 CMES, vol.35, no.1, pp.21-48, 2008

Fast BEM Solvers for 3D Poisson-Type Equations

Xuefei He[1], Kian-Meng Lim[1,2,3] and Siak-Piang Lim[1,2]

Abstract: The boundary element method (BEM) is known to have the advantage of reducing the dimension of problem by discretizing only the boundary of the domain. But it becomes less attractive for solving Poisson-type equations, due to the need to evaluate the domain integral which is computationally expensive. In this paper, we present the extension of a recently developed fast algorithm for Laplace equation, based on fast Fourier transform on multipoles (FFTM), to solve large scale 3D Poisson-type equations. We combined the Laplace solver with two fast methods for handling the domain integral based on fast Fourier transform (FFT). The first method uses the FFT on multipoles to accelerate the domain integral, while the second method solves the domain integral as a particular solution using FFT. The particular solution method is found to be faster and more accurate, and it is extended to solve non-linear Poisson-type equations. The algorithm is shown to be efficient when it is used in the inner loop of the iterative solver for the non-linear equations.

Keyword: Boundary element method (BEM), Poisson-type equation, non-linear equation, fast Fourier transform on multipoles (FFTM).

1 Introduction

The boundary element method (BEM) [Brebbia and Dominguez (1992); Becker (1992)] has the advantage of discretizing only the boundary of the computational domain for homogeneous differential equations, such as the Laplace equation and Navier equation. Traditional BEM generates a linear system with a fully populated matrix that is computationally expensive to solve. Therefore, the computational cost of BEM becomes a significant burden for large-scale problems. In recent years, the fast multipole method (FMM) [Greengard and Rokhlin (1987); Nishimura (2002)] has been used to accelerate the computational speed and reduce

[1] Singapore-MIT Alliance, E4-04-10, 4 Engineering Drive 3, Singapore 117576
[2] Department of Mechanical Engineering, National University of Singapore, Singapore 119260
[3] Corresponding author. Email: limkm@nus.edu.sg

memory storage in large-scale BEM models. These improvements are achieved by approximating far-field interactions by multipole representations. The FMM has been successful to solve electromagnetic [Nabors and White (1991); Volakis, Sertel, Jorgensen, and Kindt (2004); Chew, Song, Cui, Velarnparambil, Hastriter, and Hu (2004)] and elastostatic [Nishimura, Yoshida, and Kobayashi (1999); Wang and Yao (2005, 2008); Aoki, Amaya, Urago, and Nakayama (2004)] problems. Other numerical techniques, such as the precorrected-FFT (pFFT) technique [Phillips and White (1997)] and the adaptive cross-approximation (ACA) technique [Kurz, Rain, and Rjasanow (2002)], have also been developed, and these algorithms exploit structures and patterns present in the formulation to improve the speed and memory requirements of the BEM.

However, for Poisson-type equations, a domain integral typically needs to be evaluated. Traditional BEM [Brebbia and Dominguez (1992)] solves Poisson-type equations by discretizing the interior domain and evaluating the domain integral directly. Various approaches have been proposed to avoid or alleviate the burden due to the domain integral. Meshless methods, such as the dual reciprocity method (DRM) [Partridge, Brebbia, and Wrobel (1992)], multiple reciprocity method (MRM) [Nowak and Neves (1994)] and particular solution method (PSM) [Henry and Banerjee (1988)], are commonly used to maintain the advantage of discretizating only the boundary. Ingber [Ingber, Mammoli, and Brown (2001)] also proposed a cell-based direct domain integral scheme that gives better accuracy. In addition, when such a domain integral scheme is coupled with the FMM, it significantly improves the computational efficiency over the meshless methods. In order to apply the domain integral method in a complex domain, Mammoli [Mammoli (2002)] 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 [McKenney, Greengard, and Mayo (1995); Greengard and Lee (1996); Ethridge and Greengard (2001)] provided a series of two dimensional fast Poisson solvers based on the FMM. Ding et al. [Ding, Ye, and Gray (2005)] introduced a fast cell-based approach, based on the pFFT, that accelerates the surface integral as well as the domain integral. Ying et al. [Ying, Biros, and Zorin (2006)] handled the non-homogeneous part with a particular solution, while solving the homogeneous part with the kernel-independent 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 in the Poisson-type equation is a non-linear function of the unknown variable, the solution process becomes more complicated. Typically, iterative solvers are used, and within each iteration, a linear Poisson equation is solved. The DRM and MRM have been applied to solve such non-linear

equations. There are also some modified and improved methods [Xu and Kamiya (1998); Pollandt (1998)], based on the DRM. Liao [Liao (1998)] applied the general boundary element method to solve strongly non-linear problems. 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 non-linear problems with small number of degrees of freedom, but rarely in three dimensional problems. Recently, Ding and Ye [Ding and Ye (2006)] had applied the pFFT to solve three dimensional weakly non-linear problems, where the number of degrees of freedom reaches 4000.

Table 1: Different BEM methods for solving linear and non-linear Poisson-type equations

	Linear Poisson equation	Non-linear Poisson equation
Conventional method	DRM [Partridge, Brebbia, and Wrobel (1992)] MRM [Nowak and Neves (1994)] PSM [Henry and Banerjee (1988)]	DRM [Partridge, Brebbia, and Wrobel (1992)] MRM [Nowak and Neves (1994)] Liao [Liao (1998)]
Fast method	Ingber et al. [Ingber, Mammoli, and Brown (2001)] Group of Greengard [McKenney, Greengard, and Mayo (1995), Greengard and Lee (1996), and Ethridge and Greengard (2001)] Ding et al. [Ding, Ye, and Gray (2005)] Ying et al. [Ying, Biros, and Zorin (2006)]	Ding and Ye [Ding and Ye (2006)]

The methods discussed are summarized in Table 1. The conventional methods have been applied to many problems, but usually limited to a small number of the degrees of freedom. 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 in the Poisson equation or non-linear equation by an accelerated domain integral. Ingber et al. [Ingber, Mammoli, and Brown (2001)] and some of Greengard's work [McKenney, Greengard, and Mayo (1995); Ethridge and Greengard (2001)] accelerated the domain integral by the FMM, while Ding et al. [Ding, Ye, and Gray (2005)] and Ding and Ye [Ding and Ye (2006)], by the pFFT technique. The others [Greengard and Lee (1996); Ying, Biros, and Zorin (2006)] calculated a particular solution in a fast manner. Greengard and Lee [Greengard and Lee (1996)] calculated particular solutions with spectral method in a decomposed domain and patches the solutions together with the FMM. Ying et al. [Ying, Biros, and Zorin (2006)] obtained a particular solution with the FFT. In [Ingber, Mammoli, and

Brown (2001)], 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.

In this paper, we extend our recent work on a fast solver for Laplace equation based on FFT to multipoles (FFTM) [Ong, Lim, Lee, and Lee (2003); Lim, He, and Lim (2008)] to solve Poisson-type equations. We combine the FFTM with two fast methods for the domain integral based on the FFT. In the first method the FFT on multipole algorithm is used to accelerate domain integral directly, and it is seamlessly integrated with the Laplace solver. The second method uses the FFT to calculate the particular solution rapidly, and then combines that with the homogeneous solution obtained by the fast Laplace solver. The FFT accelerated particular solution method was found to be faster and more accurate after several benchmark tests using linear Poisson equations. Hence, this method is extended to solve non-linear Poisson-type equations. A Richardson iterative scheme is used, with each iteration involving a fast solution of a linear Poisson equation. The numerical examples demonstrate that the method is capable of solving large problems (with more than 30,000 degrees of freedom) efficiently.

2 Methodology

2.1 Standard methods for Poisson-type equation

The Poisson-type equation,

$$\nabla^2 u(\mathbf{x}) = f, \quad \mathbf{x} \in \Omega, \tag{1}$$

in which f is a function of coordinate $\mathbf{x}$ and possibly the unknown variable u, can be rewritten into a direct boundary integral formulation

$$c(\mathbf{x})u(\mathbf{x}) + \int_S H(\mathbf{x},\mathbf{y})u(\mathbf{y})dS(\mathbf{y}) + \int_\Omega G(\mathbf{x},\mathbf{y})fd\Omega(\mathbf{y}) = \int_S G(\mathbf{x},\mathbf{y})\frac{\partial u(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})}dS(\mathbf{y}). \tag{2}$$

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

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

where $r = |\mathbf{y}-\mathbf{x}|$. The surface integrals, corresponding to the Laplace equation, can be calculated rapidly by fast algorithms, such like FMM [Greengard and Rokhlin

(1987); Nishimura (2002)], pFFT [Phillips and White (1997)] or FFTM [Ong, Lim, Lee, and Lee (2003); Lim, He, and Lim (2008)]. All of them have comparable $O(N)$ or $O(N\log N)$ efficiency. In this paper, we will use the FFTM to evaluate these surface integrals. The details are given in [Ong, Lim, Lee, and Lee (2003); Lim, He, and Lim (2008)], and briefly summarized in Appendix. The non-homogeneous term f gives rise to a domain integral term, which requires discretization of the interior domain. As this is a computationally expensive step, we will use a newly developed accelerated domain integral algorithm based on FFT on multipoles (similar to the fast Laplace solver). This algorithm will be discussed in details in Section 2.2.

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

$$u(\mathbf{x}) = u_p(\mathbf{x}) + u_h(\mathbf{x}). \tag{4}$$

The particular solution u_p satisfies the Poisson equation, but not necessarily the boundary conditions.

$$\nabla^2 u_p(\mathbf{x}) = f, \quad \mathbf{x} \in \Omega \tag{5}$$

The homogeneous solution u_h satisfies the corresponding Laplace equation

$$\nabla^2 u_h(\mathbf{x}) = 0, \quad \mathbf{x} \in \Omega \tag{6}$$

and combines with the particular solution to enforce the boundary conditions of the original Poisson equation. For Dirichlet boundary condition $u = f_1$ given on the boundary S_1,

$$u_h(\mathbf{x}) = f_1(\mathbf{x}) - u_p(\mathbf{x}), \quad \mathbf{x} \in S_1, \tag{7}$$

and for Neumann boundary condition $\frac{\partial u}{\partial \mathbf{x}} = f_2$ given on the boundary S_2,

$$\frac{\partial u_h(\mathbf{x})}{\partial \mathbf{n}} = f_2(\mathbf{x}) - \frac{\partial u_p(\mathbf{x})}{\partial \mathbf{n}}, \quad \mathbf{x} \in S_2. \tag{8}$$

In this paper, the particular solution will be obtained rapidly by performing a FFT on f (as detailed in Section 2.3), and the homogeneous part is solved using the FFTM algorithm for the Laplace equation.

2.2 FFT-Multipole accelerated domain integral

To evaluate the domain integral, the solution domain needs to be discretized into volume elements. In the traditional BEM implementation, the domain integral is

performed over all the volume elements, and this is added to the right hand side of the Laplace solver. With the use of FFTM as the fast Laplace solver, the solution domain is already being divided into cells for the formation of multipoles and local expansions. These cells provide a ready form of volume discretization, which will be used to evaluate the domain integral using the Gaussian quadrature. The domain integral is easily evaluated when the cell lies entirely in the domain of computation. However, when a cell intersects the boundary, as shown in Figure 1, special treatment needs to be taken in evaluating the domain integral over that cell. For the present method, 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 domain integral can be treated as evaluation of potential due to volume sources, and it is accelerated by multipoles just like for the Laplace equation with boundary sources. The multipoles corresponding to the volume sources are readily combined with the multipoles from boundary sources, and the FFTM algorithm for the Laplace solver can be used to evaluate the potentials and solve the Poisson equation. This method accelerates both the surface and domain integrals, and the multipole representation allows both to be treated together seamlessly.

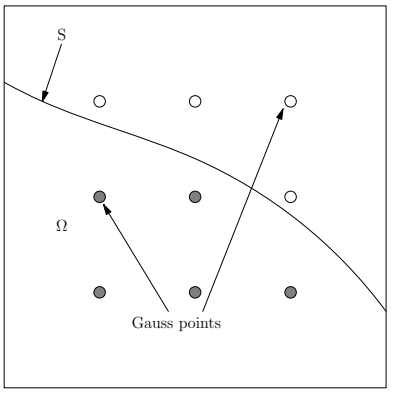


Figure 1: Identification of Gauss points in the interior of the domain, when a cell intersects boundary. Values of sources at Gauss points outside the boundary are set to zero.

For the FFTM Laplace solver, it is desirable to use relatively large cells (encompassing a fair number of boundary elements) to ensure a good computational efficiency for the surface integral [Ong, Lim, Lee, and Lee (2003)]. However, for the domain integral, small volume elements or cells are preferred for good accuracy.

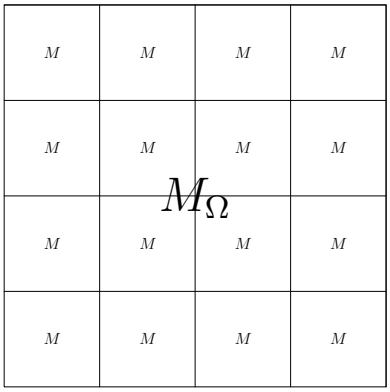


Figure 2: A cell being sub-divided into smaller cells to improve the accuracy of domain integral via FFTM. The multipoles M in the smaller cells are transformed to the initial cell center.

For the present implementation, the large cells used for the FFTM are sub-divided into smaller cells when the domain integral is being evaluated, as shown in Figure 2. The multipole moments for the domain sources are obtained at the smaller cell centers, and then translated to the cell centers of the bigger cells,

$$M^m_{\Omega,n} = \sum_{n'=0}^{n} \sum_{m'=-n'}^{n'} R^{m'}_{n'} M^{m-m'}_{n-n'}. \tag{9}$$

These multipole moments for the domain sources (M_Ω) are then combined with those from the boundary sources.

For the near field (shaded area in Figure 3), the domain integral is performed by Gaussian quadrature directly. When the evaluation node point is within or close to a cell that the integral is performed, a weak singularity appears, and it is regularized by the following coordinate transform

$$\begin{aligned} \int_\Omega G(\mathbf{x},\mathbf{y}) f(\mathbf{y}) d\Omega(\mathbf{y}) &= \int_{y_{11}}^{y_{12}} \int_{y_{21}}^{y_{22}} \int_{y_{31}}^{y_{32}} \frac{f(y_1,y_2,y_3)}{4\pi r(y_1,y_2,y_3)} dy_1 dy_2 dy_3 \\ &= \int_0^{r_b(\theta,\phi)} \int_0^{\pi} \int_0^{2\pi} \frac{1}{4\pi} f(r,\theta,\phi) r \sin\theta dr d\theta d\phi. \end{aligned} \tag{10}$$

2.3 FFT accelerated particular solution

The particular solution provides an alternative method of handling the domain integral for Poisson-type equations. The solution of the Poisson-type equations can be

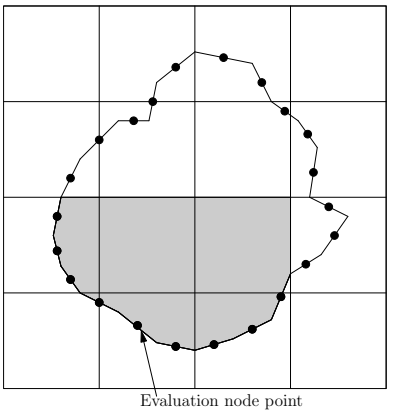


Figure 3: Standard domain integral is used for calculating the near field (shaded area) interaction.

separated into a homogeneous solution and a particular solution. The particular solution is calculated rapidly using FFT, while the homogeneous solution is obtained by solving the Laplace equation using FFTM.

In the present implementation, the non-homogeneous function f is extended to the rectangular region that bounds the solution domain, as defined by $[a_1,b_1]\times[a_2,b_2]\times[a_3,b_3]$ in the FFTM method. The rectangular domain 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. This grid is used to calculate the particular solution via the FFT. The non-homogeneous term f is represented by its Fourier coefficients $\hat{f}_{lmn}$, as given by

$$f(x_1,x_2,x_3) = \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\hat{f}_{lmn}\sin\left(\frac{x_1-a_1}{b_1-a_1}l\pi\right)\sin\left(\frac{x_2-a_2}{b_2-a_2}m\pi\right)\sin\left(\frac{x_3-a_3}{b_3-a_3}n\pi\right), \quad (11)$$

where the coefficients are obtained by the fast Fourier sine transform

$$\hat{f}_{lmn} = \frac{8}{(b_1-a_1)(b_2-a_2)(b_3-a_3)}\int_{a_1}^{b_1}\int_{a_2}^{b_2}\int_{a_3}^{b_3} f(x_1,x_2,x_3)\sin\left(\frac{x_1-a_1}{b_1-a_1}l\pi\right)\sin\left(\frac{x_2-a_2}{b_2-a_2}m\pi\right)\sin\left(\frac{x_3-a_3}{b_3-a_3}n\pi\right)dx_1dx_2dx_3. \quad (12)$$

Similar to the previous section, the values of $f(\mathbf{x})$ are set to zero when the grid point $\mathbf{x}$ falls outside the domain of the problem Ω. The particular solution can readily be

obtained by

$$u_p(x_1,x_2,x_3) = \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\hat{u}_{p,lmn}\sin\left(\frac{x_1-a_1}{b_1-a_1}l\pi\right)\sin\left(\frac{x_2-a_2}{b_2-a_2}m\pi\right)\sin\left(\frac{x_3-a_3}{b_3-a_3}n\pi\right), \quad (13)$$

where,

$$\hat{u}_{p,lmn} = -\frac{\hat{f}_{lmn}/\pi^2}{\frac{l^2}{(b_1-a_1)^2}+\frac{m^2}{(b_2-a_2)^2}+\frac{n^2}{(b_3-a_3)^2}}. \quad (14)$$

The particular solution values on the grid points are obtained by inverse fast Fourier sine transform on $\hat{u}$. The particular solution at the the nodes on the boundary are then obtained from the values at the grid points using a three-dimension 64-point Lagrange interpolation.

2.4 Iterative solver for non-linear Poisson-type equations

When the non-homogeneous term is a non-linear function of the variable to be solved, an iterative scheme needs to be used. Here, we have chosen a simple Richardson iterative scheme to illustrate the incorporation of the fast solvers into solution method.

At each iteration step (t), the variable and normal derivatives, denoted by u_t and $\partial u_t/\partial \mathbf{n}$, are known from the previous iteration or an initial guess at the first step. The iteration step involves solving the following equation

$$\begin{aligned} \nabla^2 u_{t+1}(\mathbf{x}) &= f(u_t), \quad \mathbf{x}\in\Omega \\ u_{t+1}(\mathbf{x}) &= f_1(\mathbf{x}), \quad \mathbf{x}\in S_1 \\ \frac{\partial u_{t+1}(\mathbf{x})}{\partial \mathbf{n}} &= f_2(\mathbf{x}), \quad \mathbf{x}\in S_2 \end{aligned} \quad (15)$$

for the values u_{t+1} and $\partial u_{t+1}/\partial \mathbf{n}$. The solution at the boundary is first obtained, followed by the domain solution. The iterative process is repeated until the two successive sets of solution, at t and $t+1$, are within a preset tolerance. The particular solution method was found to be more efficient than the FFT-multipole accelerated domain integral method (see the numerical examples in the next section), so the particular solution method is implemented in the present solution for non-linear Poisson-type equations.

The details of each iteration step in fast non-linear algorithm are given as follows.

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 boundary conditions for the Laplace equation.

3. Obtain the homogeneous solution $u_{h,t+1}$ and $\partial u_{h,t+1}/\partial \mathbf{n}$ on the boundary nodes by solving the corresponding Laplace equation with the FFTM (as outlined in the Appendix).

4. Evaluate the homogeneous solution $u_{h,t+1}$ at the interior grid points

$$u_{h,t+1}(\mathbf{x}) = \int_S G(\mathbf{x},\mathbf{y}) \frac{\partial u_{h,t+1}(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})} dS(\mathbf{y}) - \int_S H(\mathbf{x},\mathbf{y}) u_{h,t+1}(\mathbf{y}) dS(\mathbf{y}). \tag{16}$$

5. Combine the particular solution and the homogeneous solution to obtain 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 the values u_{t+1} and $\partial u_{t+1}/\partial \mathbf{n}$ with the previous set of u_t and $\partial u_t/\partial \mathbf{n}$. 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 domain integral in Equation (16), the surface integrals are rapidly evaluated using the FFTM procedure. When the distance between a evaluation point and a boundary element is very small (smaller than half size of a typical element), the analytical formula for nearly singular integration [Hayami (1992)] is used to obtain good accuracy.

3 Numerical examples

In this section, several numerical examples are given to investigate the accuracy and computational efficiency of the fast Poisson solvers. For all the problems, the computational domain Ω is a sphere with radius 0.5 with the sphere center placed at the origin. Dirichlet boundary conditions are prescribed on the sphere as specified in each problem. The normal derivative of the variable is calculated, and the measure of the error is defined in the L_2 norm as

$$Error = \sqrt{\frac{\sum_{i=1}^{N} |\partial u(\mathbf{x}_i)/\partial \mathbf{n} - \partial u^*(\mathbf{x}_i)/\partial \mathbf{n}|^2}{\sum_{i=1}^{N} |\partial u^*(\mathbf{x}_i)/\partial \mathbf{n}|^2}}, \tag{17}$$

where u^* is the analytical solution.

For surface integral, 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 is 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 nodes being 4858, 8566, 19234 and 33884. When implementing the FFTM to accelerate the surface integral, the rectangular domain is discretized into $16 \times 16 \times 16$ cells, and the multipole and local expansion orders of $p = 4$ and $p = 6$ are used for comparison.

To compare the two methods in Sections 2.2 and 2.3, the number of Gauss points used to perform the domain integral is the same as the number of grid points used to calculate the particular solution in the rectangular domain. This does not guarantee that the number of the Gauss points and the number of grid points inside the computational domain Ω to be the same, but they only differ slightly. In the following examples, three different sets of Gauss/grid points are studied, namely $128 \times 128 \times 128$, $256 \times 256 \times 256$ and $512 \times 512 \times 512$. The domain integral is calculated with $8 \times 8 \times 8$ Gauss points in each cell. With $16 \times 16 \times 16$ cells used for accelerating the surface integral, the number of Gauss points used for the accelerated domain integral 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 domain integral, the number of Gauss points is increased to $(32 \times 8) \times (32 \times 8) \times (32 \times 8) = 256 \times 256 \times 256$ and $(64 \times 8) \times (64 \times 8) \times (64 \times 8) = 512 \times 512 \times 512$. Also, two multipole and local expansion orders, $p = 4$ and $p = 6$, are used to study the accuracy of multipole transform for interior sources.

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 differential equation is given as follows

$$\nabla^2 u(\mathbf{x}) = 1, \quad \mathbf{x} \in \Omega, \tag{18}$$

with the prescribed boundary condition

$$u(\mathbf{x}) = \frac{x_1^2}{2}, \quad \mathbf{x} \in S, \tag{19}$$

on the surface of the sphere. The analytical solution for the normal derivative on the sphere is given by

$$\frac{\partial u^*}{\partial \mathbf{n}} = x_1 n_1, \quad \mathbf{x} \in S, \tag{20}$$

where x_i and n_i $(i = 1,2,3)$ are the components of the position coordinate $\mathbf{x}$ and normal $\mathbf{n}$ to the surface, respectively.

Figures 4 and 5 show the scaling of the computational time in handling the non-homogeneous term (by standard domain integral, multipole accelerated domain integral, and FFT accelerated particular solution method) against the number of Gauss/grid points and the number of nodes, respectively.

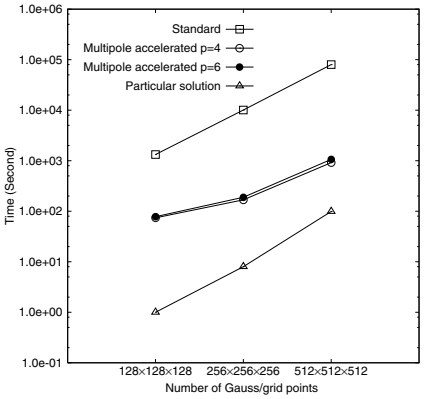


Figure 4: Computational time for domain integral using standard method (Standard), multipole accelerated domain integral method (Multipole accelerated) and FFT accelerated particular solution method (Particular solution) against the number of Gauss/grid points for a fixed number of nodes (33884).

Both fast algorithms are found to reduce the computational time significantly compared to the standard method. In addition, the particular solution is faster than the multipole accelerated domain integral by an order of magnitude. Increasing the expansion order p from 4 to 6 only increases the computational time of the accelerated domain integral marginally. For all the three methods, using more Gauss/grid points results in longer computational time, as shown in Figure 4. The time of the particular solution method scales as $O(N_g \log N_g)$ with respect of the number of grid points (N_g), which comes from the complexity of the FFT. For both the standard and multipole accelerated volume integral methods, the computational time scales as the number of Gauss points $O(N_g)$ used. The multipole method provides significant speed up for far field calculations. The near field interactions need to be

evaluated directly, and some of these involved weakly singular integral that needs to be regularized. When a small number of Gauss points are used, the proportion of these near field calculations is larger; hence, the computational time for the multipole accelerated method is slightly above the linear trend (in Figure 4) when $128 \times 128 \times 128$ Gauss points are used.

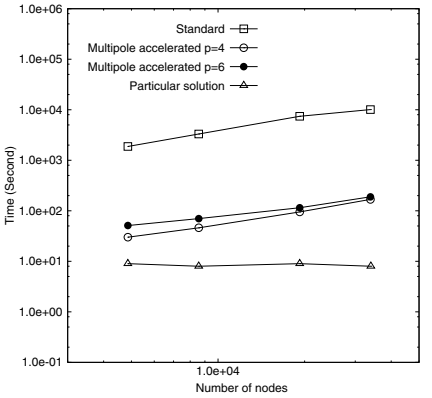


Figure 5: Computational time for domain integral using standard method (Standard), multipole accelerated domain integral method (Multipole accelerated) and FFT accelerated particular solution method (Particular solution) with a fixed number of Gauss/grid points ($256 \times 256 \times 256$).

Figure 5 shows the time needed to calculate the non-homogeneous term against the number of nodes in the BEM. This non-homogeneous term is evaluated at each boundary node, and it forms the "source" term in each boundary integral equation written at each node. For the particular solution method, the computational time is virtually unaffected by the number of nodes, since this method depends only on the number of grid points used in the FFT. The non-homogeneous term is evaluated at all the grid points together, and then the values at the boundary nodes are interpolated from those at the grid points. The computational time needed for interpolation of values at the nodes is insignificant compared to the FFT process, hence there appears to be no dependence of the computational time on the number of boundary nodes.

For the standard and multipole accelerated domain integral methods, the computational time scales almost linearly with the number of boundary nodes present. This is expected for the standard method, since the non-homogeneous term is evaluated directly at each node on the boundary. For the multipole accelerated domain integral, the computational time is reduced significantly be separating the calculation into the near- and far field sources. The far field calculation is accelerated using

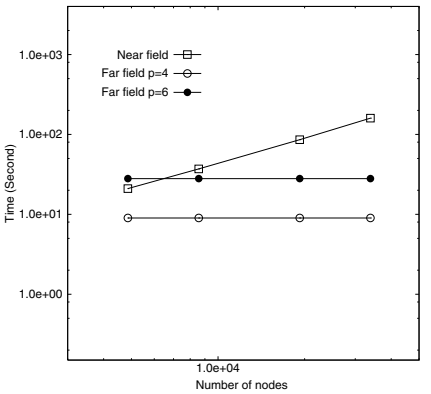
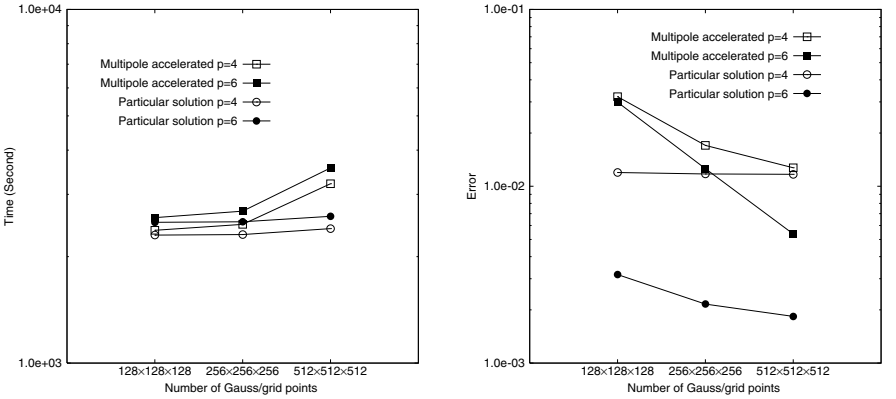


Figure 6: Comparison of computational time for near field and far field calculation with the multipole accelerated domain integral method ($256 \times 256 \times 256$ Gauss points)

multipoles, which depends mainly on the number of cells or Gauss points used. The near field calculation involves direct computation between the nodes and the sources, and hence scales linearly with the number of nodes. The near field calculation takes up a more significant amount of computational time than the far field calculations, resulting in an overall computational time that scales linearly with the number of nodes. This situation is illustrated in Figure 6, which shows the split of computational time taken for the near and far field calculations. The near field computation time scales linearly with the number of nodes, while far field computation time is practically constant, similar to the FFT accelerated particular solution method. Hence, for a problem with a large number of nodes, the near field computation time becomes dominant, and the multipole accelerated domain integral method becomes less efficient. It also is noted that using a higher order of multipole expansion (from $p = 4$ to $p = 6$) increases the far field computational time, but this is still insignificant compared to the near field computational time. Hence, the overall computational times for both orders of multipole expansion differ only slightly in Figure 5.

Figure 7(a) shows the total time taken to solve the Poisson equation using both the accelerated solvers: (i) FFT accelerated particular solution method with FFTM Laplace solver for the homogeneous solution, and (ii) multipole accelerated domain integral merged with the original FFTM Laplace solver. The computational time for the largest problem (with 33884 nodes) is reported, and its scaling with different number of Gauss/grid points is shown. The computational time for the multipole accelerated domain integral with FFTM solver increases when more Gauss points



(a) Total time (b) Accuracy

Figure 7: Comparison of multipole accelerated domain integral method (Multipole accelerated) and FFT accelerated particular solution method (Particular solution). (a) Total computational time for solving the Poisson equation. (b) Error of numerical solution as compared with analytical solution. (constant non-homogeneous term, 33884 nodes)

are used, which is consistent with Figure 4. The computational time for the particular solution method with FFTM Laplace solver is now closer to that of the multipole accelerated domain integral with FFTM, in contrast with the vast difference in computational time for just calculating the non-homogeneous part (in Figure 4). This shows that the FFTM Laplace solver constitutes a dominant part of the computational time in the entire solution process of the Poisson equation using the particular and homogeneous solutions method. Also, a higher order of multipole expansion p increases the total time taken, as expected for the FFTM Laplace solver. Nevertheless, the particular solution method is still faster than the multipole accelerated domain integral method when they are combined with the FFTM Laplace solver.

Figure 7(b) shows the accuracy of the two accelerated methods with respect to the order of multipole expansion and number of Gauss/grid points used. When more Gauss/grid points are used, the error in both methods decreases, since the non-homogeneous part is evaluated more accurately. Similarly, the accuracy improves when a higher order of multipole expansion is used in the FFTM Laplace solver and multipole accelerated domain integral. This improvement in accuracy, when p is increased from 4 to 6, is more significant for the particular solution method, with the error decreasing by about 5 times. This shows that the FFTM Laplace solver contributes to a significant part of the total error in the entire solution process, and particular solution method used in calculating the non-homogeneous part incurs

only a small portion of the overall error. Lastly, for the same number of Gauss/grid points used, the particular solution method is always more accurate than the multipole accelerated domain integral method.

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

This example is similar to the first example, except that the non-homogeneous term is no longer constant, but given as a function of spatial coordinate

$$\nabla^2 u(\mathbf{x}) = (2x_2^3 + 6x_2)e^{x_1+x_3}, \quad \mathbf{x} \in \Omega. \tag{21}$$

The prescibed boundary condition on the sphere is

$$u(\mathbf{x}) = x_2^3 e^{x_1+x_3}, \quad \mathbf{x} \in S, \tag{22}$$

and the analytical solution for the normal derivative on the sphere is

$$\frac{\partial u^*}{\partial \mathbf{n}} = x_2^2 e^{x_1+x_3}(x_2 n_1 + 3n_2 + x_2 n_3), \quad \mathbf{x} \in S. \tag{23}$$

Since the change of the non-homogeneous term does not influence the computational time, the timings should be the same with the previous example. So only the accuracy of the solution procedures is discussed to confirm the findings in the previous example. Figure 8 shows the error of the solution against the number of Gauss/grid points used. A similar behavior in convergence of the solution is observed. In this example, the non-homogeneous term being non-constant results in slightly higher errors, especially when lesser Gauss/grid points are used. With sufficient Gauss/grid points, the good accuracy is achieved as the previous example. The results obtained using the particular solution method are consistently more accurate than those from the accelerated domain integral method, and they show faster convergence with more grid points used.

The above two examples illustrate the efficiency of the FFT accelerated particular solution method over the FFT-multipole accelerated domain integral method. The particular solution method is not only faster; it is also more accurate for the same number of Gauss/grid points used. Moreover, it is computationally less expensive to improve the accuracy of the solution by using more grid points, since it has a better rate of convergence. Hence, the FFT-accelerated particular solution method will be used for solving the non-linear Poisson equations. This method also has the advantage of not evaluating the domain integral explicitly, which tends to be computationally expensive. In the following examples, $256 \times 256 \times 256$ grid points are utilized, after balancing the needs for computational speed and accuracy.

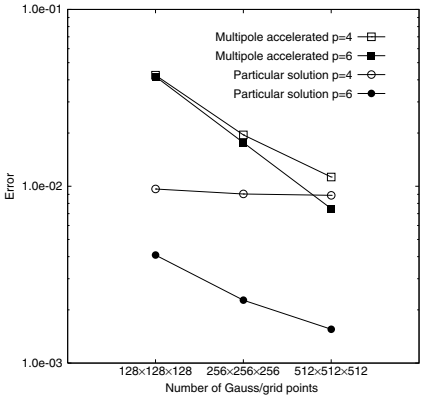


Figure 8: Comparison of the accuracy of the multipole accelerated domain integral method (Multipole accelerated) and FFT accelerated particular solution method (Particular solution). (non-constant non-homogeneous term, 33884 nodes)

3.3 Non-homogeneous modified Helmholtz equation

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

$$\nabla^2 u(\mathbf{x}) - k^2 u(\mathbf{x}) = h(\mathbf{x}), \tag{24}$$

with $k^2 = 1$. This linear equation can be solved by the same method for linear Poisson equation, by using the appropriate kernels for this Helmholtz equation. However, we are hereby using it as a preliminary test problem for our iterative Richardson scheme for non-linear Poisson-type equations. The equation is re-cast as a Poisson-type equation

$$\nabla^2 u(\mathbf{x}) = u + h(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{25}$$

For the function h given by

$$h(\mathbf{x}) = 4x_1^2 + 4x_2^2 + 12x_1x_2 - 3x_1^3x_2 - 2x_1^2x_2^2 + x_1x_2^3 - x_3 \tag{26}$$

and boundary condition of

$$u(\mathbf{x}) = 3x_1^3x_2 + 2x_1^2x_2^2 - x_1x_2^3 + x_3, \quad \mathbf{x} \in S, \tag{27}$$

on the surface of the sphere, the analytical solution for the normal derivative is given by

$$\frac{\partial u^*}{\partial \mathbf{n}} = (9x_1^2 + 4x_1x_2^2 - x_2^3)n_1 + (3x_1^3 + 4x_1^2x_2 - 3x_1x_2^2)n_2 + n_3, \quad \mathbf{x} \in S. \tag{28}$$

In contrast with the previous examples, the interior source term in this example needs to be calculated at each iteration. This involves an additional step of evaluation of the sources at the interior grid points in the solution process.

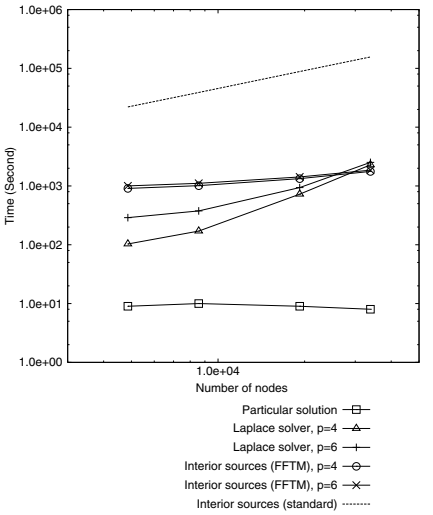
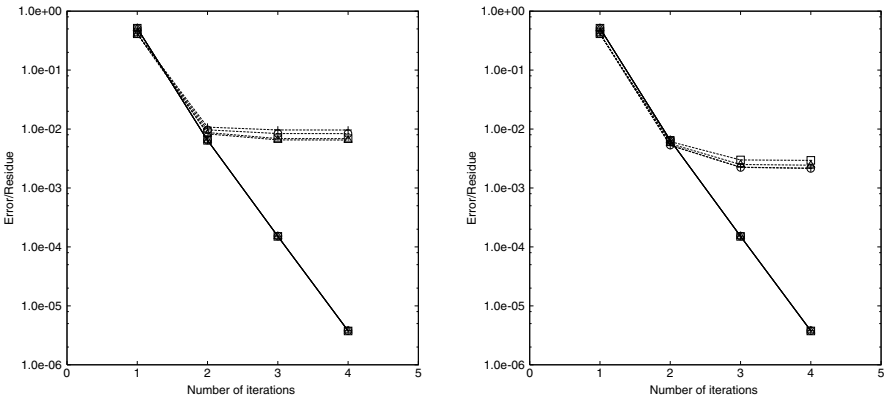


Figure 9: Computational timings for the main steps (Section 2.4) in each time iteration. Step 2: calculating a particular solution accelerated by FFT (Particular solution); Step 3: solving a Laplace equation with FFTM, $p = 4, 6$, (Laplace solver); Step 4: updating the interior values with standard integral method (Interior sources (standard)) and FFTM, $p = 4, 6$, (Interior sources (FFTM)).

The FFT accelerated particular solution method together with the FFTM Laplace solver will be used in this example since it was shown to be computationally more efficient and accurate. Figure 9 shows the time taken for the various steps of computation in each iteration.

The interior source values depend on the solution of u from the previous iteration. This value of u at the grid point is obtained from the particular solution (which is readily available), and the homogeneous solution using the boundary integral given by Equation (16). The computational times taken to evaluate this boundary integral using the standard method and FFTM are shown in Figure 9. The standard method is shown to be computationally expensive, requiring more than 43 hours for largest problem with 33,884 nodes. The FFTM reduces the computational time to evaluate these interior sources significantly. The FFTM Laplace solver for solving the homogeneous part of the solution is included for comparison. It can be seen that the interior source calculation using FFTM takes slightly more time than the

FFTM Laplace solver. Also, the interior source calculation does not scale linearly with the number of nodes as this calculation is dominated by the number of grid points ($256 \times 256 \times 256$ in this case), which is much larger than the largest number of nodes present ($N = 33,884$). In comparison, the evaluation of the particular solution takes the least time in the entire solution process. Hence, the evaluation of interior sources and the Laplace solver for homogeneous solution are the time-determining steps in the entire solution process.



(a) $p = 4$ (b) $p = 6$

Figure 10: Convergence behavior for solving $\nabla^2 u = u + h(\mathbf{x})$. Dahed line: Error; Solid line: Residue; $\square$: 4858 nodes; $\triangle$: 8566 nodes; $\bigcirc$: 19234 nodes; $+$: 33884 nodes.

Figure 10 shows the convergence behavior of the solution process. The residue, summed over all nodes, between two successive iterations (as shown by the solid lines) is defined as

$$\textit{Residue} = \sqrt{\frac{\sum_{i=1}^{N} |\partial u_{t+1}(\mathbf{x}_i)/\partial \mathbf{n} - \partial u_t(\mathbf{x}_i)/\partial \mathbf{n}|^2}{\sum_{i=1}^{N} |\partial u_t(\mathbf{x}_i)/\partial \mathbf{n}|^2}}. \tag{29}$$

The problem takes four iterations to reach a residue of less than 1×10^{-5}. The convergence for this problem is very fast and 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 of the numerical results compared to the analytical solution 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 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 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%

3.4 Non-linear Poisson-type equation

In this example, a truly non-linear Poisson-type equation (Equation (30)) is considered,

$$\nabla^2 u(\mathbf{x}) = u + u^3, \quad \mathbf{x} \in \Omega. \tag{30}$$

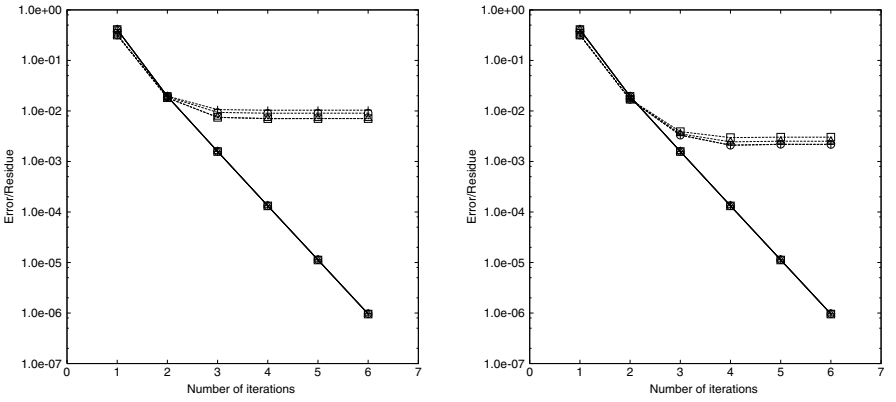
For the boundary condition

$$u(\mathbf{x}) = \tan(\frac{x_1 + x_2 + x_3}{\sqrt{6}}), \quad \mathbf{x} \in S, \tag{31}$$

prescribed on the surface of the sphere, the analytical solution is given by

$$\frac{\partial u^*}{\partial \mathbf{n}} = \frac{1+u^2}{\sqrt{6}}(n_1 + n_2 + n_3), \quad \mathbf{x} \in S. \tag{32}$$

For this non-linear problem, the convergence (Figure 11) is slower than that in the previous 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 neither dependent on the number of nodes, nor the expansion order p. Table 3 shows the results after four iterations. The results are similar with Table 2, and again $p = 6$ is preferred for large problems.



(a) $p = 4$ (b) $p = 6$

Figure 11: Convergence behavior for solving $\nabla^2 u = u + u^3$. Dahed line: Error; Solid line: Residue; $\square$: 4858 nodes; $\triangle$: 8566 nodes; $\bigcirc$: 19234 nodes; $+$: 33884 nodes.

Table 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%

3.5 Burger's equation

In this example, the static Burger's equation, which is used in the study of fluid mechanics, is solved.

$$\nabla^2 u(\mathbf{x}) = \alpha u \frac{\partial u}{\partial x_3}, \quad \mathbf{x} \in \Omega \tag{33}$$

The equation is solved for the following prescribed boundary condition

$$u(\mathbf{x}) = n_1(\mathbf{x}) + n_2(\mathbf{x}) + n_3(\mathbf{x}), \quad \mathbf{x} \in S. \tag{34}$$

The parameter α is used to adjust the degree of non-linearity in the problem, with a larger value of α denoting a stronger non-linearity. Within the boundary, the solution for $\partial u/\partial x_3$ is calculated. This is obtained using a 4th order finite difference on the values of u at the interior grid points. From the previous examples, it is noted that the results 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 12 shows the number of iterations that is needed to reduce the residue to less than the tolerance. With higher α, more iterations are needed for the solution to reach convergence due to the higher degree of non-linearity. Although the current simple Richardson iterative method appears acceptable, better iteration scheme, like the Newton's method, should be used to obtain better convergence performance. Figure 13 gives the solution u of the Burger's equation on the x_1x_2 plane. The solution is observed to change gradually, as the parameter α is increased.

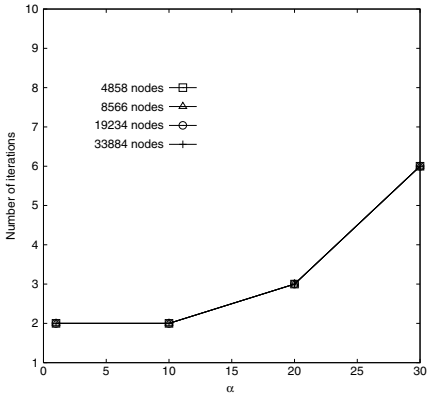
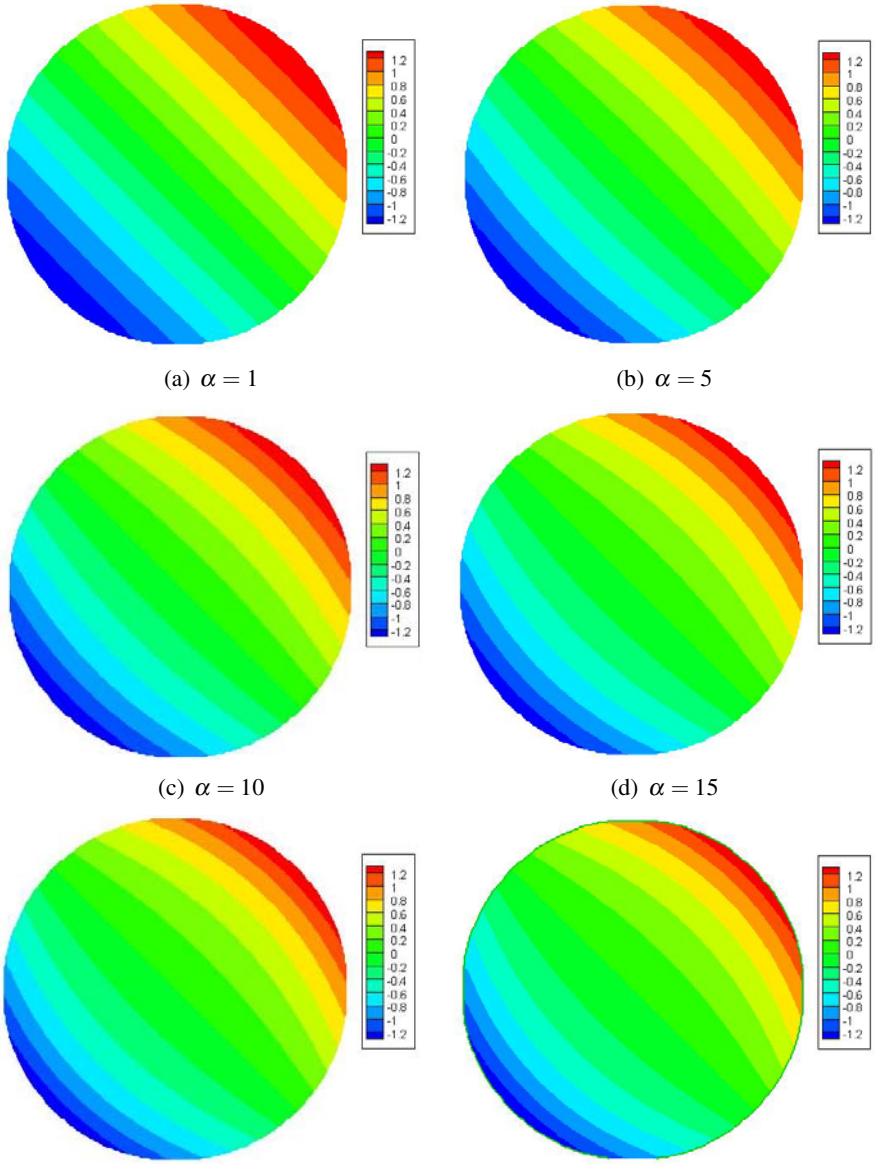


Figure 12: Number of iterations needed to achieve a residue of less than 1×10^{-3} for increasing non-linearity (α) in the problem $\nabla^2 u(\mathbf{x}) = \alpha u \frac{\partial u}{\partial x_3}$.



(a) $\alpha = 1$ (b) $\alpha = 5$

(c) $\alpha = 10$ (d) $\alpha = 15$

(e) $\alpha = 20$ (f) $\alpha = 25$

Figure 13: The contour of solution u on the x_1x_2 plane ($x_3 = 0$)

4 Conclusion

In this paper, we have combined the fast Laplace solver using FFT on multipoles with two fast methods for domain integrals to implement a fast solver for 3D Poisson-type equations. The FFT accelerated particular solution method is found to be superior to the FFT-multipole accelerated domain integral method. The former is extended and successfully applied to non-linear equations where each iterative solve involves a linear Poisson-type equation. In comparing of the Laplace FFTM solver and the domain integral, the Laplace solver is found to be the bottle-neck of the entire fast algorithm. The computational time needed for the domain integral in both methods is less than that taken by the Laplace solver. For non-linear equations, the current implementation uses the Richardson iteration. In future developments, better iterative schemes, like the Newton's method, should be used. Nevertheless, the current fast method is shown to be capable of handling large scale non-linear problems (more than 30,000 degrees of freedom) with good computational efficiency.

Appendix — FFTM for solving Laplace equation

By discretizing the boundary of the solution domain into elements, the BEM converts the boundary integral equations into a system of algebraic equations. This system of algebraic equations can be solved efficiently by iterative methods, such as GMRES. For the Laplace equation, a fast algorithm FFTM [Ong, Lim, Lee, and Lee (2003); Lim, He, and Lim (2008)], is used to accelerated the matrix-vector product within each iteration of the GMRES solver.

The FFTM algorithm uses the FFT on the multipole and local expansions to approximate the far-field potential calculations. The multipole and local expansions can be obtained from the expression for $1/r(\mathbf{x},\mathbf{y})$ in terms of the solid harmonics R_n^m and S_n^m,

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

where

$$R_n^m(\overrightarrow{\mathbf{Oy}}) = \frac{1}{(n+m)!} P_n^m(\cos\alpha) e^{im\beta} \rho^n, \tag{A-2}$$

$$S_n^m(\overrightarrow{\mathbf{Oy}}) = (n-m)! P_n^m(\cos\alpha) e^{im\beta} \frac{1}{\rho^{n+1}}, \tag{A-3}$$

and $\mathbf{O}$ is the cell center, with (ρ,α,β) being the relative spherical coordinates of the point $\mathbf{y}$ from $\mathbf{O}$. P_n^m is the associated Legendre function of the first kind. The FFTM

performs three translations, namely source to multipole moment (*S*2*M*), multipole moment to local expansion (*M*2*L*) and local expansion to destination (*L*2*D*). The single layer source ($\partial u(\mathbf{y})/\partial \mathbf{n}(\mathbf{y})$) and double layer source ($u(\mathbf{y})$) are translated into multipole moment(*S*2*M*) by

$$M_n^m(\mathbf{O}) = \int_S \frac{\partial R_n^m(\overrightarrow{\mathbf{Oy}})}{\partial \mathbf{n}(\mathbf{y})} u(\mathbf{y}) dS_y - \int_S R_n^m(\overrightarrow{\mathbf{Oy}}) \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})} dS_y. \tag{A-4}$$

This multipole moment is translated to local expansion defined at a point $\mathbf{O}'$ (*M*2*L*) by

$$L_{n'}^{m'}(\mathbf{O}') = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} (-1)^{n'} \overline{S_{n+n'}^{m+m'}}(\overrightarrow{\mathbf{OO}'}) M_n^m(\mathbf{O}). \tag{A-5}$$

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

$$L_{n'}^{m'}(x_1, x_2, x_3) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} [\sum_{x_1'} \sum_{x_2'} \sum_{x_3'} (-1)^{n'} \overline{S_{n+n'}^{m+m'}}(x_1 - x_1', x_2 - x_2', x_3 - x_3') M_n^m(x_1', x_2', x_3'), \tag{A-6}$$

where 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 is accelerated by the FFT. The free software FFTW (Fastest Fourier Transform in the West), provided by Frigo and Johnson [Frigo and Johnson (2005)] is used.

Lastly, the field value at the destination point is obtained from the local expansion coefficients L_n^m by (*L*2*D*):

$$\int_S H(\mathbf{x}, \mathbf{y}) u(\mathbf{y}) dS(\mathbf{y}) - \int_S G(\mathbf{x}, \mathbf{y}) \frac{\partial u(\mathbf{y})}{\partial \mathbf{n}(\mathbf{y})} dS(\mathbf{y}) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} R_n^m(\overrightarrow{\mathbf{O}'\mathbf{x}}) L_n^m(\mathbf{O}'). \tag{A-7}$$

References

Aoki, S.; Amaya, K.; Urago, M.; Nakayama, A. (2004): Fast multipole boundary element analysis of corrosion problems. *CMES: Computer Modeling in Engineering & Sciences*, vol. 6, no. 2, pp. 123–131.

Becker, A. A. (1992): *Boundary Element Method in Engineering: a Complete Course*. McGraw-Hill.

Brebbia, C. A.; Dominguez, J. (1992): *Boundary Elements An Introductory Course*. Computational Mechanics Publications, 2nd edition.

Chew, W. C.; Song, J. M.; Cui, T. J.; Velarnparambil, S.; Hastriter, M. L.; Hu, B. (2004): Review of large scale computing in electromagnetics with fast integral equation solvers. *CMES: Computer Modeling In Engineering & Sciences*, vol. 5, no. 4, pp. 361–372.

Ding, J.; Ye, W. J. (2006): A grid-based integral approach for quasilinear problems. *Computational Mechanics*, vol. 38, no. 2, pp. 113–118.

Ding, J.; Ye, W. J.; Gray, L. J. (2005): An accelerated surface discretization-based BEM approach for non-homogeneous linear problems in 3-D complex domains. *International Journal for Numerical Methods in Engineering*, vol. 63, no. 12, pp. 1775–1795.

Ethridge, F.; Greengard, L. (2001): A new fast-multipole accelerated Poisson solver in two dimensions. *SIAM Journal on Scientific Computing*, vol. 23, no. 3, pp. 741–760.

Frigo, M.; Johnson, S. G. (2005): The design and implementation of FFTW3. *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231.

Greengard, L.; Lee, J. Y. (1996): A direct adaptive Poisson solver of arbitrary order accuracy. *Journal of Computational Physics*, vol. 125, no. 2, pp. 415–424.

Greengard, L.; Rokhlin, V. (1987): A fast algorithm for particle simulations. *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348.

Hayami, K. (1992): *A projection transformation method for nearly singular surface boundary element integrals*. Springer-Verlag.

Henry, D. P.; Banerjee, P. K. (1988): A new boundary element formulation for two- and three-dimensional thermoelasticity using particular integrals. *International Journal for Numerical Methods in Engineering*, vol. 26, no. 9, pp. 2061–2077.

Ingber, M. S.; Mammoli, A. A.; Brown, M. J. (2001): A comparison of domain integral evaluation techniques for boundary element methods. *International Journal for Numerical Methods in Engineering*, vol. 52, no. 4, pp. 417–432.

Kurz, S.; Rain, O.; Rjasanow, S. (2002): The adaptive cross-approximation technique for the 3-D boundary-element methods. *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 421–424.

Liao, S. J. (1998): On the general boundary element method. *Engineering Analysis with Boundary Elements*, vol. 21, no. 1, pp. 39–51.

Lim, K. M.; He, X. F.; Lim, S. P. (2008): Fast Fourier Transform on Multipoles (FFTM) Algorithm for Laplace Equation with Direct and Indirect Boundary Element Method. *Computational Mechanics*, vol. 41, no. 2, pp. 313–323.

Mammoli, A. A. (2002): Solution of non-linear boundary integral equations in complex geometries with auxiliary integral subtraction. *International Journal for Numerical Methods in Engineering*, vol. 55, no. 9, pp. 1115–1128.

McKenney, A.; Greengard, L.; Mayo, A. (1995): A fast Poisson solver for complex geometries. *Journal of Computational Physics*, vol. 118, no. 2, pp. 348–355.

Nabors, K.; White, J. (1991): FastCap: a multipole accelerated 3-D capacitance extraction program. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 11, pp. 1447–1459.

Nishimura, N. (2002): Fast multipole accelerated boundary integral equation methods. *Applied Mechanics Reviews*, vol. 55, no. 4, pp. 299–324.

Nishimura, N.; Yoshida, K.; Kobayashi, S. (1999): A fast multipole boundary integral equation method for crack problems in 3D. *Engineering Analysis with Boundary Elements*, vol. 23, no. 1, pp. 97–105.

Nowak, A. J.; Neves, A. C. (1994): *The Multiple Reciprocity Boundary Element Method*. Computational Mechanics Publications.

Ong, E. T.; Lim, K. M.; Lee, K. H.; Lee, H. P. (2003): A fast algorithm for three-dimensional potential fields calculation: fast Fourier transform on multipoles. *Journal of Computational Physics*, vol. 192, no. 1, pp. 244–261.

Partridge, P. W.; Brebbia, C. A.; Wrobel, L. C. (1992): *The Dual Reciprocity Boundary Element Method*. Computational Mechanics Publications.

Phillips, J. R.; White, J. K. (1997): A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1059–1072.

Pollandt, R. (1998): 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*, vol. 78, no. 8, pp. 545–553.

Volakis, J. L.; Sertel, K.; Jorgensen, E.; Kindt, R. W. (2004): Hybrid finite element and volume integral methods for scattering using parametric geometry. *CMES: Computer Modeling In Engineering & Sciences*, vol. 5, no. 5, pp. 463–476.

Wang, H. T.; Yao, Z. H. (2005): 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*, vol. 7, no. 1, pp. 85–95.

Wang, H. T.; Yao, Z. H. (2008): A rigid-fiber-based boundary element model for strength simulation of carbon nanotube reinforced composites. *CMES: Computer Modeling in Engineering & Sciences*, vol. 29, no. 1, pp. 1–13.

Xu, S. Q.; Kamiya, N. (1998): A formulation and solution for boundary element analysis of inhomogeneous-nonlinear problem. *Computational Mechanics*, vol. 22, no. 5, pp. 367–374.

Ying, L. X.; Biros, G.; Zorin, D. (2006): A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains. *Journal of Computational Physics*, vol. 219, no. 1, pp. 247–275.