direct solver for elliptic partial differential equations on adaptively refined meshes. *SIAM J. Sci.Comput. 21*, 4 (1999), 1551–1566.

[22] Kress, R. *Linear Integral Equations*, vol. 82 of *Applied Mathematical Sciences*. Springer, Berlin, Heidelberg, New York, 1989.

[23] Mason, N., and Tausch, J. Quadrature for parabolic Galerkin BEM with moving surfaces. *Computers and Mathematics with Applications 77*, 1 (2019), 1–14.

[24] Mayergoyz, I. D. Solution of the nonlinear Poisson equation of semiconductor device theory. *Journal of Applied Physics 59*, 1 (1986), 195–199.

[25] Mayo, A. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Numer. Anal. 21*, 2 (1984), 285–299.

[26] Mayo, A., and Greenbaum, A. Fast parallel iterative solution of Poisson's and biharmonic equations on irregular regions. *SIAM J. Sci. and Stat. Comput., 13*, 1 (1992), 101–118.

[27] McKenney, A., Greengard, L., and Mayo, A. A fast Poisson solver for complex geometries. *J. Computational Physics 118*, 2 (1995), 348–355.

[28] Pfefferer, J., and Winkler, M. Finite element error estimates for normal derivatives on boundary concentrated meshes. *SIAM J. Numer. Anal. 57*, 5 (2019), 2043–2073.

[29] Popinet, S. A tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Computational Physics 190* (2003), 570–600.

[30] Saranen, J., and Vainikko, G. *Periodic Integral and Pseudodifferential Equations with Numerical Approximation.* Springer, 2002.

[31] Sauter, S., and Schwab, C. *Boundary Element Methods.* Springer, London New York, 2004.

[32] Sauter, S., and Schwab, C. *Randelementmethoden: Analyse, numerik and implementierung schneller algorithmen.* Teubner, Stuttgart, 2004.

[33] Schlick, T. *Molecular Modeling and Simulation: An Interdisciplinary Guide.* Springer, New York, 2010.

[34] Stein, D., Guy, R., and Thomases, B. Immersed boundary smooth extension: A higher order method for solving PDE on arbitrary smooth domains using fourier spectral method. *J. Computational Physics 304* (2016), 252–274.

[35] Steinbach, O. *Numerical Approximation Methods for Elliptic Boundary Value Problems: Finite and Boundary Elements.* Springer, 2008.

[36] Steinbach, O., and Urthaler, P. Fast evaluation of volume potentials in boundary element methods. *SIAM J. Sci. Comput. 22*, 2 (2010), 585–602.

[37] Strauss, H. Nonlinear, three-dimensional magnetohydrodynamics of noncircular tokamaks. *Physic Fluids 19* (1976), 134.

[38] Tausch, J. The fast multipole method for arbitrary Green's functions. In *Current Trends in Scientific Computing*, R. G. Z. Chen and K. Li, Eds., Contemporary Mathematics. American Mathematical Society, 2003, pp. 307–314.

[39] Tausch, J. Sparse BEM for potential theory and Stokes flow using variable order wavelets. *Computational Mechanics 4-6*, 32 (2003), 312–318.

[40] Tausch, J. The variable order fast multipole method for boundary integral equations of the second kind. *Computing 72* (2004), 1–22.

[41] Tausch, J., and White, J. Multiscale bases for the sparse representation of boundary integral operators on complex geometry. *SIAM J. Sci. Comput. 24*, 5 (2003), 1610–1629.

[42] Zhilin, L. *The Immersed interface method: A numerical approach for partial differential equations with interfaces.* PhD thesis, 1994.

[43] Zhou, Y., Zhao, S., Feig, M., and Wei, G. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *J. Computational Physics 213* (2006), 1–30.