FAST FOURIER TRANSFORM ON MULTIPOLES ALGORITHM FOR ELASTICITY AND STOKES FLOW

HE XUEFEI

(*B.Sc., USTC*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2007

Acknowledgements

Many people have helped me in the research during my pursuing PhD degree. First and foremost, I gladly acknowledge my debt to my supervisors, Associate Professor **Lim Siak Piang** and Assistant Professor **Lim Kian Meng**. I would like to thank their invaluable guidance, continuous encouragement and great patient throughout my study. Their influence on me is beyond this thesis and will benefit me in my whole life. I would also like to thank Dr. **Carlos Rosales Fernandez** and Dr. **Chen Puqing** for their help on Linux system and C language. And many thanks are conveyed for all my friends. Lastly, I would especially like to thank my loving wife **Zhang Xiaoshan**, for her unconditional support and constant encouragement.

Contents

Summary

In this thesis, the fast Fourier transform on multipole (FFTM) is used to accelerate the matrix-vector product in the boundary element method (BEM) for solving three dimensional Laplace equation, Navier equation, Stokes equation and non-linear Poisson-type equation. The FFTM method uses multipole moments and local expansions, together with the fast Fourier transform (FFT), to accelerate the far field computation. The FFTM algorithm was initially developed to solve the indirect BEM formulation for the Laplace equation. In this work, a new formulation for handling the double layer kernel using the direct formulation is presented. The FFTM algorithm shows different computational performances in the direct and indirect formulations. These differences are compared and analyzed.

The FFTM algorithm is extended to solve elasticity problems, governed by the Navier equation. The memory requirement of original FFTM algorithm tends to be high. In addition, the Navier equation involves vector quantities, which makes the memory requirement worse. To reduce the memory cost, a new compact storage of the translation matrices is proposed. This reduces the memory usage significantly, allowing large elasticity problems to be solved efficiently. To demonstrate its accuracy and efficiency, the FFTM is compared with the commonly used FMM in terms of efficiency, accuracy and memory cost. Then it is applied to the calculation of the effective Young's modulus of material containing numerous voids.

To extend the FFTM to solve the Stokes equation, the same technique, as that for the Navier equation, is used to derive the translation operators. The resulting multipole translations for Stokes equation are similar to the Navier equation, with the same number of multipole moments and local expansions used, due to the similarity between the boundary integral formulations of the Navier equation and the Stokes equation. In addition, the same compact storage technique for the translation matrices is employed. After it is verified with a simple example, the fast Stokes solver is applied to calculate the average drag force on numerous randomly distributed spherical particles inside a cylinder.

The BEM becomes less attractive when used to solve non-linear equation, because expensive volume integration and evaluation of interior values are involved. In this thesis, the non-linear Poisson-type equation, including a Laplace operator and a non-linear term, is solved by the FFTM. An iterative scheme is used in the fast non-linear solver. In each iteration, a Poisson equation is solved and the interior values are evaluated. To handle the non-homogeneous term in the Poisson equation, two different fast methods are compared. One uses the multipole to accelerate the volume integration, while the other obtains a particular solution through the FFT. The second method is faster and more accurate, and adopted in the fast non-linear algorithm. Several numerical examples are presented to show the improvement in computational efficiency.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Partial differential equation

In mathematics, a partial differential equation is a type of differential equation that involves an unknown function of several independent variables and the partial derivatives with respect to those variables. In this thesis, several important partial differential equations, namely, Laplace equation, Navier equation, Stokes equation and non-linear Poisson-type equation, are investigated with a power tool, fast Fourier transform on multipoles (FFTM).

Laplace equation is a partial differential equation named after its discoverer, Pierre-Simon Laplace. The scalar form of Laplace equation is

$$\nabla^2 \Phi(\mathbf{x}) = 0. \tag{1.1}$$

The partial differential operator, ∇^2, is called the Laplace operator, or just the

Laplacian. The commonly used boundary conditions for the Laplace equation are Dirichlet boundary condition (first-type boundary condition or essential boundary condition) and Neumann boundary condition (second-type boundary condition or natural boundary condition). The Dirichlet boundary condition prescribes the value of Φ on the boundary, while the Neumann boundary condition prescribes the value of $\partial\Phi/\partial\mathbf{n}$. The Laplace equation is important in many areas in science and engineering, such as astronomy and electrostatics.

The elasticity problem is governed by the Navier equation,

$$\frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{1}{1-2\nu}\frac{\partial^2 u_j}{\partial x_i \partial x_j} = 0, \tag{1.2}$$

where u is the displacement, x_i is the spatial coordinates ($i = 1, 2, 3$), and ν is the Poisson ratio. The Navier equation is the equation of equilibrium expressed in terms of displacements. It can be obtained by substituting the stress-strain relationship

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2\mu\varepsilon_{ij} \quad (k = 1, 2, 3) \tag{1.3}$$

into equation of equilibrium

$$\frac{\partial \sigma_{ij}}{\partial x_j} = 0 \tag{1.4}$$

and using the strain-displacement relationship

$$\varepsilon_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}). \tag{1.5}$$

In these equations, σ is the stress, ε is the strain, δ_{ij} is the Kronecker delta, μ is the shear modulus and $\lambda = 2\nu\mu/(1-2\nu)$ is the Lame constant. The Navier equation describes the linear elasticity relationship in solids, which is the starting point for many numerical schemes.

For fluid undergoing Stokes flow, the inertial force in the fluid is small compared to the viscous force, as indicated by the low the Reynolds number ($Re << 1$). In this case, the governing equation for steady Stokes flow is given by

$$\begin{aligned}\mu\frac{\partial^2 u_i(\mathbf{x})}{\partial x_j \partial x_j} &= \frac{\partial P(\mathbf{x})}{\partial x_i}, \quad \mathbf{x} \in \Omega;\\ \frac{\partial u_i(\mathbf{x})}{\partial x_i} &= 0,\end{aligned} \tag{1.6}$$

where u_i is the fluid velocity, x_i is the spatial coordinate ($i = 1, 2, 3$), P is the pressure and μ is the dynamic viscosity. Typically, the Reynolds number is low, when the fluid velocities are very slow, the viscosity is very large, or the length-scales are very small. Such conditions are commonly found in micro fluidic devices or Micro-Electro-Mechanical Systems (MEMS).

In science and engineering, many problems are modeled as the following partial differential equation, such as heat transfer and electrostatics:

$$\nabla^2 u(\mathbf{x}) = f(u). \tag{1.7}$$

If the solution of the equation satisfies both of the following properties, additivity and homogeneity, the equation is a linear equation. Otherwise, it is non-linear equation. Additivity means that if $u_1(\mathbf{x})$ and $u_2(\mathbf{x})$ are both solutions of the equation, then $u_1(\mathbf{x}) + u_2(\mathbf{x})$ must also be a solution. Homogeneity means that if $u_1(\mathbf{x})$ is one solution, then $\alpha u_1(\mathbf{x})$ (where α is a constant) is also one solution. These two rules, taken together, are often referred to as the principle of superposition. Because of the lack of simple superposed solutions, the non-linear equations are more complex and harder to understand and solve than the linear ones.

1.2 Boundary element method (BEM) and fast algorithms

In most cases, it is difficult to obtain an analytical solution of the partial differential equations. Hence, the partial differential equations are normally solved by numerical methods, such as finite difference method (FDM) [30], finite element method (FEM) [69] and boundary element method (BEM) [5]. In contrast with FDM and FEM, both of which need to discreitze the whole computational domain, BEM discretizes only the boundary of the domain. Consequently, the number of degrees of freedom in the problem is decreased, and the difficulties of disretizing the whole domain are avoided. This advantage enables BEM to become popular since the 1980s, and it has been applied successfully in many areas in science and engineering including heat transfer [32], fluid mechanics [68, 67], acoustics [14, 84], electromagnetics [63] and solid mechanics [1].

The critical concept in the BEM is to express the solution of the partial differential equation in terms of boundary distributions of fundamental solutions (also called Green's functions). There are two approaches to the derivation of an integral equation formulation for the partial differential equation. The first is the direct method, and the integral equations are derived through the application of Green's second theorem. The other technique is the indirect method. This is based on the assumption that the solution can be expressed in terms of a source density function defined on the boundary.

The BEM produces a full and asymmetric matrix, which poses challenges in storing the coefficient matrix and solving the linear system for large problems. The memory required for storing the matrix is $O(N^2)$, and the computational time solving the linear system with Gauss elimination method is $O(N^3)$, where N is the number of degrees of freedom. Hence, it is not practical to solve large problem with the traditional BEM owing to the limit of memory and long computational time. The generalized minimal residual (GMRES) method [71] can improve the computational efficiency of solving the linear equations from $O(N^3)$ to $O(N^2)$. In addition, in each GMRES iteration, only the matrix-vector multiplication is needed, so that the storage of the full matrix can be avoided. However, without storage of the matrix, all the coefficients in the matrix are calculated in each GMRES iteration, which means long computational time.

Such matrix-free BEM can be accelerated by performing the matrix-vector multiplication in a faster manner. There are mainly two categories of such fast algorithms. The first one is the fast multipole method (FMM) [70, 24, 52], which uses multipole and local expansions to approximate the source densities that are at places far away from the evaluation point. The efficiency of FMM comes from the effective usage of the multipole and local expansions, which are employed repeatedly in a hierarchical manner through a series of translations. The other algorithm is based on the fast Fourier transform (FFT), and the most popular is the precorrected-FFT (pFFT) introduced by Philips and White [62]. This method approximates a given distribution of charges by an equivalent system of smoothed source distribution that falls on a regular grid. Subsequently, the potential at the grid points produced by the smoothed source distribution is derived by discrete convolution, which can be done rapidly using FFT algorithms. Recently, Ong et al. [56, 59] introduced an alternative fast algorithm, fast Fourier transform on multipoles (FFTM), that combines the use of the multipole and FFT. The FFTM comes from the observation that potential evaluation using multipole to local expansion translation operator can be expressed as a series of discrete convolutions, where FFT can be employed to evaluate the discrete convolution quickly.

1.3 Objectives of the thesis

The main objectives of this thesis are to review the FFTM algorithm in solving the Laplace equation and to extend the FFTM to solve a larger group of partial differential equations, namely Navier equation, Stokes equation and non-linear Poisson-type equation. When solving the direct and indirect BEM formulation of the Laplace equation with the FFTM, different multipole translations are employed. The different performances are compared and analysed. To extend the FFTM to solve the Navier equation and Stokes equation, with vector variables, the original FFTM needs excessive memory usage. A memory-saving strategy is developed to reduce the memory usage significantly. It is always a tough task to solve non-linear equation with BEM. With the help of the FFTM, an efficient scheme is investigated to solve the non-linear Poisson-type equation.

1.4 Original contributions of the thesis

A new multipole translation is presented to solve the direct BEM formulation of the Laplace equation. The new method has more physical meanings and is related theoretically to the commonly used method. The different performances of the FFTM in solving the direct and indirect BEM formulation of the Laplace equation are compared and investigated.

The FFTM is extended to solve the Navier equation and Stokes equation. A new compact storage of the translation matrices is developed to reduce the memory cost of the original FFTM significantly. Consequently, the FFTM is efficient to solve large practical problems. When solving the Navier equation, the performance of FFTM is compared with the FMM in terms of efficiency, accuracy and memory cost. The fast Navier solver is applied to calculate the effective Young's modulus of a material with many voids. The effects of the number, size, position and shape of the voids are discussed. The fast Stokes solver is employed to compute the average drag force on many randomly distributed spheres inside a cylinder. The influence of the cylinder wall is studied.

To handle the non-homogeneous term of the Poisson equation, two fast methods are compared in terms of efficiency and accuracy. One uses the multipole to accelerate the volume integration, while the other obtains a particular solution through the FFT. Since the second method is better, it is adopted in the new fast scheme. The scheme includes calculating a particular solution with the FFT, solving the resulted Laplace equation with the FFTM and updating the interior values also with the

FFTM.

1.5 Organization of the thesis

In this chapter, a brief introduction of several partial differential equations, BEM and fast algorithms is provided, followed by the objectives, contributions and organization of the thesis.

Chapter 2 gives a literature review on the most commonly used fast algorithms.

Chapter 3 describes the implementations of the FFTM algorithm in solving the direct and indirect BEM formulations of the Laplace equation with various boundary conditions. The comparison of the different performances is illustrated by several numerical examples.

Chapter 4 shows the steps of the implementation of the FFTM algorithm in the Navier equation and gives the details of how to reduce the memory usage of storing the translation matrices. The FFTM is compared with the FMM, and then is applied to an example to calculate the effective Young's modulus of porous materials.

In Chapter 5, the FFTM algorithm is extended to solve the Stokes equation. It gives the detailed derivation of the translations for the direct BEM formulation of Stokes equation. This algorithm is applied to a practical problem, calculating the average drag force on many spherical particles inside a cylinder.

A new numerical scheme is proposed in Chapter 6 to solve the non-linear Poisson-type equation. The FFTM is used to evaluate the interior values and to solve the resulted Laplace equation. The non-homogeneous term is treated by obtaining a particular solution with the FFT. This scheme is verified by solving several equations with different non-homogeneous or non-linear terms.

Finally, some concluding remarks are given in Chapter 7.

Chapter 2

Overview of fast algorithms

2.1 Fast multipole method (FMM)

In 1980s, early fast algorithms, such as the tree algorithm [3, 4], were invented to model the gravitation of N-body problem that is governed by the Laplace equation. They implemented a hierarchical grouping of interactions, so that the number of operations is reduced from $O(N^2)$ to $O(N \log N)$. The hierarchical structure is inherited by the famous fast multipole method (FMM) that was first introduced by Rokhlin to solve the two dimensional Laplace equation [70], and then applied to three dimensional N-body problems with Coulombic potential by Greengard and Rokhlin [24]. The details of this FMM's early version can be found in Greengard's PhD thesis [26]. This original FMM algorithm can be summarised in the following steps:

1. define a hierarchical tree partitioning of the computational domain;
2. accumulate the multipole moments for the far field by a postorder traversal of the hierarchical tree;
3. translate the multipole moments to the local expansions;
4. accumulate the local expansions by a preorder traversal of the tree;
5. evaluate the far field action at the field point using local expansion;
6. add the near field action.

Nabors and White [48] were the first who applied the FMM to engineering applications. They developed FastCap to compute the capacitance of a complicated three dimensional geometry of ideal conductors in a uniform dielectric. Further improvements to FMM were investigated to obtain better performance. Nabors et al. [47] modified the FMM by combining precondition and adaptation to reduce both the computation and memory storage to $O(N)$. The precondition decreases the number of iterations of their iterative solver and the adaptation avoids operations in empty domain. White and Head-Gordon [83] introduced the multipole to Taylor transform operator to yield simpler and more efficient transforms. In [13], the mathematical theory was summarised and extended by Epton and Dembart for the multipole translation operators of the three dimensional Laplace and Helmholtz equations. Subsequently, Wang and LeSar [80] presented an efficient FMM algorithm, using a multipole expansion based on the solid harmonics instead of the more common spherical harmonics to calculate long range interactions in three dimensional Coulombic system. The solid harmonics not only increase the efficiency of FMM, but also lead to more compact translations that makes it easier to derive the multipole translation formulations for more complex kernels, such as those in the Navier equation and Stokes equation. Greengard and Rokhlin [25] presented a new version of the FMM that is based on a diagonal form for translation operators. This extra diagonal translation accelerates the fast algorithm further with higher accuracy. This new version of FMM was further improved by Cheng et al. [8]. They introduced adaptation to the algorithm to handle the non-uniform charge distributions and used a compressed version of the translation operators to reduce the computational time. Ying et al. [86] invented a kernel-independent adaptive FMM that needs the hierarchical structure, but does not require the implementation of multipole expansion of the kernel. The far field evaluations are approximated with the singular value decomposition in two dimension and the FFT in three dimension. Many researchers have solved different problems governed by the Laplace equation with the FMM, such as [75, 53, 90, 76, 17, 55].

The FMM has been extended to solve the Navier equation, which tends to be more complicated as the variables are vector quantities instead of scalar quantities in the Laplace equation. Fu et al. [20] applied the FMM to solve three-dimensional elasticity problems that involve a large number of particles embedded in a binder. They decomposed the original three dimensional elasticity kernels into a set of Laplace kernels, which results in four and twelve sets of multipole moments for the displacement and traction kernels, respectively. Yoshida et al. [89] adopted the solid harmonics, originated in [80], to solve three dimensional elastostatic crack problems, using four sets of multipole moments for both the displacement and traction kernels. Due to the concise form of the solid harmonics, it is very simple to perform derivatives on the multipole and local translations. The work in [89] and its related work [53, 90] were summarised in Yoshida's PhD thesis [88]. Except the harmonics functions, the Taylor series can also be employed in the FMM. Popov and Power [65] solved three dimensional elasticity problems with a Taylor series-based FMM. Later, two different implementations were compared by Popov et al. [66]. Lai and Rodin [37] employed the FMM method in [20] to solve problems involving many cracks imbedded in linearly elastic isotropic solids. Recently, the FMM is adopted to solve composite materials. The solid harmonics in [80] and diagonal translation in [25] was combined by Wang and Yao [79] to solve three dimensional particle-reinforced composites. Liu et al. [40] analyzed fiber-reinforced composites with the FMM based on a rigid-inclusion model, in which the number of degrees of freedom exceeds ten millions.

Some work has been done to implement the FMM to solve the Stokes equation. Sangani and Mo [72] was the first to introduce the FMM to solve Stokes flow problems, in which they approximated the interactions among the particles in suspension mechanics with multipoles. The sources in their application are the suspension particles, but not the elements on the boundary. Due to the similarity of the governing equations of elasticity and Stokes flow, the same method, as solving the elasticity problem, can be used to solve the Stokes flow problem. The Taylor series-based FMM was applied by Gomez and Power [22] to solve two dimensional cavity flow problems. The decomposition method in [20] was also employed to solve Stokes problem by Fu and Rodin [21]. Zinchenko and Davis [92] developed a new FMM algorithm to simulate the interaction among many deformable drops in Stokes fluid. Their algorithm is quite different from the traditional FMM in treating both near and remote interactions. The near field is calculated by multipole expansions, further accelerated by rotational transformation, while the far field is treated by Taylor expansions. Later, they [93] applied this fast algorithm to simulate close interaction of slightly deformable drops. If the Stokes equation is solved with vorticity formulation, the domain integral is needed. Brown et al. [6] accelerated the evaluation of the domain integral with the FMM. In the design of MEMS, the damping force on the structure is evaluated by solving an exterior Stokes problem. Frangi [16] solved such problem with the qualocation mixed-velocity-traction approach accelerated with the FMM. This method was improved by Frangi et al. [18], in which they gave detailed derivations of multipole translations. Wang et al. [78] implemented FMM to solve Stokes problem with the direct BEM formulation. They derived the multipole translations for the two kernels in the direct BEM formulation and gave the same translations as Frangi et al. [18].

Fewer work of fast algorithms has been done on Poisson equation. Ingber [33] proposed that when a volume integration scheme is coupled with FMM, it significantly improves the computational efficiency. The FMM was applied by the Greengard's group [46, 23, 15], providing a series of two dimensional fast Poisson solvers. Some of their work [46, 15] accelerated the volume integration by the FMM, and Greengard and Lee [23] calculated particular solutions with spectral method in a decomposed domain and patches the solutions together with the FMM. Ying et al. [87] handled the non-homogeneous part with particular solution calculated from the FFT, while solving the homogeneous part with the kernel-independent FMM.

2.2 Precorrected-FFT (pFFT)

Another commonly used fast algorithm is the pFFT method [62] that was first introduced to solve the problem of coupled capacitance extraction in complicated three dimensional geometries. The pFFT algorithm represents the long range part of the Coulomb potential by point charges lying on a uniform grid, rather than a series expansions as in FMM. The grid representation allows the FFT to be used to perform potential computations efficiently. Since the calculation using the FFT on the grid does not accurately approximate the nearby interactions, the precorrection is needed to modify the nearby interactions. Since its appearance, the pFFT algorithm has been applied to solve many problems governed by the Laplace equation. It was employed by Newman and Lee [51] and Newman [50] to perform hydrodynamic analysis of very large floating structures. Hu et al. [31] simulated large industrial circuits with up to 121,000 inductors and over 7 billion mutual inductive couplings with the pFFT. Tissari and Rahola [77] adopted the pFFT to accurately localize the brain activity recorded by magnetoencephalography (MEG). The pFFT was improved by Zhu et al. [91] to analyze wide-band electromagnetic effects in very complicated geometries of conductors.

To extend the pFFT to equations with vector variables, such as Navier equation and Stokes equation, the projection procedure is more complex than the Laplace equation. Masters and Ye [45] extended the pFFT to solve the Navier equation and solved coupled three dimensional electrostatic and linear elastic problems. The pFFT was also applied to solve the Stokes equation, evaluating the damping force on the complicated structures in MEMS [81, 9, 82]. In his PhD thesis [81], Wang developed an incompressible FastStokes solver and a compressible FastStokes solver. The compressible solver solves the linearized compressible Stokes equation to capture weak air compression effect in MEMS. With the help of the FastStokes, Ding and Ye [9] compared two slip models in the simulation of rarefied gas flows in MEMS. Recently, the FastStokes was applied to simulate several practical micromachined devices [82].

The pFFT was also implemented for Poisson equation and non-linear equation. Ding et al. [11] introduced a fast cell-based approach, based on the pFFT technique, that accelerates the surface integration as well as the volume integration. Later, the same technique was extended by Ding and Ye [10] to solve some three dimensional weakly non-linear problems, in which the number of freedom reached 4000.

2.3 Fast Fourier transform on multipoles (FFTM)

Since the FMM and pFFT have advantages and drawbacks in different aspects, some researchers tried to retain the benefits of both methods by combining the two methods. One combination is particle-particle-particle-mesh/ multipole-expansion (PPPM/MPE) method that was developed for the bio-molecular simulations by

Shimada et al. [73, 74]. This method was not further exploited mainly due to its expensive memory usage. Elliott and Board [12] proposed another combined method, which performs the FFT to accelerate the multipole and local translations. In their method, the convolution variables are the indexes of the translation operators. Yet, this method becomes unstable numerically for high expansion order.

Recently, Ong et al. [56, 59] introduced a new combined fast algorithm, fast Fourier transform on multipoles (FFTM). In 2004, the FFTM was introduced for three-dimensional electrostatics analysis [56]. This fast algorithm uses the FFT to rapidly evaluate the discrete convolutions in potential calculations via multipole expansions. After the potentials at the cell centers are computed, the potentials at other desired locations are obtained by interpolation. But such interpolation procedure brings extra errors. To resolve this problem, local expansion was introduced to calculate the three-dimensional potential fields more accurately [59]. This improvement comes from the observation that potential evaluation using multipole to local expansion translation operator can be expressed as a series of discrete convolutions, where the FFT can be employed to evaluate the discrete convolution very fast. This new fast algorithm is different from Elliott and Board's method in that the convolution variables in the FFTM are the spatial coordinates of the source and field points instead of being the indices of translation operators as in Elliott and Board's method. The FFTM partially resolves the memory storage issue which was present in the method used by Shimada et al. This is achieved through the exploitation of the symmetry relations of the spherical harmonics [59]. Compared with the FMM, the FFTM is easier to implement and is more accurate with a relatively low order of expansion.

Later, Ong et al. [58] showed that the parallel implementation of the FFTM could further accelerate the algorithm with the speedup factor at 5.0-6.4. The FFTM was also applied in acoustics problems by solving the Helmholtz equations [57, 39]. Since the FFTM forgoes the hierarchical structure in the FMM, the wave-number radius criterion has less impact on the FFTM. More over, the FFTM implementation for the Helmholtz equation is rather straightforward compared with the FMM. Recently, the FFTM has also showed its efficiency in solving micromagnetics problems [43, 41, 42] and in modeling multiple bubbles dynamics [7]. Both the magnetostatic field and the inviscid, incompressible and irrotational fluid field can be formulated as a scalar potential field that is governed by the same partial differential equation (Laplace equation) as in electrostatic analysis and potential field calculation. Hence, such problems can be solved by an easy extension of former work of the FFTM. In the work of Bui et al. [7], they found that the FFTM becomes less efficient when dealing with spatially sparse bubble distribution. In order to overcome this deficiency, a new version of FFTM with clustering was proposed, named FFTM Clustering. This new FFTM is as accurate as the original version and the efficiency is less dependent on the distribution of sources in the problem domain.

Up to now, the FFTM only solved two kinds of partial differential equations, the Laplace equation and Helmholtz equation, both of which have well-developed multipole and local translation formulas. Other partial differential equations, such as the Navier equation, Stokes equation and non-linear Poisson-type equation, are also very important in science and engineering problems. In this thesis, the FFTM algorithm is extended to solve the above mentioned partial differential equations.

2.4 Other methods

There are also some methods that exploit the fact that a large part of the dense matrix from the BEM is numerically low rank and apply the singular value decomposition to obtain a sparse representation of the original dense matrix. IES^3 [35] recursively partitions the matrix, and compresses the submatrices with the singular value decomposition. The FFTSVD [2] decomposes the matrix into different length scales. The FFT is used to diagonalize the translation operation that computes the long range interactions.

Chapter 3

Laplace equation

In this chapter, the FFTM is reviewed to solve the Laplace equation with the indirect BEM formulation. Subsequently, the FFTM is implemented in the direct BEM formulation. In order to apply multipole methods in the direct BEM formulation, most researchers [76, 17, 55] used solid harmonics and their derivatives to treat the double layer kernel, following Yoshida's method [88, 52]. Here, an alternative method is introduced for the direct BEM formulation, which is based on the physical interpretation of monopole and dipole sources. Different implementations of the FFTM in the direct and indirect BEM formulations have different influences on the accuracy of the BEM results. The performances are compared and analyzed, showing that the effect of FFTM is secondary to the inherent accuracy of the standard direct and indirect BEM. This means that the FFTM accelerates the computation without much loss of accuracy.

3.1 BEM for Laplace equation

3.1.1 Indirect formulation

The representation of the harmonic potential ($\Phi(\mathbf{x})$) by single-layer potentials is the foundation of the indirect boundary integral equation formulation [34, 67]. The single-layer potential is the potential associated with a continuous distribution of simple sources of density σ extending over the surface S, which is the form

$$\Phi(\mathbf{x}) = \int_S G(\mathbf{x},\mathbf{y})\sigma(\mathbf{y})dS(\mathbf{y}) = \frac{1}{4\pi}\int_S \frac{1}{r(\mathbf{x},\mathbf{y})}\sigma(\mathbf{y})dS(\mathbf{y}), \tag{3.1}$$

where $r(\mathbf{x},\mathbf{y})$ is the distance between the source point $\mathbf{y}$ and the field point $\mathbf{x}$ and $G(\mathbf{x},\mathbf{y})$ is the single layer kernel. The normal derivatives of the single-potential Φ at the point $\mathbf{x}$ for interior problem (denoted by subscript i) and exterior problem (denoted by subscript e) are given by

$$(\frac{\partial\Phi}{\partial\mathbf{n}}(\mathbf{x}))_i = \frac{1}{2}\sigma(\mathbf{x}) + \int_S K(\mathbf{x},\mathbf{y})\sigma(\mathbf{y})dS(\mathbf{y}), \tag{3.2}$$

$$(\frac{\partial\Phi}{\partial\mathbf{n}}(\mathbf{x}))_e = -\frac{1}{2}\sigma(\mathbf{x}) + \int_S K(\mathbf{x},\mathbf{y})\sigma(\mathbf{y})dS(\mathbf{y}), \tag{3.3}$$

respectively, where $\mathbf{n}$ is the outward normal vector, pointing from the interior to the exterior, at the field point, with the derivative of the single layer kernel at the point $\mathbf{x}$ defined as

$$K(\mathbf{x},\mathbf{y}) = \frac{1}{4\pi}\frac{\partial}{\partial\mathbf{n}(\mathbf{x})}(\frac{1}{r}) = \frac{1}{4\pi r^3}(\mathbf{y}-\mathbf{x})\cdot\mathbf{n}(\mathbf{x}). \tag{3.4}$$

If Dirichlet boundary condition ($\Phi(\mathbf{x})$ given) is prescribed at the point $\mathbf{x}$, $\sigma(\mathbf{y})$ is computed from Equation (3.1) and then substituted into Equation (3.2) or (3.3) to obtain the normal derivative. On the other hand, if Neumann boundary condition is given at point $\mathbf{x}$, Equation (3.2) or (3.3) are used to compute $\sigma(\mathbf{y})$ and Equation (3.1) is used to calculate $\Phi(\mathbf{x})$. In order to solve for $\sigma(\mathbf{y})$ with Equation (3.1), (3.2) or (3.3), the boundary integral equation needs to be discretized to obtain a system of linear equations. In this chapter, constant triangular elements with one node at the element center are used. The numerical integration is performed over these elements using local intrinsic coordinates. When $\mathbf{x}$ and $\mathbf{y}$ are on different elements, the standard Gaussian quadrature (with 7 Gauss points over each element) is applied to perform the integration. When $\mathbf{x}$ and $\mathbf{y}$ are on the same element ($\mathbf{x} = \mathbf{y}$), weak ($1/r$) or strong ($1/r^2$) singularities appear. The weak singularity is removed by transforming the triangular elements to a quadrilateral domain on which 7×7 Gauss points are used for Gauss quadrature. For constant element, the integration over the strong singularity is equal to zero due to orthogonality. Although the analytical integration can be employed for the constant planar element to obtain better accuracy, as used by Masters and Ye [45], the more general Gauss quadrature method is chosen for future extension of this algorithm to quadratic element.

The resulting linear system can be solved using Gaussian elimination which takes $O(N^3)$ operations, where N is the number of unknowns. Also, the $N \times N$ matrix has to be constructed explicitly with the use of Gaussian elimination. When N is large, the amount of computational time and memory needed by Gaussian elimination may become exorbitantly large. This can be alleviated by using iterative linear solvers, such as GMRES [71], which typically require $O(N^2)$ operations to solve the linear system, and also provide the possibility of not forming the $N \times N$ matrix explicitly. Within each iteration, only the matrix-vector multiplication needs to be performed, and this corresponds to calculating Φ (Equation (3.1)) or $\partial\Phi/\partial\mathbf{n}$ (Equation (3.2) or (3.3)) at all the node points. Within each iteration, guesses for values of σ are used to calculate Φ or $\partial\Phi/\partial\mathbf{n}$ at all points, and then the differences between the calculated values with the boundary condition values are used by the iterative solver to obtain better guesses for next iteration. This iterative process is repeated until the difference is smaller than a prescribed tolerance.

3.1.2 Direct formulation

The direct boundary element formulation is given by the following integral equation

$$c(\mathbf{x})\Phi(\mathbf{x}) + \int_S H(\mathbf{x},\mathbf{y})\Phi(\mathbf{y})dS(\mathbf{y}) = \int_S G(\mathbf{x},\mathbf{y})\frac{\partial\Phi}{\partial\mathbf{n}}(\mathbf{y})dS(\mathbf{y}). \tag{3.5}$$

Here, $H(\mathbf{x},\mathbf{y})$ corresponds to the double layer kernel

$$H(\mathbf{x},\mathbf{y}) = \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial\mathbf{n}(\mathbf{y})} = \frac{1}{4\pi r^3}(\mathbf{x}-\mathbf{y})\cdot\mathbf{n}(\mathbf{y}). \tag{3.6}$$

where $\mathbf{n}$ is the outward normal vector at the point $\mathbf{y}$. The free term $c(\mathbf{x})$ need not be calculated explicitly in the direct BEM; it can be obtained by physical considerations such as arbitrary shifting of datum in potential problems or arbitrary rigid body motion in mechanics problems. This technique enables the free term and the strongly singular integrals in the direct BEM formulation to be calculated together.

Traditionally, both matrices $[H]$ and $[G]$ are constructed, and these two matrices are rearranged according to the prescribed boundary conditions. The unknowns are gathered on one side of the equation, and the resulting linear system can be solved, by Gaussian elimination or iterative methods, like GMRES. Equation (3.5) can also be solved by iterative methods without forming the matrices explicitly like the indirect method. However, there is a slight difference in the implementation. Now, there are two sets of variables, Φ and $\partial\Phi/\partial\mathbf{n}$; Φ are known on the nodes where Dirichlet boundary conditions are prescribed, and $\partial\Phi/\partial\mathbf{n}$ are known on the nodes where Neumann boundary conditions are prescribed. To handle this, all the unknown quantities are set to zero, and $([G][\frac{\partial\Phi}{\partial\mathbf{n}}]-[H][\Phi])$ is calculated. This result serves as the negative of the right hand side of the linear system that is typically used in the iterative solvers. Subsequently, guesses are made for unknowns, and the expression $([G][\frac{\partial\Phi}{\partial\mathbf{n}}]-[H][\Phi])$ is evaluated again, with the known quantities now set to zero. This procedure allows the conventional iterative solvers, such as GMRES, to be used, just like the case for the indirect BEM. For this direct formulation, two matrix-vector product and one vector subtraction need to be performed within each iteration.

3.2 FFTM for Laplace equation

3.2.1 Indirect formulation

The FFTM has been developed for solving the Laplace equation with Dirichlet boundary conditions [59, 56, 58]. It is used to accelerate the matrix-vector multiplication in the inner loops of GMRES, reducing the operations from $O(N^2)$ to $O(N \log N)$. Here, a brief description of the FFTM algorithm used in the indirect BEM formulation with Dirichlet boundary condition is given before showing its extension to handle Neumann boundary condition. Figure 3.1 illustrates the four steps of FFTM in two dimensions, where M and L stand for multipole moments and local expansions, respectively.

In the first step (A), a rectangular domain enclosing the entire computational domain is defined, and it is divided into numerous regularly spaced cells. The cells separate the contribution of sources as near and far field sources. For a given node, its near field sources consist of those in its own cell and the near cells; the sources in the other cells are considered far field sources. In the present case, the neighbouring cells that share at least one vertex with the node's own cell are considered near cells. There is a possibility of including more layers of neighbouring cells as near cells, as discussed in [56].

In the second step (B) the integral on every element within each cell is converted to multipole moments, following

$$M_n^m(\mathbf{O}) = \frac{1}{4\pi} \int_{S_y} \rho^n Y_n^{-m}(\alpha, \beta)\sigma(\mathbf{y})dS(\mathbf{y}), \tag{3.7}$$

where $\mathbf{O}$ is the multipole cell center, $(\rho,\ \alpha,\ \beta)$ are the spherical coordinates of $\mathbf{y}$ relative to $\mathbf{O}$ and

$$Y_n^m(\alpha, \beta) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}(\cos\alpha)e^{im\beta}. \tag{3.8}$$

Here, the special functions P_n^m are the associated Legendre functions, which can be defined by Rodrigues' formula

$$P_n^m(x) = \frac{(-1)^m}{2^n n!}(1-x^2)^{m/2}\frac{d^{n+m}}{dx^{n+m}}(x^2-1)^n. \tag{3.9}$$

$P_n^m(\cos\alpha)$ are normally evaluated recursively by

$$(n-m)P_n^m(\cos\alpha) = (2n-1)\cos\alpha P_{n-1}^m(\cos\alpha) - (n+m-1)P_{n-2}^m(\cos\alpha), \tag{3.10}$$

for $0 \le m \le n-2$, and

$$P_m^m(\cos\alpha) = \frac{(2m)!}{2^m m!}(-\sin\alpha)^m, \tag{3.11}$$

$$\text{and} \quad P_{m+1}^m(\cos\alpha) = (2m+1)\cos\alpha P_m^m(\cos\alpha), \tag{3.12}$$

for $m \ge 0$.

In the third step (C), the local expansion coefficients at cell centers due to the multipole moments in the far field are evaluated,

$$L_j^k(\mathbf{O}') = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} \frac{i^{|k-m|-|k|-|m|}A_n^m A_j^k Y_{j+n}^{m-k}(\theta,\phi)M_n^m(\mathbf{O})}{(-1)^n A_{j+n}^{m-k} r^{j+n+1}}, \tag{3.13}$$

where $\mathbf{O}'$ is the local expansion cell center, $(r,\ \theta,\ \phi)$ are the spherical coordinates of $\mathbf{O}$ relative to $\mathbf{O}'$ and A_n^m defined by

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}. \tag{3.14}$$

This process can be written as a series of three-dimensional discrete convolutions

$$\begin{aligned} L_j^k(x_1,x_2,x_3) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}[\sum_{x_1'}\sum_{x_2'}\sum_{x_3'} M_n^m(x_1',x_2',x_3') \\ T_{j,n}^{m,k}(x_1-x_1',x_2-x_2',x_3-x_3')], \end{aligned} \tag{3.15}$$

where

$$T_{j,n}^{m,k} = \frac{i^{|k-m|-|k|-|m|}A_n^m A_j^k Y_{j+n}^{m-k}}{(-1)^n A_{j+n}^{m-k} r^{j+n+1}} \tag{3.16}$$

is the response function that relates the multipole moment M_n^m and the local expansion L_j^k, and the indexes (x_1, x_2, x_3) and (x_1', x_2', x_3') denote the locations of the local expansion and the multipole moment respectively. The calculation of the convolution can be accelerated by the FFT. The free software FFTW (Fastest Fourier Transform in the West), provided by Frigo and Johnson [19], is used.

The final step (D) is to compute the potential $\Phi(\mathbf{x})$ at the field node point. The far field's contribution due to the multipoles can be calculated by

$$\Phi(\mathbf{x}) = \sum_{j=1}^{\infty} \sum_{k=-j}^{j} L_j^k(\mathbf{O}') Y_j^k(\theta, \phi) r^j, \tag{3.17}$$

where $(r,\ \theta,\ \phi)$ are the spherical coordinates of $\mathbf{x}$ relative to $\mathbf{O}'$. The near field contributions due to the sources in the near cells (including the cell containing the node) are calculated by the standard BEM technique.

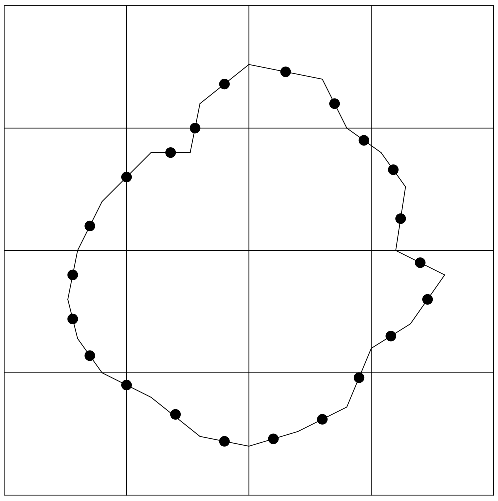
To handle Neumann boundary condition, some modifications to the last step (D) need to be made. Here, the normal derivative, $\partial\Phi/\partial\mathbf{n}(\mathbf{x})$, is obtained by differentiating Equation (3.17). Since this equation is expressed in spherical coordinate, it is convenient to calculate the derivatives of $\frac{\partial\Phi}{\partial r}$, $\frac{1}{r}\frac{\partial\Phi}{\partial\theta}$ and $\frac{1}{r\sin\theta}\frac{\partial\Phi}{\partial\phi}$ as follows.

$$\frac{\partial\Phi}{\partial r} = \sum_{j=1}^{\infty} \sum_{k=-j}^{j} L_j^k Y_j^k(\theta, \phi) j r^{j-1}, \tag{3.18}$$

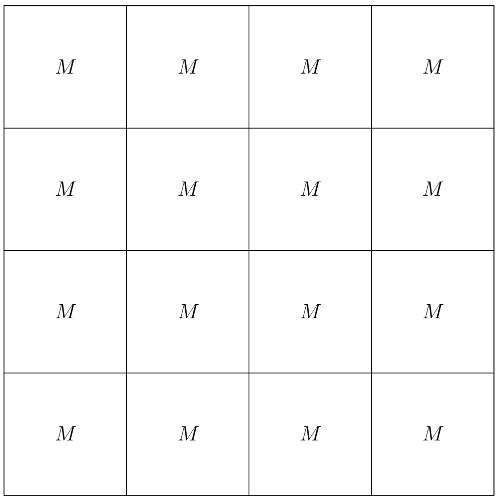
$$\frac{1}{r}\frac{\partial\Phi}{\partial\theta} = \sum_{j=1}^{\infty} \sum_{k=-j}^{j} L_j^k \sqrt{\frac{(j-|k|)!}{(j+|k|)!}} \frac{\partial P_j^{|k|}(\cos\theta)}{\partial\theta} e^{ik\phi} r^{j-1} \tag{3.19}$$

$$\frac{1}{r\sin\theta}\frac{\partial\Phi}{\partial\phi} = \sum_{j=1}^{\infty} \sum_{k=-j}^{j} L_j^k Y_j^k \frac{ik}{\sin\theta} r^{j-1}. \tag{3.20}$$

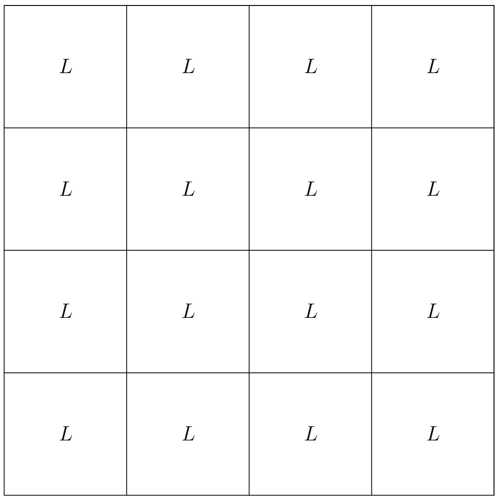
In Equation (3.19), $\frac{\partial P_j^{|k|}(\cos\theta)}{\partial\theta}$ is obtained from differentiating Equations (3.10) to



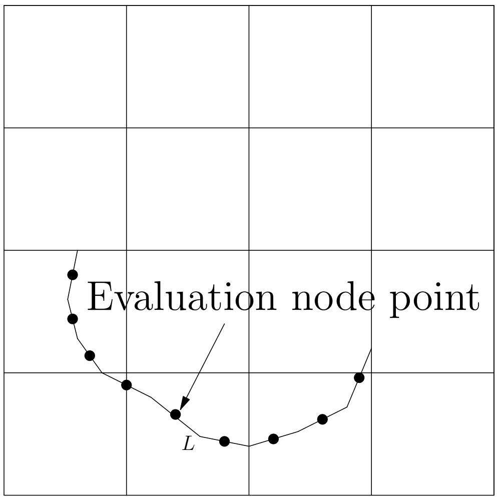
(a) Step A



(b) Step B



(c) Step C



(d) Step D

Figure 3.1: Two-dimensional pictorial representation of FFTM for Laplace equation. Step A: Discretization of domain into cells. Step B: Transformation of sources to multipoles, S2M (S denotes source, monopole, dipole or their combination). Step C: Transformation of multipoles to local expansions, M2L. Step D: Transformation of local expansions to potentials or potential gradients at destinations, L2D (D denotes destination's Φ or $\partial\Phi/\partial\mathbf{n}(\mathbf{x})$)

(3.12). For $0 \leq k \leq j-2$,

$$(j-k)\frac{\partial P_j^k(\cos\theta)}{\partial\theta} = (2j-1)(-\sin\theta)P_{j-1}^k(\cos\theta) + - \\ (2j-1)\cos\theta\frac{\partial P_{j-1}^k(\cos\theta)}{\partial\theta}(j+k-1)\frac{\partial P_{j-2}^k(\cos\theta)}{\partial\theta}. \quad (3.21)$$

For $k = j$,

$$\frac{\partial P_k^k(\cos\theta)}{\partial\theta} = -\frac{(2k)!}{2^k k!}k(-\sin\theta)^{k-1}\cos\theta. \quad (3.22)$$

For $k = j-1$,

$$\frac{\partial P_j^k(\cos\theta)}{\partial\theta} = \frac{\partial P_{k+1}^k(\cos\theta)}{\partial\theta} = -(2k+1)\sin\theta P_k^k(\cos\theta) \\ +(2k+1)\cos\theta\frac{\partial P_k^k(\cos\theta)}{\partial\theta}. \quad (3.23)$$

Lastly, we perform a coordinate transform to obtain

$$\begin{bmatrix} \frac{\partial\Phi}{\partial x_1} \\ \frac{\partial\Phi}{\partial x_2} \\ \frac{\partial\Phi}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \sin\theta\cos\phi & \sin\theta\sin\phi & \cos\theta \\ \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ -\sin\phi & \cos\phi & 0 \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial\Phi}{\partial r} \\ \frac{1}{r}\frac{\partial\Phi}{\partial\theta} \\ \frac{1}{r\sin\theta}\frac{\partial\Phi}{\partial\phi} \end{bmatrix}, \quad (3.24)$$

and the derivative in the normal direction is

$$\frac{\partial\Phi}{\partial\mathbf{n}(\mathbf{x})} = n_1(\mathbf{x})\frac{\partial\Phi}{\partial x_1} + n_2(\mathbf{x})\frac{\partial\Phi}{\partial x_2} + n_3(\mathbf{x})\frac{\partial\Phi}{\partial x_3}. \quad (3.25)$$

For Neumann boundary condition, this value of $\partial\Phi/\partial\mathbf{n}$ is used at the field node, instead of Φ as used in the case of Dirichlet boundary condition.

3.2.2 Direct formulation

The direct formulation involves integrals of the two kernels $G(\mathbf{x},\mathbf{y})$ and $H(\mathbf{x},\mathbf{y})$ as given in Equation (3.5). The double layer kernel $H(\mathbf{x},\mathrm{y})$ is the response function of a dipole source. Hence, it can be considered that a dipole source with strength μ is placed along the direction of the outward normal to the boundary $\mathbf{n}$. The multipole representation of this dipole source is

$$\begin{aligned} M_1^{-1}(\mathbf{y}) &= (-\frac{n_1}{\sqrt{2}} - \frac{n_2}{\sqrt{2}}i)\mu, \\ M_1^0(\mathbf{y}) &= n_3\mu, \\ M_1^1(\mathbf{y}) &= (-\frac{n_1}{\sqrt{2}} + \frac{n_2}{\sqrt{2}}i)\mu, \end{aligned} \tag{3.26}$$

and

$$M_n^m(\mathbf{y}) = 0, \quad \text{for} \quad n \neq 1, \tag{3.27}$$

where n_1, n_2 and n_3 are the components of $\mathbf{n}$. The effect of the single layer kernel $G(\mathbf{x}, \mathbf{y})$, which corresponds to a monopole source q, can be readily included. The sum of these two integrals in Equation (3.5) can thus be represented by the combined multipole representation

$$\begin{bmatrix} M_0^0(\mathbf{y}) \\ M_1^{-1}(\mathbf{y}) \\ M_1^0(\mathbf{y}) \\ M_1^1(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{n_1}{\sqrt{2}} - \frac{n_2}{\sqrt{2}}i \\ 0 & n_3 \\ 0 & -\frac{n_1}{\sqrt{2}} + \frac{n_2}{\sqrt{2}}i \end{bmatrix} \begin{bmatrix} -q \\ \mu \end{bmatrix} \tag{3.28}$$

and

$$M_n^m(\mathbf{y}) = 0, \quad \text{for} \quad n \neq 0, n \neq 1. \tag{3.29}$$

With this multipole representation of the source on the boundary, the multipole moments at the cell centers can be calculated with the multipole translation operation

$$M_j^k(\mathbf{O}) = \sum_{n=0}^{j} \sum_{m=-n}^{n} M_{j-n}^{k-m}(\mathbf{y}) \cdot i^{|k|-|m|-|k-m|} \cdot A_n^m \cdot A_{j-n}^{k-m} \cdot \rho^n \cdot Y_n^{-m} / A_j^k. \tag{3.30}$$

The implementation of FFTM in the direct BEM formulation is again similar to that of the indirect formulation (Section 3.2.1), except for step (B), where a combination of monopole and dipole is transformed to the multipole moment at the cell center in this case, instead of just a single layer source (monopole) in the indirect method.

3.2.3 Alternative formulation

The above translations are based on the spherical harmonics used by Greengard and Rokhlin [24]. Although these formulations appear different from those using solid harmonics, which were given by Wang and LeSar [80] and adopted by Yoshida [88] to handle the derivatives of kernels, the two sets of formulae are identical to each other. The solid harmonics are given by

$$R_n^m(\overrightarrow{O\mathbf{y}}) = \frac{1}{(n+m)!} P_n^m(\cos\alpha) e^{im\beta} \rho^n, \tag{3.31}$$

$$S_n^m(\overrightarrow{O\mathbf{x}}) = (n-m)! P_n^m(\cos\theta) e^{im\phi} \frac{1}{r^{n+1}}. \tag{3.32}$$

where $(r,\ \theta,\ \phi)$ and $(\rho,\ \alpha,\ \beta)$ are the spherical coordinates of two points $\mathbf{x}$ and $\mathbf{y}$, respectively. The definition of associated Legendre function in Equation (3.31) and (3.32) is slightly different from Equation (3.9). For solid harmonics, the term $(-1)^m$ is omitted, giving

$$P_n^m(x) = \frac{1}{2^n n!}(1-x^2)^{m/2} \frac{d^{n+m}}{dx^{n+m}}(x^2-1)^n, \quad m \geq 0, \tag{3.33}$$

and

$$P_n^{-m}(\cos\theta) = (-1)^m \frac{(n-m)!}{(n+m)!} P_n^m(\cos\theta). \tag{3.34}$$

The similarity of the spherical and solid harmonics can be seen from their relation to the distance between two points $\mathbf{x}$ and $\mathbf{y}$:

$$\begin{aligned}\frac{1}{|\mathbf{x}-\mathbf{y}|} &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n} Y_n^m(\theta,\phi)Y_n^{-m}(\alpha,\beta)\frac{\rho^n}{r^{n+1}} \\ &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n} R_n^m(\overrightarrow{O\mathbf{y}})\overline{S_n^m(\overrightarrow{O\mathbf{x}})}. \end{aligned} \tag{3.35}$$

The solid harmonics R_n^m and $\overline{S_n^m}$ correspond directly to the spherical harmonics $\rho^n Y_n^m$ and $\frac{Y_n^{-m}}{r^{n+1}}$, respectively. Since the two sets of translations are the same, the new method can also be expressed in terms of solid harmonics to compare with Yoshida's formulation. The formulations can be rewritten in solid harmonics, and the derivatives in Equation (3.24) are given by

$$\begin{aligned}\frac{\partial\Phi}{\partial x_1} &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{\partial R_n^m}{\partial x_1}L_n^m(\mathbf{O}'), \\ \frac{\partial\Phi}{\partial x_2} &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{\partial R_n^m}{\partial x_2}L_n^m(\mathbf{O}'), \\ \frac{\partial\Phi}{\partial x_3} &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\frac{\partial R_n^m}{\partial x_3}L_n^m(\mathbf{O}'), \end{aligned} \tag{3.36}$$

where

$$\begin{aligned}\frac{\partial R_n^m}{\partial x_1} &= \frac{1}{2}(R_{n-1}^{m-1}-R_{n-1}^{m+1}), \\ \frac{\partial R_n^m}{\partial x_2} &= \frac{i}{2}(R_{n-1}^{m-1}+R_{n-1}^{m+1}), \\ \frac{\partial R_n^m}{\partial x_3} &= R_{n-1}^m. \end{aligned} \tag{3.37}$$

which are the same with those from [88].

Also, the multipole to multipole translation written in solid harmonics is

$$M_j^k(\mathbf{O}) = \sum_{n=0}^{j}\sum_{m=-n}^{n} R_n^m M_{j-n}^{k-m}(\mathbf{y}). \tag{3.38}$$

When this is used for the translation of a dipole whose representation is

$$\begin{aligned} M_1^{-1}(\mathbf{y}) &= \frac{\mu}{2}(-n_1+n_2 i), \\ M_1^{0}(\mathbf{y}) &= n_3\mu, \\ M_1^{1}(\mathbf{y}) &= \frac{\mu}{2}(n_1+n_2 i), \end{aligned} \tag{3.39}$$

the lower index of $M_{j-n}^{k-m}(\mathbf{y})$ in Equation (3.38) is limited to $j-n=1$, and the upper index $k-m=-1,0,$ or 1. Therefore, the dipole to multipole translation operator is

$$\begin{aligned} M_j^k(\mathbf{O}) &= R_{j-1}^{k+1}M_{-1}^{1}(\mathbf{y}) + R_{j-1}^{k+1}M_{0}^{1}(\mathbf{y}) + R_{j-1}^{k-1}M_{1}^{1}(\mathbf{y}) \\ &= \frac{\mu}{2}(-n_1+n_2 i)R_{j-1}^{k+1} + n_3\mu R_{j-1}^{k} + \frac{\mu}{2}(n_1+n_2 i)R_{j-1}^{k-1} \\ &= \frac{1}{2}(R_{j-1}^{k-1}-R_{j-1}^{k+1})n_1\mu + \frac{i}{2}(R_{j-1}^{k+1}+R_{j-1}^{k+1})n_2\mu + R_{j-1}^{k}n_3\mu. \end{aligned} \tag{3.40}$$

Using the relations given in Equation (3.37), the following equation can be obtained,

$$M_j^k(\mathbf{O}) = (\frac{\partial R_j^k}{\partial x_1}n_1 + \frac{\partial R_j^k}{\partial x_2}n_2 + \frac{\partial R_j^k}{\partial x_3}n_3)\mu = \frac{\partial R_j^k}{\partial \mathbf{n}(\mathbf{y})}\mu, \tag{3.41}$$

where n_1, n_2 and n_3 are the direction of the dipole, and also the outward normal direction at the point $\mathbf{y}$. The result in Equation (3.41) is exactly the same as the formulation that Yoshida used to handle the kernel $H(\mathbf{x},\mathbf{y})$. Here, the formulation is obtained from a physical interpretation of the dipole associated with the kernel $H(\mathbf{x},\mathbf{y})$.

In contrast, Yoshida [88] obtained exactly the same formulation by differentiating Equation (3.35). For a double layer kernel that is associated with a dipole of source strength μ, the potential is given by

$$\mu\frac{\partial}{\partial \mathbf{n}(\mathbf{y})}\frac{1}{|\mathbf{x}-\mathbf{y}|} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\mu\frac{\partial}{\partial \mathbf{n}(\mathbf{y})}R_n^m(\overrightarrow{O\mathbf{y}})\overline{S_n^m(\overrightarrow{O\mathbf{x}})}. \tag{3.42}$$

Hence, the multipole (the coefficients of $\overline{S_n^m(\overrightarrow{O\mathbf{x}})}$), associated with the double layer kernel, are given as

$$M_j^k(\mathbf{O}) = \frac{\partial R_j^k}{\partial \mathbf{n}(\mathbf{y})}\mu, \tag{3.43}$$

similar with Equation (3.41).

3.3 Numerical examples

3.3.1 Accuracy of translation operators

Here, the accuracy of the translations is investigated. First, the new method of handling the kernels $H(\mathbf{x}, \mathbf{y})$ is compared with Yoshida's method. A source consisting of a dipole $(d : \mu, \mathbf{n})$ and its combination with a monopole (q), is placed at $\mathbf{y} = (1.02, 1.03, 1.05)$, and then the potential at $\mathbf{x}$ = (0.03,0.04,-0.1) due to the sources is calculated. The exact solution for the two cases $(q = 0, \mu = 1)$ and $(q = -1, \mu = 1)$, with $\mathbf{n}(\mathbf{y}) = (1/2, 1/2, 1/\sqrt{2})$ are given by

$$\begin{aligned} \Phi(\mathbf{x})^d_{exact} &= H(\mathbf{x}, \mathbf{y}) = -0.3032, \\ \Phi(\mathbf{x})^{d+q}_{exact} &= H(\mathbf{x}, \mathbf{y}) - G(\mathbf{x}, \mathbf{y}) = -0.8551, \end{aligned} \tag{3.44}$$

where the permittivity in free space is given by $\epsilon = 1$. The potential is also obtained, using the translation operators $S2M, M2L$ and $L2D$. The centers where the multipoles M and local expansion coefficients L are evaluated are given by $O_M = (1, 1, 1)$ and $O_L = (0, 0, 0)$ respectively. The measure of the error is defined in the L_2 norm as

$$Error = \left(\frac{\sum_{i=1}^{k} |u(\mathbf{x}_i) - u_{exact}(\mathbf{x}_i)|^2}{\sum_{i=1}^{k} |u_{exact}(\mathbf{x}_i)|^2} \right)^{1/2}, \tag{3.45}$$

where u may represent the potential Φ or its derivative $\partial\Phi/\partial\mathbf{n}$. The new method is compared with those of Yoshida's method for various orders of expansion (p), and the results from the two methods are identical (Figure 3.2). It is also noted that the error decreases with higher orders of expansion used.

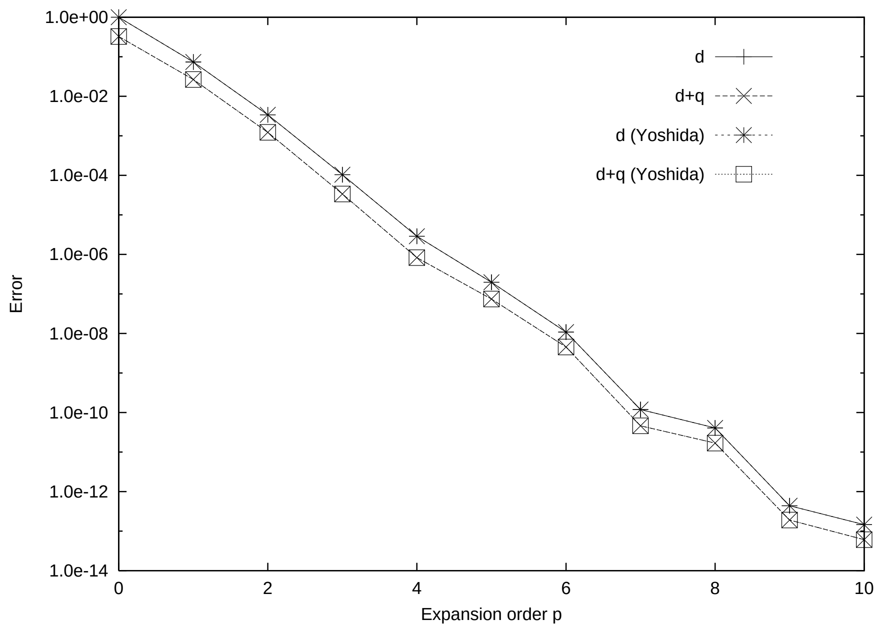


Figure 3.2: Comparison of error between the new method handling the double layer kernel and Yoshida's method for increasing order of multipole expansion.

Next, the accuracy for different translation operators: $q \rightarrow \Phi$, $d \rightarrow \Phi$ and $q \rightarrow \partial\Phi/\partial\mathbf{n}$ are compared. These translations correspond to the different kernels in the direct and indirect BEM formulation (as shown in Table 3.1).

The exact solutions of the translation operators used in the above example are

Table 3.1: Different translations used in the direct and indirect BEM.

Direct formulation	
$q \rightarrow \Phi$	G kernel
$d \rightarrow \Phi$	H kernel
Indirect formulation	
$q \rightarrow \Phi$	G kernel
$q \rightarrow \partial\Phi/\partial\mathbf{n}$	K kernel

given by:

$$\begin{aligned}\Phi(\mathbf{x})^q &= G(\mathbf{x},\mathbf{y}) = 0.5519,\\ \Phi(\mathbf{x})^d &= H(\mathbf{x},\mathbf{y}) = -0.3032,\\ \frac{\partial\Phi}{\partial\mathbf{n}}(\mathbf{x})^q &= K(\mathbf{x},\mathbf{y}) = 0.3032, \qquad (3.46)\end{aligned}$$

where $\mathbf{n}(\mathbf{x}) = (1/2, 1/2, 1/\sqrt{2})$, and the superscript ($q$ or d) denotes the source. Figure 3.3 shows the error of each translation operator for various orders of expansion p. It can be seen that the translation $q \rightarrow \Phi$ is the most accurate, followed by $d \rightarrow \Phi$, and then $q \rightarrow \partial\Phi/\partial\mathbf{n}$.

For a particular order of expansion, the translation $q \rightarrow \Phi$ has one more term than the other two translations, hence it is more accurate. For the dipole case, $d \rightarrow \Phi$, the monopole term is absent. For the calculation of derivative of potential $q \rightarrow \partial\Phi/\partial\mathbf{n}$, the differentiation removes the zeroth order term from the expansion, which can be seen from Equation (3.18) to (3.20), and Equation (3.37). Since the multipole (Step B) and local expansion (Step D) are performed up to a fixed order of expansion ($p = 4$ or 6 typically), there are truncation errors in the series expansion. For $q \rightarrow \Phi$, $(p+1)^2$ terms are employed in the translations, $S2M$ (Step B), $M2L$ (Step C) and $L2D$ (Step D). In contrast, $d \rightarrow \Phi$ gets one term less in the multipole expansion, $(p+1)^2 - 1$ terms; $q \rightarrow \partial\Phi/\partial\mathbf{n}$ only considers $(p+1)^2 - 1$ terms in local expansion. It seems that the errors in both cases should be similar to each other. However, this is not the case for the numerical results in Figure 3.3. The reason is that the reduction by one term occurs at different steps: for $d \rightarrow \Phi$, it occurs in $S2M$ (Step B), and for $q \rightarrow \partial\Phi/\partial\mathbf{n}$, in $L2D$ (Step D). The accuracy of these transforms is also dependent on the distance between the two points of transformation. Hence, it is not surprising that the errors in both cases are different.

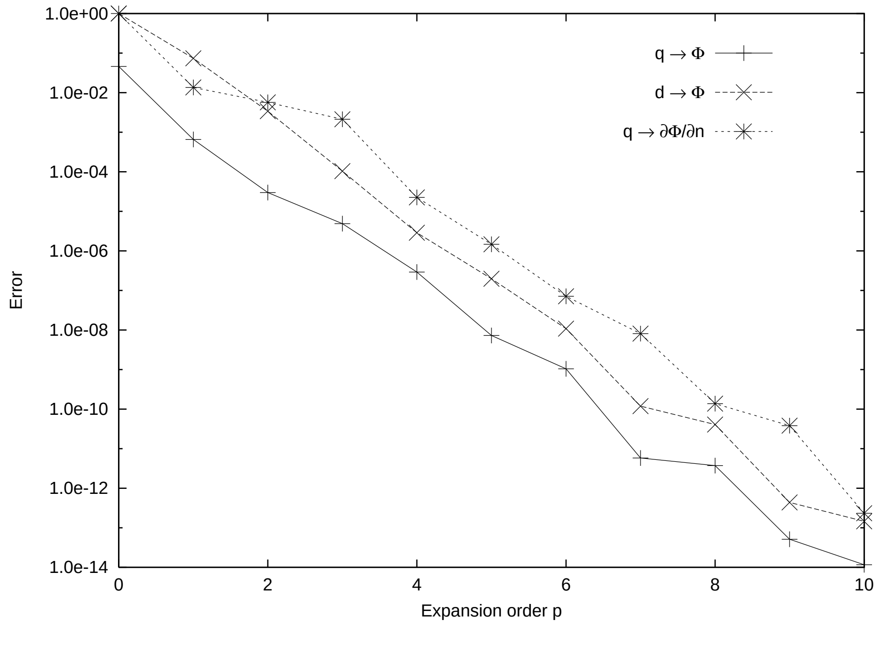


Figure 3.3: Comparison of accuracy for various source to potential/gradient translation operators with increasing order of multipole expansion.

3.3.2 Thermal conduction in a sphere

In this example, the thermal conduction in a sphere is considered. The temperature distribution on the surface of the sphere (with radius 0.5) is given by

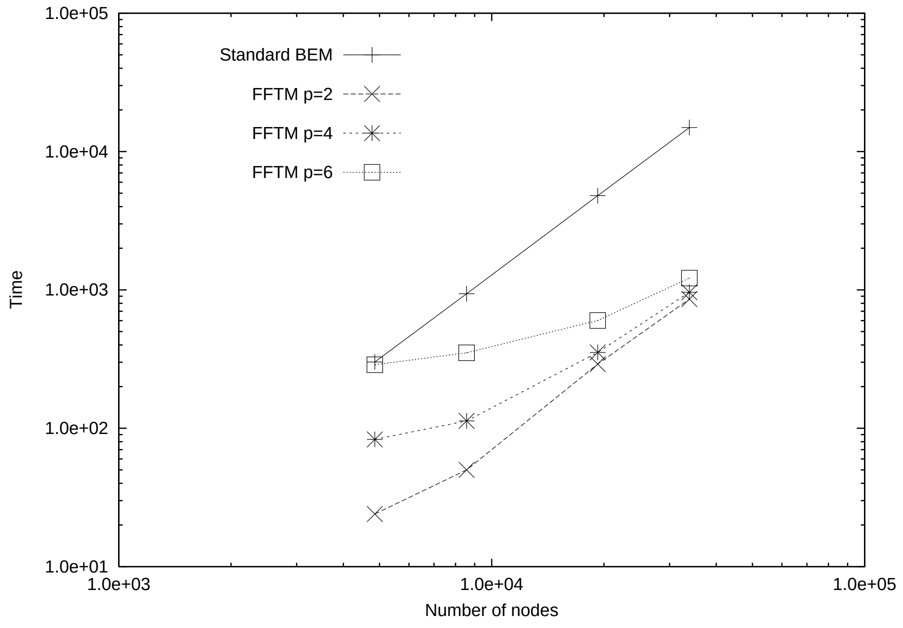
$$\Phi(r,\theta,\phi) = \frac{1}{3} + r^2(\cos^2\theta - \frac{1}{3}) = \frac{1}{4}(\cos^2\theta + 1). \tag{3.47}$$

The spherical coordinates (r,θ,ϕ) are used, and the origin coincides with the center of the sphere. The analytical solution for the temperature gradient is given by

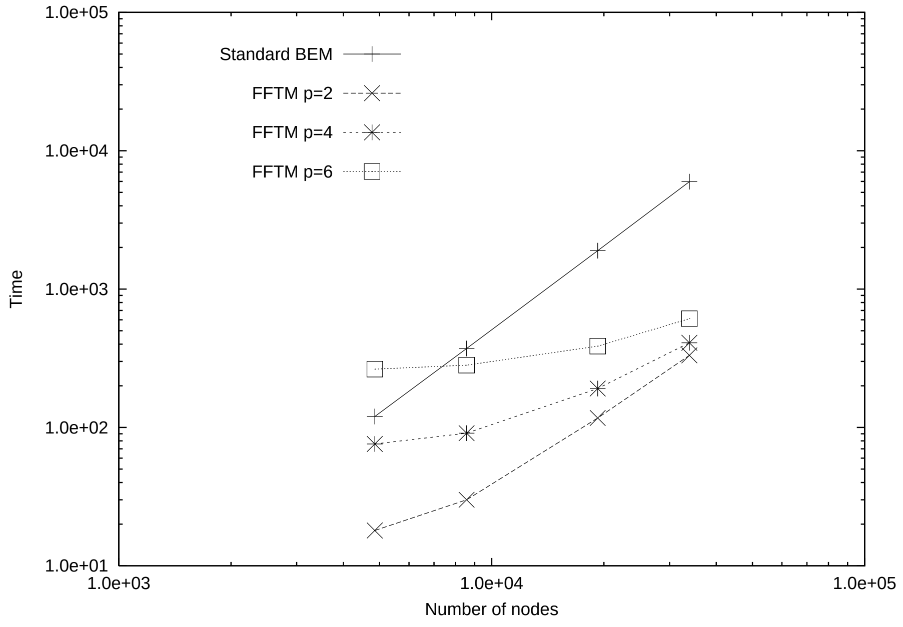
$$\frac{\partial\Phi(r,\theta,\phi)}{\partial\mathbf{n}} = 2r(\cos^2\theta - \frac{1}{3}) = \cos^2\theta - \frac{1}{3}. \tag{3.48}$$

Four different surface discretizations are used, with the total number of nodes being 4858, 8566, 19234, and 33884. To investigate the influence of the parameters of FFTM on the accuracy and computational time, various cell discretizations ($8\times8\times8$, $16\times16\times16$, $24\times24\times24$ and $32\times32\times32$) and orders of expansion ($p=2$, $p=4$ and $p=6$) are tested. The numerical experiments are performed on a computer with AMD Opteron processor (2.2 GHz).

The computational timings for the direct and indirect methods are illustrated in Figures 3.4 and 3.5. It can be seen that the direct method typically takes more time than the indirect method, because the direct method has to handle two kernels while the indirect method handles only one kernel. The FFTM reduces the computational time significantly when the problem becomes large. The standard BEM using GMRES shows the typical computational complexity of $O(N^2)$, while the computational complexity of FFTM is lower, especially when higher order of expansion p and finer cell discretization are used. The FFTM algorithm includes

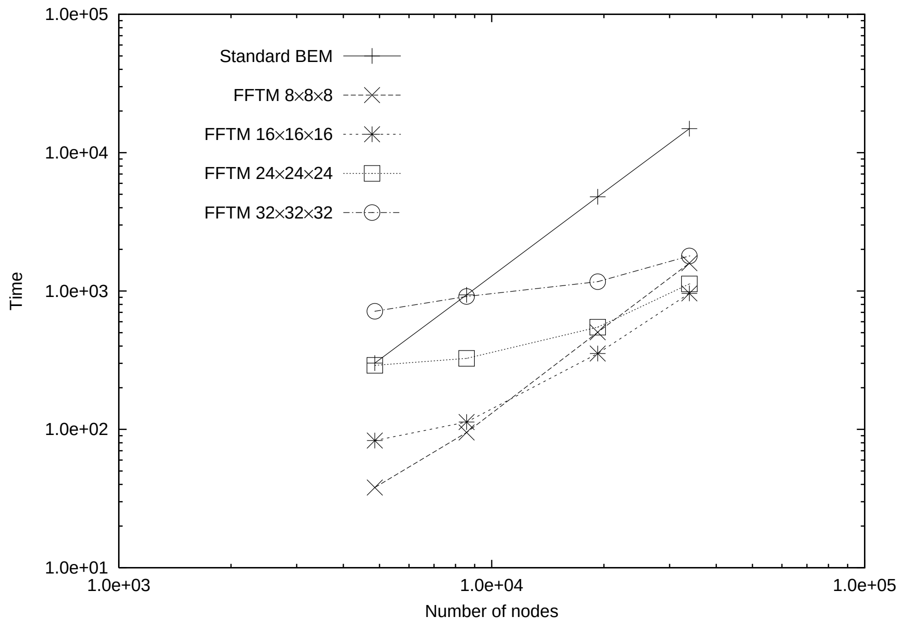


(a) direct formulation

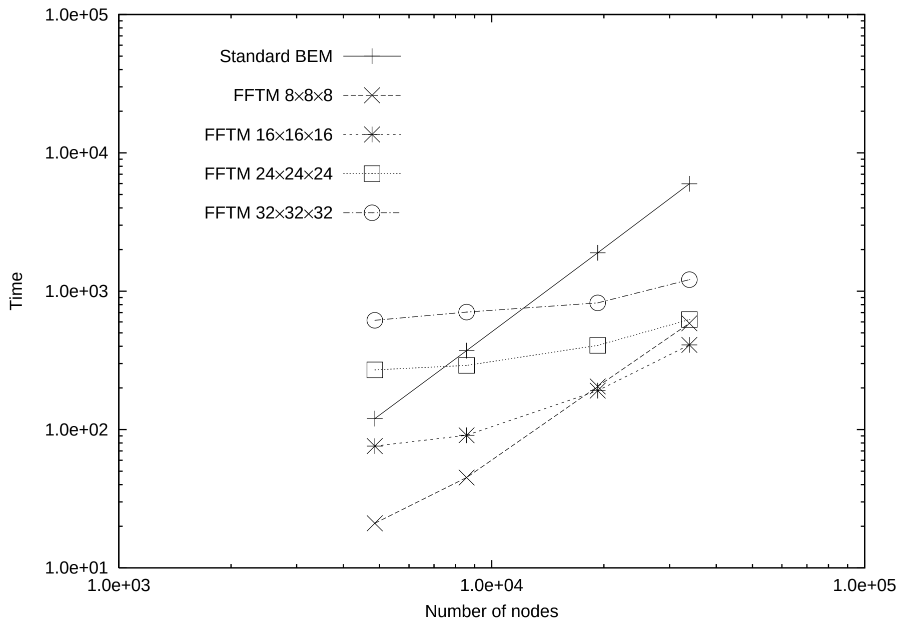


(b) indirect formulation

Figure 3.4: Computational time for various orders of expansion with cell discretization of $16 \times 16 \times 16$ (Dirichlet problem).



(a) direct formulation



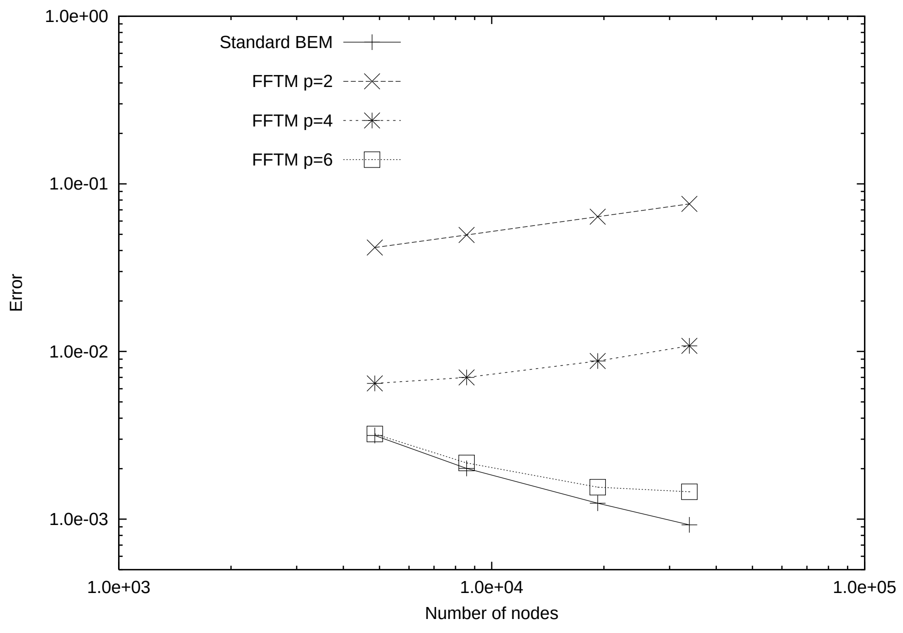
(b) indirect formulation

Figure 3.5: Computational time for various cell discretizations with $p = 4$ (Dirichlet problem).

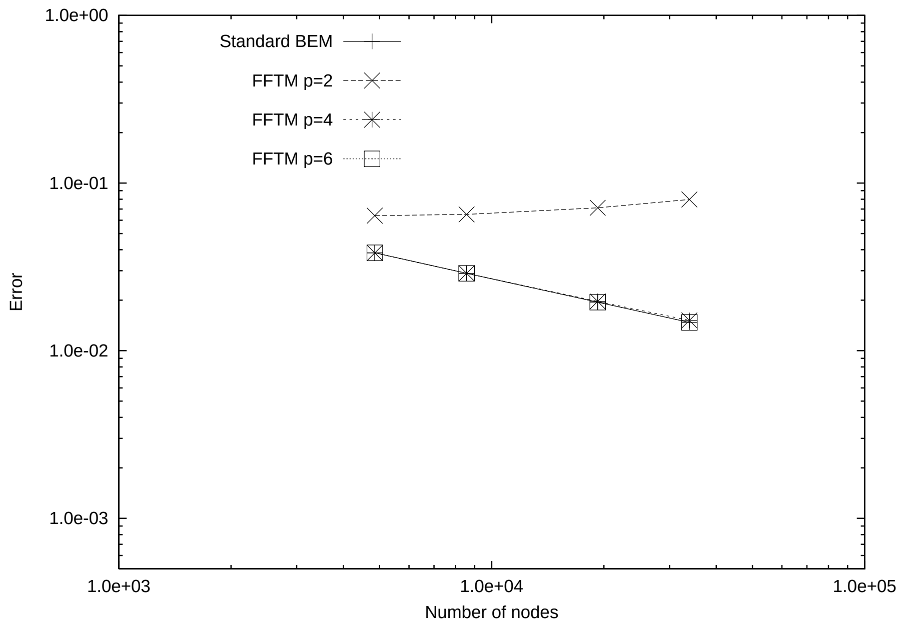
the steps of $S2M$, $M2L$, $L2D$, and calculation of near field, whose computational complexities are $O(N)$, $O(N \log N)$, and $O(N)$, respectively. So the total cost of the FFTM is of the order of $N \log N$. More detailed complexity analysis is given in [59]. The FFTM introduces additional computational overheads, resulting in slightly higher or comparable computational time as the standard BEM for problems with a small number of nodes. However, when the number of nodes increases (to about 40,000), the FFTM reduces the computational time by almost one order of magnitude, compared to the standard BEM.

When a higher order of expansion is used, more operations are needed, and the computation time is longer, as shown in Figure 3.4. However, for large problems, the time increase is not significant compared with the total computational time, as the computational complexity decreases with the order of expansion p. Figure 3.5 shows that the computational complexity decreases with finer cell discretization. But a finer cell discretization also results in higher overhead, and thus this may not improve the computational efficiency when the problem size is small.

Figures 3.6 and 3.7 show the error of the results obtained by standard BEM and FFTM with the exact solutions given in Equation (3.48). The errors of the direct formulation is almost one order of magnitude lower than those of the indirect formulation, since the strongly singular integral can be calculated more accurately in the direct method. In the indirect method, evaluation of the integral involving the K kernel is less accurate. For the direct method, as the order of expansion p goes from 2 to 4, and 4 to 6, there is a significant improvement in accuracy (Figure 3.6). When $p = 6$ is used, it is very close to the standard BEM, for the panel

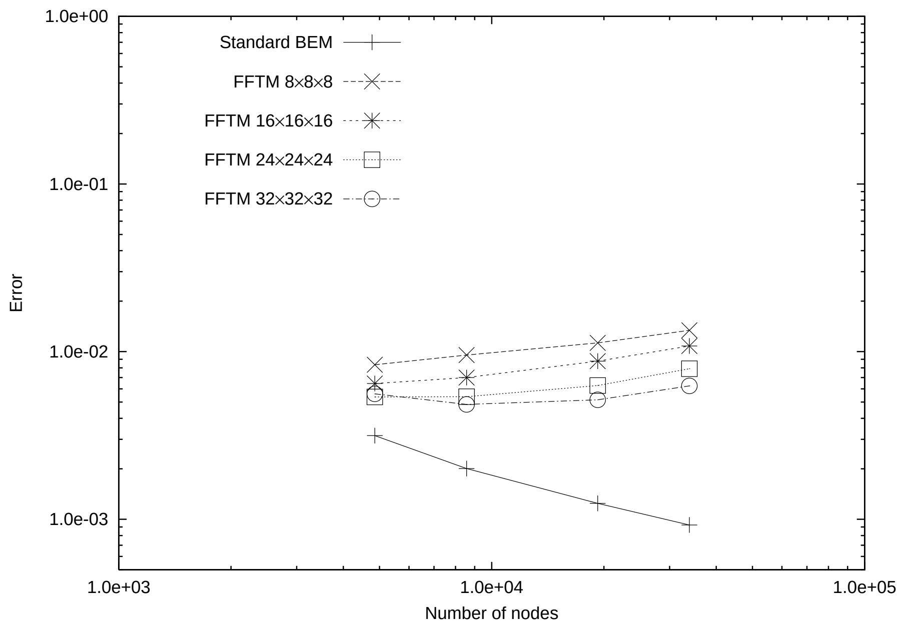


(a) direct formulation

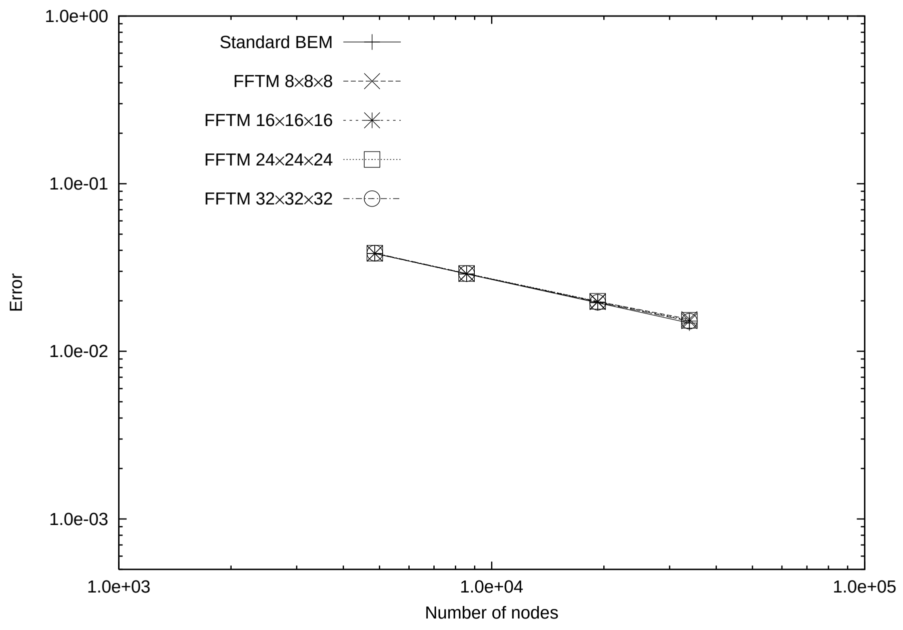


(b) indirect formulation

Figure 3.6: Error for various orders of expansion with cells discretization of $16 \times 16 \times 16$ (Dirichlet problem).



(a) direct formulation



(b) indirect formulation

Figure 3.7: Error for various cell discretizations with $p = 4$ (Dirichlet problem).

discretizations used. For the indirect method, $p = 2$ gives a large error, but $p = 4$ and $p = 6$ give errors that are close to the standard BEM. It is noted that the indirect BEM tends to be less accurate than the direct BEM, hence a lower order of expansion, like $p = 4$ instead of $p = 6$, may be sufficient for the FFTM to achieve the accuracy of the standard BEM. Also, when the number of panels used in the BEM is increased, a higher order of expansion p, for a fixed cell discretization, is normally needed to maintain an accuracy close to the standard BEM.

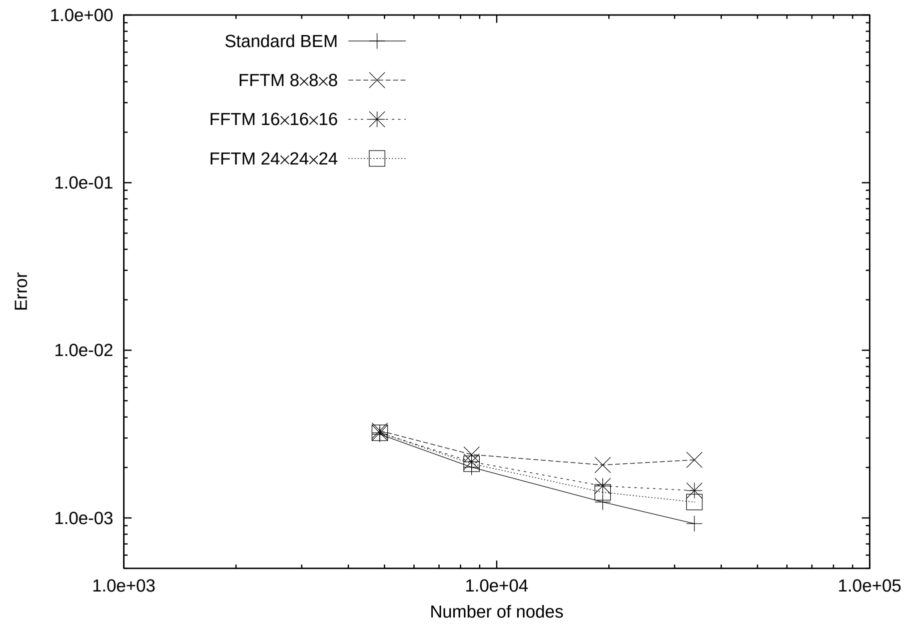


Figure 3.8: Error for various cell discretizations with $p = 6$ using the direct method (Dirichlet problem).

Figure 3.7 illustrates the effects of cell discretization on the accuracy of FFTM (with $p = 4$). For the indirect method, there is no improvement since the standard BEM is not accurate to begin with. For the direct method, the error generally decreases with finer cell discretization, but the improvement is slow – the lines bunch together as finer cell discretization is used. In fact, the error increases with the number of panels used, which is on contrary with the standard BEM. This is because the accuracy is not only dependent on the number of panels, but also the cell discretization and expansion order. Since the number of cells is fixed in each case in Figure 3.7, increasing the number of panels results in accumulating more truncation error for a given multipole expansion in each cell. Consequently, using more panels does not lead to better accuracy; a higher multipole expansion order should be used to maintain the same degree of approximation and accuracy. When a higher order of expansion ($p = 6$) is used, the error improves when both finer cell and panel discretizations are used (Figure 3.8). The above example shows that the order of expansion p has a more significant role than the cell discretization in improving the accuracy of the FFTM algorithm.

Balancing between the computational time and accuracy, it can be seen that $16 \times 16 \times 16$ cells and $p = 4$ are preferred for most cases. With such parameters, the computational complexity of the FFTM is about $O(N^{1.2})$, shown in Figures 3.4 and 3.5. The memory usage for all the discretizations ranges from $390M$ to $480M$. Most of the memory (about $330M$) is used to store the $M2L$ translation matrix. Hence, the memory requirement is determined mainly by the number of cells and expansion order p. Apart from the $330M$, the rest of the memory usage scales linearly with the number of nodes. Based on the memory requirement in this example, the largest problem that can be solved with the FFTM, using a computer with $1GB$ of RAM, is estimated to have 200,000 degrees of freedom.

3.3.3 Sphere moving in potential flow

In this example, an exterior problem consisting of a sphere moving through a fluid (modeled as a potential flow) is considered. The sphere moves at a constant of velocity U, and a non-penetrating boundary condition (of the Neumann type) on the surface of the sphere is imposed.

$$\frac{\partial \Phi}{\partial \mathbf{n}} = U \cos \theta, \tag{3.49}$$

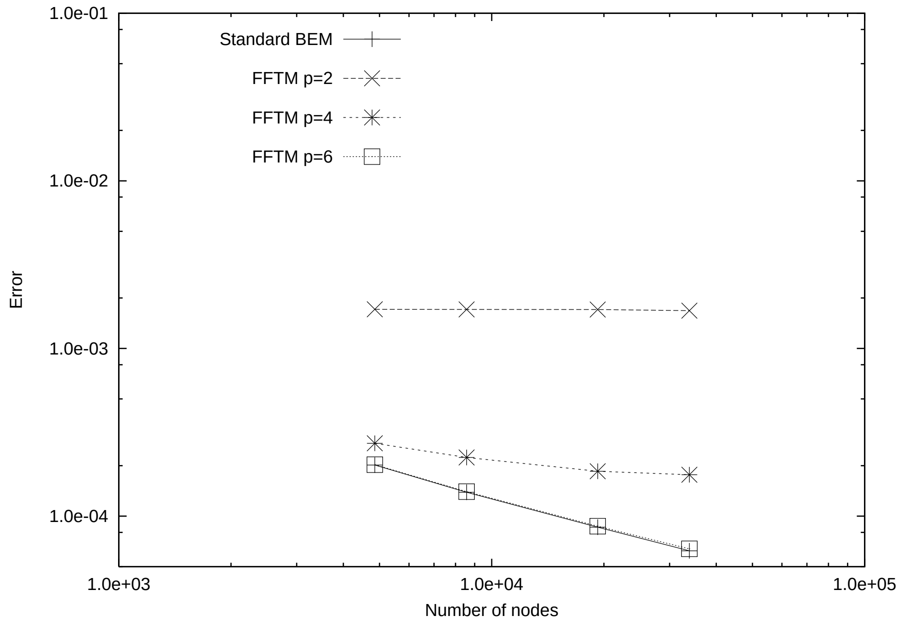
where Φ is the velocity potential. The analytical solution for Φ is

$$\Phi = -\frac{1}{2} U r_0 \cos \theta, \tag{3.50}$$

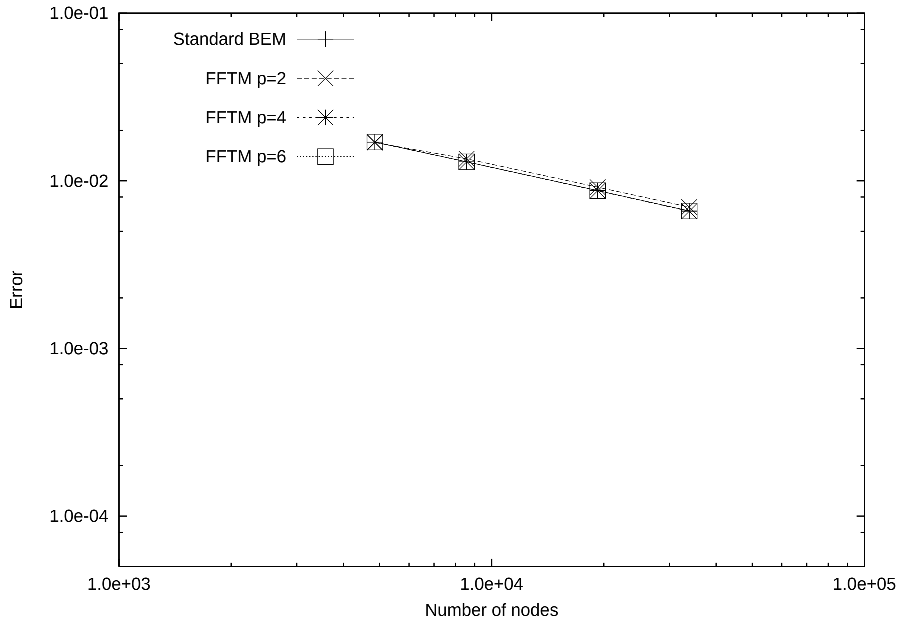
where $r_0 = 0.5$ is the radius of the sphere.

The computational timings for this problem are very similar with those in the previous example with Dirichlet boundary conditions, so they are not presented here. For the accuracy of FFTM, the errors (referenced to the analytical solution in Equation (3.50)) in this Neumann boundary condition problem are generally lower than those in the previous thermal conduction (Dirichlet boundary condition) example. Figure 3.9 shows the accuracy of FFTM for various orders of expansion p used. Similar to the previous example, the FFTM needs $p = 6$ to achieve accuracy close to the standard BEM when the direct formulation is used. For the indirect formulation, $p = 2, 4$ and 6 are able to achieve the same accuracy as the standard BEM, as the standard indirect BEM is not very accurate (probably due to the difficulty to evaluate the strongly singular integral that involves the kernel K).

Figure 3.10 shows the accuracy of the FFTM with various cell discretizations.

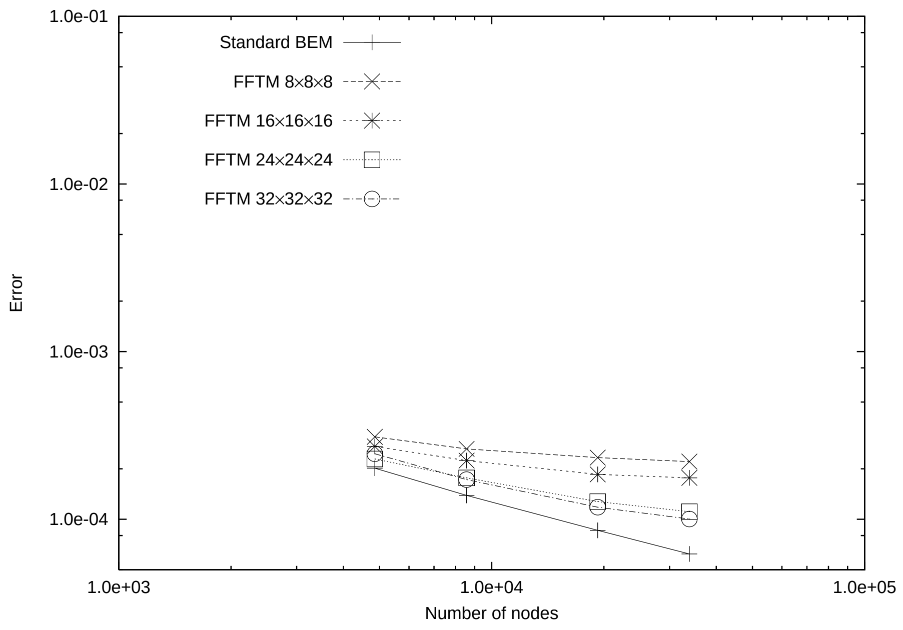


(a) direct formulation

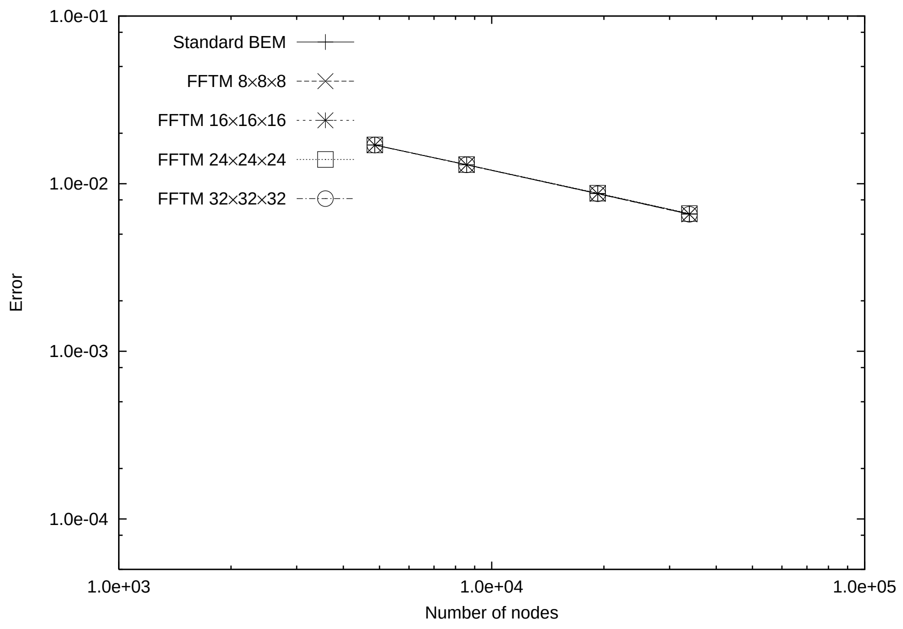


(b) indirect formulation

Figure 3.9: Error for various orders of expansion with cell discretization of $16 \times 16 \times 16$ (Neumann problem).



(a) direct formulation



(b) indirect formulation

Figure 3.10: Error for various cell discretizations with $p = 4$ (Neumann problem).

Similar to the previous example, the accuracy does not improve significantly with finer cell discretization. For the indirect method, the poor accuracy of standard BEM dominates the error so that the FFTM cell discretization error is insignificant. For the direct formulation, finer cell discretization improves the accuracy of FFTM, but rather slowly. In this example, $p = 4$ is sufficient to show improvement in error when both cell and panel discretizations are refined, as compared to the previous example that needs $p = 6$.

3.4 Summary of FFTM for Laplace equation

In this chapter, the FFTM algorithm is used to accelerate the BEM when solving Laplace equation. This is implemented in both the indirect and direct formulations of the BEM. The translations for the indirect BEM formulation with Dirichlet boundary condition are provided, followed by detailed derivation of the translations for the Neumann problem. For the direct BEM formulation, a new formulation for handling the double layer kernel is obtained. This new method is based on the physical interpretation of monopole and dipole sources, and it is obvious to be related theoretically to the method given by Yoshida. Hence, the results from this new method are the same with Yoshida's method. The performances are illustrated and analyzed with two simple numerical examples. The direct formulation tends to take more computational time due to the evaluation of an extra integral. The direct formulation is more accurate than the indirect formulation because the direct formulation has the advantage of avoiding the calculations of the free term and the strongly singular integral explicitly. The multipole and local translations introduce approximation errors, but these are not significant compared with the discretization error in the direct or indirect BEM formulation. Since the constant boundary element is used, the discretization error of BEM is high. FFTM can achieve the same error as the standard BEM with sufficient high multipole expansion order used.

Chapter 4

Navier equation

In this Chapter, the conventional BEM for elasticity problems will be introduced, followed by the details of the implementation of the FFTM. As the variables are now vectors (displacement and tractions), to solve the governing equation, which is the Navier equation (Equation (1.2)), the kernels are more complicated than those of the Laplace equation. More translation operators are needed to map vectors to their multipole representations. In this chapter, the translation operators in Yoshida's [88, 89] work are adopted. Only four sets of multipole moments are needed. The memory storage requirement of the FFTM algorithm is dominated by the storage of kernels for the multipole to local expansion transform. Using the original FFTM method, the memory requirement is of $O(p^4)$, where p is the expansion order. For elasticity problems, where the variables are vector quantities, the memory requirement for the translation operators is more than that for scalar equations, such as the Laplace equation. Hence, reducing the memory requirement is critical for implementing the FFTM in large elasticity problems. Here, a more concise form is presented to store the transformation matrices, and the memory requirement is reduced to $O(p^2)$. The performance of the improved FFTM is compared with the standard method, as well as the commonly used FMM. Lastly, several case studies are presented to highlight the computational accuracy and efficiency of the FFTM.

4.1 BEM for Navier equation

The Navier equation (Equation (1.2)) can be rewritten in its integral form

$$c_{ij}u_i(\mathbf{x}) = \int_{S_y} T_{ij}(\mathbf{x},\mathbf{y})u_j(\mathbf{y})dS_y - \int_{S_y} U_{ij}(\mathbf{x},\mathbf{y})t_j(\mathbf{y})dS_y, \tag{4.1}$$

where S_y is the boundary and c_{ij} is the free term. The single layer kernel U_{ij} (displacement kernel) and the double layer kernel T_{ij} (traction kernel) are defined as ([5])

$$U_{ij}(\mathbf{x},\mathbf{y}) = \frac{1}{16\pi\mu(1-\nu)}\frac{1}{r(\mathbf{x},\mathbf{y})}[(3-4\nu)\delta_{ij} + \frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_i}\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_j}], \tag{4.2}$$

and

$$\begin{aligned} T_{ij}(\mathbf{x},\mathbf{y}) \quad = \quad & \frac{-1}{8\pi(1-\nu)r^2(\mathbf{x},\mathbf{y})}\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}[(1-2\nu)\delta_{ij} + 3\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_i}\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_j}] \\ & -\frac{1-2\nu}{8\pi(1-\nu)r^2(\mathbf{x},\mathbf{y})}[\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_j}n_i(\mathbf{y}) - \frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_i}n_j(\mathbf{y})], \end{aligned} \tag{4.3}$$

where r is the distance between the source point $\mathbf{y}$ and the evaluation point $\mathbf{x}$, and $\mathbf{n}(\mathbf{y})$ is the outward normal direction at the source point.

For BEM, the boundary of the domain is discretized into elements, and a system of equations involving the nodal displacements $[u]$ and tractions $[t]$ is obtained,

$$[T][u] - [U][t] = 0, \tag{4.4}$$

where $[T]$ and $[U]$ are coefficient matrices obtained by performing the integrations in Equation (4.1) over the boundary elements. In this chapter, the same constant element and numerical integration scheme, as the previous chapter, are used to calculate the coefficients. Equation (4.4) can be solved similarly as shown in Section 3.1.2. One method is to rearrange the system of equations into the form $[A][x] = [b]$, where $[A]$ is a big full matrix and $[b]$ is a vector obtained from known boundary conditions. The resulting linear system (with matrix $[A]$ and vector $[b]$ known) can be solved by Gauss elimination or GMRES. The other method is to solve Equation (4.4) by GMRES without forming the matrix $[A]$ explicitly. In each iteration of GMRES scheme, guess values of the unknowns and the boundary conditions are used to perform the matrix-vector multiplication $[T][u] - [U][t]$ without storing the matrices $[T]$ and $[U]$. The entries in these two matrices are calculated in every iteration. For the case of mixed boundary conditions, the GMRES solver needs a preconditioner to achieve fast convergence. In this chapter, a simple but effective diagonal preconditioner is used by post-multiplying the $[T]$ and $[U]$ matrices by the inverse of the corresponding diagonal matrices, $[\text{diag}T]^{-1}$ and $[\text{diag}U]^{-1}$, respectively,

$$[T][u] - [U][t] = \left\{[T][\text{diag}T]^{-1}\right\}[\text{diag}T][u] - \left\{[U][\text{diag}U]^{-1}\right\}[\text{diag}U][t]. \tag{4.5}$$

The intermediate variables, $[\tilde{u}] = [\text{diag}T][u]$ and $[\tilde{t}] = [\text{diag}U][t]$, are solved, from which the original variables can readily be obtained. As the matrices $[T]$ and $[U]$

are diagonally dominant, the resulting system with coefficient matrices $[T][\text{diag}T]^{-1}$ and $[U][\text{diag}U]^{-1}$ are well-conditioned for the use with iterative solvers, especially when mixed boundary conditions are prescribed. In all the numerical examples, convergence to reasonable accuracy can be achieved within 40 iterations.

4.2 FFTM for Navier equation

The integral equation (Equation (4.1)) can be interpreted as a field calculation due to single and double layer sources as expressed by the kernels U_{ij} and T_{ij}, respectively. The FFTM algorithm provides an acceleration of this field calculation by grouping of the sources into equivalent multipole and local expansion representations. This is accurate especially when the source point $\mathbf{y}$ and field point $\mathbf{x}$ are far apart. To evaluate the field at a node, the sources are divided into two groups, near sources and far sources, based on the distance between the field point and the sources points. This is done by dividing the computational domain into a grid of regularly spaced cells that contain all the nodes and elements. The near sources consist of those in the same cell as the field point and the neighbouring cells that share at least one vertex with the cell containing the field point. This is equivalent to assigning one layer of surrounding cells as the near sources. It is also possible to use more layers of surrounding cells as near sources, as implemented in [56]. The field due to the near sources are calculated directly using the standard BEM, while the contribution from far sources are calculated using the multipole and local expansions.

Similar to the case of the Laplace equation, the FFTM needs three translations in the Navier equation, namely source to multipole moment ($S2M$), multipole moment to local expansion ($M2L$) and local expansion to destination ($L2D$). For the Navier equation, the variables at the source and destination are both 3×1 vectors, and the translation operators are more complicated than those for scalar variables in the Laplace equations. Here, Yoshida's formulation [88], which uses only four sets of multipole moments (and local expansions) to represent each group of sources, is employed.

The expression for $1/r(\mathbf{x}, \mathbf{y})$ is expanded in terms of solid harmonics R_n^m and S_n^m,

$$\frac{1}{r(\mathbf{x}, \mathbf{y})} = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \overline{S_n^m}(\overrightarrow{\mathbf{Ox}}) R_n^m(\overrightarrow{\mathbf{Oy}}). \tag{4.6}$$

where

$$R_n^m(\overrightarrow{\mathbf{Oy}}) = \frac{1}{(n+m)!} P_n^m(\cos\alpha) e^{im\beta} \rho^n, \tag{4.7}$$

$$S_n^m(\overrightarrow{\mathbf{Oy}}) = (n-m)! P_n^m(\cos\alpha) e^{im\beta} \frac{1}{\rho^{n+1}}, \tag{4.8}$$

$\mathbf{O}$ is the cell center, with (ρ, α, β) being the relative spherical coordinates of the point $\mathbf{y}$ from $\mathbf{O}$, and P_n^m is the Legendre polynomial. With the usage of solid harmonics, the Legendre polynomial defined as in Equation (3.33) is slightly different from Equation (3.9). The single layer kernel can then be expressed in terms of solid harmonics

$$\begin{aligned} U_{ij}(\mathbf{x}, \mathbf{y}) = \frac{1}{16\pi\mu(1-\nu)} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (3-4\nu)[\delta_{ij} \overline{S_n^m}(\overrightarrow{\mathbf{Ox}}) R_n^m(\overrightarrow{\mathbf{Oy}}) - \\ (\overrightarrow{\mathbf{Ox}})_j \frac{\partial}{\partial x_i} \overline{S_n^m}(\overrightarrow{\mathbf{Ox}}) R_n^m(\overrightarrow{\mathbf{Oy}}) + \frac{\partial}{\partial x_i} \overline{S_n^m}(\overrightarrow{\mathbf{Ox}}) (\overrightarrow{\mathbf{Oy}})_j R_n^m(\overrightarrow{\mathbf{Oy}})]. \end{aligned} \tag{4.9}$$

And the double layer kernel can be obtained from the single layer kernel

$$T_{ij}(\mathbf{x},\mathbf{y}) = C_{cjdl}\frac{\partial U_{id}(\mathbf{x},\mathbf{y})}{\partial y_l}n_c(\mathbf{y}), \tag{4.10}$$

giving

$$T_{ij}(\mathbf{x},\mathbf{y}) = \frac{1}{16\pi\mu(1-\nu)}\sum_{n=0}^{\infty}\sum_{m=-n}^{n} C_{cjdl}(3-4\nu)[\delta_{id}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}})\frac{\partial}{\partial y_l}R_n^m(\overrightarrow{\mathbf{Oy}}) - (\overrightarrow{\mathbf{Ox}})_d\frac{\partial}{\partial x_i}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}})\frac{\partial}{\partial y_l}R_n^m(\overrightarrow{\mathbf{Oy}}) + \frac{\partial}{\partial x_i}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}})\frac{\partial}{\partial y_l}((\overrightarrow{\mathbf{Oy}})_d R_n^m(\overrightarrow{\mathbf{Oy}}))]n_c(\mathbf{y}), \tag{4.11}$$

where

$$C_{ijkl} = \frac{2\mu\nu}{1-2\nu}\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}). \tag{4.12}$$

So the multipole moments at the cell centers are defined by ($S2M$)

$$M_{n,j}^m(\mathbf{O}) = \int_{S_y} C_{cdjl}\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_l}u_d(\mathbf{y})n_c(\mathbf{y})dS_y - \int_{S_y} R_n^m(\overrightarrow{\mathbf{Oy}})t_j(\mathbf{y})dS_y,$$
$$M_n^m(\mathbf{O}) = \int_{S_y} C_{cdjl}\frac{\partial}{\partial y_l}[(\overrightarrow{\mathbf{Oy}})_j R_n^m(\overrightarrow{\mathbf{Oy}})]u_d(\mathbf{y})n_c(\mathbf{y})dS_y - \int_{S_y}(\overrightarrow{\mathbf{Oy}})_j R_n^m(\overrightarrow{\mathbf{Oy}}) t_j(\mathbf{y})dS_y. \tag{4.13}$$

where $M_{n,j}^m$ consists of three components ($j = 1, 2, 3$), and M_n^m is a scalar.

These four sets multipole moments can be translated to four sets of local expansions defined at another point $\mathbf{O}'$ ($M2L$) by

$$\begin{aligned} L_{n',j}^{m'}(\mathbf{O}') &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^{n'}\overline{S_{n+n'}^{m+m'}}(\overrightarrow{\mathbf{OO}'})M_{n,j}^m(\mathbf{O}) \\ L_{n'}^{m'}(\mathbf{O}') &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^{n'}\overline{S_{n+n'}^{m+m'}}(\overrightarrow{\mathbf{OO}'})[M_n^m(\mathbf{O}) - (\overrightarrow{\mathbf{OO}'})_j M_{n,j}^m(\mathbf{O})]. \end{aligned} \tag{4.14}$$

The $M2L$ translation operators are functions of only the distance between the points $\mathbf{O}$ and $\mathbf{O}'$. This can be seen as a convolution in space coordinates, and hence the FFT can be used to accelerate the calculation.

Lastly, the field at the destination point can be obtained from the local expansion coefficients $L^m_{n,j}$ and L^m_n by ($L2D$):

$$\int_{S_y} T_{ij}(\mathbf{x},\mathbf{y})u_j(\mathbf{y})dS_y - \int_{S_y} U_{ij}(\mathbf{x},\mathbf{y})t_j(\mathbf{y})dS_y = \frac{1}{16\pi\mu(1-\nu)}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}$$

$$\{[(3-4\nu)\delta_{ij}R^m_n(\overrightarrow{\mathbf{O'x}}) - (\overrightarrow{\mathbf{O'x}})_j\frac{\partial R^m_n(\overrightarrow{\mathbf{O'x}})}{\partial x_i}]L^m_{n,j}(\mathbf{O'}) + \frac{\partial R^m_n(\overrightarrow{\mathbf{O'x}})}{\partial x_i}L^m_n(\mathbf{O'})\}. \quad (4.15)$$

The application of FFTM in Navier equation can be summarized in the following steps, shown in Figure 4.1:

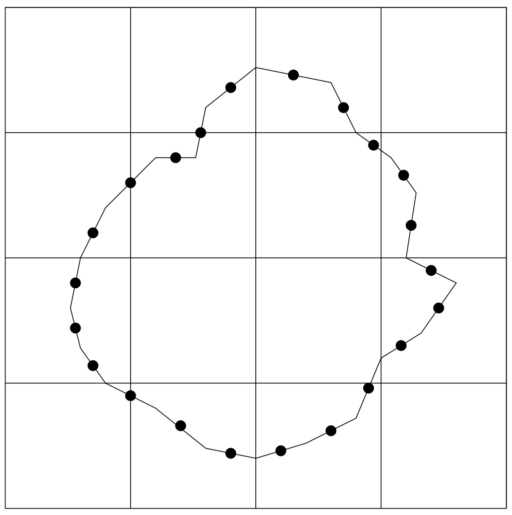
A. Define a rectangular domain that contains the whole computational domain and discretize the spatial domain into many smaller cells.

B. Convert sources ($\mathbf{u}$ and $\mathbf{t}$) within each cells to multipole moments ($S2M$), using Equation (4.13).

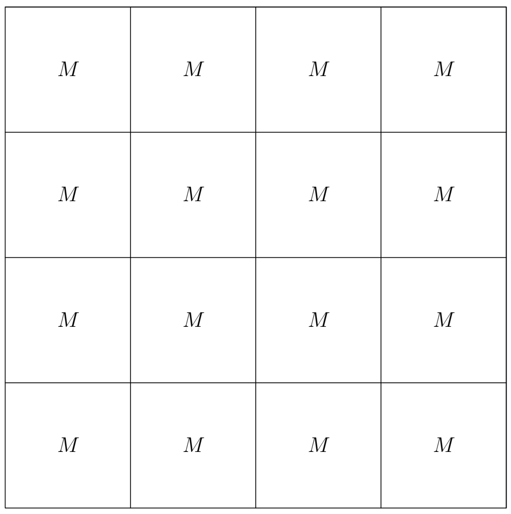
C. Evaluate the local expansion coefficients at cell centers due to the multipole moments at other cells (Equation (4.14)). This process is a series of discrete convolutions that are accelerated by the FFT ($M2L$).

D. Compute the field quantity ($\mathbf{u}$) at the nodal locations using local expansions ($L2D$) (Equation (4.15)), which only account for the distant charges contributions, and also add the contributions from the near sources.

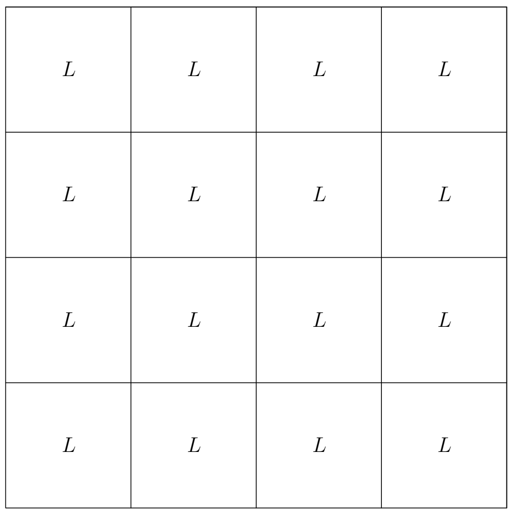
From Equation (4.14), the translation operators $\overline{S^{m+m'}_{n+n'}}$ and $\overline{S^{m+m'}_{n+n'}}(\overrightarrow{\mathbf{OO'}})_j$ for the $M2L$ translation are actually matrices that need to be stored. For example, the



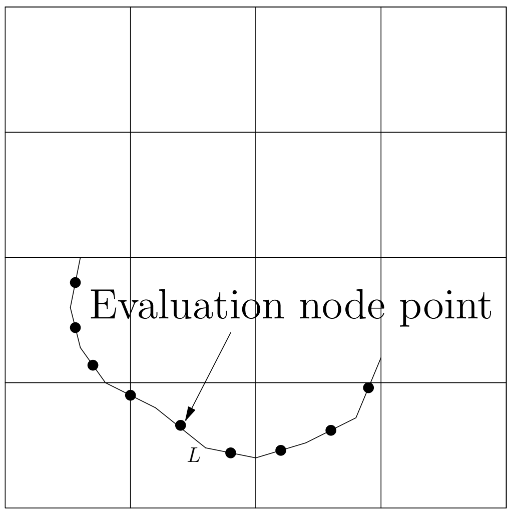
(a) Step A



(b) Step B



(c) Step C



(d) Step D

Figure 4.1: Two dimensional pictorial representation of FFTM for Navier equation. Step A: discretization of domain, Step B: sources to multipoles translation (S2M), Step C: multipoles to local expansions translation (M2L) accelerated by FFT, Step D: field evaluation by local expansions (L2D) and direct calculation.

operator $\overline{S_{n+n'}^{m+m'}}$ between two cells can be written explicitly in the matrix form

$$\begin{bmatrix} \overline{S_0^0} & \overline{S_1^{-1}} & \overline{S_1^0} & \overline{S_1^1} & \overline{S_2^{-2}} & \overline{S_2^{-1}} & \cdots \\ \overline{S_1^{-1}} & \overline{S_2^{-2}} & \overline{S_2^{-1}} & \overline{S_2^0} & \overline{S_3^{-3}} & \overline{S_3^{-2}} & \cdots \\ \overline{S_1^0} & \overline{S_2^{-1}} & \overline{S_2^0} & \overline{S_2^1} & \overline{S_3^{-2}} & \overline{S_3^{-1}} & \cdots \\ \overline{S_1^1} & \overline{S_2^0} & \overline{S_2^1} & \overline{S_2^2} & \overline{S_3^{-1}} & \overline{S_3^0} & \cdots \\ \overline{S_2^{-2}} & \overline{S_3^{-3}} & \overline{S_3^{-2}} & \overline{S_3^{-1}} & \overline{S_4^{-4}} & \overline{S_4^{-3}} & \cdots \\ \overline{S_2^{-1}} & \overline{S_3^{-2}} & \overline{S_3^{-1}} & \overline{S_3^0} & \overline{S_4^{-3}} & \overline{S_4^{-2}} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}, \tag{4.16}$$

and the dimension of this matrix is $(p+1)^2 \times (p+1)^2$. However, many entries in this matrix are the repeated. A more compact storage

$$\begin{bmatrix} \overline{S_0^0} & \overline{S_1^{-1}} & \overline{S_1^0} & \cdots & \overline{S_{2p}^{2p-1}} & \overline{S_{2p}^{2p}} \end{bmatrix} \tag{4.17}$$

can be obtained by only storing the distinct terms. The dimension of this vector is $(2p+1)^2$, compared to the original $(p+1)^4$. This reduces the memory storage considerably. For example, when $p = 4$, the new scheme needs to store 81 complex numbers for each multipole/local expansion component, compared with 625 complex numbers using the original scheme. The operators $\overline{S_{n+n'}^{m+m'}}(\overrightarrow{\mathbf{OO'}})_j$ can be stored compactly in a similar manner.

4.3 Numerical examples

In this section, several examples are provided to demonstrate the accuracy and efficiency of the new algorithm. A hydrostatically loaded sphere is first presented, with a comparison of the numerical solutions with the analytical solution and the FMM. Next, the new algorithm is applied to calculation of the effective stiffness in materials with voids of various shapes and sizes. The latter examples, with all the voids modeled explicitly, involve a large number of degrees of freedom, and they benefit greatly from the present fast algorithm in terms of memory storage and computational time.

4.3.1 Hydrostatically loaded sphere

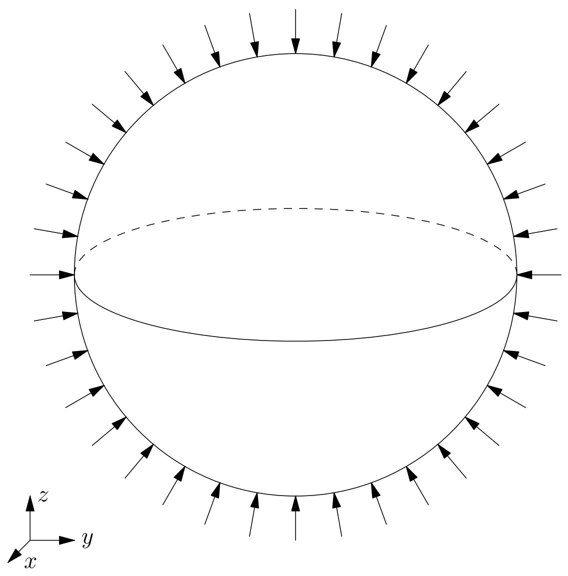


Figure 4.2: Hydrostatically loaded sphere

In this example, illustrated in Figure 4.2, a sphere under hydrostatic load is considered. Dirichlet boundary condition is prescribed on the sphere surface with the displacement in the radial direction given to be δ. For a small displacement δ, the analytical solution for the normal traction $\bar{t}$ is then given by

$$\bar{t} = \frac{\delta E}{r(1-2\nu)}. \tag{4.18}$$

In this problem the radius of the sphere r is 0.5 m, the Young's modulus E is 165 MPa, and the Poisson's ratio ν is 0.3. The measure of the error is defined in the L_2 norm as

$$Error = \sqrt{\frac{\sum_{i=1}^{N} |t(\mathbf{x}_i) - \bar{t}(\mathbf{x}_i)|^2}{\sum_{i=1}^{N} |\bar{t}(\mathbf{x}_i)|^2}}, \tag{4.19}$$

where t is the numerical solution of the normal traction.

To illustrate the advantages of the FFTM algorithm, we compare our results with the standard BEM, as well as the FMM. In the literature, researchers compared their fast algorithms with three different standard BEMs. They are referred in this work as Standard BEM (GE), Standard BEM (GMRES) and Standard BEM (matrix-free). In the Standard BEM (GE) and Standard BEM (GMRES), the full coefficient matrix is constructed and the resulted linear equations are solved by Gauss elimination method and GMRES method, respectively. In the Standard BEM (matrix-free), the construction of the big matrix is avoided with the help of the GMRES, but the coefficients in the matrix are calculated in every iteration in GMRES scheme.

To compare with the FMM, we select Wang and Yao's work [79] which presents the most recent and detailed results. In addition, Wang and Yao used constant element, and ran their program on a single CPU computer, which make it easy and reasonable for us to compare. For comparison, their data in the Figure 5 (a) in their paper are scaled, so that the time of the standard method (Standard BEM (GE)) in their paper is the same with ours. Such scaling is based on the fact that the computational time of the standard method, in [79] and current work, should be equivalent approximately. They solved a mixed boundary condition problem to obtain the data and used about 30 GMRES iterations to get the FMM algorithm converged to a residue less than 10^{-5}. Although for the current pure Dirichlet problem about 20 GMRES iterations are sufficient to obtain the same residue, we still run 30 iterations to compare the computational time fairly with the FMM. The extra iterations do not affect the accuracy and memory usage very much. In the FFTM, $16 \times 16 \times 16$ cells are used to discretize the spatial domain and expansion order $p = 4$ and $p = 6$ are tested.

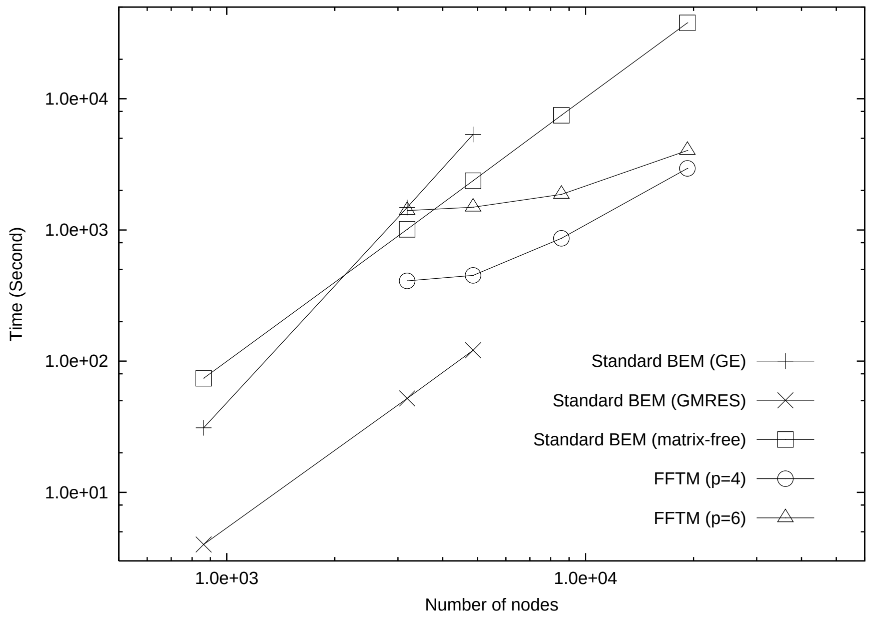


Figure 4.3: Computational time compared with standard methods

Figure 4.3 shows the timings of the FFTM algorithm, compared with the three standard methods. The Standard BEM (GE) shows the typical computational complexity of $O(N^3)$, and the computational complexity of the Standard BEM (GMRES) and Standard BEM (matrix-free) is $O(N^2)$, while the complexity of the FFTM is lower. FFTM keeps the same computational complexity, $O(N \log N)$, when it is used to solve the Navier equation. The matrix-vector multiplication in GMRES constitutes the bulk of the timing. In each multiplication, the complexities of $S2M$, $M2L$ and $L2D$ are $O(N)$, $O(N \log N)$ and $O(N)$, respectively. When the problem becomes large (more than 10,000 panels), the FFTM reduces the computational time by almost one order of magnitude, compared with the Standard BEM (matrix-free). For large problems, the use of the Standard BEM (GE) and Standard BEM (GMRES) is not possible on personal computer due to the large memory requirement (typically more than 2 GB). For smaller problems, the Standard BEM (GMRES) is the fastest, as the matrix is found and stored in memory. Figure 4.4 gives the memory usage comparison of the FFTM and standard methods. The Standard BEM (GE) and Standard BEM (GMRES) need $O(N^2)$ memory to store the full coefficient matrix. With 2 GB memory, the two methods can solve problems with less than 5,000 panels. With the help of the compact memory storage, explained in the above section, the FFTM ($p = 6$) can solve the largest problem with less than 400 MB memory. This is acceptable for most PCs. Most of the memory, $O(N_c \times p^2)$, is used to store the $M2L$ translation matrices, where N_c is the number of cells. The other part grows linearly with the number of panels (N). The Standard BEM (matrix-free) uses less memory, but it needs to recompute the matrix entries during the iterations, and hence slows down the computation.

In Figure 4.5, the efficiency of the FFTM is compared with that of the FMM. For the FMM, the expansion order for multipole and local expansion is equal to 10, and the exponential expansion order is 9. Both the FMM and FFTM reduce the computational time when the problem becomes large. The standard BEM (GE)

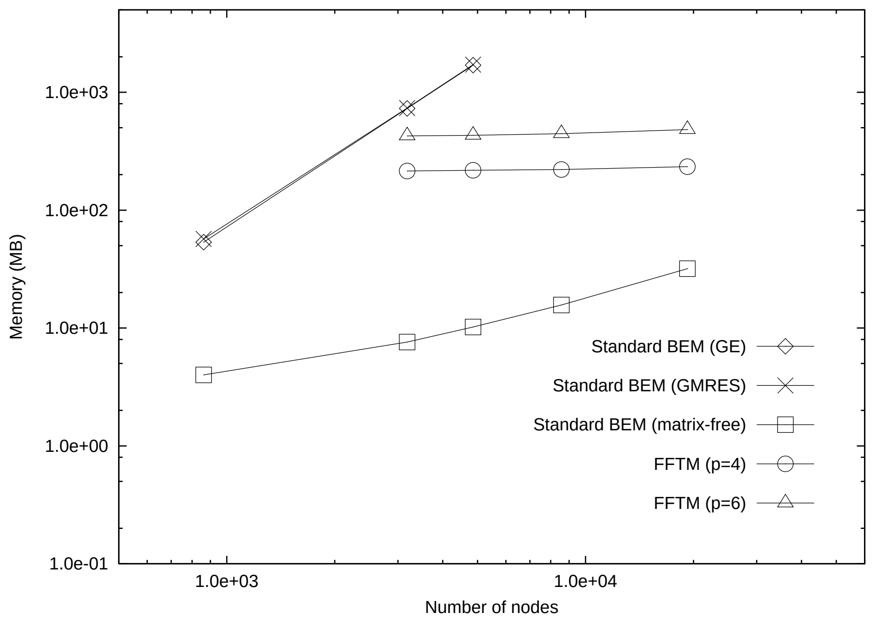


Figure 4.4: Memory usage compared with standard methods

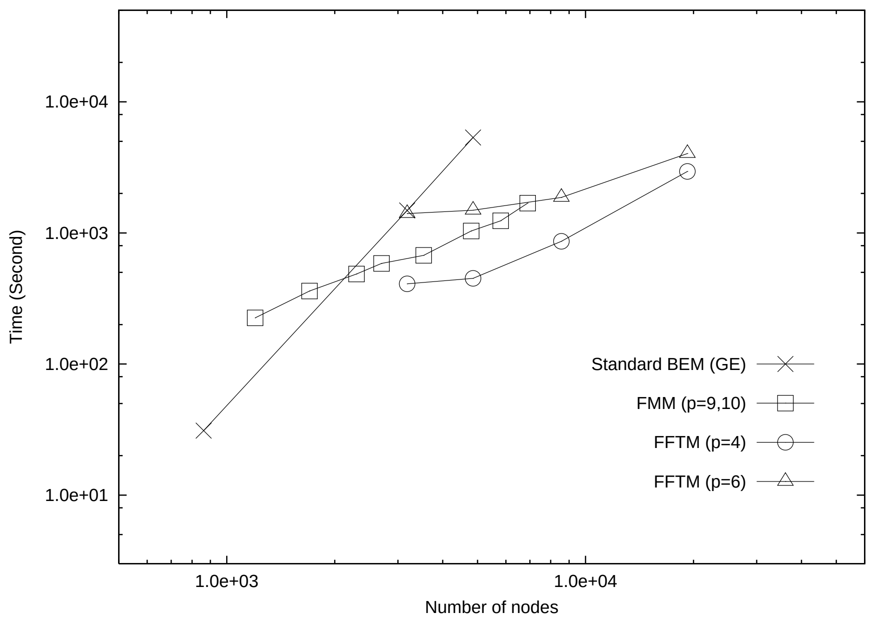


Figure 4.5: Computational time compared with FMM

has a complexity of $O(N^3)$, and the complexity of the FMM is $O(N)$ [79]. The FFTM method has similar complexity with the FMM. The timings of the FMM, with $p = 9, 10$, are comparable with those of FFTM, with $p = 6$, for large problems. When the number of nodes is less than 8000, the FMM uses less time than FFTM ($p = 6$), but longer time than FFTM ($p = 4$).

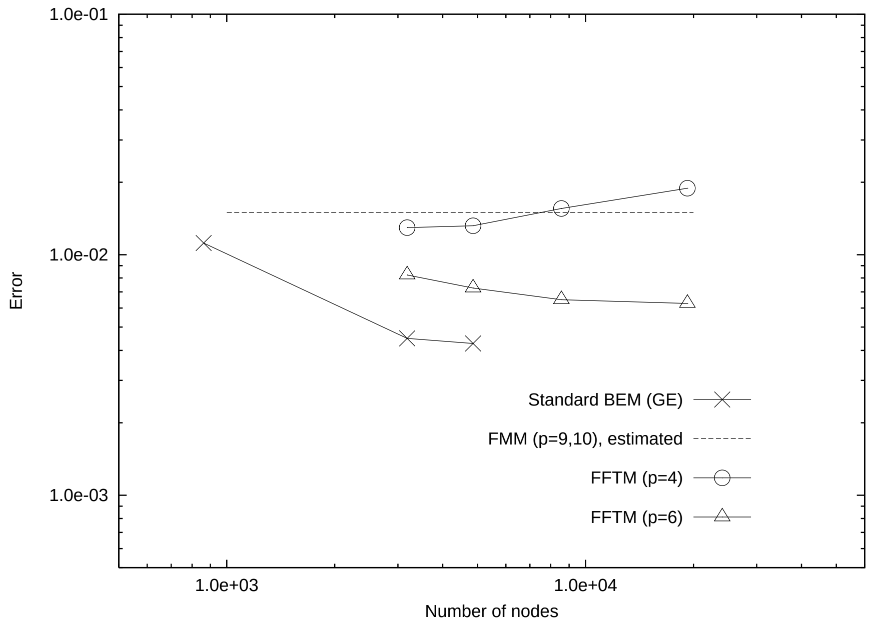


Figure 4.6: Accuracy compared with FMM

Figure 4.6 shows the accuracy of the fast algorithms. The FFTM provides relatively good accuracy. With higher expansion order ($p = 6$), the FFTM's results are closer to the standard BEM than $p = 4$. Since Wang and Yao did not provide detailed results to show the accuracy in [79], it is difficult to compare the accuracy directly. But when Wang et al. [78] extended their FMM algorithm to solve the Stokes equation, they gave the accuracy of with various expansion orders. When the exponential expansion order equals to 9, the error is a little higher than 10^{-2}, regardless of the multipole expansion used. Hence it is reasonable to estimate the accuracy of the FMM, in solving Navier equation, to be about 10^{-2}, as shown in Figure 4.6. With $p = 9, 10$, the accuracy of the FMM is comparable with the FFTM ($p = 4$). This is also consistent with the results of the early FFTM work from Ong et al. [59]. The FFTM is able to give comparable accuracy with lower expansion order than the FMM. When $p = 4$, the error of FFTM does not decrease with the number of nodes. This is the same behavior as that in Section 3.3.2. Again, with a fixed number of cells, more nodes are present in a cell and this leads to a higher truncation error for a given order of multipole expansion. A higher expansion order should be used when more nodes are present to reduce the truncation error. When a higher expansion order of $p = 6$ is used, the errors are shown to reduce.

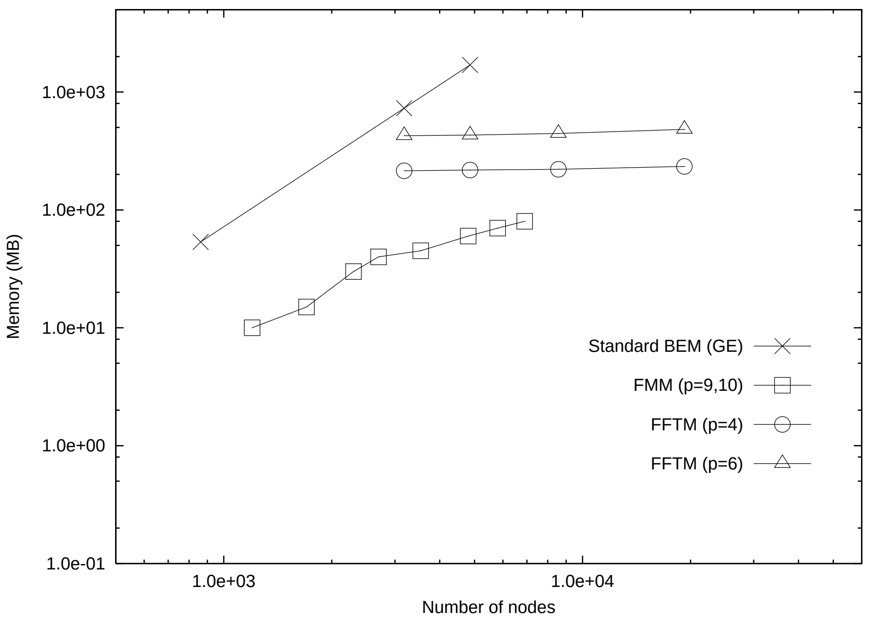


Figure 4.7: Memory usage compared with FMM

In Figure 4.7, the memory cost of the FMM, from Figure 5(b) in [79], grows linearly with number of panels. Even with the compact storage method, the FFTM consumes more memory than the FMM, especially when higher expansion order is used. So in summary, the FFTM obtains comparable accuracy with shorter computational time, but with higher memory requirment.

4.3.2 Effective Young's modulus with uniformly distributed spherical voids

In this example, a simulation of an elastic material containing voids undergoing a uniaxial tension, as shown in Figure 4.8, is considered. The voids provide the advantage of reducing the mass in cases where the weight of the material is crucial, but the stiffness and strength of the material also decrease. Hence, it is important to quantify the stiffness and strength of such materials when they are used in load bearing structures. In many applications, a numerical model containing all the voids will result in a large number of degrees of freedom. Even with the BEM that applies only boundary discretization, the computational resources needed may be formidable. O'Rourke et al. [60] have used a parallel implementation of BEM (with 162 processors) to compute the effective modulus of porous materials. Here, the FFTM is applied to accelerate the BEM computation.

In this numerical example, the bulk material has Young's modulus, $E = 165MPa$ and Poisson's ratio $\nu = 0.3$. For a cube of height L, a displacement in the z direction, $d = 0.01L$, is applied at the top boundary, and the bottom boundary is fixed in the z direction. All other boundaries, including the wall of the cube and the surfaces on the voids, are traction free. In the elastic regime, only small deformation is allowed, and the shape of the voids are not changed. The resultant

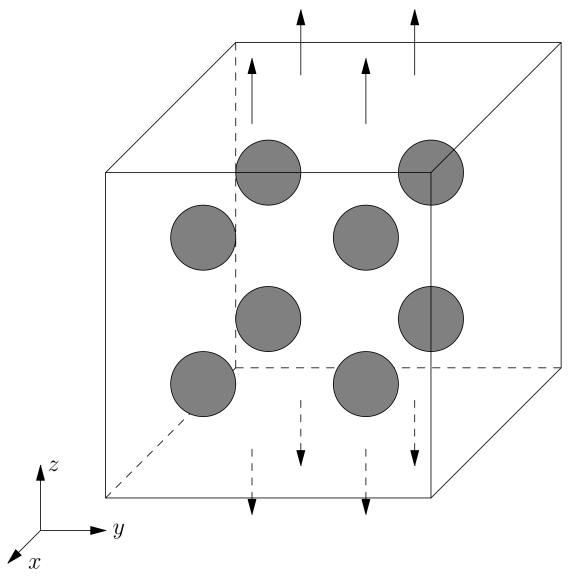


Figure 4.8: Axially loaded cube with uniformly distributed spherical voids

force F can be calculated at the top surface of the cube,

$$F = \int_{top} t_z dS, \tag{4.20}$$

and the effective Young's modulus E_{eff} of the porous material is defined by

$$E_{eff} = \frac{FL}{Ad}, \tag{4.21}$$

where A is the area over which the displacement d is applied. The normalized effective Young's modulus is then compared with the results from Nemat-Nasser et al. [49]. In this numerical example, the number of voids in the model ranges from 8 to 125. Table 4.1 gives the number of panels on the cube and the voids, with the largest number of degrees of freedom being 103650. For all the cases, a $16 \times 16 \times 16$ cell discretization of the computational domain and an expansion order of $p = 4$ are used. Using an AMD Opteron processor running at 2.2 GHz, the computational time ranges from 20 minutes to 3 hours. The maximum memory needed for the FFTM is about 250 MB, which is much lesser than that for the standard BEM (estimated to be 80 GB).

Table 4.1: Number of panels in each case

Number of voids	Panels on the cube	Panels on each void	Total
8	4800	238	6704
27	4800	238	11226
64	4800	238	20032
125	4800	238	34550

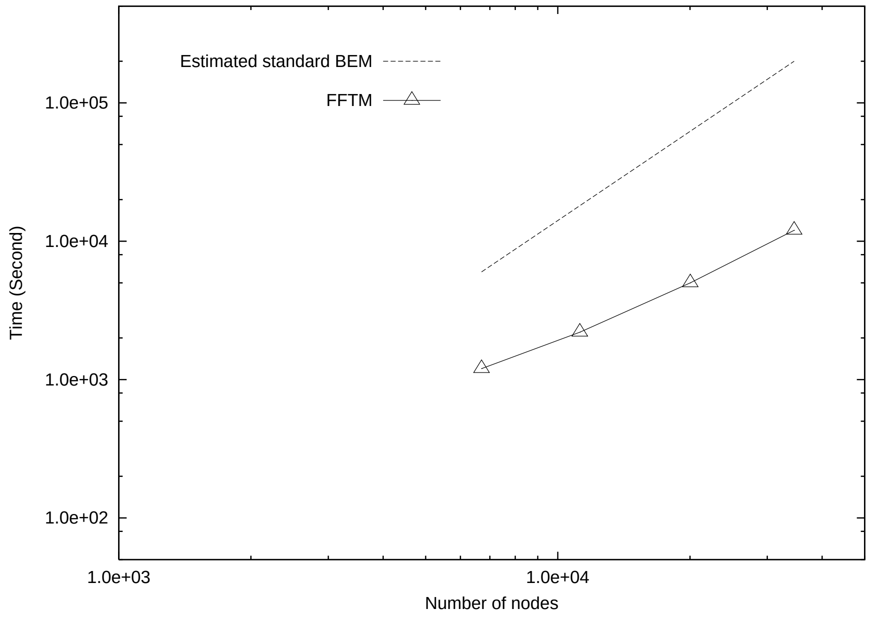


Figure 4.9: Typical FFTM computational time compared with that estimated for standard BEM

Figure 4.9 gives the typical computational time of the FFTM for the four cases with different voids. The computational time of the standard BEM is estimated from the timings in the above example, based on the $O(N^2)$ complexity of GMRES. The FFTM algorithm shows considerable savings in computational time, and also a

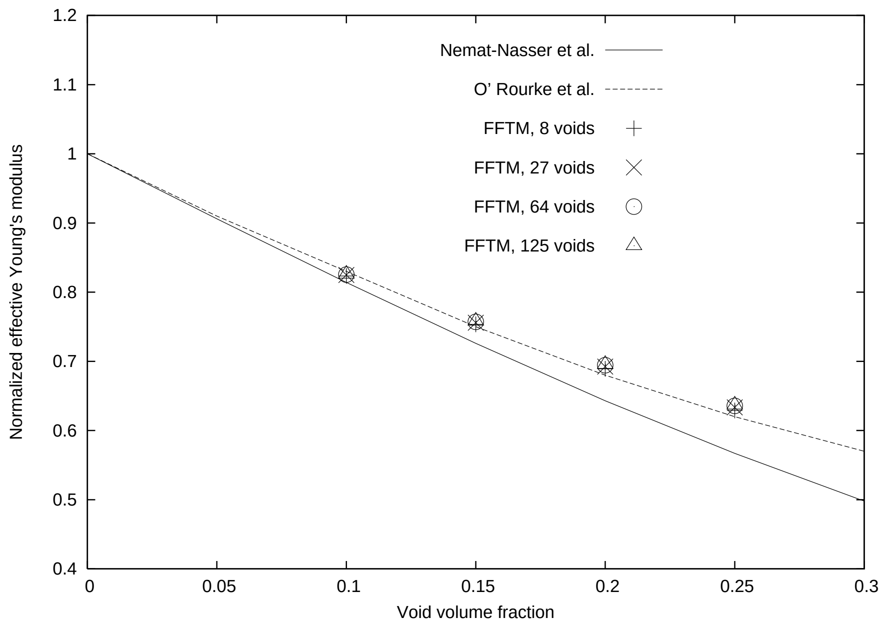


Figure 4.10: Normalized effective Young's modulus as a function of void volume fraction (uniformly distributed spherical voids)

lower computational complexity. For a fixed number of voids or degrees of freedom, computational timings for different void volume fractions are obtained, and these timings do not defer very much. The average computational times are given in Figure 4.9.

The effective Young's modulus of the porous material, normalized by the bulk Young's material, is given in Figure 4.10 as a function of the void volume fraction. Increase in the void volume fraction results in a reduced stiffness of the material. The effective Young's modulus is not sensitive to the number of the voids, but only dependent on the void volume fraction. The FFTM results deviate from the series solution of Nemat-Nasser et al. [49] as the void volume fraction increases. However, they are close to the numerical results from O'Rourke et al. [60].

4.3.3 Effective Young's modulus with randomly distributed spherical voids

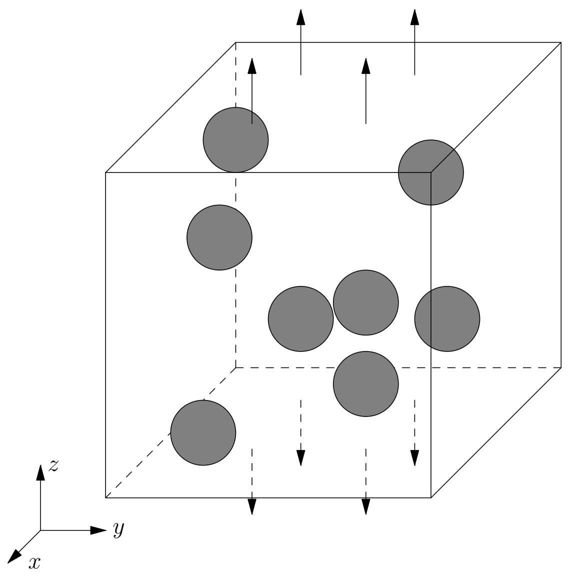


Figure 4.11: Axially loaded cube with randomly distributed spherical voids

This example investigates the effect of randomly distributed voids (shown in Figure 4.11). The same material in the previous example, $E = 165MPa$ and $\nu = 0.3$, is used and the effective Young's modulus of the cube is calculated. A small and random shift of the position of each void from its regularly spaced position is given. The shift is kept small to avoid two spheres from getting too close or intersecting with each other. The distribution is pseudo random with the voids being distributed over most parts of the cube. To avoid getting an extremely fine mesh that takes long computational time, the voids are not allowed to get too close to each other and those cases where voids are concentrated in a small volume are not considered. All the cases in Figure 4.10 are repeated, and the calculations are performed for ten random configurations in each case. Without the fast algorithm, it will be very time consuming to run so many cases. Figure 4.12 shows the average

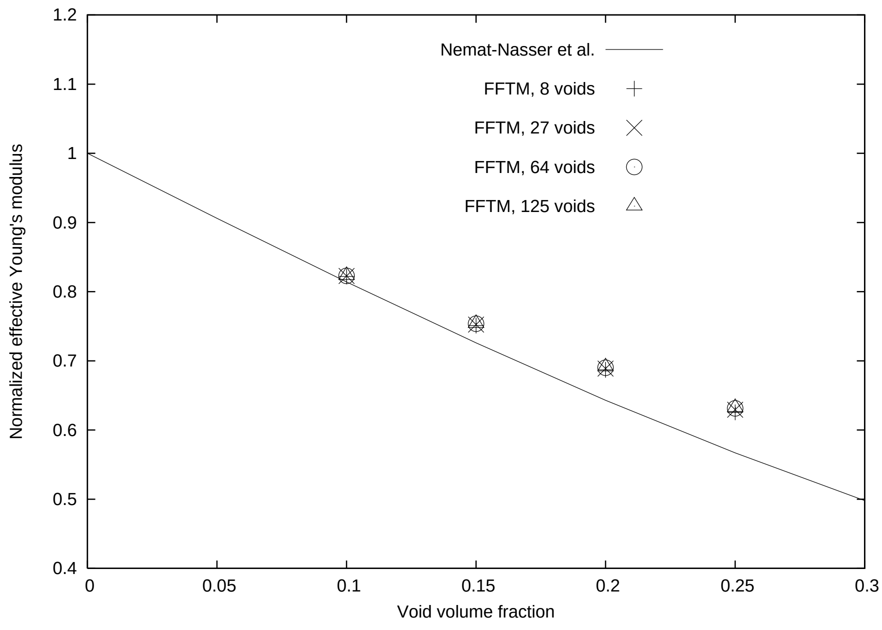


Figure 4.12: Average normalized effective Young's modulus as a function of void volume fraction (randomly distributed spherical voids)

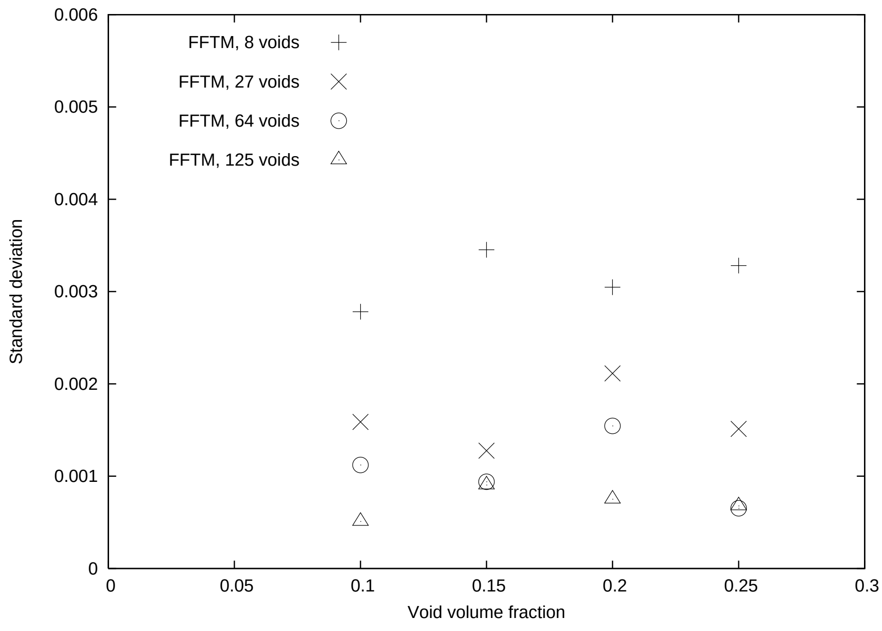
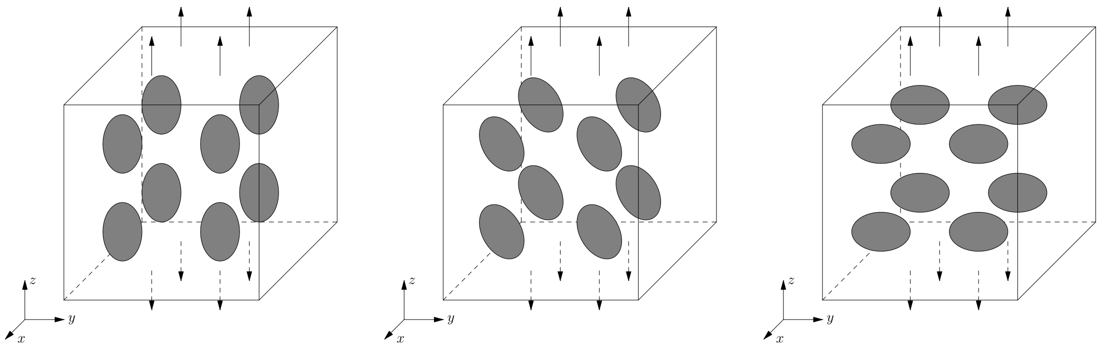


Figure 4.13: The standard deviation of the effective Young's modulus for random void configurations

results of the ten random configurations. The results are very similar with those in Figure 4.10. In addition, the standard deviations of the effective Young's modulus, shown in Figure 4.13, are very small, suggesting that the positions of the voids have little influence on the effective Young's modulus of the porous material. For a fixed void volume fraction, more voids lead to a smaller standard deviation of E.

4.3.4 Effective Young's modulus with uniformly distributed ellipsoidal voids



(a) Angle = 0 (b) 0 < Angle < 90 (c) Angle = 90

Figure 4.14: Axially loaded cube with uniformly distributed ellipsoidal voids

In this example, the effect of the void shape is considered. Here the material properties used are the same as the previous example. Figure 4.14 shows ellipsoidal voids with different orientations. The distribution of the ellipsoidal voids is the same as those cases in Section 4.3.2. Here, there are 64 ellipsoidal voids inside the cube with 20032 nodes in the model. The ellipsoids are distributed uniformly and the angular orientation of all the ellipsoidal voids are the same. The lengths of three semi-axes are represented by a, b and c, and three types of ellipsoid are analyzed: $\frac{4}{5}a = b = c$, $\frac{2}{3}a = b = c$ and $\frac{1}{2}a = b = c$.

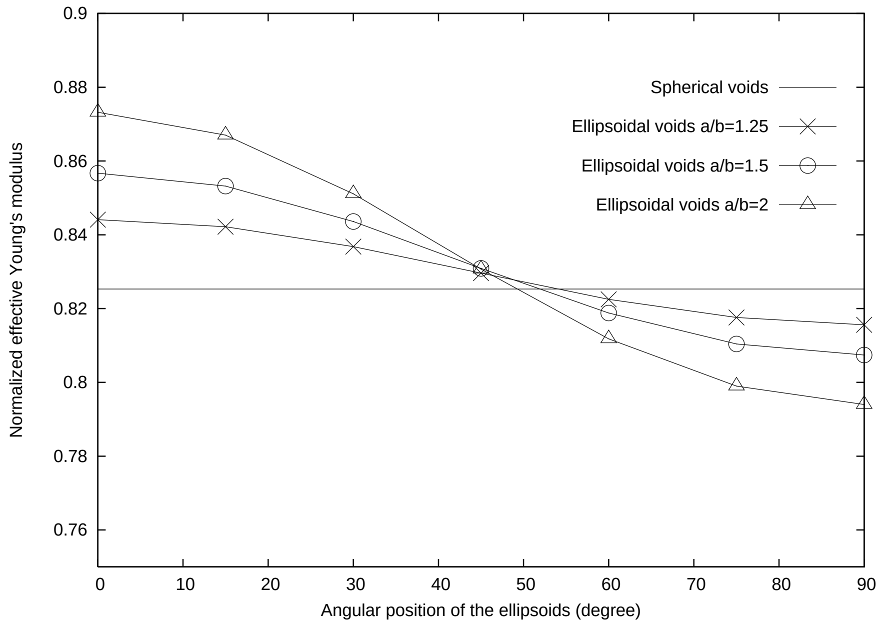


Figure 4.15: Normalized effective Young's modulus as a function of ellipsoid angular orientation (64 ellipsoidal voids and the void volume fraction is 0.1)

Figure 4.15 illustrates the normalized effective Young's modulus of different materials with 64 voids when the void volume fraction is 0.1. The horizontal line in Figure 4.15 corresponds to the numerical result for spherical voids. The effective stiffness of the material changes with the aspect ratio and orientation of the ellipsoid voids. Compared with the case with spherical voids, the material with ellipsoidal voids becomes stiffer, when the angle between the major axis and the z-axis is small (Figure 4.14(a)). When the angle is large (Figure 4.14(c)), the material becomes softer. When the aspect ratio a/b becomes bigger, the ellipsoid deviates more from the sphere, and the material stiffness also differ more from the spherical case. It is interesting to note that the effective Young's modulus is almost the same for different aspect ratios when the angular orientation is 45^o.

4.4 Summary of FFTM for Navier equation

In this chapter, the FFTM algorithm is applied to solve elastostatics problems governed by the Navier equation. Since the Navier equation involves vector quantities, the multipole translations are more complex than the Laplace equation. The translation formulations, derived by Yoshida, are adopted. Both the displacement kernel and the traction kernel can be represented by four sets of multipoles. These sets of multipoles result in more translation operations, as well as a tougher memory requirement than the Laplace equation. A compact storage of the matrix of the multipole to local expansion is introduced to reduce the memory usage, allowing large elasticity problems to be solved efficiently. In addition, a simple diagonal preconditioner is employed to achieve a faster convergence of GMRES algorithm for mixed boundary condition problems. With these techniques, the FFTM is compared with the FMM. The FFTM is faster and requires more memory than the FMM to obtain comparable accuracy. Then the fast algorithm is applied to the calculation of effective Young's modulus of a material containing numerous voids of various positions, sizes, shapes and orientations. For a material with spherical voids, the void volume fraction plays a significant role in determining the effective Young's modulus. The number of the voids or the positions of the voids do not have an obvious influence on the effective modulus. For a material with ellipsoidal voids, the effective Young's modulus is found to be dependent on the shape and orientation of the voids.

Chapter 5

Stokes equation

In this chapter, FFTM is extended to solve Stokes equation following the formulations provided by Frangi et al. [18] and Wang et al. [78]. The Stokes equation is the governing equation for fluid flow where the inertia force is small compared with the viscous force. The integral form of the Stokes equation is very similar with that of the Navier equation. Hence the technique for solving the Navier equation using the FFTM can easily be extended to solve the Stokes equation. However, the translation operators of Stokes equation differ from those of Navier equation (due to different kernels) and the formulations are present is this chapter. The same compact storage method of the translation matrices and the diagonal preconditioner are employed. The fast algorithm is verified with a simple example, calculating the drag force on a sphere centered in a cylinder tube. Lastly, the algorithm is applied to simulate Stokes flow in a cylinder tube with many spherical suspensions.

5.1 BEM for Stokes equation

The Stokes equation (Equation (1.6)) can be rewritten in its integral form

$$c_{ij}u_i(\mathbf{x}) = \int_{S_y} T_{ij}(\mathbf{x},\mathbf{y})u_j(\mathbf{y})dS_y - \int_{S_y} U_{ij}(\mathbf{x},\mathbf{y})t_j(\mathbf{y})dS_y, \tag{5.1}$$

where S_y is the boundary and c_{ij} is the free term. The single layer kernel U_{ij} and double layer kernel T_{ij} are defined as

$$U_{ij}(\mathbf{x},\mathbf{y}) = -\frac{1}{8\pi\mu}\frac{1}{r(\mathbf{x},\mathbf{y})} \times [\delta_{ij} + \frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_i}\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_j}], \tag{5.2}$$

and

$$T_{ij}(\mathbf{x},\mathbf{y}) = \frac{3}{4\pi}\frac{1}{r^2}\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_i}\frac{\partial r(\mathbf{x},\mathbf{y})}{\partial y_j}, \tag{5.3}$$

where r is the distance between the source point $\mathbf{y}$ and the evaluation point $\mathbf{x}$, $\mathbf{n}(\mathbf{y})$ is the outward normal direction at the source point, and δ_{ij} is the Kronecker delta.

Except for the expressions of the kernels, the standard BEM implementation for the Stokes equation is the same as the Navier equation.

5.2 FFTM for Stokes equation

The implementation of the FFTM in the Stokes equation is the same as the Navier equation (see Figure 4.1). The computational domain is discretized with numerous small cells, and then the translations of $S2M$, $M2L$ and $L2D$ are employed to accelerate the calculation. In addition, the compact storage of the translation in Equation (4.17) and the preconditioner in Equation (4.5) are also used.

The expression for $1/r(\mathbf{x},\mathbf{y})$ is expanded in terms of solid harmonics R_n^m and S_n^m,

$$\frac{1}{r(\mathbf{x},\mathbf{y})} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} \overline{S_n^m}(\overrightarrow{\mathbf{Ox}})R_n^m(\overrightarrow{\mathbf{Oy}}). \tag{5.4}$$

where

$$R_n^m(\overrightarrow{\mathbf{Oy}}) = \frac{1}{(n+m)!}P_n^m(\cos\alpha)e^{im\beta}\rho^n, \tag{5.5}$$

$$S_n^m(\overrightarrow{\mathbf{Oy}}) = (n-m)!P_n^m(\cos\alpha)e^{im\beta}\frac{1}{\rho^{n+1}}, \tag{5.6}$$

$\mathbf{O}$ is normally the cell center, with (ρ,α,β) being the relative spherical coordinates of the point $\mathbf{y}$ from $\mathbf{O}$, and P_n^m is the Legendre polynomial. The single layer kernel can then be expressed in terms of solid harmonics

$$\begin{aligned} U_{ij}(\mathbf{x},\mathbf{y}) &= -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\{[\delta_{ij}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}}) - (\overrightarrow{\mathbf{Ox}})_j\frac{\partial}{\partial x_i}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}})]R_n^m(\overrightarrow{\mathbf{Oy}}) + \\ &\qquad \frac{\partial}{\partial x_i}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}})(\overrightarrow{\mathbf{Oy}})_j R_n^m(\overrightarrow{\mathbf{Oy}})\} \\ &= -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}[F_{ij}^1(\overrightarrow{\mathbf{Ox}})R_n^m(\overrightarrow{\mathbf{Oy}}) + F_i^2(\overrightarrow{\mathbf{Ox}})(\overrightarrow{\mathbf{Oy}})_j R_n^m(\overrightarrow{\mathbf{Oy}})], \end{aligned} \tag{5.7}$$

giving

$$\begin{aligned} \int_{S_y} U_{ij}(\mathbf{x},\mathbf{y})t_j(\mathbf{y})dS_y &= -\frac{1}{8\pi\mu}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}[F_{ij}^1(\overrightarrow{\mathbf{Ox}})\int_{S_y}R_n^m(\overrightarrow{\mathbf{Oy}})t_j(\mathbf{y})dS_y + \\ &\qquad F_i^2(\overrightarrow{\mathbf{Ox}})\int_{S_y}(\overrightarrow{\mathbf{Oy}})_j R_n^m(\overrightarrow{\mathbf{Oy}})t_j(\mathbf{y})dS_y], \end{aligned} \tag{5.8}$$

where

$$\begin{aligned} F_{ij}^1(\overrightarrow{\mathbf{Ox}}) &= \delta_{ij}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}}) - (\overrightarrow{\mathbf{Ox}})_j\frac{\partial}{\partial x_i}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}}) \\ F_i^2(\overrightarrow{\mathbf{Ox}}) &= \frac{\partial}{\partial x_i}\overline{S_n^m}(\overrightarrow{\mathbf{Ox}}). \end{aligned} \tag{5.9}$$

And the double layer kernel can be obtained from the single layer kernel

$$T_{ij}(\mathbf{x},\mathbf{y}) = [\frac{\delta_{jk}}{4\pi}\frac{\partial}{\partial x_i}(\frac{1}{r(\mathbf{x},\mathbf{y})}) + \mu(\frac{\partial U_{ij}(\mathbf{x},\mathbf{y})}{\partial y_k} + \frac{\partial U_{ik}(\mathbf{x},\mathbf{y})}{\partial y_j})]n_k(\mathbf{y}), \tag{5.10}$$

giving

$$\int_{S_y} T_{ij}(\mathbf{x},\mathbf{y})u_j(\mathbf{y})dS_y = -\frac{1}{8\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\{F_{ij}^1(\overrightarrow{\mathbf{Ox}})\int_{S_y}\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}[u_j(\mathbf{y})n_k(\mathbf{y}) +$$

$$u_k(\mathbf{y})n_j(\mathbf{y})]dS_y + F_i^2(\overrightarrow{\mathbf{Ox}})\int_{S_y}(\overrightarrow{\mathbf{Oy}})_j\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}[u_j(\mathbf{y})n_k(\mathbf{y}) + u_k(\mathbf{y})n_j(\mathbf{y})]dS_y\}(5.11)$$

In the derivation, the following expressions are used, corresponding to the three terms in Equation (5.10),

$$\int_{S_y}\frac{\delta_{jk}}{4\pi}\frac{\partial}{\partial x_i}(\frac{1}{r(\mathbf{x},\mathbf{y})})u_j(\mathbf{y})n_k(\mathbf{y})dS_y = \frac{1}{4\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}F_i^2(\overrightarrow{\mathbf{Ox}})\int_{S_y}2\delta_{jk}R_n^m(\overrightarrow{\mathbf{Oy}})$$

$$u_j(\mathbf{y})n_k(\mathbf{y})dS_y, (5.12)$$

$$\int_{S_y}\mu\frac{\partial U_{ij}(\mathbf{x},\mathbf{y})}{\partial y_k}u_j(\mathbf{y})n_k(\mathbf{y})dS_y = -\frac{1}{8\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\{F_{ij}^1(\overrightarrow{\mathbf{Ox}})\int_{S_y}\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}u_j(\mathbf{y})$$

$$n_k(\mathbf{y})dS_y + F_i^2(\overrightarrow{\mathbf{Ox}})\int_{S_y}[\delta_{jk}R_n^m(\overrightarrow{\mathbf{Oy}}) + (\overrightarrow{\mathbf{Oy}})_j\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}]u_j(\mathbf{y})n_k(\mathbf{y})dS_y\}(5.13)$$

$$\int_{S_y}\mu\frac{\partial U_{ik}(\mathbf{x},\mathbf{y})}{\partial y_j}u_j(\mathbf{y})n_k(\mathbf{y})dS_y = -\frac{1}{8\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\{F_{ik}^1(\overrightarrow{\mathbf{Ox}})\int_{S_y}\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_j}u_j(\mathbf{y})$$

$$n_k(\mathbf{y})dS_y + F_i^2(\overrightarrow{\mathbf{Ox}})\int_{S_y}[\delta_{jk}R_n^m(\overrightarrow{\mathbf{Oy}}) + (\overrightarrow{\mathbf{Oy}})_k\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_j}]u_j(\mathbf{y})n_k(\mathbf{y})dS_y\}$$

$$= -\frac{1}{8\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\{F_{ij}^1(\overrightarrow{\mathbf{Ox}})\int_{S_y}\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}u_k(\mathbf{y})$$

$$n_j(\mathbf{y})dS_y + F_i^2(\overrightarrow{\mathbf{Ox}})\int_{S_y}[\delta_{jk}R_n^m(\overrightarrow{\mathbf{Oy}}) + (\overrightarrow{\mathbf{Oy}})_j\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}]u_k(\mathbf{y})n_j(\mathbf{y})dS_y\}(5.14)$$

By summing up the above three equations, all the terms with δ_{jk} cancel each other.

From Equations (5.8) and (5.11), the multipole moments can be calculated from the sources on the boundary, following

$$M_{n,j}^m(\mathbf{O}) = \int_{S_y}\frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}[u_j(\mathbf{y})n_k(\mathbf{y}) + u_k(\mathbf{y})n_j(\mathbf{y})]dS_y - \frac{1}{\mu}\int_{S_y}R_n^m(\overrightarrow{\mathbf{Oy}})t_j(\mathbf{y})dS_y,$$

$$M_n^m(\mathbf{O}) = \int_{S_y} (\overrightarrow{\mathbf{Oy}})_j \frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial y_k}[u_j(\mathbf{y})n_k(\mathbf{y}) + u_k(\mathbf{y})n_j(\mathbf{y})]dS_y - \frac{1}{\mu}\int_{S_y} (\overrightarrow{\mathbf{Oy}})_j R_n^m(\overrightarrow{\mathbf{Oy}})t_j(\mathbf{y})dS_y, \quad (5.15)$$

where $M_{n,j}^m$ consists three components and M_n^m is a scalar. The four sets of multipole moments can be translated to four sets of local expansions defined at another point $\mathbf{O}'$ by

$$\begin{aligned} L_{n',j}^{m'}(\mathbf{O}') &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^{n'}\overline{S_{n+n'}^{m+m'}}(\overrightarrow{\mathbf{OO}'})M_{n,j}^m(\mathbf{O}); \\ L_{n'}^{m'}(\mathbf{O}') &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^{n'}\overline{S_{n+n'}^{m+m'}}(\overrightarrow{\mathbf{OO}'})(M_n^m(\mathbf{O}) - (\overrightarrow{\mathbf{OO}'})_j M_{n,j}^m(\mathbf{O})). \quad (5.16) \end{aligned}$$

The $M2L$ translation operators in the Stokes equation are exactly same as those in the Navier equation (Equation (4.14)). They are functions of only the distance between the points $\mathbf{O}$ and $\mathbf{O}'$. This can be seen as a convolution in space coordinates, and hence the FFT can be used to accelerate the calculation. Finally, the field at the destination point can be obtained from the local expansion coefficients $L_{n,j}^m$ and L_n^m by:

$$\int_{S_y} T_{ij}(\mathbf{x},\mathbf{y})u_j(\mathbf{y})dS_y - \int_{S_y} U_{ij}(\mathbf{x},\mathbf{y})t_j(\mathbf{y})dS_y = -\frac{1}{8\pi}\sum_{n=0}^{\infty}\sum_{m=-n}^{n} \{[\delta_{ij}R_n^m(\overrightarrow{\mathbf{O}'\mathbf{x}}) - (\overrightarrow{\mathbf{O}'\mathbf{x}})_j\frac{\partial}{\partial x_i}R_n^m(\overrightarrow{\mathbf{O}'\mathbf{x}})]L_{n,j}^m(\mathbf{O}') + \frac{\partial}{\partial x_i}R_n^m(\overrightarrow{\mathbf{O}'\mathbf{x}})L_n^m(\mathbf{O}')\}. \quad (5.17)$$

5.3 Numerical examples

5.3.1 Drag force on a fixed sphere in a tube

In this example, a Stokes problem consisting of a stationary rigid sphere in a cylinder is considered, as shown in Figure 5.1. The fluid flows from the left to the right and the drag force on the sphere is calculated. The cylinder has length $L = 1$ and radius $R = 1/3$. The radius of the sphere is r, ranging from 1/18 to 1/6. Non-slip boundary condition is prescribed on the surface of the sphere and cylinder. A pressure drop of 1 is prescribed: at the inlet, the tractions $t_y = 1$, and $t_x = t_z = 0$; and at the outlet, the tractions $t_x = t_y = t_z = 0$, where y direction is the flow direction. The resulting velocity profile at the inlet and outlet in the cylinder is quadratic, given by Figure 5.2. Drag force is normalized by that in infinite fluid

$$F = \frac{\int t_y dS}{6\pi\mu r U}, \tag{5.18}$$

where t_y is the traction in the flow direction and U is the average velocity in the flow direction at the inlet. The analytical solution, from Haberman and Sayre [27] using an algebraic stream function approach, is used to validate the numerical results,

$$\begin{aligned}\bar{F} = 2(1 - 0.66667\frac{r^2}{R^2} - 0.20217\frac{r^5}{R^5})/(1 - 2.1050\frac{r}{R} + 2.0865\frac{r^3}{R^3} \\ -1.7068\frac{r^5}{R^5} + 0.72603\frac{r^6}{R^6}).\end{aligned} \tag{5.19}$$

To solve this problem, three sets of mesh on the boundary of the computational domain are tested (Table 5.1), ranging from coarse to fine. The tolerance of the

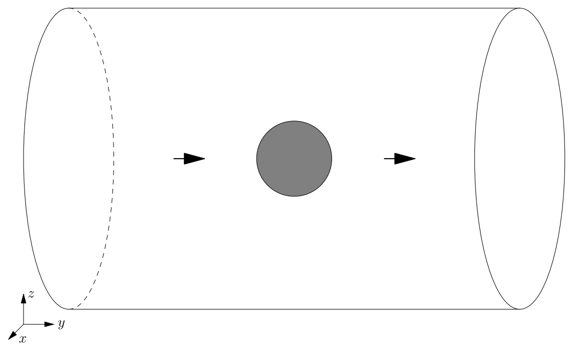


Figure 5.1: Stokes fluid flows from left to right. The drag force is calculated on the inside sphere.

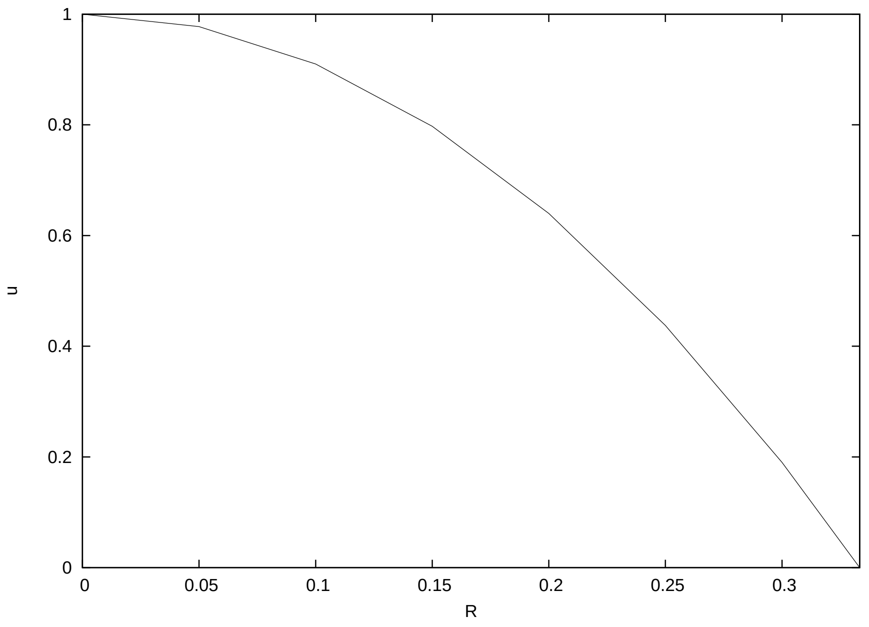
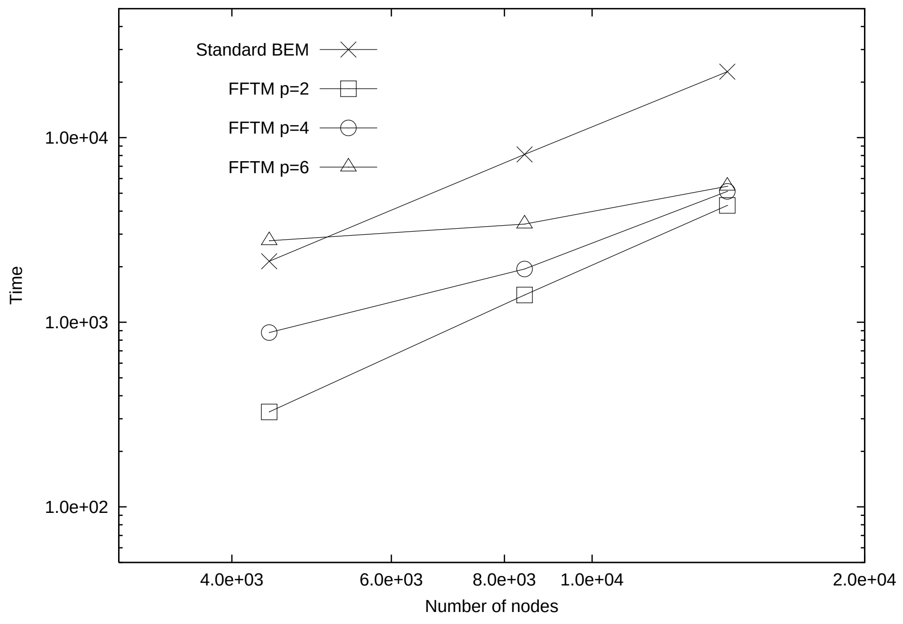
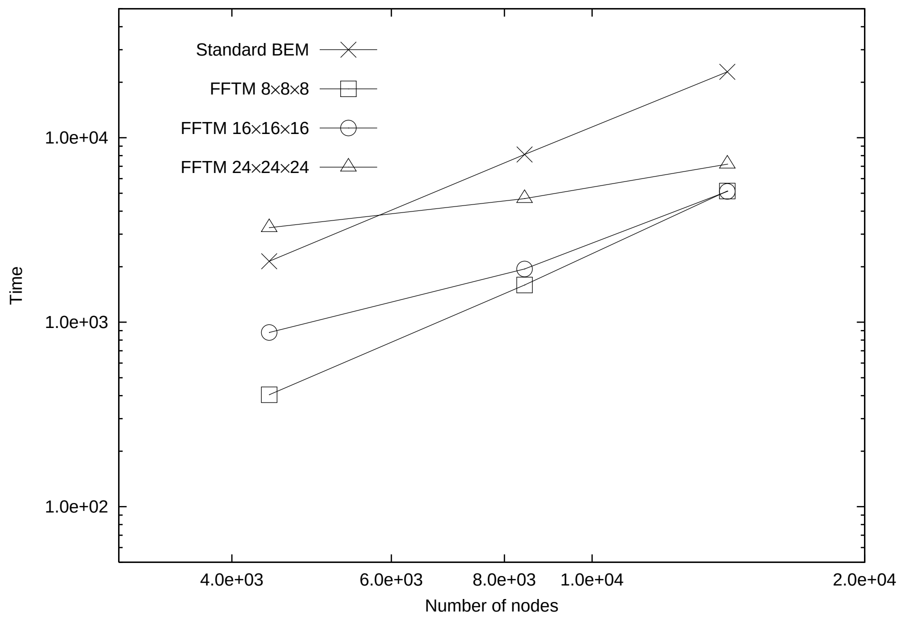


Figure 5.2: Normalized velocity u in the flow direction at the inlet and outlet. Since the velocity distribution is axis-symmetric, only the results on one arbitrary radius R are shown.



(a) With fixed number of cells $16 \times 16 \times 16$



(b) With fixed $p = 4$

Figure 5.3: Computation timings of FFTM and standard BEM

Table 5.1: Three sets of meshes are used to calculate the drag force on a single sphere inside a cylinder cube.

	Panels on the cylinder	Panels on the sphere	Total
Coarse	4160	238	4398
Medium	7560	862	8422
Fine	10920	3182	14102

residue in the GMRES is set as 10^{-4}. The largest case (with 14102 panels) needs 40 iterations in the GMRES procedure to converge. To show the efficiency of the fast algorithm, we run 40 iterations for all the cases. Figure 5.3 shows the timings of the FFTM, compared with the standard BEM, with different cell discretizations and expansion order. The FFTM reduces the computational time significantly when the problem size becomes large. The standard BEM using GMRES shows the typical computational complexity of $O(N^2)$, while the computational complexity of FFTM is lower, especially when higher order of expansion p and finer cell discretization are used. The FFTM introduces additional computational overhead, resulting in slightly higher or comparable computational time as the standard BEM for problems with a small number of panels. When a higher order of expansion is used, more operations are needed, and the computation time is longer, as shown in Figure 5.3(a). However, for large problems, the time increase is not significant compared with the total computational time, as the computational complexity decreases with the order of expansion p. Figure 5.3(b) shows that the computational complexity decreases with finer cell discretization. But a finer cell discretization also results in extra overhead, and thus this may not improve the computational efficiency when the problem size is small.

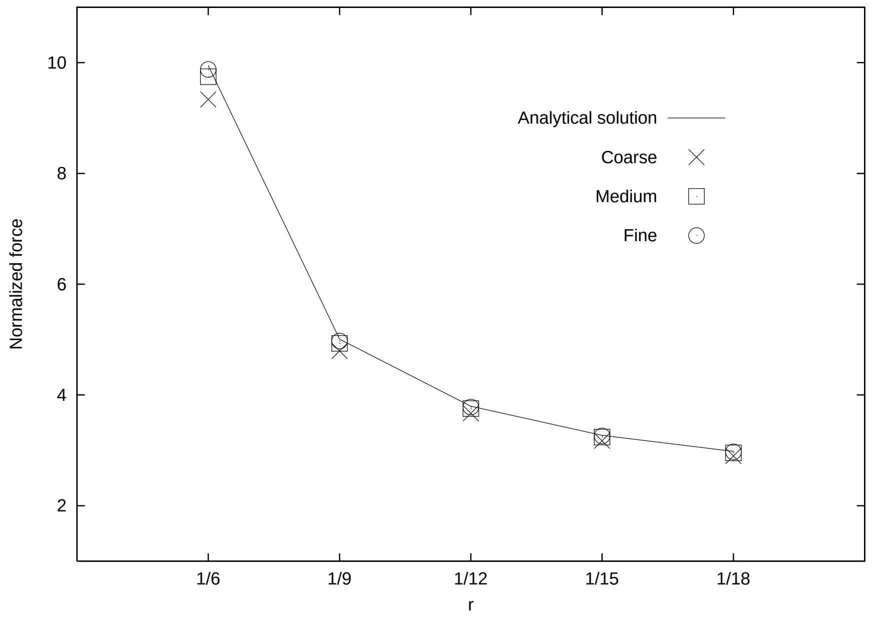


Figure 5.4: The drag force changes with the radius of the sphere.

Figures 5.4 and 5.5 shows the accuracy of the FFTM. The computational domain is discretized into $16 \times 16 \times 16$ cells and the expansion order (p) of the translations is equal to 4. In Figure 5.4, it can be seen that the normalised drag force on the sphere decreases when the sphere becomes smaller. For various dimensions of sphere, the results of FFTM agree very well with the analytical solution, especially when the fine mesh is employed. Figure 5.5 shows the actual errors in the numerical scheme. With finer mesh, the results are more accurate. In addition, the accuracy is improved with smaller sphere, since the absolute size of the elements is smaller on a smaller sphere.

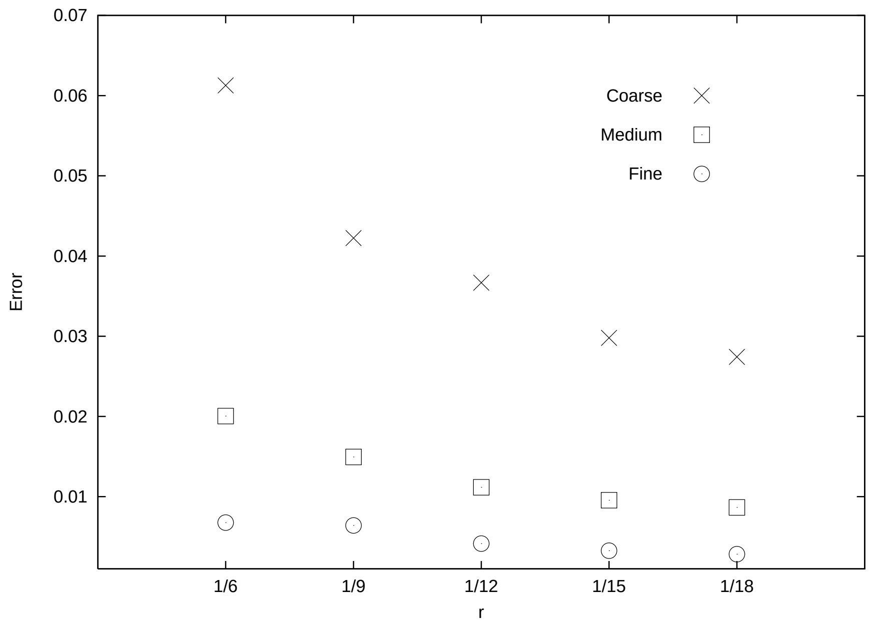


Figure 5.5: The accuracy of drag force changes with the radius of the sphere.

5.3.2 Drag force on numerous spheres in a tube

In this example, a more complex problem is considered, shown in Figure 5.6. The dimensions of the cylinder tube are the same with the previous example, $L = 1$ and $R = 1/3$. Numerous small spheres are placed randomly in the cylinder and kept stationary. The distribution of spheres is pseudo random, similar with that in Section 4.3.3. Again high concentration of spheres in a small region is not considered to avoid a very fine mesh. The boundary conditions are also the same with the previous example. Non-slip boundary condition is prescribed on the surface of the spheres and cylinder, and pressure drop is 1 between the inlet and outlet.

We look at two cases with different number of spheres, as shown in Table 5.2. In each case, 4160 panels are used to discretize the cylinder and 238 panels for each

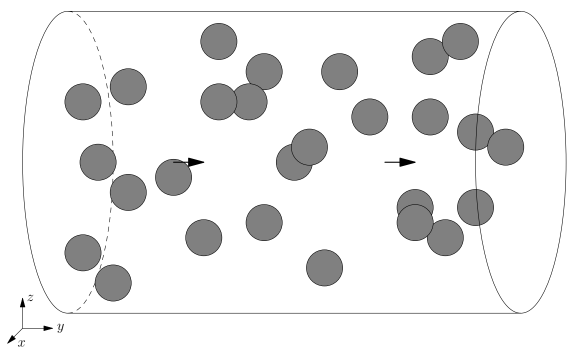


Figure 5.6: Stokes fluid flows from left to right. The average drag force is calculated on the inside spheres. The spheres are randomly distributed inside the tube.

sphere (238). In the second case, there are more panels, as there are more spheres. So it needs more GMRES iterations to obtain convergence. The simulations of the two cases need about three and nine hours approximately on an AMD Opteron processor running at 2.2 GHz.

Table 5.2: Case studies with 63 and 105 spheres

	Spheres (n)	Panels	Cells	p	Iterations	Time (Hour)
1	63	19154	$16 \times 16 \times 16$	4	60	~ 3
2	105	29150	$16 \times 16 \times 16$	4	100	~ 9

For an infinite number of randomly distributed spheres in an unconfined domain, the average drag force on the spheres is related to the spheres' solid volume fraction ϕ. Kim and Russel [36] derived the expression of the average drag force

$$F_{Kim} = 1 + \frac{3}{\sqrt{2}}\sqrt{\phi} + \frac{135}{64}\phi \ln \phi + 16.456\phi. \tag{5.20}$$

The spheres in this example are confined in a cylinder tube, so ϕ is defined as

$$\phi = \frac{4/3 n \pi r^3}{\pi L R^2} = \frac{4nr^3}{3LR^2}, \tag{5.21}$$

where n is the number of spheres inside the cylinder tube, and $L = 1$ in this example. The interest here is to investigate whether the relation of average drag force and volume fraction (ϕ), Equation (5.20), is still valid in this problem.

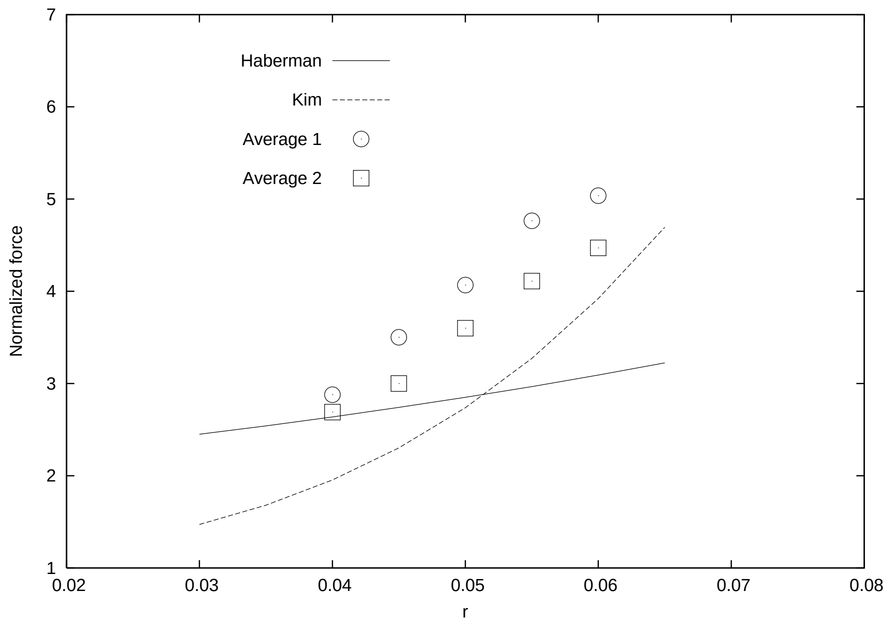


Figure 5.7: Average drag force on 63 spheres. Harberman: drag force on a single sphere in a cylinder (Equation (5.19)); Kim: drag force on randomly distributed spheres in an unconfined domain (Equation (5.20)); Average 1: the average of all the spheres inside the cylinder; Average 2: the average of the spheres whose distance from the axis is less than $0.6R$

Figures 5.7 and 5.8 show the average drag force on the spheres. The present results of the average force on the numerous spheres in a cylinder are compared with the drag force on a single sphere in a cylinder and the average drag force on spheres in an unconfined domain. In this work, two average forces, Average 1 and Average 2, are discussed. The Average 1 is calculated from all the spheres inside the cylinder and the Average 2 is calculated only from the spheres whose distance from the

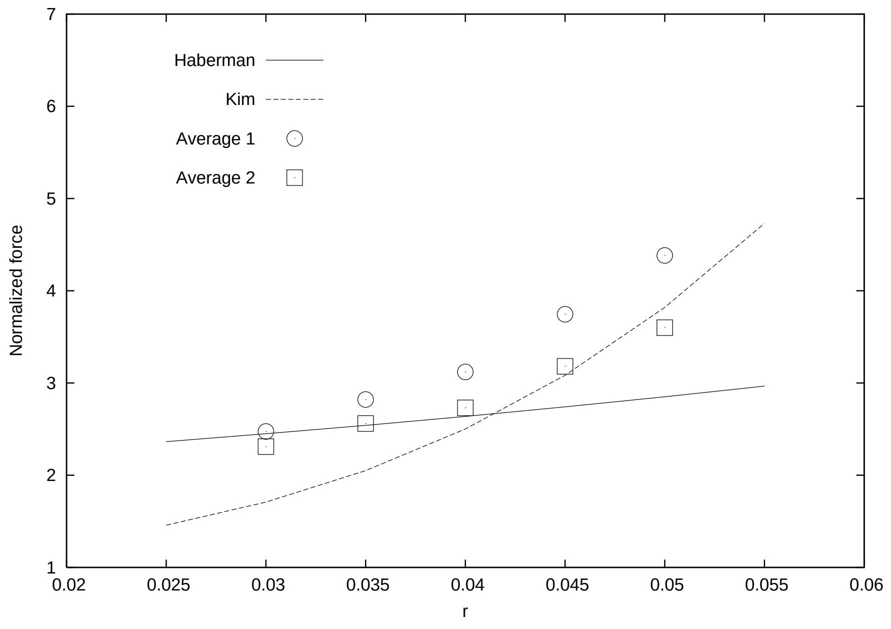


Figure 5.8: Average drag force on 105 spheres. Harberman: drag force on a single sphere in a cylinder (Equation (5.19)); Kim: drag force on randomly distributed spheres in unconfined domain (Equation (5.20)); Average 1: the average of all the spheres inside the cylinder; Average 2: the average of the spheres whose distance from the axis is less than $0.6R$

cylinder axis is less than $0.6R$. This distance is arbitrary chosen so that spheres near the wall are not considered, and it reduces the wall effects. It better shows the fluid force on spheres in an unconfined domain, and compares better with Kim and Russel approximation [36]. When the sphere radius is small, the interactions among the spheres are negligible. The average drag force is close to the Harberman and Sayre's results that are for one sphere located in the center line of the cylinder. Since the Average 2 calculates the spheres nearer to the center line, it gives better estimation than Average 1. With bigger spheres, the interactions among the spheres become more dominant. When the radius r of the sphere is larger (0.052 in Figure 5.7 and 0.041 in Figure 5.8), the results get nearer to those of many randomly distributed spheres (Kim and Russel). The results of the Average 1 are larger than Kim and Russel's expression. This is beacause the cylinder wall will induce additionally drag force on the spheres near the wall (which is not present in the calculation of Kim and Russel). The spheres nearer to the axis resemble the spheres in the Kim and Russel's model. Hence, the Average 2 gives results closer to Kim and Russel's results. With more spheres, the problem approximates the model used by Kim and Russel better, so the results in Figure 5.8 are closer to the results of Kim and Russel.

5.4 Summary of FFTM for Stokes equation

In this chapter, the FFTM algorithm is extended to solve the Stokes equation. The same methods of derivation and implementation as in solving the Navier equation can be employed to solve Stokes equation. There are only slight modifications to the translation operators compared with the Navier equation. The fast Stokes algorithm reduces the computational time when the problem becomes large, while maintaining good accuracy. In calculating the average drag force, we have demonstrated cases with more than 100 spheres and the total number of degrees of freedom exceeds 90,000. When the spheres are small, the interactions among the spheres can be ignored and the average drag force can be approximated by that of the single sphere inside a cylinder, given by Haberman and Sayre. When the spheres get bigger, the interactions among the spheres are dominant. In this case, the average drag force on the spheres get closer to the estimation, given by Kim and Russel, for randomly distributed spheres in an unconfined domain. But this approximation is only good for the sphere away from the wall of the cylinder.

Chapter 6

Non-linear Poisson-type equation

Non-linear equation is very important in engineering, such as Navier-Stokes equation and Navier equation with non-linear material properties. To illustrate the application to non-linear problem, the FFTM is employed to solve non-linear Poisson-type equation in this chapter. The BEM becomes less attractive when the partial differential equation has a non-linear term, as expensive volume integration and interior evaluation are required. In the solution procedure of the non-linear equation, the Poisson equation needs to be solved. To handle the non-homogeneous term in the Poisson equation, two fast methods are compared. One method introduces the multipole to accelerate the volume integration, and the other calculates a particular solution through the FFT. Since the second method is faster and more accurate, it is adopted in the new fast non-linear scheme. In each iteration of the scheme, there are mainly three steps, namely calculating a particular solution with the FFT, solving the resulted Laplace equation with the FFTM and updating the interior values also with the FFTM. The new scheme is tested by several non-linear Poisson-type equations with various non-linear terms.

6.1 Overview of BEM solving non-homogeneous and non-linear equations

When solving the non-homogeneous equation, such as Poisson equation, with the BEM, various approaches have been proposed to avoid or alleviate the burden due to the non-homogeneous term. The meshless methods, such as the dual reciprocity method (DRM) [61], multiple reciprocity method (MRM) [54] and particular solution method (PSM) [29], were commonly used to maintain the advantage of boundary only discretization in the BEM. However, Ingber [33] proposed that a cell-based direct volume integration scheme can be used to give better accuracy. In addition, when such a volume integration scheme is coupled with the FMM, it significantly improves the computational efficiency over the meshless methods. In order to apply the volume integration method in complex domain, Mammoli [44] developed the auxiliary domain method (ADM) to simplify the mesh generation. Various fast algorithms have been applied to accelerate the solution procedure. The group of Greengard [46, 23, 15] provided a series of two dimensional fast Poisson solver based on the FMM. Ding et al. [11] introduced a fast cell-based approach, based on the pFFT technique, that accelerates the surface integration as well as the volume integration. Ying et al. [87] handled the non-homogeneous part with a particular solution, while solving the homogeneous part with FMM. They used the FFT to calculate the particular solution, which is much faster than the global shape function method used in the PSM.

When the non-homogeneous term is a non-linear function of the unknown solution, the equation becomes a non-linear equation. The DRM and MRM have been applied to solve such non-linear equation. There were also some modified and improved methods [85, 64], based on the DRM. Liao [38] applied the general boundary element method to solve strongly non-linear problem. With the help of the homotopy analysis method (HAM), the general boundary element method is valid even for governing equations and boundary conditions that do not contain any linear terms. Most of the above algorithms were applied to solve two dimensional problems with small number of freedom, but rarely in three dimensional ones. Recently, Ding and Ye [10] applied the pFFT to solve some three dimensional weakly non-linear problems, where the number of degrees of freedom reaches 4000.

Table 6.1: Different BEM methods solving Poisson and non-linear equation

	Poisson equation	Non-linear equation
Conventional method	DRM [61] MRM [54] PSM [29]	DRM [61] MRM [54] Liao [38]
Fast method	Ingber et al. [33] Group of Greengard [46, 23, 15] Ding et al. [11] Ying et al. [87]	Ding and Ye [10]

The above methods are summarized in Table 6.1. The conventional methods have been applied in solving many problems, but the degrees of freedom of the problems cannot be very large. The fast methods are comparatively new and can be used to solve large scale problems. Most of the fast algorithms treat the non-homogeneous term of the Poisson equation or non-linear equation by accelerated volume integration. Ingber et al. [33] and some of Greengard's work [46, 15] accelerated the volume integration by the FMM, while Ding et al. [11] and Ding and Ye [10], by the pFFT technique. The others calculated a particular solution in a fast manner. Greengard and Lee [23] calculated particular solutions with spectral method in a decomposed domain and patches the solutions together with the FMM. Ying et al. [87] obtained a particular solution with the FFT. In this chapter, two methods handling the non-homogeneous part in the Poisson equation are compared, namely multipole accelerated volume integration and method of particular solution obtained from FFT. In [33], Ingber et al. calculated the particular solution using radial basis functions and claimed that multipole accelerated method is both faster and more accurate than the particular solution method. However, if the FFT is adopted, in place of the radial basis function method, to calculate a particular solution, it is found that this method has great advantages in efficiency and accuracy over the multipole accelerated volume integration method.

Table 6.1 indicates that relatively few work has been done on the fast non-linear solver. In this chapter, a fast non-linear algorithm is presented, based on the FFTM, to solve large three dimensional non-linear problem. The non-linear equation is Poisson-type equation, in which the linear term is the Laplace operator and the non-homogeneous term is a non-linear function of the unknown solution. Simple Richardson iterative scheme is used to solve the non-linear equation. Each iteration includes calculating a particular solution through fast Fourier transform (FFT), solving the resulting Laplace equation with the FFTM and evaluating the interior values also with the FFTM. With the above ideas, large non-linear problems can be solved efficiently. The numerical examples demonstrate the method for problems with number of degrees of freedom exceeding 30,000.

6.2 Methodology

To solve a non-linear equation

$$\nabla^2 u(\mathbf{x}) = f(u), \tag{6.1}$$

where $f(u)$ is a non-linear function of u, two approaches can be employed. One approach introduces u at interior points as additional unknowns and collocates integral formulation at these points. The other approach gives a initial guess for the intierior value of u. A Poisson equation is then solved to find the unknowns on the boundary. The obtained boundary values together with the given boundary conditions are then used to update interior value of u and iteration continues until a convergence tolerance is met. The second scheme is easier to implement and less expensive in memory cost, but slower to converge, compared with the first scheme. In this chapter, the second scheme is implemented.

6.2.1 BEM for Poisson equation

In each iteration of non-linear solver, the Poisson equation can be rewritten into a direct boundary integral formulation

$$c(\mathbf{x})u(\mathbf{x}) + \int_S H(\mathbf{x},\mathbf{y})u(\mathbf{y})dS(\mathbf{y}) + \int_\Omega G(\mathbf{x},\mathbf{y}) f d\Omega(\mathbf{y}) = \int_S G(\mathbf{x},\mathbf{y})\frac{\partial u(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}dS(\mathbf{y}). \tag{6.2}$$

Here, c is the free term, and $G(\mathbf{x},\mathbf{y})$ and $H(\mathbf{x},\mathbf{y})$ correspond to the single layer kernel and double layer kernel, respectively,

$$\begin{aligned} G(\mathbf{x},\mathbf{y}) &= \frac{1}{4\pi r}, \\ H(\mathbf{x},\mathbf{y}) &= \frac{\partial G(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})} = \frac{1}{4\pi r^3}(\mathbf{x}-\mathbf{y})\cdot\mathbf{n}(\mathbf{y}), \end{aligned} \tag{6.3}$$

where $r = |\mathbf{y}-\mathbf{x}|$. The surface integrals, corresponding to homogeneous part (Laplace equation), can be calculated rapidly by fast algorithms, such like FMM, pFFT or FFTM. All of them have comparable $(N \log N)$ efficiency. In this chapter, the FFTM with the solid harmonics is adopted, since the use of solid harmonics results in a more compact storage of the translation matrices, as shown in Section 4.2. Except the choice of harmonic functions, the implementation procedure is the same as that in Section 3.2.2. The non-homogeneous term is transformed into a volume integration term in the integral equation. It is very expensive to discretize the interior domain and perform the volume integration with traditional BEM. However, the volume integration can be accelerated by multipole method.

An alternative way of solving the Poisson equation is to separate the solution into the particular solution u_p and homogeneous solution u_h,

$$u(\mathbf{x}) = u_p(\mathbf{x}) + u_h(\mathbf{x}). \tag{6.4}$$

The particular solution u_p satisfies the Poisson equation, but not necessarily the boundary conditions. The homogeneous solution u_h satisfies the corresponding Laplace equation and enforces the boundary conditions of the original Poisson equation,

$$\begin{aligned}
\nabla^2 u_p(\mathbf{x}) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega \\
\nabla^2 u_h(\mathbf{x}) &= 0, \quad \mathbf{x} \in \Omega \\
\frac{\partial u_h(\mathbf{x})}{\partial \mathbf{n}} &= f_1(\mathbf{x}) - \frac{\partial u_p(\mathbf{x})}{\partial \mathbf{n}}, \quad \mathbf{x} \in S_1 \\
u_h(\mathbf{x}) &= f_2(\mathbf{x}) - u_p(\mathbf{x}), \quad \mathbf{x} \in S_2,
\end{aligned} \tag{6.5}$$

where f_1 and f_2 are the boundary conditions of the original Poisson equation. The homogeneous part is also solved with the FFTM algorithm, and the particular solution is obtained by finding the Fourier transform of f.

6.2.2 Multipole accelerated volume integration

For the standard BEM implementation, the volume integration is performed for every node point, and then the result is added to the right hand side of the Laplace solver. To avoid the interior discretization, the cells that are used in the FFTM algorithm are used as interior discretization for the volume integration. The volume integral is approximated by the Gaussian quadrature. When a cell intersects the boundary, shown in Figure 6.1, the values at the Gauss points outside the boundary are set to zero. This process provides a simple way of handling the intersection of the boundary and the cells. Accuracy can be improved by sub-dividing the original cell into smaller cells or increasing the number of Gauss points used.

The volume integration can be accelerated by the FFTM, which includes representing the sources located at the interior Gauss points into multipoles, translating the multipoles to local expansions, and lastly calculating the potential at the destination node points. However, there is an even faster method to perform the volume integration. In the current method, the interior sources are represented into multipoles, and then summed up with the multipoles from the interior sources and surface sources. The resulted multipoles are substituted into the fast Laplace solver to give the results for the Poisson equation. In other words, the surface and volume integrals can be treated uniformly by multipole representations.

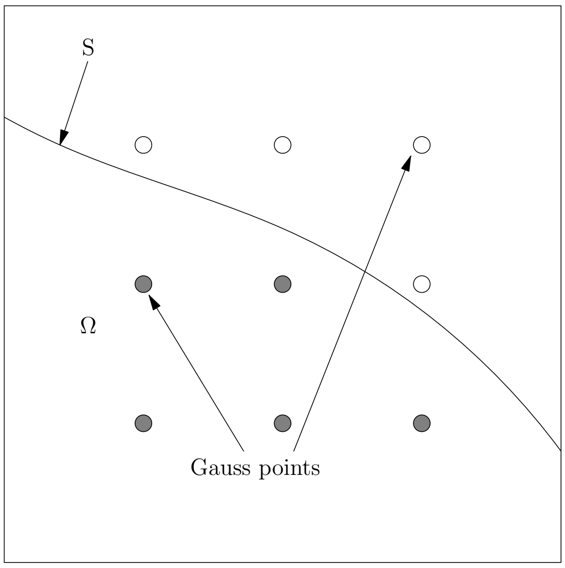


Figure 6.1: When a cell intersects with the boundary, the values of the Gauss points outside the boundary are set to zero.

Very small cell is not always viable to keep computational efficiency for the surface

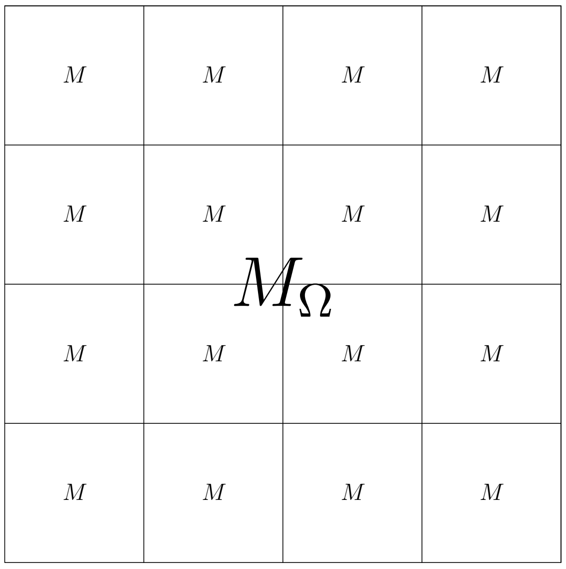


Figure 6.2: A cell being sub-divided into smaller cells to improve the accuracy of volume integration via FFTM. The multipoles M in the smaller cells are transformed to initial cell center's M_Ω.

integration, as shown by [59], while small cell is always preferred for the volume integration in the far field. So, two sets of cells are used: bigger cells for the surface integration and smaller cells for volume integration. The cells used for surface integral are further divided into smaller cells, as shown in Figure 6.2. The multipole moments for the volume sources are obtained at the smaller cell centers, and then translated to the cell centers of the bigger cells,

$$M^m_{\Omega,n} = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R^{m'}_{n'} M^{m-m'}_{n-n'}. \tag{6.6}$$

The multipole moments for the volume sources (M_Ω) are then combined with those from the surface sources.

For the near field (shaded area in Figure 6.3), the volume integral is performed by Gaussian quadrature. When the evaluation node point is in or close to a cell

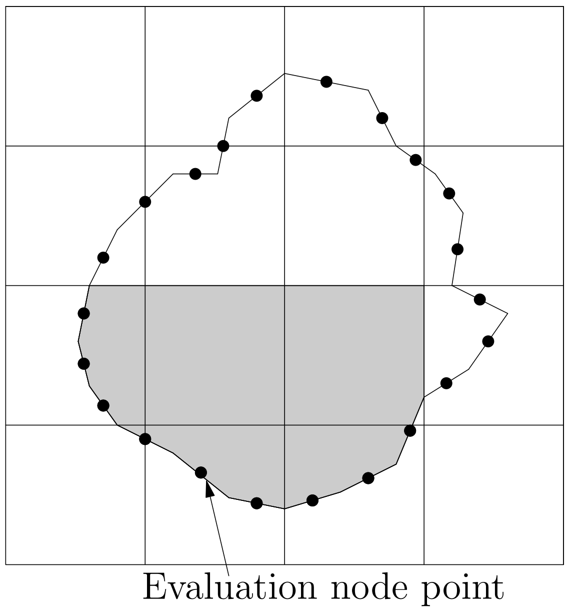


Figure 6.3: For the near field (shaded area), the standard volume integration is needed.

that the integration is performed, the weakly singularity appears. The weakly singularity can be regularized by the coordinate transform

$$\begin{aligned}\int_\Omega G(\mathbf{x},\mathbf{y})f(\mathbf{y})d\Omega(\mathbf{y}) &= \int_{y_{11}}^{y_{12}}\int_{y_{21}}^{y_{22}}\int_{y_{31}}^{y_{32}}\frac{f(y_1,y_2,y_3)}{4\pi r(y_1,y_2,y_3)}dy_1dy_2dy_3 \\ &= \int_0^{r_b(\theta,\phi)}\int_0^{\pi}\int_0^{2\pi}\frac{1}{4\pi}f(r,\theta,\phi)r\sin\theta drd\theta d\phi. \quad (6.7)\end{aligned}$$

6.2.3 Particular solution method with FFT

Except the volume integration, there is also another way to handle the non-homogeneous term in the Poisson equation. The solution of the Poisson equation can be separated into a homogeneous solution and a particular solution. The homogeneous solution is obtained by solving the Laplace equation and the particular solution is calculated fast with the help of the FFT.

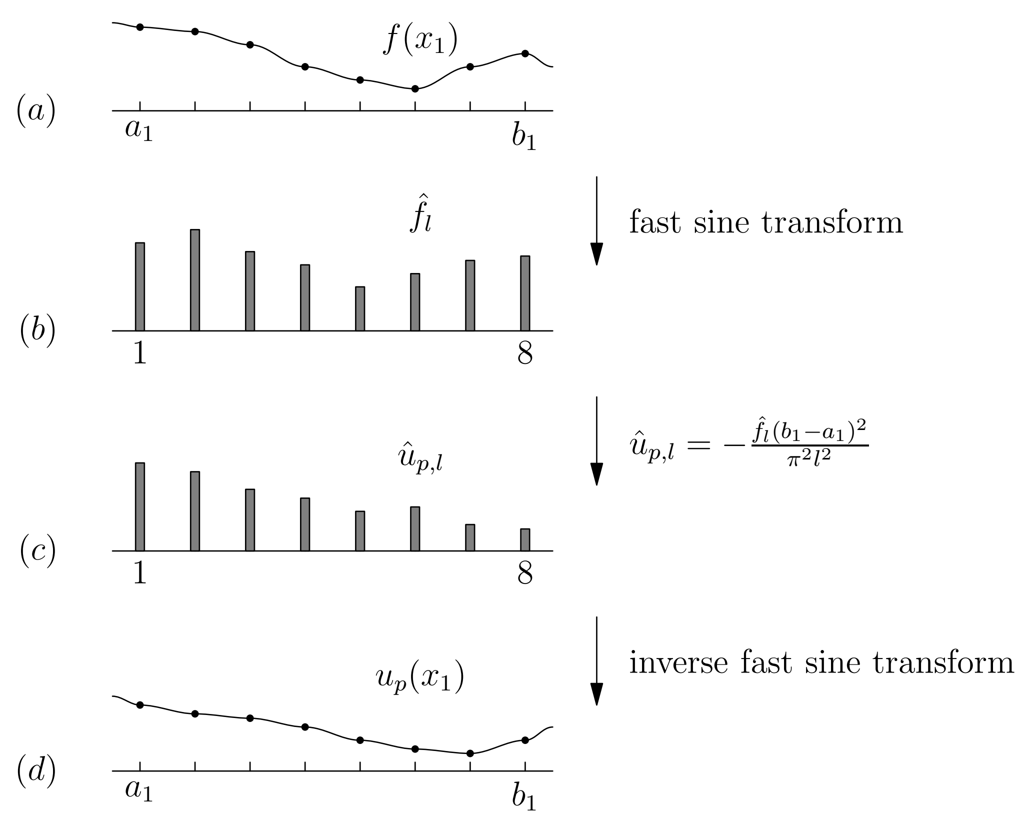


Figure 6.4: One dimensional illustration of how to obtain a particular solution from FFT

In the present implementation, $f(\mathbf{x})$ is extended to the rectangular domain C defined as $[a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$, used in FFTM. The rectangular domain C is discretized by a regular grid, preferably $2^p \times 2^q \times 2^r$ for implementing FFT, where p, q and r are positive integers. The grid is used to calculate the particular solution via the FFT. Figure 6.4 illustrates the main steps to calculate a particular solution using FFT for a 1D example. Figure 6.4(a) shows the non-homogeneous term f of the Poisson equation. The Fourier coefficients $\hat{f}$, given in Figure 6.4(b), can be obtained by the fast Fourier sine transform. In 3D, the coefficients $\hat{f}_{lmn}$ are given by

$$f(x_1, x_2, x_3) = \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \hat{f}_{lmn} \sin(\frac{x_1 - a_1}{b_1 - a_1} l\pi) \sin(\frac{x_2 - a_2}{b_2 - a_2} m\pi) \sin(\frac{x_3 - a_3}{b_3 - a_3} n\pi), \tag{6.8}$$

and

$$\begin{aligned}\hat{f}_{lmn} = \frac{8}{(b_1 - a_1)(b_2 - a_2)(b_3 - a_3)} \int_{a_1}^{b_1}\int_{a_2}^{b_2}\int_{a_3}^{b_3} f(x_1, x_2, x_3) \sin(\frac{x_1 - a_1}{b_1 - a_1} l\pi) \\ \sin(\frac{x_2 - a_2}{b_2 - a_2} m\pi) \sin(\frac{x_3 - a_3}{b_3 - a_3} n\pi) dx_1 dx_2 dx_3.\end{aligned} \tag{6.9}$$

Again, $f(\mathbf{x}) = 0$, when the grid point $\mathbf{x}$ falls outside the domain of the problem Ω. The particular solution to the Poisson equation can readily be obtained by

$$u_p(x_1, x_2, x_3) = \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \hat{u}_{p,lmn} \sin(\frac{x_1 - a_1}{b_1 - a_1} l\pi) \sin(\frac{x_2 - a_2}{b_2 - a_2} m\pi) \sin(\frac{x_3 - a_3}{b_3 - a_3} n\pi), \tag{6.10}$$

where,

$$\hat{u}_{p,lmn} = -\frac{\hat{f}_{lmn}}{\pi^2} / [\frac{l^2}{(b_1 - a_1)^2} + \frac{m^2}{(b_2 - a_2)^2} + \frac{n^2}{(b_3 - a_3)^2}]. \tag{6.11}$$

Equation (6.11) is the simple algebraic operation in frequency domain to obtain $\hat{u}_p$. This is also illustrated in Figure 6.4(c) for the 1D example. The inverse fast Fourier sine transform is then used to calculate the particular solution u_p, as illustrated in Figure 6.4(d).

The particular solutions on the grid points are obtained from inverse fast Fourier sine transform on $\hat{u}$. The particular solutions at the nodes on the boundary are then obtained by three-dimension 64-point Lagrange interpolation. In Figure 6.5, a one-dimension example is given to show how the Lagrange interpolation works. The interpolation formulation is

$$u_p(x) = \sum_{k=0}^{3} u_p(x_k) L_k(x). \tag{6.12}$$

So

$$\begin{aligned}
L_0(x) &= -\frac{1}{6h^3}(x-x_1)(x-x_2)(x-x_3), \\
L_1(x) &= \frac{1}{2h^3}(x-x_0)(x-x_2)(x-x_3), \\
L_2(x) &= -\frac{1}{2h^3}(x-x_0)(x-x_1)(x-x_3), \\
L_3(x) &= \frac{1}{6h^3}(x-x_0)(x-x_1)(x-x_2),
\end{aligned} \tag{6.13}$$

in which h is the equidistant step size used. The derivative of u_p, which is needed

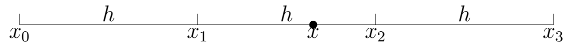


Figure 6.5: One dimensional 4-point Lagrange interpolation

in Equation (6.5), is obtained from performing derivative on both sides of Equation

(6.12), as follows

$$\frac{\partial u_p(x)}{\partial x} = \sum_{k=0}^{3} f(x_k) \frac{\partial L_k(x)}{\partial x}, \tag{6.14}$$

where

$$\begin{aligned}
\frac{\partial L_k(x)}{\partial x} &= -\frac{1}{6h^3}[(x-x_2)(x-x_3)+(x-x_1)(x-x_3)+(x-x_1)(x-x_2)], \\
\frac{\partial L_k(x)}{\partial x} &= \frac{1}{2h^3}[(x-x_2)(x-x_3)+(x-x_0)(x-x_3)+(x-x_0)(x-x_2)], \\
\frac{\partial L_k(x)}{\partial x} &= -\frac{1}{2h^3}[(x-x_1)(x-x_3)+(x-x_0)(x-x_3)+(x-x_0)(x-x_1)], \\
\frac{\partial L_k(x)}{\partial x} &= \frac{1}{6h^3}[(x-x_1)(x-x_2)+(x-x_0)(x-x_2)+(x-x_0)(x-x_1)].
\end{aligned} \tag{6.15}$$

6.2.4 FFTM for non-linear Poisson-type equation

When solving the non-linear equation, an iterative scheme is used. At the beginning of each iteration (t), the u_t and $\partial u_t/\partial \mathbf{n}$ at the boundary nodes and u_t at the interior area are known. In each iteration, the following Poisson equation is solved to calculate u_{t+1} and $\partial u_{t+1}/\partial \mathbf{n}$ on the boundary:

$$\begin{aligned}
\nabla^2 u_{t+1}(\mathbf{x}) &= f(u_t), \quad \mathbf{x} \in \Omega \\
\frac{\partial u_{t+1}(\mathbf{x})}{\partial \mathbf{n}} &= f_1(\mathbf{x}), \quad \mathbf{x} \in S_1 \\
u_{t+1}(\mathbf{x}) &= f_2(\mathbf{x}), \quad \mathbf{x} \in S_2.
\end{aligned} \tag{6.16}$$

From the new values on the boundary, u_{t+1} inside the boundary is updated. The advantages of the method of particular solution in Section 6.2.3 will be seen clearly from the comparison in the following numerical examples in the next section. So this method is chosen to solve the non-linear equation. The fast non-linear algorithm needs the following steps in each iteration:

1. calculate $f(u_t)$ at the interior grid points;

2. calculate the particular solution $u_{p,t+1}$ on the grid points from $f(u_t)$ through the FFT and then interpolate on the node points to obtain $u_{p,t+1}$ and $\partial u_{p,t+1}/\partial \mathbf{n}$. These are used to obtain the new boundary conditions for the homogeneous Laplace equation (details in Section 6.2.3);

3. solve the homogeneous part, a Laplace equation, to obtain $u_{h,t+1}$ and $\partial u_{h,t+1}/\partial \mathbf{n}$ on the boundary nodes with the FFTM (details in Section 3.2.2);

4. evaluate the interior values $u_{h,t+1}$ due to $u_{h,t+1}$ and $\partial u_{h,t+1}\partial/\mathbf{n}$ on the boundary using the FFTM

$$u_{h,t+1}(\mathbf{x}) = \int_S G(\mathbf{x},\mathbf{y})\frac{\partial u_{h,t+1}(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}dS(\mathbf{y}) - \int_S H(\mathbf{x},\mathbf{y})u_{h,t+1}(\mathbf{y})dS(\mathbf{y}); \quad (6.17)$$

5. sum up the particular solution and the homogeneous solution to get u_{t+1} and $\partial u_{t+1}/\partial \mathbf{n}$ on the boundary nodes and u_{t+1} on the grid points;

6. compare u_t, $\partial u_t/\partial \mathbf{n}$ and u_{t+1}, $\partial u_{t+1}/\partial \mathbf{n}$ on the node points. If the difference is smaller than a pre-set tolerance, the scheme is deemed to have converged.

In the step 4, we need to calculate the unknown solution at all the interior grid points. Since there is no volume integral in Equation (6.17), the fast surface integration algorithm is employed without any change for the interior values. If the distance between a evaluation point and a boundary element is very small (smaller than half size of a typical element), the analytical nearly singular integration algorithm in [28] is implemented to obtain good accuracy.

6.3 Numerical examples

In this section, five numerical examples are given to investigate different aspects of the fast non-linear solver. For all the problems, the computational domain Ω is a sphere with radius 0.5 and the sphere center is the origin of the $x_1x_2x_3$ coordinate. The boundary condition is the Dirichlet boundary condition. The measure of the error is defined in the L_2 norm as

$$Error = \sqrt{\frac{\sum_{i=1}^{N} |\partial u(\mathbf{x}_i)/\partial \mathbf{n} - \partial u^*(\mathbf{x}_i)/\partial \mathbf{n}|^2}{\sum_{i=1}^{N} |\partial u^*(\mathbf{x}_i)/\partial \mathbf{n}|^2}}, \tag{6.18}$$

when an analytical solution u^* is available.

For surface integration, constant triangular elements (plane panels) with one node at the element center are used. The numerical integration is performed over these elements using local intrinsic coordinates. When $\mathbf{x}$ and $\mathbf{y}$ are on different elements, the standard Gaussian quadrature (with 7 Gauss points over each element) is applied to perform the integration. When $\mathbf{x}$ and $\mathbf{y}$ are on the same element ($\mathbf{x} = \mathbf{y}$), weak ($1/r$) or strong ($1/r^2$) singularities appear. The weak singularity is removed by transforming the triangular elements to a quadrilateral domain on which 8×8 Gauss points are used for Gauss quadrature. The free term c does not need to be calculated explicitly in the direct BEM; it can be obtained by physical considerations such as arbitrary shifting of datum in potential problems or arbitrary rigid body motion in mechanics problems. This technique enables the free term and the strongly singular integrals in the direct BEM formulation to be calculated together. Four different surface discretizations are used, with the total number of nods being 4858, 8566, 19234 and 33884. When implementing the FFTM to accelerate the surface integration, the rectangular domain C is discretized into $16 \times 16 \times 16$ cells, and two expansion orders $p = 4$ and $p = 6$ are compared.

To compare the two methods in Sections 6.2.2 and 6.2.3 fairly, the same number of Gauss points are used to perform the volume integration as the number of grid points to calculate the particular solution in the rectangular domain C. This also makes the number of the Gauss points and the number of grid points inside the computational domain Ω to be almost the same. For convenience, the Gauss points used in the volume integration are also referred to as "grid points". This means that regularly distributed grid points are used in calculating the particular solution from FFT, but irregularly distributed grid points are used in performing the volume integration. In the following examples, three different sets of grid points are studied, namely $128 \times 128 \times 128$, $256 \times 256 \times 256$ and $512 \times 512 \times 512$. The volume integration is calculated with $8 \times 8 \times 8$ grid points (Gauss points) in each cell. With $16 \times 16 \times 16$ cells used for accelerating the surface integration, the number of grid points, used for the accelerated volume integration, is $(16 \times 8) \times (16 \times 8) \times (16 \times 8) = 128 \times 128 \times 128$. When the $16 \times 16 \times 16$ cells are further divided to perform the volume integration, the number of grid points can be $(32 \times 8) \times (32 \times 8) \times (32 \times 8) = 256 \times 256 \times 256$ and $(64 \times 8) \times (64 \times 8) \times (64 \times 8) = 512 \times 512 \times 512$. Also, two expansion order $p = 4$ and $p = 6$ are used to study the accuracy of multipole transform for interior sources.

6.3.1 Poisson equation with a constant non-homogeneous term

In this example, a Poisson equation with a constant non-homogeneous term is considered. The equation, boundary condition and analytical solution are given in Equation (6.19)

$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= 1, \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= \frac{x_1^2}{2}, \\ \frac{\partial u^*}{\partial \mathbf{n}} &= x_1 n_1, \quad \mathbf{x} \in S, \end{aligned} \tag{6.19}$$

where x_1 is the first component of the coordinate and n_1 is the first component of the normal direction.

Figures 6.6 and 6.7 show the timings of different methods handling the non-homogeneous term. The standard method performs the volume integration with the Gaussian quadrature for all the node points, the multipole method translates the far sources into multipoles and performs the standard integration for the near field , and the particular method calculates the particular solution with the FFT for all the node points. Both of the two fast algorithms save a lot of computational time. Calculating the particular solution is about one order of magnitude faster than the multipole accelerated volume integration and about three orders of magnitude faster than the standard method with the same number of grid points. Increasing the expansion order p from 4 to 6 only increases the computational time of the accelerated volume integration marginally. For all the three methods,

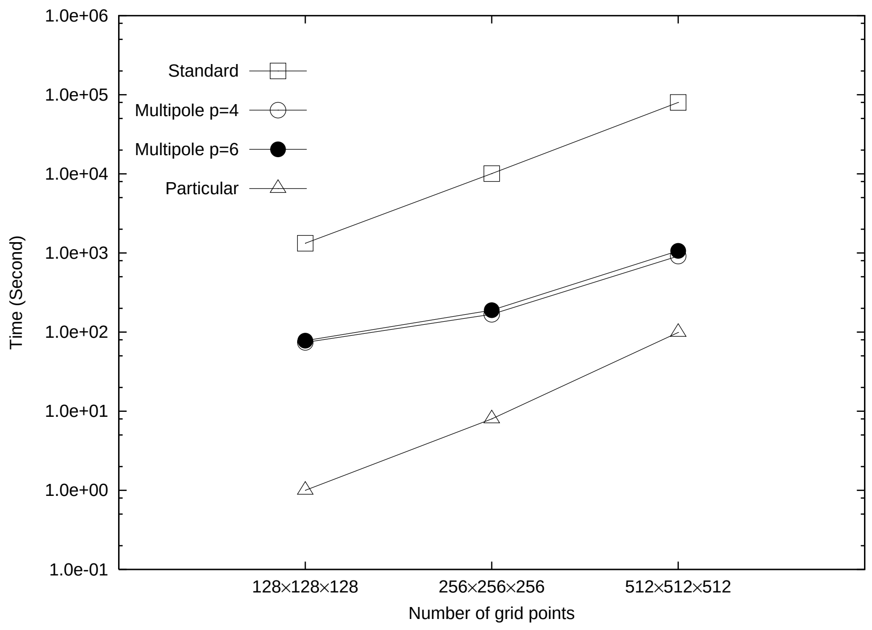


Figure 6.6: Timings for the three methods, standard volume integration (Standard), multipole accelerated volume integration (Multipole) and particular solution method from FFT (Particular), to handle the non-homogeneous term with fixed number of nodes (33884)

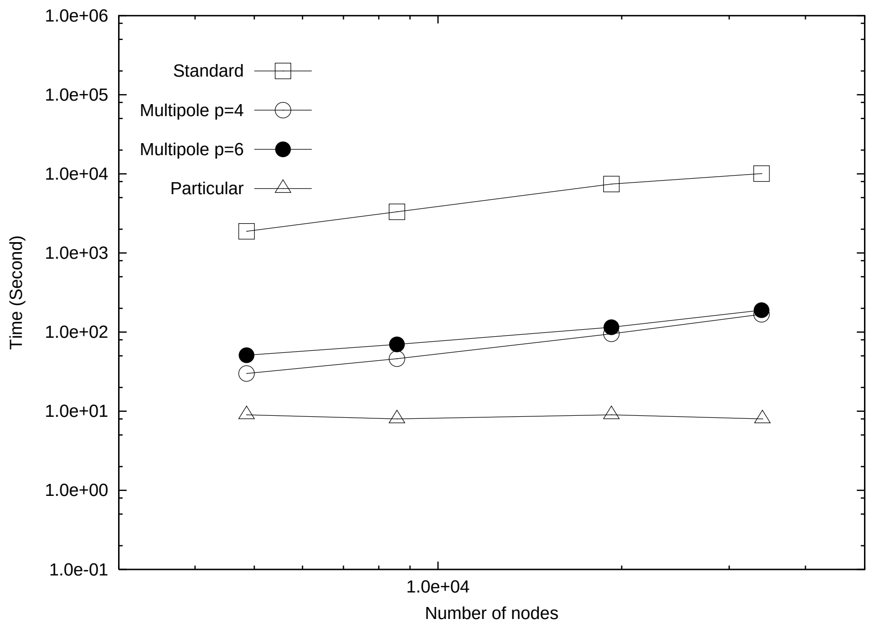


Figure 6.7: Timings for the three methods, standard volume integration (Standard), multipole accelerated volume integration (Multipole) and particular solution method from FFT (Particular), to handle the non-homogeneous term with fixed number of grid points ($256 \times 256 \times 256$)

more grid points result in longer computational time, as shown in Figure 6.6. The time of the particular method increases at a rate of $O(N_g \log N_g)$ with respect of the number of grid points (N_g), which comes from the complexity of the FFT. The complexity of the standard method is $O(N_g)$, only slightly lower than the particular method. The complexity of the multipole method to translate the interior sources in the far field is also $O(N_g)$. However, with $128 \times 128 \times 128$ grid points, the volume integration of the whole near field needs to be performed by regularizing the weakly singular integral following Equation (6.7). In contrast, with more grid points, the near field volume integration is performed with smaller cells and the regularization of the weakly singular integration is only needed for smaller fraction of near field. The implementation time of Equation (6.7) is longer than the standard Gaussian quadrature. Hence, when the number of grid points is small, the complexity of the calculation for the near field is higher than $O(N_g)$, so that the total complexity of the multipole method is lower than $O(N_g)$.

Not all the three methods use longer time to handle the non-homogeneous term with more number of nodes. The standard BEM performs volume integration for every node point, so the slope of the curve from the standard BEM is about one ($O(N)$). The particular solution is calculated from FFT in the whole rectangular domain C with FFT, and then interpolated at every node. Since the interpolation procedure is not time consuming, the computational time does not increase with number of nodes. The accelerated volume integration approximates the far sources with multipoles and performs standard volume integration for the near sources. The computational time of volume integration in the near field is dependent lin-

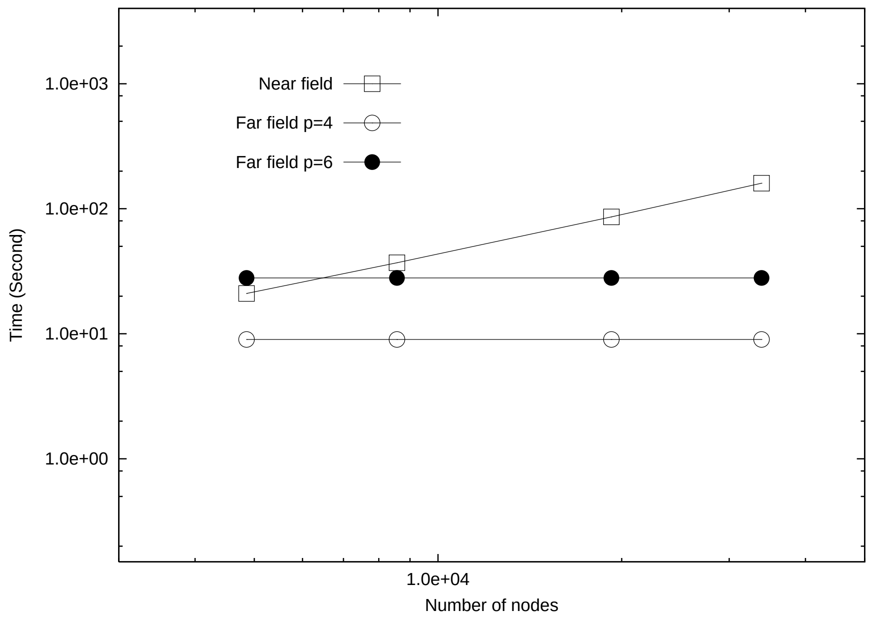
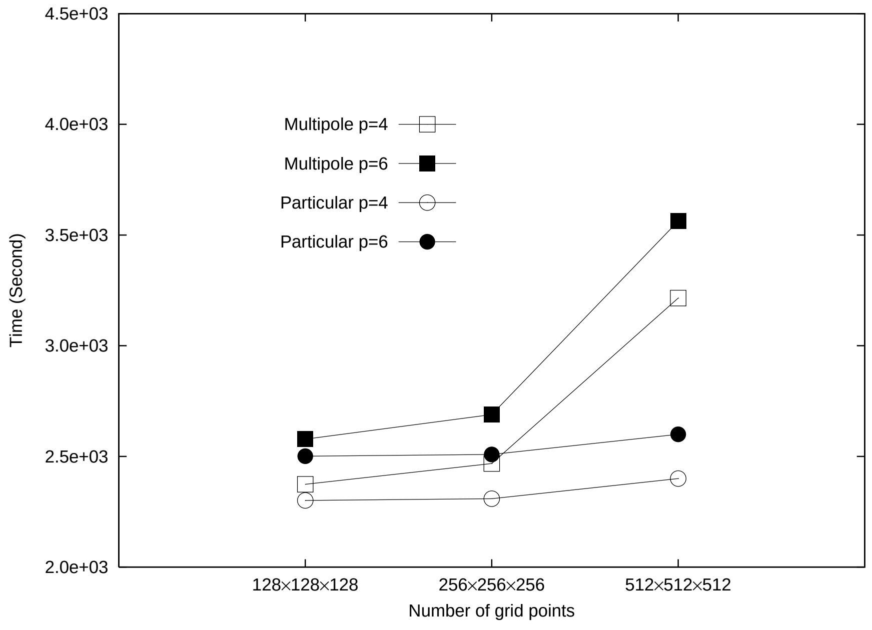


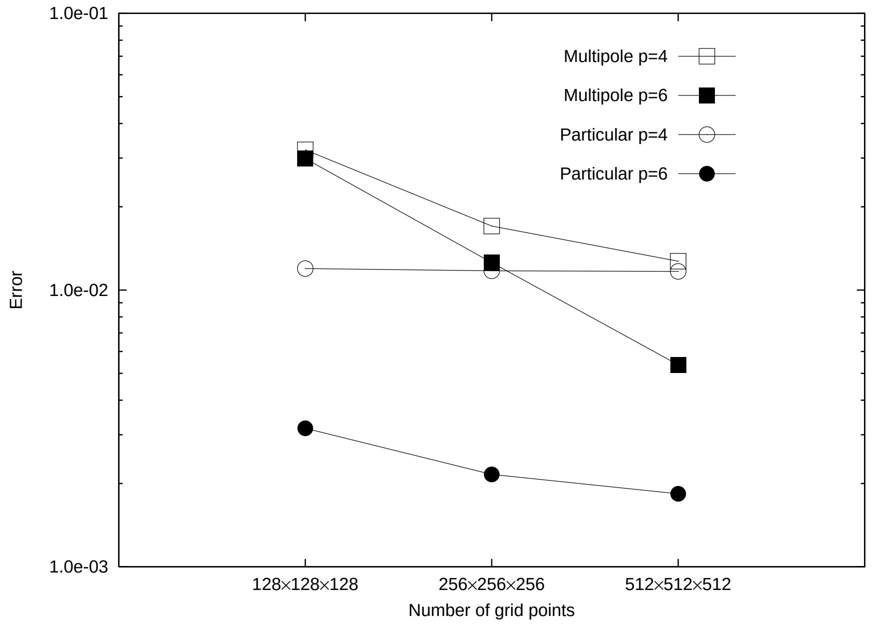
Figure 6.8: The accelerated volume integration method includes tow parts, namely, performing standard volume integration for the near field (Near field) and preparing the multipoles for the far field (Far field). ($256 \times 256 \times 256$ grid points)

early on the number of nodes, while the time of translating the far sources is only dependent on the number of grid points. In Figure 6.7, the increasing rate of the accelerated volume integration is nearly one, which may indicates that the time for the near field in dominant. The timings of the two parts of the accelerated volume integration are demonstrated in Figure 6.8. For large problems, more time is spent on calculating the volume integration for the near field than transforming the far sources. The time for the near field increases with the number of nodes, which makes the accelerated volume integration less efficient than calculating a particular solution from FFT for large problems. Higher expansion order p only results in longer time for the far field approximation.

Figure 6.9 gives the results of the two fast algorithms for the largest problem stud-



(a) Total time



(b) Accuracy

Figure 6.9: Results for solving the Poisson equation with two methods, namely multipole accelerated volume integration (Multipole) and calculating a particular solution from FFT (Particular) (constant non-homogeneous term, 33884 nodes)

ied in this example with 33884 nodes. The total time in Figure 6.9(a) includes solving a Laplace equation and handling the non-homogeneous term. The computational time of accelerated volume integration method increases faster than the particular solution method in terms of the number of grid points, which is also shown in Figure 6.6. In the particular solution method, higher p only increases the computational time for solving the Laplace equation. Yet in the accelerated volume integration method, higher p also results in more time for preparing the multipole for the far field. So the computational time of accelerated volume integration method also increases more than the particular solution method with the expansion order. Figure 6.9(b) shows that the accuracy changes with the expansion order p and the number of grid points. Higher expansion order and more grid points produce higher accuracy. Higher expansion order gives better approximation of multipoles for solving the Laplace equation and preparing the multipoles for the interior sources. More grid points approximate the non-homogeneous term more accurately. However, with the same p and number of grid points, the particular solution method is always more accurate than the accelerated volume integration method. In addition, the accelerated volume integration method converges slower than the particular solution method in terms of grid points. For $p = 4$, both the method converge to about 1×10^{-2}; for $p = 6$, they tend to converge to about 1.5×10^{-3}. When $p = 4$, the accelerated domain integral method needs more grid points to obtain the accuracy of 1×10^{-2}, whereas the particular solution method can achieve that with as few as $128 \times 128 \times 128$ grid points. This is because the particular solution method gives good accuracy, such that the error in the total solution is dominated by the homogeneous solution. This can be seen from the typical errors of the Laplace solver, shown in Figures 3.6(a) and 3.7(a), which are comparable with the total error displayed in Figure 6.9(b). In fact, the error 1×10^{-2} in the particular solution method comes mainly from the Laplace solver. The accelerated domain integral method gives lower accuracy for a coarse grid, and the total error only approaches that of the Laplace solver (1×10^{-2}) when $512 \times 512 \times 512$ grid points are used.

6.3.2 Poisson equation with a non-constant non-homogeneous term

In this Poisson equation, the non-homogeneous term is no longer constant, but a function of interior position as shown in the following equation

$$\begin{aligned}
\nabla^2 u(\mathbf{x}) &= (2x_2^3 + 6x_2)e^{x_1+x_3}, \quad \mathbf{x} \in \Omega, \\
u(\mathbf{x}) &= x_2^3 e^{x_1+x_3}, \\
\frac{\partial u^*}{\partial \mathbf{n}} &= x_2^2 e^{x_1+x_3}(x_2 n_1 + 3n_2 + x_2 n_3), \quad \mathbf{x} \in S, \qquad (6.20)
\end{aligned}$$

where $\mathbf{x} = (x_1, x_2, x_3)$ is the coordinate and $\mathbf{n} = (n_1, n_2, n_3)$ is the normal direction.

Since the change of the non-homogeneous term does not influence the computational time, the timings should be almost the same with the first example. So only the accuracy of the solution procedures is discussed in this example. Figure 6.10 shows a similar behavior in convergence as the first example. The non-constant non-homogeneous term only decreases the accuracy with course grid. With suffi-

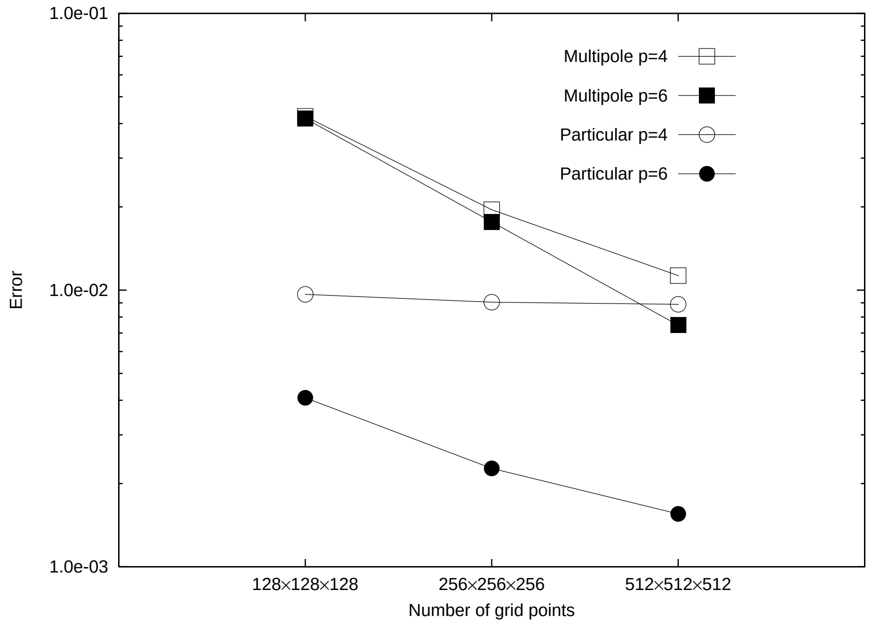


Figure 6.10: Accuracy for solving the Poisson equation with different methods and number of interior values (variational non-homogeneous term, 33884 nodes)

cient grid points, the accuracy is as good as that in the previous example. All the results from the particular solution method are still more accurate than those from the accelerated volume integration method and the convergence is faster as more grid points are used. Similar with the previous example, when $p = 4$, the dominant error from FFTM Laplace solver limits any improvement in accuracy when more grid points are used.

From the above two examples, the advantages of the particular solution method, obtained from FFT, over accelerated volume integration method are quite clear. The former is not only faster than the latter, but also more accurate with the same number of grid points. Moreover, increasing the accuracy of the former is less expensive. Hence, the particular solution method is used, when the non-linear equations are solved in the following examples. With this method, there is another big advantage that volume integration is avoided when function variable is evaluated at the interior grid points. In the following examples, $256 \times 256 \times 256$ grid points are utilized, after balancing the computational cost and accuracy.

6.3.3 Non-homogeneous modified Helmholtz equation

The equation in this example is a non-homogeneous modified Helmholtz equation

$$\nabla^2 u(\mathbf{x}) - k^2 u(\mathbf{x}) = h(\mathbf{x}), \tag{6.21}$$

with $k^2 = 1$. The equation, boundary condition and analytical solution are

$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= u + h(\mathbf{x}), \quad \mathbf{x} \in \Omega \\ u(\mathbf{x}) &= 3x_1^3 x_2 + 2x_1^2 x_2^2 - x_1 x_2^3 + x_3, \\ \frac{\partial u^*}{\partial \mathbf{n}} &= (9x_1^2 + 4x_1 x_2^2 - x_2^3)n_1 + (3x_1^3 + 4x_1^2 x_2 - 3x_1 x_2^2)n_2 + n_3, \\ \mathbf{x} \in S, & \end{aligned} \tag{6.22}$$

where $h(\mathbf{x}) = 4x_1^2 + 4x_2^2 + 12x_1x_2 - 3x_1^3x_2 - 2x_1^2x_2^2 + x_1x_2^3 - x_3$. This equation is a linear equation. With the Helmholtz kernels, this equation can be solved in the same method as the Poisson equation. Yet, there is also an alternative way to solve the non-homogeneous modified Helmholtz equation. It can be considered as a non-linear Poisson-type equation and solved with an iterative scheme.

In each iteration, the right hand side, which is a function of the unknown solution, needs to be evaluated at all the interior grid points. Most of the $256 \times 256 \times 256$ grid points are inside the boundary. Since the number of the evaluation points is very big, it is very expensive to calculate using the standard integration method, as

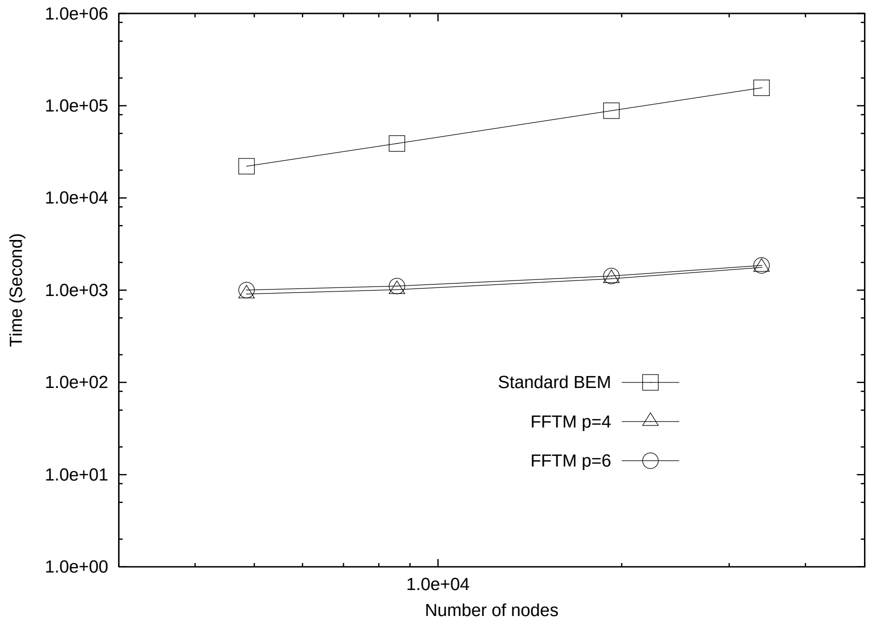


Figure 6.11: Computational time for evaluation of interior points using the standard BEM and FFTM

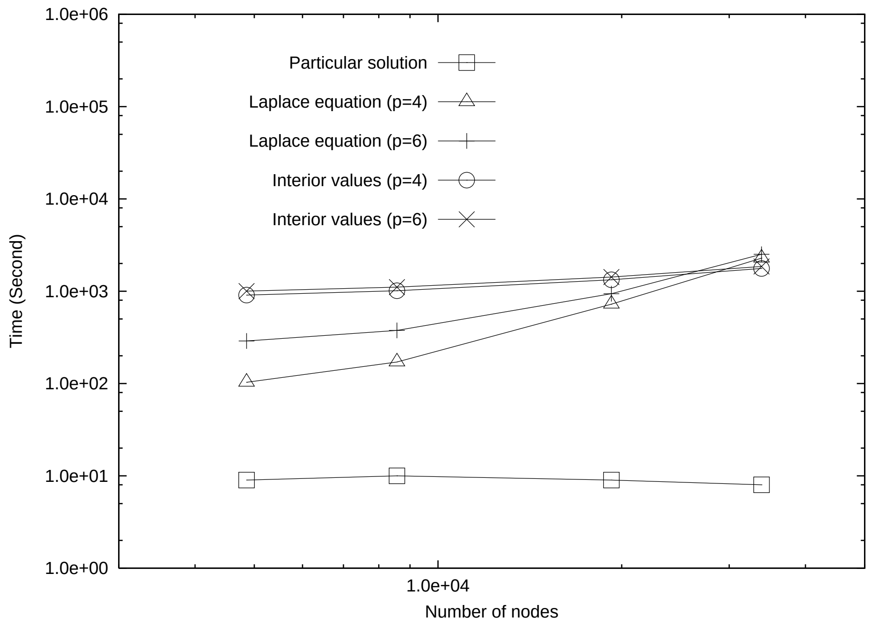
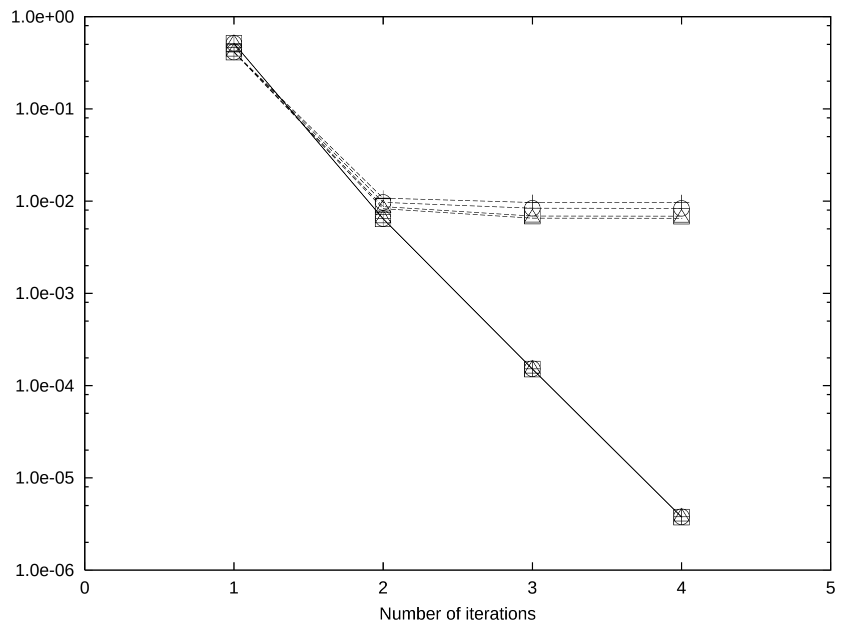


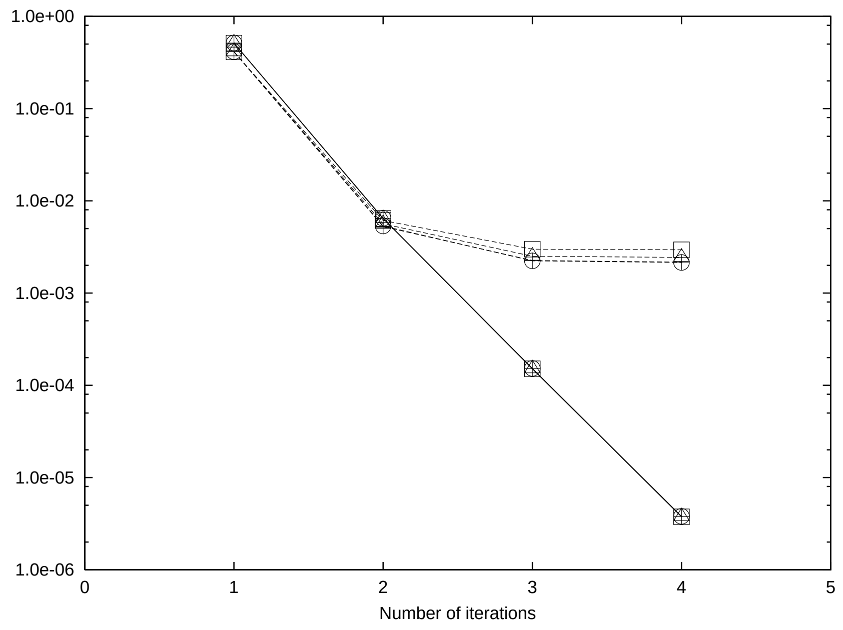
Figure 6.12: Computational time needed for particular solution, Laplace equation and interior values in each iteration

shown in Figure 6.11. It compares the computational time for calculating the values of u at the interior grid points between the standard method and the FFTM (the step 4 in Section 6.2.4). The time of updating the interior values with the standard method increases linearly with the number of nodes, as the complexity is $O(NN_g)$, where N is the number of nodes and N_g is the number of grid points. When solving the largest problem, with 33884 nodes, more than 43 hours are needed to update all the interior values once. If the FFTM algorithm is employed to update the interior value, the computational time is two-order less time than the standard method. In the procedure of the FFTM, the complexities of the $S2M$, $M2L$ and $L2D$ are $O(N)$, $O(N \log N)$ and $O(N_g)$, respectively. Consequently, the total complexity is $O(N \log N + N_g)$. In addition, N_g is about 10^7, much larger than $N = 33884$. Hence, the total complexity of the FFTM updating the interior values, appeared in the Figure 6.11, is lower than $O(N)$. Increasing the expansion order p from 4 to 6 does not increase the computational time very much.

In each iteration, the calculation of the particular solution, solution of the Laplace equation and evaluation of the interior values take up most of the computational time. Figure 6.12 shows the timings of the three procedures in one iteration. The calculation of the particular solution is much faster than the others. The time for evaluating the interior points does not increase very much with more number of nodes. When the problem becomes very large, solving the Laplace equation takes more time than evaluating the interior values. Higher p results in longer computational time solving the Laplace equation and updating the interior values, but the increase is marginal.



(a) $p = 4$



(b) $p = 6$

Figure 6.13: Dahed line: Error; Solid line: Residue; □: 4858 nodes; △: 8566 nodes; ◯: 19234 nodes; +: 33884 nodes. Convergence procedures for different number of nodes, when solving $\nabla^2 u = u + h(\mathbf{x})$

Figure 6.13 shows the convergence procedures. The residue is defined as

$$Residue = \sqrt{\frac{\sum_{i=1}^{N} |\partial u_{t+1}(\mathbf{x}_i)/\partial \mathbf{n} - \partial u_t(\mathbf{x}_i)/\partial \mathbf{n}|^2}{\sum_{i=1}^{N} |\partial u_t(\mathbf{x}_i)/\partial \mathbf{n}|^2}}. \tag{6.23}$$

The problem takes four iterations to reach a residue of less than 1×10^{-5}. The convergence for this problem is very fast and is not dependent on the number of nodes or expansion order p. After three iterations, when the residues become less than 1×10^{-3}, the errors for all the cases converge to values in the region of 10^{-2}. More iterations reduce the residue, but do not improve the accuracy of the results further. This is because the error of the problem is limited by the discretization error of the BEM and the truncation error of multipole translations. Table 6.2 gives the computational timings and errors after three iterations. With higher p, the computational time becomes longer and the accuracy becomes better. However, the timings increase by a little, while the accuracy improves considerably, especially for large problems. Since the number of cells is fixed, increasing the number of nodes results in accumulating more truncation error of multipole expansion in each cell. Consequently, more nodes do not always mean better accuracy. Yet, higher expansion order reduces the truncation error, which gives better accuracy convergence.

Table 6.2: Numerical results of the FFTM with different number of nodes after three iterations

Number of nodes	Computational time (Second)		Error	
	$p = 4$	$p = 6$	$p = 4$	$p = 6$
4858	2158	3007	0.69%	0.30%
8566	2582	3440	0.65%	0.25%
19234	4990	5834	0.84%	0.22%
33884	10291	11549	0.97%	0.22%

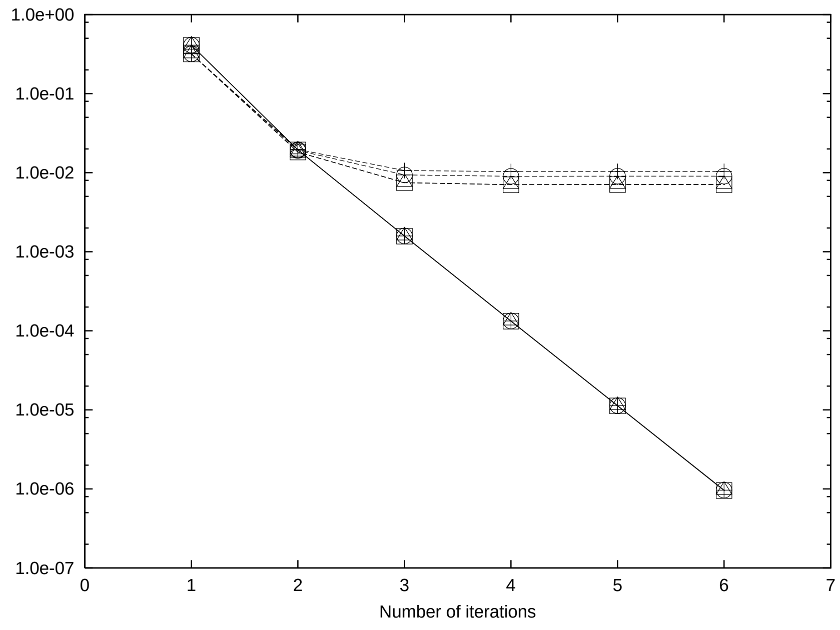
6.3.4 Non-linear Poisson-type equation

In this example, a truly non-linear Poisson-type equation (Equation (6.24)) is considered.

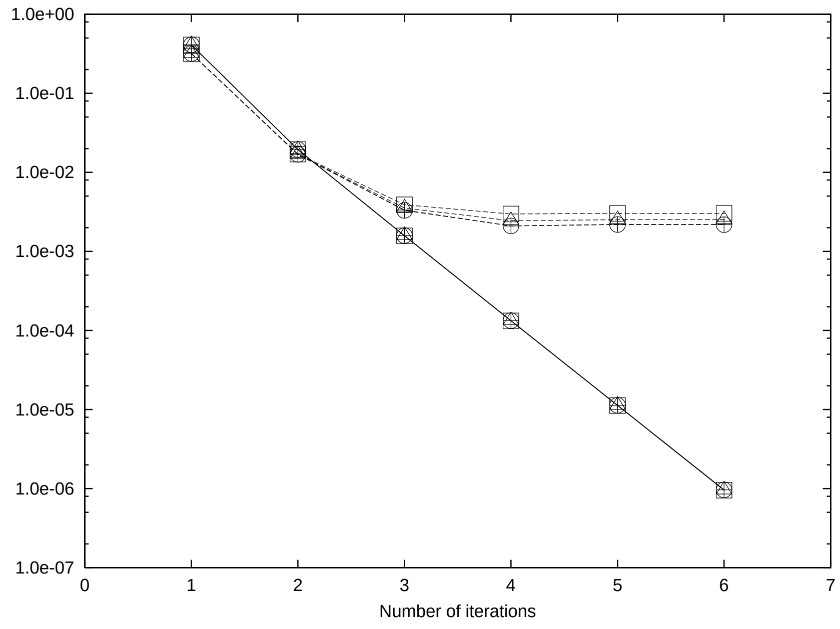
$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= u + u^3, \quad \mathbf{x} \in \Omega \\ u(\mathbf{x}) &= \tan(\frac{x_1 + x_2 + x_3}{\sqrt{6}}), \\ \frac{\partial u^*}{\partial \mathbf{n}} &= \frac{1+u^2}{\sqrt{6}}(n_1 + n_2 + n_3), \quad \mathbf{x} \in S, \end{aligned} \tag{6.24}$$

where $\mathbf{x} = (x_1, x_2, x_3)$ is the coordinate and $\mathbf{n} = (n_1, n_2, n_3)$ is the normal direction.

With stronger non-linearity, the convergence, shown in Figure 6.14, is slower than that in the above example. Now, six iterations are needed to reduce the residues to less than 1×10^{-5}. After four iteration, the residues are less than 1×10^{-3} and the errors can not be decreased further. The convergence is still neither dependent on the number of nodes, nor the expansion order p. Table 6.3 shows the results after four iterations. The results are similar with Table 6.2, and again $p = 6$ is preferred



(a) $p = 4$



(b) $p = 6$

Figure 6.14: Dahed line: Error; Solid line: Residue; □: 4858 nodes; △: 8566 nodes; ○: 19234 nodes; +: 33884 nodes. Convergence procedures of for different number of nodes, when solving $\nabla^2 u = u + u^3$

for large problems.

Table 6.3: Numerical results of the FFTM with different number of nodes after four iterations

Number of nodes	Computational time (Second)		Error	
	$p = 4$	$p = 6$	$p = 4$	$p = 6$
4858	3953	4851	0.70%	0.30%
8566	3989	4914	0.71%	0.24%
19234	7557	8035	0.90%	0.21%
33884	14188	15661	1.0%	0.21%

6.3.5 Burger's equation

In this example, the static Burger's equation is solved. Burgers' equation is a fundamental partial differential equation from fluid mechanics. The equation and boundary condition are given as

$$\begin{aligned} \nabla^2 u(\mathbf{x}) &= \alpha u \frac{\partial u}{\partial x_3}, \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= n_1(\mathbf{x}) + n_2(\mathbf{x}) + n_3(\mathbf{x}), \quad \mathbf{x} \in S. \end{aligned} \tag{6.25}$$

Higher α presents stronger non-linearity. Inside the boundary, the term $\partial u/\partial x_3$ is calculated by 4th order finite difference for all the values at the interior grid points. This method is a little less accurate than the direct evaluation from the integral equation, but much faster. From the previous examples, it is noted that the results

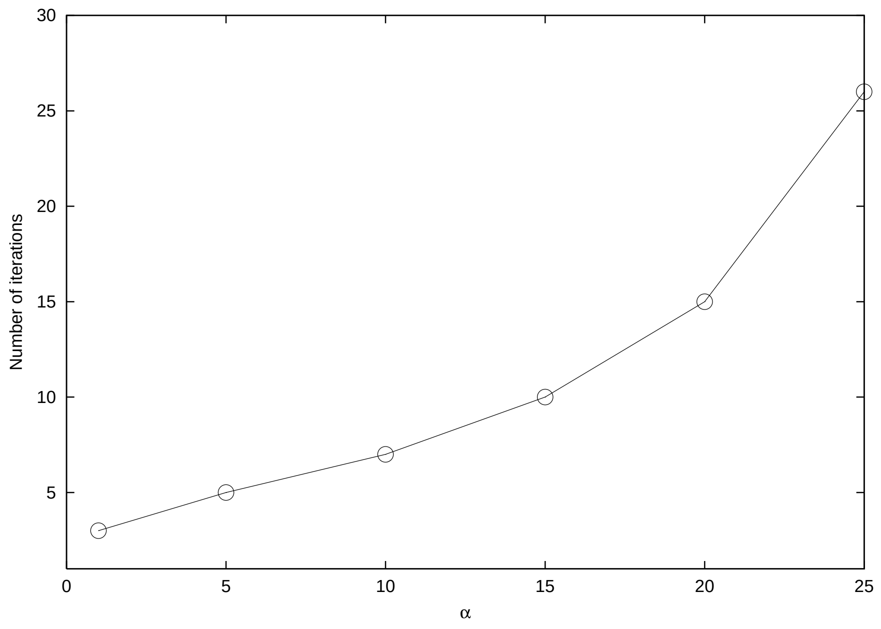
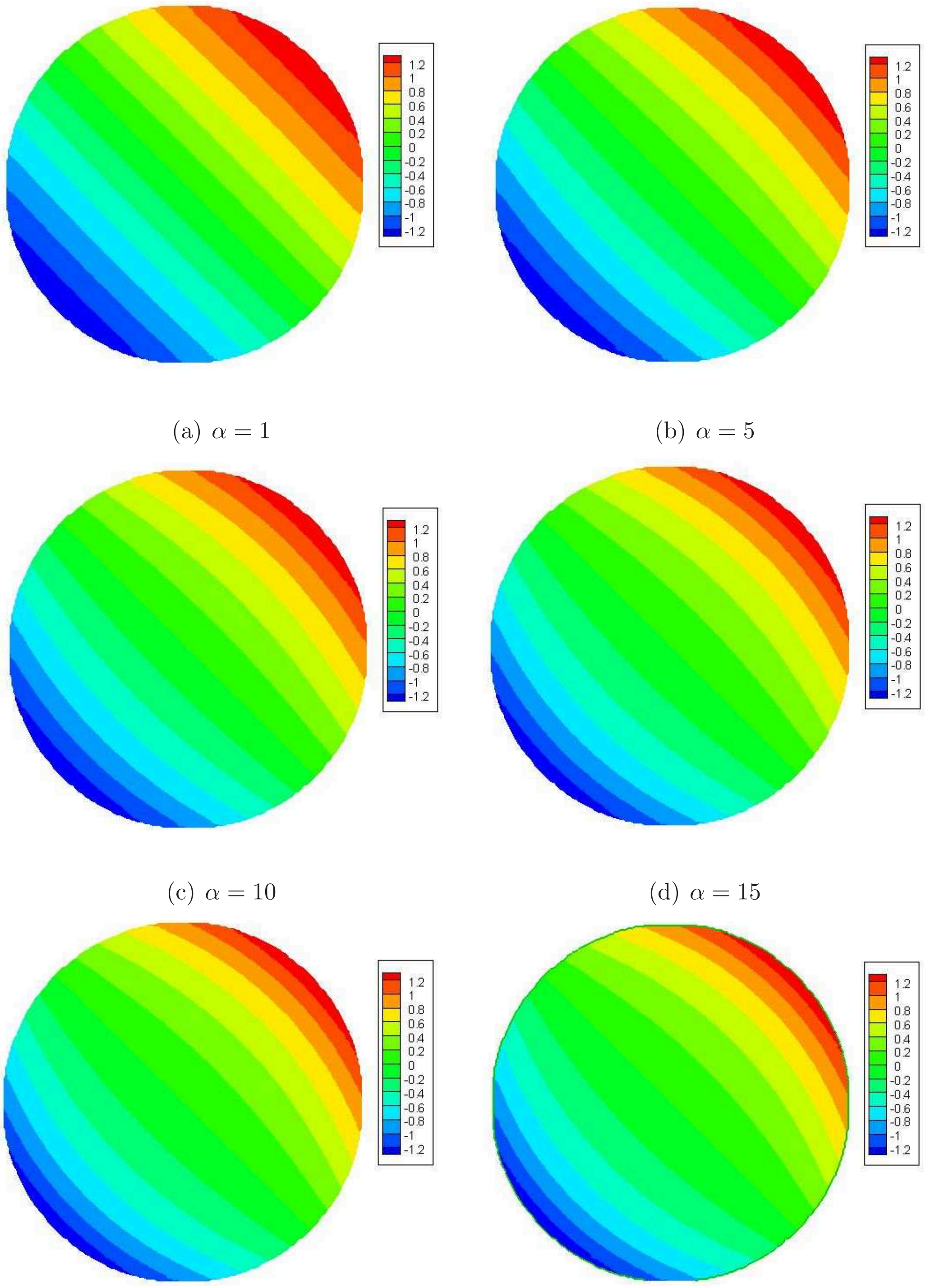


Figure 6.15: The number of iterations that is needed to enable the residue less than 1×10^{-3}

converge when the residue is less than 1×10^{-3}. So, in this example, the tolerance of the iterative scheme is set 1×10^{-3}. Figure 6.15 shows the number of iterations that are needed to reduce the residue to less than the tolerance. With higher α, the convergence needs more iterations. When $\alpha > 25$, the current scheme is not efficient to converge. Figure 6.16 gives the solution u of the Burger's equation on the x_1x_2 plane. With the α increasing, the solution changes gradually.

6.4 Summary of FFTM for non-linear Poisson-type equation

In this chapter, a fast non-linear solver based on the FFTM algorithm is presented. A simple Richardson iterative scheme is adopted. In each iteration, a Poisson equa-



(a) $\alpha = 1$

(b) $\alpha = 5$

(c) $\alpha = 10$

(d) $\alpha = 15$

(e) $\alpha = 20$

(f) $\alpha = 25$

Figure 6.16: The contour of solution u on the x_1x_2 plane ($x_3 = 0$)

tion is solved and the values inside the domain are updated. Two fast methods of handling the non-homogeneous term of the Poisson equation are presented and compared, namely accelerating the volume integration with multipoles and calculating a particular solution with FFT. With the same number of interior grid points, the particular solution method is faster and more accurate than the accelerated volume integration method. So, in each iteration of the fast algorithm, a particular solution is calculated by the FFT, the resulted Laplace equation is solved by the FFTM and the interior values are updated also by the FFTM. Such a fast solver is applied to solve several non-linear Poisson-type equations with different non-linear term. It can converge very fast with weak and moderate non-linearity. Higher expansion order of the FFTM can improve the accuracy of the algorithm, but cannot speedup the convergence. The convergence rate is only dependent on the property of the non-linear term.

Chapter 7

Conclusion

The main purpose of this thesis is to apply a fast algorithm, the fast Fourier transform on multipoles (FFTM), based on the boundary element method (BEM), to solve the Laplace equation, Navier equation, Stokes equation and non-linear Poisson-type equation. The FFTM solve these different kinds of partial different equations fast and accurately. For each equation, several numerical examples are given to show the advantages of the FFTM.

The FFTM is extended to solve the direct BEM formulation of Laplace equation. In the implementation of direct formulation, a new method is developed to perform the translations for the double layer kernel. This new method presents a physical interpretation of Yoshida's method used in the literature. In both of the direct and indirect formulations, the FFTM accelerates the computational process significantly and provides reasonable accuracy. In the two formulations, the FFTM uses different translations that introduce different approximation errors to the results.

However, the error from the FFTM is less significant than the error introduced by the choice of the direct or indirect formulation.

Unlike the Laplace equation, there are no simple translations for the kernels in the Navier equation. Following the Yoshida's method, the FFTM is applied in the Navier equation. The translations in the Navier equation are more complex and cost more memory than those in the Laplace equation. Storing the multipole-to-local translation matrices costs most of the memory usage. A compact storage of the matrix is proposed to reduce the memory usage of the FFTM significantly. The fast algorithm is shown to be efficient for large scale problems, as demonstrated in the calculation of effective Young's modulus of porous materials.

For Stokes equation, the multipole translations are quite similar with those in the Navier equation. In addition, the compact matrix storage, used in solving the Navier equation, is also used to solve Stokes equation. This fast Stokes solver is employed to simulate the Stokes flow in a cylinder with many spherical particles inside. The average drag force on the spheres is calculated and compared with various theoretical solutions.

When solving non-linear Poisson-type equation, a Poisson equation needs to be solved. Two methods of handling the non-homogeneous term in Poisson equation are compared. It is found that calculating a particular solution through the FFT is faster and more accurate than accelerating the volume integration by the FFTM. Further more, the particular solution method avoids volume integration when evaluating the non-linear function at the interior points. So this method is employed in the new fast non-linear solver. The non-linear scheme includes calculating a particular solution through fast Fourier transform (FFT), solving the resulting Laplace equation with the FFTM and evaluating the interior values with the FFTM. The iterative scheme converges fast with weak and moderate non-linearity. The higher expansion order, such as $p = 6$, is preferred, when solving large problems, as it improves accuracy significantly and only increases the computational time slightly. With such a fast non-linear solver, large three dimensional non-linear problems are solved efficiently.

Appendix A

Code structure of the FFTM

In this appendix, the implementation details of the FFTM are illustrated. The same code structure is implemented for Laplace equation, Navier equation and Stokes equation. The following pseudo-codes, including the `main()` function and the `MVmulti()` function, show how to implement the FFTM.

```
main() {

input(); // to input the coordinates of node, node-element connection
and boundary condition

getCellInfo(); // to define the spatial domain in Figure 3.1(a); to set
number to each cell (This three-digit number can indicate the position
of every cell.); to obtain the connection between the nodes and cell

getS2M(); // to calculate translation matrices used to perform $S2M$
(Step B in Figure 3.1(b)), following Equations (3.7), (4.13) and (5.15)
getM2L(); // to construct the matrices used in Step C $M2L$ in Figure
3.1(c), following Equations (3.13), (4.14) and (5.16)

getFFTM2L(); // to perform FFT on the $M2L$ translation matrices, whose
results are used to compute the local expansion coefficients rapidly in
each iteration of GMRES solver.

getRHS(); // to evaluate the right hand side of the linear system (For
the indirect formulation, the right hand side comes from the boundary
condition directly; while for the direct formulation, it comes from one
matrix-vector multiplication that is shown in the following MVmulti()
function.)

GMRES(); // to solve the linear system (In each iteration of GMRES solver,
one matrix-vector multiplication is performed rapidly with the FFTM, whose
details are shown in the following MVmulti() function.)

output(); // to output the results

}
```

The translation matrices $S2M$ and $M2L$ are constructed and stored outside the GMRES solver. They are calculated only once. The following function to do the matrix-vector multiplication is called in every GMRES iteration.

```
MVmulti() {

getM(); // to obtain the multipoles from sources and $S2M$ matrices
getL(); // to calculate the local expansions from the multipoles and M2L
matrices using FFT

getDfar(); // to evaluate at the field point from the local expansions
for the far field, following Equations (3.17), (4.15) and (5.17)

getDnear(); // to evaluate the near field from standard boundary element
method

}
```

Bibliography

[1] F Aliabadi. *The Boundary Element Method : Applications in Solids and Structures.* Wiley, 2002.

[2] M D Altman, J P Bardhan, B Tidor, and J K White. FFTSVD: A fast multiscale boundary-element method solver suitable for bio-MEMS and biomolecule simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(2):274–284, 2006.

[3] A W Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6(1):85–103, 1985.

[4] J Barnes and P Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.

[5] C A Brebbia and J Dominguez. *Boundary Elements An Introductory Course.* Computational Mechanics Publications, 2nd edition, 1992.

[6] M J Brown, A A Mammoli, and M S Ingber. Parallel multipole implementation of the generalized Helmholtz decomposition for solving viscous flow problems.

International Journal for Numerical Methods in Engineering, 58(11):1617–1635, 2003.

[7] T T Bui, E T Ong, B C Khoo, E Klaseboer, and K C Hung. A fast algorithm for modeling multiple bubbles dynamics. *Journal of Computational Physics*, 216(2):430–453, 2006.

[8] H Cheng, L Greengard, and V Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155(2):468–498, 1999.

[9] J Ding and W J Ye. A fast integral approach for drag force calculation due to oscillatory slip stokes flows. *International Journal for Numerical Methods in Engineering*, 60(9):1535 – 1567, 2004.

[10] J Ding and W J Ye. A grid-based integral approach for quasilinear problems. *Computational Mechanics*, 38(2):113–118, 2006.

[11] J Ding, W J Ye, and L J Gray. An accelerated surface discretization-based BEM approach for non-homogeneous linear problems in 3-D complex domains. *International Journal for Numerical Methods in Engineering*, 63(12):1775–1795, 2005.

[12] W D Elliott and J A Board. Fast Fourier transform accelerated fast multipole algorithm. *SIAM Journal on Scientific Computing*, 17(2):398–415, 1996.

[13] M A Epton and B Dembart. Multipole translation theory for the three-dimensional Laplace and Helmholtz equations. *SIAM Journal on Scientific Computing*, 16(4):865–897, 1995.

[14] O Von Estorff, editor. *Boundary Elements in Acoustics: Advances and Applications*. WIT Press, 2000.

[15] F Ethridge and L Greengard. A new fast-multipole accelerated Poisson solver in two dimensions. *SIAM Journal on Scientific Computing*, 23(3):741–760, 2001.

[16] A Frangi. A fast multipole implementation of the qualocation mixed-velocity-traction approach for exterior Stokes flows. *Engineering Analysis with Boundary Elements*, 29(11):1039–1046, 2005.

[17] A Frangi, P Faure-Ragani, and L Ghezzi. Magneto-mechanical simulations by a coupled fast multipole methodfinite element method and multigrid solvers. *Computers & Structures*, 83(10-11):718–726, 2005.

[18] A Frangi, G Spinola, and B Vigna. On the evaluation of damping in MEMS in the slip-flow regime. *International Journal for Numerical Methods in Engineering*, 68(10):1031 – 1051, 2006.

[19] M Frigo and S G Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216231, 2005.

[20] Y H Fu, K J Klimkowski, G J Rodin, E Berger, J C Browne, J K Singer, R A van de Geijn, and K S Vemaganti. A fast solution method for three-dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering*, 42(7):1215–1229, 1998.

[21] Y H Fu and G J Rodin. Fast solution method for three-dimensional Stokesian many-particle problems. *Communications in Numerical Methods in Engineering*, 16(2):145–149, 2000.

[22] J E Gomez and H Power. A multipole direct and indirect BEM for 2D cavity flow at low Reynolds number. *Engineering Analysis with Boundary Elements*, 19(1):17–31, 1997.

[23] L Greengard and J Y Lee. A direct adaptive Poisson solver of arbitrary order accuracy. *Journal of Computational Physics*, 125(2):415–424, 1996.

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

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

[26] L F Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, 1988.

[27] W L Haberman and R M Sayre. Motion of rigid and fluid spheres in stationary and moving liquids inside cylindrical tubes. Technical report, David Taylor Model Basin Report No. 1143, U S Navy Department, Washington D C, 1958.

[28] K Hayami. *A projection transformation method for nearly singular surface boundary element integrals*. Springer-Verlag, 1992.

[29] D P Henry and P K Banerjee. A new boundary element formulation for two- and three-dimensional thermoelasticity using particular integrals. *International Journal for Numerical Methods in Engineering*, 26(9):2061–2077, 1988.

[30] F B Hildebrand. *Finite-Difference Equations and Simulations.* Prentice-Hall, 1968.

[31] H T Hu, D T Blaauw, V Zolotov, K Gala, M Zhao, R Panda, and S S Sapatnekar. Fast on-chip inductance simulation using a precorrected-FFT method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(1):49–66, 2003.

[32] M T Ibanez, H Power, and M T Ibanez. *Advanced Boundary Elements for Heat Transfer.* WIT Press, 2003.

[33] M S Ingber, A A Mammoli, and M J Brown. A comparison of domain integral evaluation techniques for boundary element methods. *International Journal for Numerical Methods in Engineering*, 52(4):417–432, 2001.

[34] M A Jaswon and G T Symm. *Integral Equation Methods in Potential Theory and Elastostatics.* Academic Press, 1977.

[35] S Kapur and D E Long. IES3: Efficient electrostatic and electromagnetic simulation. *IEEE Computational Science & Engineering*, 5(4):60–67, 1998.

[36] S Kim and W B Russel. Modelling of porous media by renormalization of the Stokes equations. *Journal of Fluid Mechanics*, 154:269–286, 1985.

[37] Y S Lai and G J Rodin. Fast boundary element method for three-dimensional solids containing many cracks. *Engineering Analysis with Boundary Elements*, 27(8):845–852, 2003.

[38] S J Liao. On the general boundary element method. *Engineering Analysis with Boundary Elements*, 21(1):39–51, 1998.

[39] K M Lim, E T Ong, H P Lee, and S P Lim. A fast boundary element method for underwater acoustics. *Modern Physics Letters B*, 19(28-29):1679–1682, 2005.

[40] Y J Liu, N Nishimura, Y Otani, T Takahashi, X L Chen, and H Munakata. A fast boundary element method for the analysis of fiber-reinforced composites based on a rigid-inclusion model. *Journal of Applied Mechanics-Transactions of the ASME*, 72(1):115–128, 2005.

[41] Z J Liu, H H Long, E T Ong, and E P Li. A fast Fourier transform on multipole algorithm for micromagnetic modeling of perpendicular recording media. *Journal of Applied Physics*, 99(8):08B903, 2006.

[42] Z J Liu, H H Long, E T Ong, E P Li, and J T Li. Dynamic simulation of high-density perpendicular recording head and media combination. *IEEE Transactions on Magnetics*, 42(4):943–946, 2006.

[43] H H Long, E T Ong, Z J Liu, and E P Li. Fast Fourier transform on multipoles for rapid calculation of magnetostatic fields. *IEEE Transactions on Magnetics*, 42(2):295–300, 2006.

[44] A A Mammoli. Solution of non-linear boundary integral equations in complex geometries with auxiliary integral subtraction. *International Journal for Numerical Methods in Engineering*, 55(9):1115–1128, 2002.

[45] N Masters and Wenjing Ye. Fast BEM solution for coupled 3D electrostatic and linear elastic problems. *Engineering Analysis with Boundary Elements*, 28(9):1175–1186, 2004.

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

[47] K Nabors, F T Korsmeyer, F T Leighton, and J White. Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional first-kind integral equations of potential theory. *SIAM Journal on Scientific computing*, 15(3):713–735, 1994.

[48] K Nabors and J White. FastCap: a multipole accelerated 3-D capacitance extraction program. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1447–1459, 1991.

[49] S Nemat-Nasser, T Iwakuma, and M Hejazi. On composites with periodic structure. *Mechanics of Materials*, 1(3):239–267, 1982.

[50] J N Newman. Efficient hydrodynamic analysis of very large floating structures. *Marine Structures*, 18(2):169–180, 2005.

[51] J N Newman and C H Lee. Boundary-element methods in offshore structure analysis. *Journal of Offshore Mechanics and Arctic Engineering-Transactions of ASME*, 124(2):81–89, 2002.

[52] N Nishimura. Fast multipole accelerated boundary integral equation methods. *Applied Mechanics Reviews*, 55(4):299–324, 2002.

[53] N Nishimura, K Yoshida, and S Kobayashi. A fast multipole boundary integral equation method for crack problems in 3D. *Engineering Analysis with Boundary Elements*, 23(1):97–105, 1999.

[54] A J Nowak and A C Neves. *The Multiple Reciprocity Boundary Element Method*. Computational Mechanics Publications, 1994.

[55] G Of, O Steinbach, and W L Wendland. The fast multipole method for the symmetric boundary integral formulation. *IMA Journal of Numerical Analysis*, 26(2):272–296, 2006.

[56] E T Ong, H P Lee, and K M Lim. A fast algorithm for three-dimensional electrostatics analysis fast Fourier transform on multipoles (FFTM). *International Journal for Numerical Methods in Engineering*, 61(5):633–656, 2004.

[57] E T Ong, H P Lee, and K M Lim. A fast Fourier transform on multipoles (FFTM) algorithm for solving Helmholtz equation in acoustics analysis. *Journal of the Acoustical Society of America*, 116(3):1362–1371, 2004.

[58] E T Ong, H P Lee, and K M Lim. A parallel fast Fourier transform on multipoles (FFTM) algorithm for electrostatics analysis of three-dimensional

structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(7):1063–1072, 2004.

[59] E T Ong, K M Lim, K H Lee, and H P Lee. A fast algorithm for three-dimensional potential fields calculation: fast Fourier transform on multipoles. *Journal of Computational Physics*, 192(1):244–261, 2003.

[60] J P O'Rourke, M S Ingber, and M W Weiser. The effective elastic constants of solids containing spherical exclusions. *Journal of Composite Materials*, 31(9):910–934, 1997.

[61] P W Partridge, C A Brebbia, and L C Wrobel. *The Dual Reciprocity Boundary Element Method*. Computational Mechanics Publications, 1992.

[62] J R Phillips and J K White. A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1059–1072, 1997.

[63] D Poljak and C A Brebbia. *Boundary Element Methods For Electrical Engineers*. WIT Press, 2005.

[64] R Pollandt. An approximation method for solving inhomogeneous linear and nonlinear differential equations using the boundary element method and radial basis functions. *Zeitschrift Fur Angewandte Mathematik Und Mechanik*, 78(8):545–553, 1998.

[65] V Popov and H Power. An O(N) Taylor series multipole boundary element method for three-dimensional elasticity problems. *Engineering Analysis with Boundary Elements*, 25(1):7–18, 2001.

[66] V Popov, H Power, and S P Walker. Numerical comparison between two possible multipole alternatives for the BEM solution of 3D elasticity problems based upon Taylor series expansions. *Engineering Analysis with Boundary Elements*, 27(5):521–531, 2003.

[67] H Power and L C Wrobel. *Boundary Integral Methods in Fluid Mechanics.* Computational Mechanics Publications, 1995.

[68] C Pozrikidis. *Boundary Integral and Singularity Methods for Linearized Viscous Flow.* Cambridge University Press, 1992.

[69] J N Reddy. *An Introduction to the Finite Element Method.* McGraw-Hill Science, 1993.

[70] V Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187–207, 1985.

[71] Y Saad and M H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[72] A S Sangani and G B Mo. An O(N) algorithm for Stokes and Laplace interactions of particles. *Physics of Fluids*, 8(8):1990–2010, 1996.

[73] J Shimada, H Kaneko, and T Takada. Efficient calculations of Coulombic interactions in biomolecular simulations with periodic boundary conditions. *Journal of Computational Chemistry*, 14(7):867–878, 1993.

[74] J Shimada, H Kaneko, and T Takada. Performence of fast multipole methods for calculating electrostatic interactions in biomacromolecular simulations. *Journal of Computational Chemistry*, 15(1):28–43, 1994.

[75] J Tausch and J White. Capacitance extraction of 3-D conductor systems in dielectric media with high-permittivity ratios. *IEEE Transactions on Microwave Theory and Techniques*, 47(1):18–26, 1999.

[76] B Teng, D Z Ning, and Y Gou. A fast multipole boundary element method for three-dimensional potential flow problems. *Acta Oceanologica Sinica*, 23(4):747–756, 2004.

[77] S Tissari and J Rahola. A precorrected-FFT method to accelerate the solution of the forward problem in magnetoencephalography. *Physics in Medicine and Biology*, 48(4):523–541, 2003.

[78] H T Wang, T Lei, J Li, J F Huang, and Z H Yao. A parallel fast multipole accelerated integral equation scheme for 3D Stokes equations. *International Journal for Numerical Methods in Engineering*, 70(7):812–839, 2007.

[79] H T Wang and Z H Yao. A new fast multipole boundary element method for large scale analysis of mechanical properties in 3D particle-reinforced composites. *CMES-Computer Modeling in Engineering & Sciences*, 7(1):85–95, 2005.

[80] H Y Wang and R LeSar. An efficient fast-multipole algorithm based on an expansion in the solid harmonics. *Journal of Chemical Physics*, 104(11):4173–4179, 1996.

[81] X Wang. *FastStokes: a fast 3-D fluid simulation program for micro-electro-mechanical systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2002.

[82] X Wang, J Kanapka, W J Ye, N R Aluru, and J White. Algorithms in FastStokes and its application to micromachined device simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(2):248–257, 2006.

[83] C A White and M Head-Gordon. Derivation and efficient implementation of the fast multipole method. *Journal of chemical physics*, 101(8):6593–6605, 1994.

[84] T W Wu, editor. *Boundary Element Acoustics: Fundamentals and Computer Codes*. WIT Press, 2000.

[85] S Q Xu and N Kamiya. A formulation and solution for boundary element analysis of inhomogeneous-nonlinear problem. *Computational Mechanics*, 22(5):367–374, 1998.

[86] L X Ying, G Biros, and D Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, 2004.

[87] L X Ying, G Biros, and D Zorin. A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains. *Journal of Computational Physics*, 219(1):247–275, 2006.

[88] K Yoshida. *Application of fast multipole method to boundary integral equation method*. PhD thesis, Global Environment Engineering, Kyoto University, Japan, 2001.

[89] K Yoshida, N Nishimura, and S Kobayashi. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *International Journal for Numerical Methods in Engineering*, 50(3):525–547, 2001.

[90] K Yoshida, N Nishimura, and S Kobayashi. Application of new fast multipole boundary integral equation method to crack problems in 3D. *Engineering Analysis with Boundary Elements*, 25(4-5):239–247, 2001.

[91] Z H Zhu, B Song, and J K White. Algorithms in FastImp: A fast and wide-band impedance extraction program for complicated 3-D geometries. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(7):981–998, 2005.

[92] A Z Zinchenko and R H Davis. An efficient algorithm for hydrodynamical interaction of many deformable drops. *Journal of Computational Physics*, 157(2):539–587, 2000.

[93] A Z Zinchenko and R H Davis. A multipole-accelerated algorithm for close interaction of slightly deformable drops. *Journal of Computational Physics*, 207(2):695–735, 2005.