INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2007; **70**:812–839
Published online 2 November 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.1910

A parallel fast multipole accelerated integral equation scheme for 3D Stokes equations

Haitao Wang[1], Ting Lei[2], Jin Li[3], Jingfang Huang[4,*,†] and Zhenhan Yao[2]

[1]*Institute of Nuclear & New Energy Technology, Tsinghua University, Beijing, China*
[2]*Department of Engineering Mechanics, Tsinghua University, Beijing, China*
[3]*Department of Mathematics, Tsinghua University, Beijing, China*
[4]*Department of Mathematics, University of North Carolina, Chapel Hill, NC, U.S.A.*

SUMMARY

In this paper, we discuss a numerical scheme for the Stokes equations in three dimensions. It uses an integral equation formulation and is accelerated by the new version of fast multipole method first introduced by Greengard and Rokhlin in 1997 (*Acta Numerica* 1997; **6**:229–269). The code is parallelized to solve problems of extremely large size. The resulting numerical solver can be applied to Stokes flows in complex geometry and also serves as a building block for solving the Navier–Stokes equations of low to moderate Reynold's numbers. Copyright © 2006 John Wiley & Sons, Ltd.

Received 10 February 2006; Revised 25 August 2006; Accepted 13 September 2006

KEY WORDS: fast multipole method; integral equation method; Stokes equations; diagonal translation; preconditioner; parallel computing

1. INTRODUCTION

The Stokes equations

$$-\nabla P + \mu\nabla^2\mathbf{u} = 0$$
$$\nabla\cdot\mathbf{u} = 0$$

have been widely used to model incompressible 'creeping flows' where the fluid Reynold's number is very low. The corresponding numerical method also serves as a building block for the

*Correspondence to: Jingfang Huang, CB# 3250, Department of Mathematics, University of North Carolina, Chapel Hill, NC 27599-3250, U.S.A.
†E-mail: huang@amath.unc.edu

Contract/grant sponsor: Tsinghua University

incompressible Navier–Stokes equations with low to moderate Reynold's numbers. Applications include the design of fluidic micromechanical systems (MEMS), the modelling of fluid in human bodies, and the study of ground water in porous medium. The work in this paper is inspired by a project related with biofluidic MEMS design where incompressible Stokes equations are used to model fluid in a device with complicated geometric details and the goal is to optimize device structure for desired performance.

Numerical methods for Stokes flows with complex geometry have been extensively studied in the last 50 years. The finite element method describes the geometry using a set of 'elements' and the solution is represented as a linear combination of basis functions defined on each element. Because of the properties of these basis functions, the resulting linear system using the 'variational formulation' is sparse and can be solved efficiently using specially designed fast solvers for sparse linear systems, such as the multigrid method and domain decompositions. Another class of methods is the Cartesian grid finite difference schemes including the immersed boundary and immersed interface methods [1, 2]. The structured grids simplify the adaptive mesh refinement (AMR) [3–6], algorithm implementation and parallelization. However, as problem size gets larger, it becomes difficult for traditional methods to provide satisfactory simulation results: the condition numbers (reflecting how error is magnified) of linear systems from finite difference and finite element discretizations usually grow with the number of grid points, hence the numerical results quickly loose accuracy unless appropriate preconditioners can be found; and for large-scale problems, existing fast solvers for sparse systems may no longer be both efficient and stable, especially when very high accuracy is required.

In this paper, we focus on a new class of numerical methods emerged in the last 20 years. These methods use integral equation formulations based on potential theory (resulting in better accuracy and stability properties) and are accelerated by fast algorithms for dense linear systems. Potential theory has been extensively studied previously by mathematics community but mostly for the analysis of existence and regularities of solutions to differential equations. It has traditionally been neglected as a numerical scheme. The reason is that integral representations of solutions to differential equations result in dense linear systems. Direct Gauss elimination for such a system with N unknowns requires prohibitive $O(N^3)$ operations. However, the situation is changing rapidly due to the introduction of fast algorithms for dense convolution-type operators, including the $O(N \log N)$ fast Fourier transform (FFT)-based methods (e.g. particle in cell (PiC), particle–particle particle–mesh (P3M), and precorrected FFT (pFFT) [7–9]) and multipole expansion-based methods (e.g. $O(N \log N)$ tree codes [10, 11], and $O(N)$ fast multipole method (FMM) [12, 13]). When accelerated by these fast algorithms, integral equation methods (IEM) can be both efficient and accurate even for very large-scale problems, and they have been successfully used in many science and engineering fields including computational electromagnetics, molecular dynamics and astrophysics [7–11, 14–22]. More recently, IEM and fast algorithms have also been introduced to the field of computational solid and fluid mechanics. In [23–31], FMM accelerated IEM were used to simulate elastic properties of composite material with millions of inclusions; in [32–42], solutions to traditional Stokes and Navier–Stokes equations using integral equation representations were studied. Parallel iterative methods accelerated by fast algorithms have also been extensively studied in [43–49].

In this paper, we focus on an integral equation formulation for the Stokes equations in three dimensions. It uses the Stokelet, stresslet, and a direct integral representation. The numerical scheme is accelerated by the new version of FMM introduced by Greengard and Rokhlin in 1997 [13]. Compared with the original $O(N)$ FMM, the prefactor of the new version was dramatically reduced, especially in three dimensions. Main features of our software package include: (a) the geometry is imported from available CAD software packages; (b) a direct integral formulation is used, with a carefully designed preconditioner; (c) the new version of FMM is implemented for efficient matrix vector multiplications and (d) the code is parallelized in distributed memory architectures for large-scale problems. This numerical package is being used to study fluids in bioMEMS, it also reflects our first step in developing general-purpose numerical simulation toolboxes for the Navier–Stokes equations of low to moderate Reynold's numbers.

The structure of the paper is organized as follows: in Section 2, we briefly discuss integral equation formulation for the Stokes equations. Currently a direct formulation is used, and so in Section 3, we discuss how a traditional preconditioner can be designed to improve the convergence of Krylov subspace-based iterative methods. In Section 4, we generalize the new version of FMM to Stokes equations. In Section 5, we discuss how our simulation toolbox is parallelized. In Section 6, we present several numerical results to show the accuracy and efficiency of our solver, and finally in Section 7, we provide concluding remarks and discuss current research efforts for the optimization of the numerical package.

2. INTEGRAL EQUATION FORMULATION FOR 3D STOKES EQUATIONS

Consider a physical domain V with boundary S. An interior problem refers to the case when fluid flows inside V with velocity or pressure specified along S. Assume that there are no external forces, the governing equations in V are given by the Stokes equations

$$\begin{aligned} -\nabla P + \mu\nabla^2\mathbf{u} &= 0 \\ \nabla\cdot\mathbf{u} &= 0 \end{aligned} \tag{1}$$

where P denotes the pressure, $\mathbf{u}=(u_1,u_2,u_3)$ is the velocity and μ is the viscosity coefficient. For cavity-driven flows, the velocity field is specified along portions of the boundary, and no slip boundary conditions are specified on the rest. In the bioMEMS system we are studying, usually no slip boundary conditions are used except on terminals where either pressure is specified or velocity field is given. For an exterior problem, velocity field is specified at infinity, and boundary conditions on S usually depend on physical properties of V and whether V is moving in fluid. In this section, we briefly study the potential theory for the Stokes equations in three dimensions. This has been extensively studied previously and interested readers are referred to [50] for detailed discussions.

The fundamental solutions of the 3D Stokes equations are given by

$$\begin{aligned} T^*_{ij}(\mathbf{x},\mathbf{y}) &= -\frac{3}{4\pi}\frac{r_i r_j r_k n_k}{r^5} \\ U^*_{ij}(\mathbf{x},\mathbf{y}) &= \frac{1}{8\pi\mu}\left(\frac{\delta_{ij}}{r}+\frac{r_i r_j}{r^3}\right) \end{aligned} \tag{2}$$

The first tensor is usually referred to as the stresslet, and the second the Stokeslet. Using the fundamental solutions in Equation (2), the velocity field $\mathbf{u}=(u_1,u_2,u_3)$ and traction tensor $\mathbf{t}=(t_1,t_2,t_3)$ satisfy the boundary integral equation (BIE)

$$C_{ij}(\mathbf{x})u_j(\mathbf{x}) + \oint_S T^*_{ij}(\mathbf{x},\mathbf{y})u_j(\mathbf{y})\,\mathrm{d}S(\mathbf{y}) = \oint_S U^*_{ij}(\mathbf{x},\mathbf{y})t_j(\mathbf{y})\,\mathrm{d}S(\mathbf{y}) \tag{3}$$

where $\mathbf{x}=(x_1, x_2, x_3)$ and $\mathbf{y}=(y_1, y_2, y_3)$ are on the boundary S representing the target and source points, respectively. In the formula, the index $i, j=1, 2, 3$, $C_{ij}(\mathbf{x})$ is determined by the geometry of S at $\mathbf{x}$: if the outward normal vector at $\mathbf{x}$ is continuous, then $c_{ij}(\mathbf{x})=\delta_{ij}/2$, and r is the distance between $\mathbf{x}$ and $\mathbf{y}$ given by $r=\sqrt{r_i r_i}$ where $r_i=y_i-x_i$. Once $\mathbf{u}$ and $\mathbf{t}$ are derived on S by solving Equation (3) using prescribed boundary conditions, fluid velocity and traction at any point can be obtained using Green's second identity.

To discretize Equation (3), we first let the boundary S be divided into N boundary elements

$$S=\sum_{k=1}^{N} S_k \tag{4}$$

and assume that element S_k has m nodes $y_l^{(k)}$ each associated with a basis function $N_l^{(k)}(\xi_1, \xi_2)$ for $l=1, \ldots, m$ where (ξ_1, ξ_2) are the new co-ordinates on S_k. Then the velocity and traction tensors on S_k can be approximated by

$$\begin{aligned} u_i^{(k)}(\xi_1, \xi_2) &= \sum_{l=1}^{m} N_l^{(k)}(\xi_1, \xi_2) u_i(y_l^{(k)}) \\ t_i^{(k)}(\xi_1, \xi_2) &= \sum_{l=1}^{m} N_l^{(k)}(\xi_1, \xi_2) t_i(y_l^{(k)}) \end{aligned} \tag{5}$$

The discretized Equation (3) is then given by

$$\begin{aligned} C_{ij}(\mathbf{x})u_j(\mathbf{x}) &+ \sum_{k=1}^{N}\sum_{l=1}^{m}\left[\int_{S_k} T_{ij}^*(\mathbf{x}, \mathbf{y}(\xi_1, \xi_2)) N_l^{(k)}(\xi_1, \xi_2)\, \mathrm{d}S(\xi_1, \xi_2)\right] u_j(y_l^{(k)}) \\ &= \sum_{k=1}^{N}\sum_{l=1}^{m}\left[\int_{S_k} U_{ij}^*(\mathbf{x}, \mathbf{y}(\xi_1, \xi_2)) N_l^{(m)}(\xi_1, \xi_2)\, \mathrm{d}S(\xi_1, \xi_2)\right] t_j(y_l^{(k)}) \end{aligned} \tag{6}$$

where $u_j(y_l^{(k)})$ and $t_j(y_l^{(k)})$ are the velocity and traction values on node $y_l^{(k)}$, respectively. Setting $\mathbf{x}$ to be one of the element nodes, the collocation formulation gives the matrix form of Equation (6)

$$H \cdot U = G \cdot F \tag{7}$$

where U and F are discretized boundary velocity and traction vectors, and entries of matrices H and G are formed by integrals of the fundamental solutions and basis functions. Letting $S^U \cup S^F = S$, $\overline{U}_1$ and F_1 be the given velocity and unknown traction on boundaryS^U, and U_2 and $\overline{F}_2$ the unknown velocity and given traction on boundary S^F, Equation (7) can be rewritten as

$$\begin{aligned} &\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{Bmatrix} \overline{U}_1 \\ U_2 \end{Bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{Bmatrix} F_1 \\ \overline{F}_2 \end{Bmatrix} \\ &\qquad \Rightarrow \begin{bmatrix} -G_{11} & H_{12} \\ -G_{21} & H_{22} \end{bmatrix} \begin{Bmatrix} F_1 \\ U_2 \end{Bmatrix} = \begin{bmatrix} -H_{11} & G_{12} \\ -H_{21} & G_{22} \end{bmatrix} \begin{Bmatrix} \overline{U}_1 \\ \overline{F}_2 \end{Bmatrix} \Rightarrow AX=B \end{aligned} \tag{8}$$

where

$$A=\begin{bmatrix} -G_{11} & H_{12} \\ -G_{21} & H_{22} \end{bmatrix}, \quad X=\begin{Bmatrix} F_1 \\ U_2 \end{Bmatrix}, \quad B=\begin{bmatrix} -H_{11} & G_{12} \\ -H_{21} & G_{22} \end{bmatrix} \begin{Bmatrix} \overline{U}_1 \\ \overline{F}_2 \end{Bmatrix}$$

3. PRECONDITIONERS

In engineering community, formulation (3) is often referred to as a 'direct' formulation. For most problems with mixed boundary conditions, the resulting integral equation for the unknowns is not a Fredholm equation of second kind and the condition number of the resulting coefficient matrix A in Equation (8) grows as the number of discretization points increases. When Krylov subspace-based iterative methods are applied to such a system, the number of required iterations increases very quickly as problem size gets larger.

For a general ill-conditioned linear system, many 'preconditioning' techniques have been developed to improve the system's condition number and reduce the iteration number when solved by Krylov subspace-based methods [51]. In this paper, we consider a left preconditioner denoted by M, and the preconditioned form of Equation (8) becomes

$$(MA)X = (MB) \tag{9}$$

Generally speaking, M should be chosen so that (a) the matrix MA is close to identity matrix and (b) the matrix vector product $(MA)v$ can be easily carried out for any given vector v. For our fast multipole accelerated BIE method, as the matrix A is not explicitly formed (so the storage can be reduced to asymptotically optimal $O(N)$), we adapt a block diagonal matrix preconditioner M based on the FMM tree structure as in [52–54, 31]. It is constructed as follows and interested readers are referred to [54] for more detailed discussions.

Suppose that the whole boundary is discretized into N boundary elements. In our algorithm, an octal-tree structure is constructed such that each tree leaf has at most a prescribed number of elements (see [55, 56] for a discussion of the adaptive tree structures in the FMM methods). Using this tree structure, the matrix A is first approximated by a block diagonal matrix $\overline{A}$ in which the number n of the block sub-matrices equals to the number of tree leaves and entries in the kth block sub-matrix $A^{(k)}$ are formed by integrating m_k elements contained in the kth tree leaf using conventional BIE (or boundary element) methods. This idea coincides with the fast multipole BIE method in which elements in one leaf are calculated directly. The form of $\overline{A}$ is

$$\overline{A} = \begin{pmatrix} A^{(1)}_{m_1\times m_1} & & & & & 0 \\ & A^{(2)}_{m_2\times m_2} & & & & \\ & & \ddots & & & \\ & & & A^{(k)}_{m_k\times m_k} & & \\ & & & & \ddots & \\ 0 & & & & & A^{(n)}_{m_n\times m_n} \end{pmatrix} \tag{10}$$

After $\overline{A}$ is constructed, we define $M = (\overline{A})^{-1}$ which is block diagonal.

We want to mention that instead of sparse matrix-based preconditioning techniques, methods using convolution-type integral operators as preconditioners are also being actively studied. In [57–60], it was shown that hypersingular integral operators can be used as preconditioners for Fredholm first kind integral equations, and the resulting equations become Fredholm integral equations of second kind. Currently, instead of direct formulation (3), we are working on deriving better integral equation formulations and possible integral operator-based preconditioners. Results along these directions will be reported in the future.

4. NEW VERSION FAST MULTIPOLE METHOD FOR 3D STOKES FLOWS

In traditional boundary element methods (BEM), the dense matrix from numerical discretization is usually calculated and stored explicitly. Solving such a system with N unknowns requires $O(N^2)$ memory and $O(N^3)$ operations using Gauss elimination. An alternative is to use Krylov subspace-based iterative schemes such as the GMRES or BiCGSTAB [61–65]. As the iteration number is bounded by a constant for a well conditioned linear system (e.g. system (9) with properly chosen preconditioners), the amount of work is dominated by the matrix vector product which requires $O(N^2)$ operations using a direct method. In the last 20 years, thanks to the introduction of fast algorithms such as the FMM [12], the amount of work for convolution-type dense matrix vector products can be reduced to $O(N)$ and explicit storage of the matrix is no longer necessary. The resulting integral equation schemes are asymptotically optimal in memory and efficiency.

In this paper, we focus on the new version of FMM introduced by Greengard and Rokhlin in 1997 [13]. This technique was later generalized to the Helmholtz and linearized Poisson–Boltzmann equations in [66, 67]. Compared with the original FMM, by introducing a diagonal translation operator based on exponential expansions, the prefactor in the new version $O(N)$ FMM was greatly reduced, especially in three dimensions. In this section, we give a brief introduction of basic ideas and formulations of the new version of FMM for 3D steady Stokes equations. Interested readers are referred to [13, 68] where the new version of FMM was discussed under the circumstance of the Laplace equation. In the following discussions, to avoid notation clutter, we simply denote $(\mathbf{x}, \mathbf{y})$ as (x, y), and the vector pointing from x to y as $\mathbf{xy}$.

4.1. Multipole expansions

The boundary integrals containing fundamental solutions $U^*_{ij}(x, y)$, $\mathrm{T}^*_{ij}(x, y)$ in Equation (2) represent the contribution of field (source) point y to target point x. Consider a point y in a cubic box $\mathbf{A}$ with centre y_0 as shown in Figure 1, assume that the following threshold is satisfied:

$$|\mathbf{y}_0\mathbf{y}| \leqslant \tfrac{1}{2}|\mathbf{y}_0\mathbf{x}| \quad \forall y \in \mathbf{A} \tag{11}$$

The fundamental solutions $U^*_{ij}(x, y)$ and $\mathrm{T}^*_{ij}(x, y)$ can be expanded succinctly with respect to the centre y_0 using the separation of variables technique as

$$\sum_n \sum_m f_{nm}(x, y_0) g_{nm}(y, y_0) \tag{12}$$

More specifically for $U^*_{ij}(x, y)$

$$U^*_{ij}(x, y) = \frac{1}{8\pi\mu} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left(\overline{F^{1S}_{ij,n,m}}(\mathbf{y}_0\mathbf{x}) R_{n,m}(\mathbf{y}_0\mathbf{y}) + \overline{F^{2S}_{i,n,m}}(\mathbf{y}_0\mathbf{x}) (\mathbf{y}_0\mathbf{y})_j R_{n,m}(\mathbf{y}_0\mathbf{y}) \right) \tag{13}$$

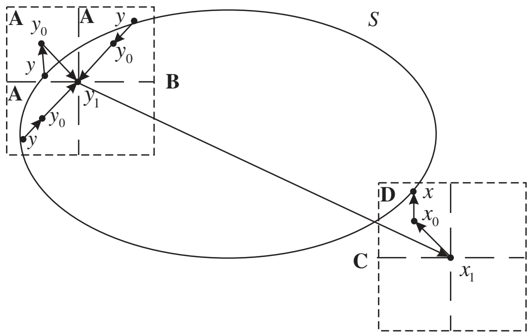

Figure 1. Four steps of FMM.

where

$$
\begin{aligned}
F^{1S}_{ij,n,m}(\mathbf{Ox}) &= \delta_{ij} S_{n,m}(\mathbf{Ox}) - (\mathbf{Ox})_j \frac{\partial}{\partial x_i} S_{n,m}(\mathbf{Ox}) \\
F^{2S}_{i,n,m}(\mathbf{Ox}) &= \frac{\partial}{\partial x_i} S_{n,m}(\mathbf{Ox})
\end{aligned}
\tag{14}
$$

$R_{n,m}$ and $S_{n,m}$ are the solid spherical harmonic functions defined as

$$
\begin{aligned}
R_{n,m}(\mathbf{Ox}) &= \frac{1}{(n+m)!} P^m_n(\cos\theta) \mathrm{e}^{\mathrm{i}m\phi} r^n \\
S_{n,m}(\mathbf{Ox}) &= (n-m)! P^m_n(\cos\theta) \mathrm{e}^{\mathrm{i}m\phi} \frac{1}{r^{n+1}}
\end{aligned}
$$

$\{r, \theta, \phi\}$ are the spherical co-ordinates of vector $\mathbf{Ox}$, P^m_n is the associated Legendre function and $\overline{(\ )}$ stands for the complex conjugate (see also [69, 30]). Substituting Equation (12) or (13) into the boundary integral $\int_S U^*_{ij}(x, y) t_j(y)\, \mathrm{d}S(y)$, one obtains

$$
\begin{aligned}
\int_S U^*_{ij}(x, y) t_j(y)\, \mathrm{d}S(y) &= \int_S \left[\sum_n \sum_m f_{nm}(x, y_0) g_{nm}(y, y_0) \right] t_j(y)\, \mathrm{d}S(y) \\
&= \frac{1}{8\pi\mu} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left(\overline{F^{1S}_{ij,n,m}}(\mathbf{y_0x}) c^{U1}_{j,n,m}(y_0) + \overline{F^{2S}_{i,n,m}}(\mathbf{y_0x}) c^{U2}_{n,m}(y_0) \right)
\end{aligned}
\tag{15}
$$

which is referred to as the multipole expansion of $\int_S U^*_{ij}(x, y) t_j(y)\, \mathrm{d}S(y)$ centred at y_0, and the multipole moments $c^{U1}_{j,n,m}(y_0)$ and $c^{U2}_{n,m}(y_0)$ are given by

$$
\begin{aligned}
c^{U1}_{j,n,m}(y_0) &= \int_S R_{n,m}(\mathbf{y_0y}) t_j(y)\, \mathrm{d}S(y) \\
c^{U2}_{n,m}(y_0) &= \int_S (\mathbf{y_0y})_j R_{n,m}(\mathbf{y_0y}) t_j(y)\, \mathrm{d}S(y)
\end{aligned}
\tag{16}
$$

Similarly for $T^*_{ij}(x, y)$, notice the relation between T^*_{ij} and U^*_{ij}

$$T^*_{ij}(x, y) = \mu \left\{ \frac{\partial U^*_{ij}(x, y)}{\partial y_k} n_k + \frac{\partial U^*_{ik}(x, y)}{\partial y_j} n_k + \frac{1}{2} \frac{\partial U^*_{kk}(x, y)}{\partial y_i} n_j \right\} \tag{17}$$

$T^*_{ij}(x, y)$ can then be expanded using Equation (13), and straightforward algebraic manipulations for $\int_S T^*_{ij}(x, y) u_j(y)\, \mathrm{d}S(y)$ give

$$\int_S T^*_{ij}(x, y) u_j(y)\, \mathrm{d}S(y) = \frac{1}{8\pi\mu} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left(\overline{F^{1S}_{ij,n,m}}(\mathbf{y}_0\mathbf{x}) c^{T1}_{j,n,m}(y_0) + \overline{F^{2S}_{i,n,m}}(\mathbf{y}_0\mathbf{x}) c^{T2}_{n,m}(y_0) \right)$$

where

$$c^{T1}_{j,n,m}(y_0) = \mu \int_S \left\{ \frac{\partial R_{n,m}(\mathbf{y}_0\mathbf{y})}{\partial y_k} n_k u_j(y) + \frac{\partial R_{n,m}(\mathbf{y}_0\mathbf{y})}{\partial y_k} n_j u_k(y) \right\} \mathrm{d}S(y)$$

$$c^{T2}_{n,m}(y_0) = \mu \int_S \left\{ (\mathbf{y}_0\mathbf{y})_k \frac{\partial R_{n,m}(\mathbf{y}_0\mathbf{y})}{\partial y_j} n_j u_k(y) + (\mathbf{y}_0\mathbf{y})_j \frac{\partial R_{n,m}(\mathbf{y}_0\mathbf{y})}{\partial y_k} n_j u_k(y) \right\} \mathrm{d}S(y)$$

Once the coefficients $c^{U1}_{j,n,m}(y_0)$, $c^{U2}_{n,m}(y_0)$, $c^{T1}_{j,n,m}(y_0)$, $c^{T2}_{n,m}(y_0)$ are calculated, the integrals $\int_S U^*_{ij}(x, y) t_j(y)\, \mathrm{d}S(y)$ and $\int_S T^*_{ij}(x, y) u_j(y)\, \mathrm{d}S(y)$ at different target point x can be evaluated using the multipole expansions. Notice that except for the upper index (T or U) for the coefficients, the multipole expansion formulas for both integrals are identical. This implies that other expansion and translation operator formulas are also identical for both integrals if they are derived using the multipole expansions. Hence to simplify the discussion, in the following, we omit the details for the second integral $\int_S T^*_{ij}(x, y) u_j(y)\, \mathrm{d}S(y)$, and neglect the upper index T or U for different expansion coefficients (e.g. using $c^1_{j,n,m}(y_0)$, $c^2_{n,m}(y_0)$ instead of $c^{U1}_{j,n,m}(y_0)$, $c^{U2}_{n,m}(y_0)$, so the formulas are also applicable to $c^{T1}_{j,n,m}(y_0)$, $c^{T2}_{n,m}(y_0)$). Further, as the multipole expansions are formulated in the same form as those in [29] for 3D elastostatic problems, the translation operators to be discussed below also can be found in [29] and references therein.

4.2. Multipole to multipole translation (M2M)

Let **B** denote the parent cube of **A** as shown in Figure 1. Assume y_1 is **B**'s centre and the following threshold is satisfied:

$$|\mathbf{y}_1\mathbf{y}| \leqslant \tfrac{1}{2} |\mathbf{y}_1\mathbf{x}| \quad \forall y \in \mathbf{B} \tag{18}$$

Using the multipole expansion of $U^*_{ij}(x, y)$ with respect to y around y_1, one can obtain an expansion similar to Equation (15) but centred at y_1 as

$$\int_S U^*_{ij}(x, y) t_j(y)\, \mathrm{d}S(y) = \frac{1}{8\pi\mu} \sum_{n=0}^{\infty} \sum_{m=-n}^{n} \left(\overline{F^{1S}_{ij,n,m}}(\mathbf{y}_1\mathbf{x}) c^1_{j,n,m}(y_1) + \overline{F^{2S}_{i,n,m}}(\mathbf{y}_1\mathbf{x}) c^2_{n,m}(y_1) \right) \tag{19}$$

Notice that instead of evaluating the integrals in Equation (16) directly, the new multipole moments $c^1_{j,n,m}(y_1)$, $c^2_{n,m}(y_1)$ can be obtained by introducing a linear mapping acting on the old moments

$c^1_{j,n,m}(y_0)$, $c^2_{n,m}(y_0)$ as given by (see also [29])

$$c^1_{j,n,m}(y_1) = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\mathbf{y}_1\mathbf{y}_0)c^1_{j,n-n',m-m'}(y_0)$$
$$c^2_{n,m}(y_1) = \sum_{n'=0}^{n}\sum_{m'=-n'}^{n'} R_{n',m'}(\mathbf{y}_1\mathbf{y}_0)\left(c^2_{n-n',m-m'}(y_0) - (\mathbf{y}_0\mathbf{y}_1)_j c^1_{j,n-n',m-m'}(y_0)\right)$$

This linear mapping can be precomputed and is referred to as the 'multipole to multipole (M2M) translation operator' in the FMM literature.

4.3. *Multipole to local translation* (*M*2*L*)

Let **C** denote a cube of the same size as **B** as shown in Figure 1. Assume x_1 is **C**'s centre and the following threshold is satisfied:

$$|\mathbf{x}_1\mathbf{x}| \leqslant \tfrac{1}{2}|\mathbf{x}_1\mathbf{y}_1| \quad \forall x \in \mathbf{C} \tag{20}$$

As the functions $f_{nm}(x, y_1)$ (representing $\overline{F^{1S}_{ij,n,m}}(\mathbf{y}_1\mathbf{x})$ and $\overline{F^{2S}_{i,n,m}}(\mathbf{y}_1\mathbf{x})$) in Equation (12) are smooth for x close to x_1, they can be Taylor expanded succinctly as

$$f_{nm}(x, y_1) = \sum_{n'}\sum_{m'} p_{n'm'}(x, x_1)q^{nm}_{n'm'}(y_1, x_1) \tag{21}$$

Substituting Equation (21) into the multipole expansion in Equation (15) or (19), one can obtain a 'local expansion' centred at x_1 as

$$\int_S U^*_{ij}(x, y)t_j(y)\,\mathrm{d}S(y)$$
$$= \frac{1}{8\pi\mu}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}\left(\overline{P^{1R}_{ij,n',m'}}(\mathbf{x}_1\mathbf{x})d^1_{j,n',m'}(x_1) + \overline{P^{2R}_{i,n',m'}}(\mathbf{x}_1\mathbf{x})d^2_{n',m'}(x_1)\right) \tag{22}$$

where $P^{1R}_{ij,n',m'}(\mathbf{x}_1\mathbf{x})$ and $P^{2R}_{i,n',m'}(\mathbf{x}_1\mathbf{x})$ are detailed expressions for $p_{n'm'}(x, x_1)$ given by

$$P^{1R}_{ij,n',m'}(\mathbf{x}_1\mathbf{x}) = \delta_{ij}R_{n',m'}(\mathbf{x}_1\mathbf{x}) - (\mathbf{x}_1\mathbf{x})_j\frac{\partial}{\partial x_i}R_{n',m'}(\mathbf{x}_1\mathbf{x})$$
$$P^{2R}_{i,n',m'}(\mathbf{x}_1\mathbf{x}) = \frac{\partial}{\partial x_i}R_{n',m'}(\mathbf{x}_1\mathbf{x})$$

The local moments $d^1_{j,n',m'}(x_1)$ and $d^2_{n',m'}(x_1)$ can be derived by a linear mapping acting on the multipole moments $c^1_{j,n,m}(y_1)$, $c^2_{n,m}(y_1)$ (see [29])

$$\begin{aligned} d^1_{j,n',m'}(x_1) &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^{n'}\overline{S_{n+n',m+m'}}(\mathbf{y}_1\mathbf{x}_1)c^1_{j,n,m}(y_1) \\ d^2_{n',m'}(x_1) &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}(-1)^{n'}\overline{S_{n+n',m+m'}}(\mathbf{y}_1\mathbf{x}_1)\left(c^2_{n,m}(y_1) - (\mathbf{y}_1\mathbf{x}_1)_j c^1_{j,n,m}(y_1)\right) \end{aligned} \tag{23}$$

This operator is referred to as the 'multipole to local (M2L) translation operator'. Notice that in the original FMM, direct calculation of $d^1_{j,n',m'}(x_1)$, $d^2_{n',m'}(x_1)$ using $c^1_{j,n,m}(y_1)$, $c^2_{n,m}(y_1)$ requires $O(P^4)$ work where terms with $n>P$ or $m>P$ in Equation (23) are truncated. Furthermore, in three dimensions, as many as 189 such translations may be required for each multipole expansion. The new version of FMM, on the other hand, reduces the number of operations from $189P^4$ to approximately $40P^2+2P^3$. This technique is discussed in Section 4.5.

4.4. Local to local translation (*L2L*)

Let **D** denote a child cube of **C** as shown in Figure 1, and x_0 be **D**'s centre. Using expansion of $f_{nm}(x, y_1)$ with respect to x around x_0, one can obtain an expansion similar to Equation (22)

$$\int_S U^*_{ij}(x,y)t_j(y)\,\mathrm{d}S(y) = \frac{1}{8\pi\mu}\sum_{n'=0}^{\infty}\sum_{m'=-n'}^{n'}\left(\overline{P^{1R}_{ij,n',m'}}(\mathbf{x}_0\mathbf{x})d^1_{j,n',m'}(x_0)+\overline{P^{2R}_{i,n',m'}}(\mathbf{x}_0\mathbf{x})d^2_{n',m'}(x_0)\right) \tag{24}$$

where new local moments can be calculated, instead of using the M2L operator, by a linear mapping acting on old ones as given by (see [29])

$$d^1_{j,n,m}(x_0) = \sum_{n'=n}^{\infty}\sum_{m'=-n'}^{n'} R_{n'-n,m'-m}(\mathbf{x}_1\mathbf{x}_0)d^1_{j,n',m'}(x_1)$$

$$d^2_{n,m}(x_0) = \sum_{n'=n}^{\infty}\sum_{m'=-n'}^{n'} R_{n'-n,m'-m}(\mathbf{x}_1\mathbf{x}_0)\left(d^2_{n',m'}(x_1)-(\mathbf{x}_1\mathbf{x}_0)_j d^1_{j,n',m'}(x_1)\right)$$

This mapping from local moments $d^1_{j,n',m'}(x_1)$, $d^2_{n',m'}(x_1)$ to $d^1_{j,n,m}(x_0)$, $d^2_{n,m}(x_0)$ can be precomputed and is referred to as the 'local to local (L2L) translation operator'.

4.5. Exponential expansion

In the new version of FMM, it is noticed that instead of the multipole and local expansions, $\int_S U^*_{ij}(x,y)t_j(y)\,\mathrm{d}S(y)$ can be approximated by an 'exponential' expansion as

$$\int_S U^*_{ij}(x,y)t_j(y)\,\mathrm{d}S(y) = \frac{1}{8\pi\mu}\sum_{n=1}^{s(\varepsilon)}\sum_{m=1}^{M_n}\left[X^1_{j,n,m}(y_0)\left(\delta_{ij}-(\mathbf{y}_0\mathbf{x})_j\frac{\partial}{\partial x_i}\right)+X^2_{n,m}(y_0)\frac{\partial}{\partial x_i}\right]$$
$$\times \mathrm{e}^{-(\lambda_n/lt)(\mathbf{y}_0\mathbf{x})_3+\mathrm{i}(\lambda_n/l)[(\mathbf{y}_0\mathbf{x})_1\cos\alpha_{mn}+(\mathbf{y}_0\mathbf{x})_2\sin\alpha_{mn}]}$$

where $s(\varepsilon)$, M_n, λ_n, α_{mn} are some constants determined by prescribed accuracy requirement (calculating these constants is non-trivial and interested readers are referred to [15] for details), l is the edge length of the cube related to an octal-tree node, and $X^1_{j,n,m}(y_0)$ and $X^2_{n,m}(y_0)$ are the exponential moments.

Using exponential expansions, the dense translation operator from multipole moments to local ones (M2L) in Equation (23) can be replaced by three steps:

(1) The multipole to exponential shift (M2E) from $c^1_{j,n,m}(y_0)$, $c^2_{n,m}(y_0)$ to $X^1_{j,n,m}(y_0)$, $X^2_{n,m}(y_0)$, in order to derive the exponential moments from multipole ones for the same cube. Note that an exponential expansion carries the same information (up to a prescribed precision depending on the number of terms used) for each box in which the original multipole expansion is to be shifted. This M2E translation is given by

$$X^1_{j,n,m}(y_0) = \frac{w_n}{M_n l} \sum_{m'=-\infty}^{\infty} (-\mathrm{i})^{m'} \mathrm{e}^{-\mathrm{i}m'\alpha_{nm}} \sum_{n'=|m'|}^{\infty} \left(\frac{\lambda_n}{l}\right)^{n'} c^1_{j,n',m'}(y_0)$$

$$X^2_{n,m}(y_0) = \frac{w_n}{M_n l} \sum_{m'=-\infty}^{\infty} (-\mathrm{i})^{m'} \mathrm{e}^{-\mathrm{i}m'\alpha_{nm}} \sum_{n'=|m'|}^{\infty} \left(\frac{\lambda_n}{l}\right)^{n'} c^2_{n',m'}(y_0)$$

where the constant w_n can be found in [13] (see [70] for its calculation).

(2) The exponential to exponential shift (E2E) from $X^1_{j,n,m}(y_0)$, $X^2_{n,m}(y_0)$ to $X^1_{j,n,m}(x_0)$, $X^2_{n,m}$ (x_0), in order to transfer exponential moments from one cube containing source points to another cube containing target points. The latter cube is referred to as a member of the 'interaction list' [12]. E2E is referred to as a diagonal translation operator because it only requires one multiplication for each moment, or in matrix form, the translation matrix is diagonal, as given by

$$X^1_{j,n,m}(x_0) = X^1_{j,n,m}(y_0)\mathrm{e}^{-\left(\frac{\lambda_n}{l}\right)(\mathbf{y}_0\mathbf{x}_0)_3+\mathrm{i}\left(\frac{\lambda_n}{l}\right)[(\mathbf{y}_0\mathbf{x}_0)_1\cos\alpha_{nm}+(\mathbf{y}_0\mathbf{x}_0)_2\sin\alpha_{nm}]}$$

$$X^2_{n,m}(x_0) = [X^2_{n,m}(y_0) - (\mathbf{y}_0\mathbf{x}_0)_k X^1_{k,n,m}(y_0)]\mathrm{e}^{-\left(\frac{\lambda_n}{l}\right)(\mathbf{y}_0\mathbf{x}_0)_3+\mathrm{i}\left(\frac{\lambda_n}{l}\right)[(\mathbf{y}_0\mathbf{x}_0)_1\cos\alpha_{nm}+(\mathbf{y}_0\mathbf{x}_0)_2\sin\alpha_{nm}]}$$

(3) The exponential to local shift (E2L) from $X^1_{j,n,m}(x_0)$, $X^2_{n,m}(x_0)$ to $d^1_{j,n,m}(x_0)$, $d^2_{n,m}(x_0)$, in order to obtain local moments from exponential ones in the same cube. The local expansions are derived by Taylor expanding the exponential expansions, and the coefficients are given by

$$d^1_{j,n,m}(x_0) = \sum_{n'=1}^{s(\varepsilon)} \sum_{m'=1}^{M_{n'}} X^1_{j,n',m'}(x_0)(-\mathrm{i})^m \left(\frac{-\lambda_{n'}}{l}\right)^n \mathrm{e}^{-\mathrm{i}m\alpha_{n'm'}}$$

$$d^2_{n,m}(x_0) = \sum_{n'=1}^{s(\varepsilon)} \sum_{m'=1}^{M_{n'}} X^2_{n',m'}(x_0)(-\mathrm{i})^m \left(\frac{-\lambda_{n'}}{l}\right)^n \mathrm{e}^{-\mathrm{i}m\alpha_{n'm'}}$$

Figure 2 shows these three steps which diagonalize the original M2L operator. Since M2L operator in FMM dominates in computer resources usage for 3D problems, its diagonalization makes the new version of FMM more efficient.

It should be mentioned that kernel functions for Stokes equations can be expanded in different forms. Interested readers are referred to [71] for a summary of existing forms. As will be shown in Figure 11, by introducing diagonal translation operator, a break-even point of approximately 2000 elements for 6 digits precision is numerically observed. Currently, a comparison of different forms is being performed and results will be reported in the future.

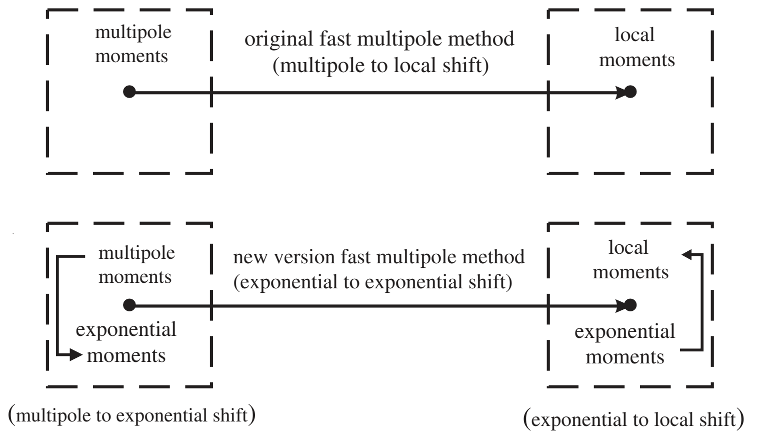
multipole moments
original fast multipole method
(multipole to local shift)
local moments
multipole moments
new version fast multipole method
(exponential to exponential shift)
local moments
exponential moments
exponential moments
(multipole to exponential shift)
(exponential to local shift)

Figure 2. Procedures of the new version of FMM.

4.6. Major steps in FMM

As a summary, several major steps in the FMM are described below.

4.6.1. Upward stage. Assume an octal-tree is constructed, the upward stage first calculates the multipole moments of a leaf by integrating the boundary elements it contains. Second, multipole moments of each tree node are transferred to those of its parent by the multipole to multipole shift (M2M).

4.6.2. Downward stage. After multipole moments are obtained for all nodes in the octal-tree structure, a downward stage is performed in order to obtain local moments for each node. The concepts of 'neighbours' and 'interaction list' are introduced. Two tree nodes are called 'neighbours' when they share at least one vertex, and the 'interaction list' of a node B contains a group of tree nodes which are not neighbours of B but their parent nodes are neighbours of B's parent. In the downward stage, information is first transferred from local moments of B's parent (representing contributions from outside of B's interaction list) by the local to local (L2L) translation operator. Then contributions from cubes in the interaction list are collected. In the original FMM, multipole moments of interaction list cubes are transferred to B using the multipole to local (M2L) translation operator. In the new version, M2L operation is decomposed into three operations: M2E, E2E and E2L, which significantly improves the efficiency of the algorithm.

4.6.3. Direct local interactions. Notice that by evaluating the local expansion in Equation (24) for each point x in a leaf node, one only obtains contributions from far-field boundary integrals. For the remaining a few 'neighbouring' boundary integrals, a numerical quadrature can be applied to evaluate the contributions directly, which is referred to as the direct local interactions.

4.6.4. Iterative solutions. In order to solve Equation (8) where matrix A is in general non-symmetric, we start from an initial guess X_0, and then an iterative method (such as GMRES, BiCGStab, or TFQMR) can be applied to the preconditioned system (9) to find the 'optimal solution' in the Krylov subspace. In each iteration, the matrix vector product is performed using the new version of FMM. The procedure continues until the solution converges within a given tolerance.

For existing Krylov subspace methods, interested readers are referred to [61, 63, 65] for a summary. Notice that GMRES gives better convergence properties in theory, however, the required storage increases linearly with the number of iterations k, and the number of multiplications scales like $O(k^2)$. On the other hand, although the convergence properties are less well understood compared with GMRES, for BiCGStab and TFQMR methods, the storage is independent of k and the number of multiplications only increases linearly. Currently a detailed study of different Krylov subspace methods is being pursued.

5. PARALLELIZATION

Compared with conventional matrix vector multiplication methods, the FMMs are more difficult to parallelize efficiently due to two problems: (a) the elements are non-uniformly distributed and hence the tree is imbalanced (adaptive) and difficult to decompose effectively; and (b) the tree is a connected data structure which is difficult to traverse compared with simple arrays. In the following, we briefly discuss several approaches we adopted.

5.1. Tree decomposition by cubes

Instead of the row, column and cyclic block partition schemes that are commonly used in conventional matrix-based algorithms, the 'tree decomposition by cubes' technique is used in our FMM solver as described in the following procedure.

Procedure 1
Tree decomposition by cubes.

Step 1: The cubes (nodes in the adaptive tree structure) and their associated elements are first sorted using the 'space filling curves' algorithm [72] which has been studied previously by Singer and Wang *et al.* [47, 54] (however in their work, elements were distributed among tasks, which is not readily applicable to our leaf-based block-preconditioned method). The index k of a cube satisfies the following property: if $k_1 < k_2 < k_3$, and k_1, k_3 share the same parent cube (node), then so does k_2.

Step 2: The cubes are weighed by their predicted run-time costs in the algorithm, as calculated by the following procedure: consider an element e in a leaf cube (node) located at level $L(e)$, the predicted run-time cost for e is then

$$T^e = \frac{(T_{L(e)-1}/n^c_{L(e)-1} + n^{\mathrm{E2E}}_{L(e)} \cdot T^{\mathrm{E2E}} + T^{\mathrm{E2L}} + T^{\mathrm{L2L}} + n^{\mathrm{N2R}}_{L(e)} \cdot T^{\mathrm{N2R}})}{n^e_{L(e)}} \tag{25}$$

where $T_l (l = 0, \ldots, L(e) - 1)$ is the predicted run-time cost of the (grand) parent cube (containing e) at level l computed hierarchically by traversing the tree structure using

$$T_0 = 0$$

$$T_l = \frac{T_{l-1}}{n^c_{l-1}} + n^{\mathrm{E2E}}_l \cdot T^{\mathrm{E2E}} + T^{\mathrm{E2L}} + T^{\mathrm{L2L}} \; (l = 1, \ldots, L(e) - 1)$$

T^{E2E}, T^{E2L} and T^{L2L} are costs for the E2E, E2L and L2L FMM operators which depend on expansion orders of the multipole, local, and exponential expansions, respectively. T^{N2R} represents direct one pair element–element computation cost which can be measured before the FMM steps. n_{l-1}^{c} is the number of non-empty children cubes at level l for the parent cube (containing e) located at level $l-1$, n_l^{E2E} is the number of E2E operations for the cube at level l, $n_{L(e)}^{\mathrm{N2R}}$ is the number of direct computations for a total of $n_{L(e)}^{e}$ elements (including e) located in the leaf cube at level $L(e)$. Note that if two elements are in the same leaf cube, then they have the same predicted run-time costs as the numbers n_{l-1}^{c}, n_l^{E2E}, $n_{L(e)}^{\mathrm{N2R}}$ and $n_{L(e)}^{e}$ are identical for both elements. The predicted run-time cost for a given cube k at level l can be obtained by summing up the costs of all the elements in k as

$$T_l^k = \sum_{e \in k} T^e \tag{26}$$

Step 3: To balance the costs in the parallel algorithm, the cubes at level *dec_level* are distributed among tasks where *dec_level* is a prescribed level which cannot be lower than the finest level of leaves in the tree structure. Suppose there are M cubes at this level, and their costs are $T_{dec\_level}^1, T_{dec\_level}^2, \ldots, T_{dec\_level}^M$. Let p be the number of tasks (or processors), we can determine a series of indices $k_j\,(j=0,1,\ldots,p)$ by setting $k_0=0$ and $k_p=M$, and requiring

$$\begin{aligned} \sum_{i=k_{j-1}+1}^{k_j} T_{dec\_level}^i &\leqslant \frac{\tilde{T}_K}{p} \\ \sum_{i=k_{j-1}+1}^{k_j+1} T_{dec\_level}^i &> \frac{\tilde{T}_K}{p} \end{aligned} \tag{27}$$

The cubes from $k_{j-1}+1$ to k_j are then assigned to task j. In the formula, $\tilde{T}_K$ represents the sum of cube costs at *dec_level* defined by

$$\tilde{T}_K = \sum_{k=1}^{M} T_{dec\_level}^k \tag{28}$$

The optimal choice of the parameter *dec_level* depends on the adaptive tree structure as well as computer architecture. In Figure 3(a), we present a global tree structure based on the model of a square plate with an eccentered hole. The tree is decomposed into four tasks using *dec_level* $=1$ (b) and 2 (c), respectively. It can be seen that the costs of the local tree in (c) is closer to a quarter of the total costs compared with the case in (b). In general, lower *dec_level* means more balanced decomposition but more complicated communication pattern. In our current implementation, highest possible *dec_level* is chosen as long as the decomposition is reasonably balanced. Main advantages of the tree decomposition by cubes scheme are that (a) the children of any cube at level below *dec_level* are in the same task, so the communication pattern in tree traversing becomes simpler than other schemes; and (b) the block preconditioning matrices can be constructed as a straightforward extension of the serial code and applied to a candidate vector without any communication in each iteration. In general, the parameter *dec_level* helps providing balanced communication overhead and work load.

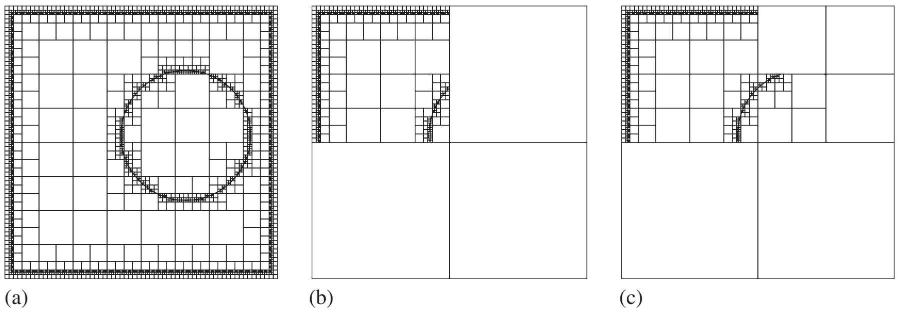

Figure 3. Task decomposition by cubes at different tree levels: (a) tree structure; (b) decomposition at $dec\_level = 1$; and (c) decomposition at $dec\_level = 2$.

5.2. Parallelization of tree traversing

For data-locality, the tree structure in each task is created based on the elements it contains. As each task is unaware of other tasks' tree structures, the tree traversing process becomes different from its serial counterpart. In the following, we focus on the downward stage in the FMM, which is the most time-consuming and complicated step in parallel formulations due to interactions with the 'interaction list' and 'neighbouring' cubes that may not be in the same task or even may not exist. Based on the decomposition scheme mentioned above, two necessary procedures are performed to establish these required communication lists for cubes in other tasks as described below.

Procedure 2
Set sending/receiving 'interaction list'.

Step 1: For every task, traverse its tree structure recursively, and find all 'possible' 'interaction list' cubes in other tasks from which it needs to receive information, as shown in Figure 4(a) for a 3-level tree decomposed at $dec\_level = 1$.

Step 2: For every task, communicate with other tasks so that they know what they need to send. All 'possible' sending cubes are then checked for existence. For example in Figure 4(b), shaded cubes are those possible sending cubes. After checking, the cubes with dotted lines are those that do not exist.

Step 3: The existing cubes are the real sending cubes and are recorded in receiver task's real receiving interaction list, as shown in Figure 4(d). For non-existing cubes, their parent cubes are located instead and are saved to the possible neighbouring list for further processing in Procedure 3 '*Set sending/receiving neighbouring list*', as they can only be neighbours (instead of interaction list members) of cubes in receiver's task.

Procedure 3
Set sending/receiving 'neighbouring list'.

Step 1: For every task, traverse its tree structure recursively, and find all 'possible' neighbour cubes in other tasks from which it needs to receive information, as shown in Figure 5(a).

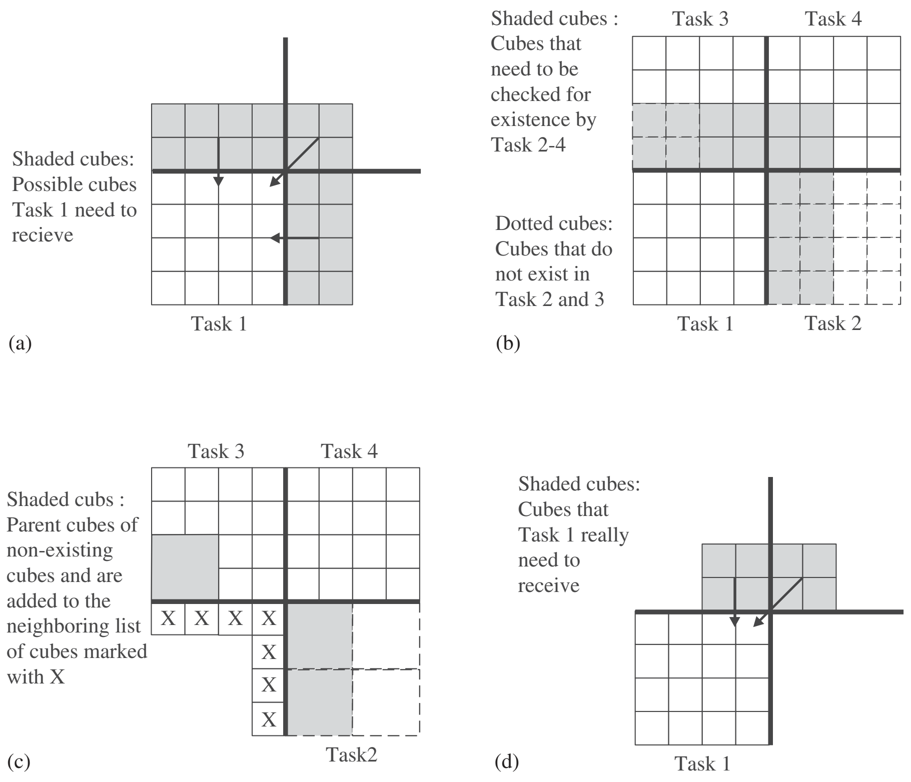

Figure 4. Set interaction list in parallel tree traversing.

Note that these possible cubes also include the parent cubes of non-existing cubes discussed in step 3 of Procedure 2.

Step 2: For every task, communicate with other tasks so that they know what they need to send. All ‘possible’ sending cubes are then checked for existence. For example in Figure 5(b), shaded cubes are those possible sending cubes. After checking, the cubes with dotted lines are those that do not exist. This step is similar to step 2 in Procedure 2.

Step 3: The existing cubes are the real sending neighbouring cubes and are recorded in receiver task’s real receiving neighbouring list, as shown in Figure 5(d). For non-existing cubes, their parent cubes are checked for existence recursively. For example in Figure 5(c), the shaded cubes in Task 2 and 3 are parent cubes of non-existing cubes.

For the new version of FMM, the sending/receiving interaction list is further divided into 4 (2-D) or 6 (3-D) sub-lists representing different directions in the E2E translation.

After the cubes’ relations between different tasks have been set up, exponential moments of the interaction list cubes and element information in the neighbouring list can be packed and sent or received using message passing interfaces (MPI). Other operations in the new version of FMM (like vector operations) are conventionally parallelized.

Finally in this section, we want to mention that the overhead due to procedures 2 and 3 is negligible since such ‘possible’ cubes are only a small portion of the total cubes in each

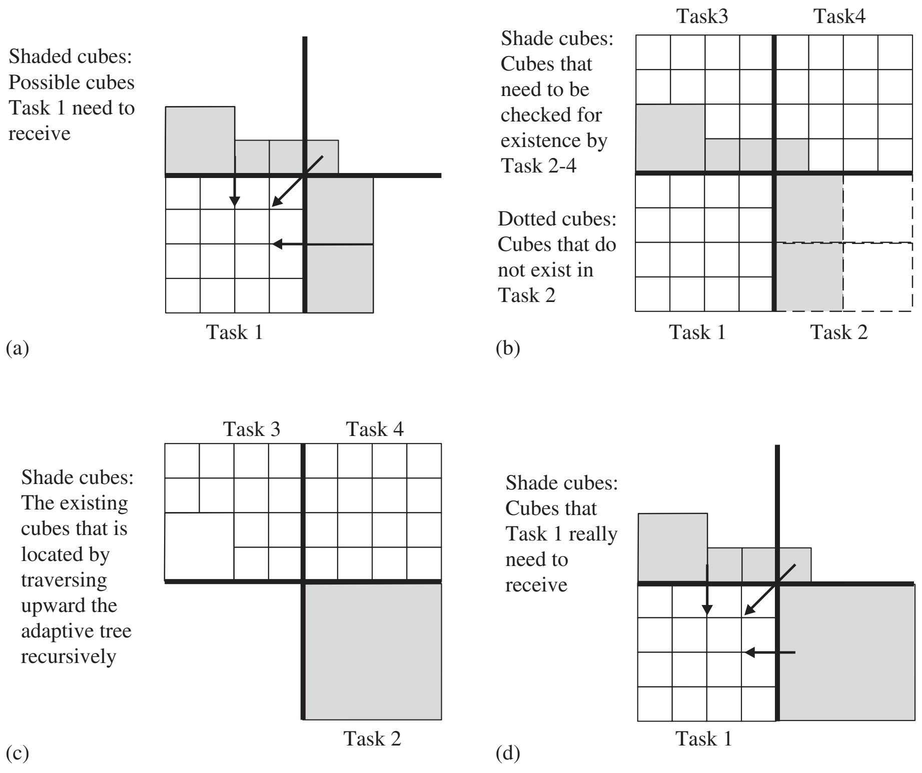
Shaded cubes: Possible cubes Task 1 need to receive
Task 1
(a)
Task3
Task4
Shade cubes: Cubes that need to be checked for existence by Task 2-4
Dotted cubes: Cubes that do not exist in Task 2
Task 1
Task 2
(b)
Task 3
Task 4
Shade cubes: The existing cubes that is located by traversing upward the adaptive tree recursively
Task 2
(c)
Shade cubes: Cubes that Task 1 really need to receive
Task 1
(d)

Figure 5. Set neighbouring list in parallel tree traversing.

task. Also, the resulting lists remain the same for each iteration in the Krylov subspace-based methods.

6. NUMERICAL RESULTS

In this section, we give several numerical results to show the efficiency and accuracy of our parallelized fast Stokes solver. In the simulation, a Pentium 4 XEON 2.8 GHz 32-processor cluster running Linux system is used. The fast multipole BIE code is written in C/C + +, and LAM-MPI library is used for communication. Compared with the original FMM and conventional BIE methods (where Lapack (serial) and ScaLAPACK (parallel) subroutines are used to solve the linear equation using Gauss elimination), the new version of FMM accelerated solver is more efficient and asymptotically optimal.

In the first set of simulations, we consider a translating sphere as shown in Figure 6 and set the physical parameters to $\mu = 1.0 \times 10^{-3}$ Pa s, $R = 1.0$ m and $U_x = 1.0$ m/s. For this simple geometry, the normal surface stress defined in spherical co-ordinate system can be derived analytically and is given by

$$t_{rr}|_{r=R} = -\frac{3}{2}\frac{\mu U_x}{R}\cos\theta - p_\infty \tag{29}$$

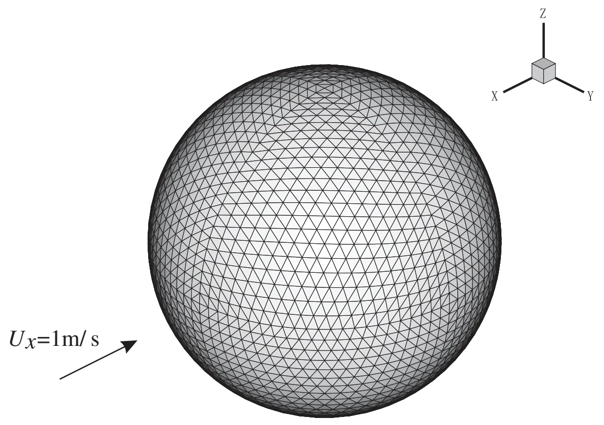

Figure 6. A translating sphere discretized using 4232 elements.

The drag force in x direction on the sphere is

$$T_x = \int_S t_x \, \mathrm{d}A = -6\pi\mu UR \tag{30}$$

6.1. *FMM accelerated matrix-vector multiplications*

We first compare the accuracy and efficiency of the original and new version of FMMs. Letting $\overline{F_P}$ represent a constant normal pressure p and considering the matrix G in the right hand side of Eq. (7), we have the following equality:

$$G\overline{F_P} = 0 \tag{31}$$

Hence, the relative error of the FMM accelerated matrix vector product can be measured by

$$e_r = \frac{\mu \|G\overline{F_P}\|_2}{p} \tag{32}$$

In Figure 7, we plot this relative error for the original and new version of FMMs with different multipole/local and exponential expansion orders (from 4 to 32). A sphere with 12 168 elements is used and numerical results are compared with results using analytically integration. In Figures 8 and 9, the computation time and memory requirements are presented. From these figures, it can be seen that the new version of FMM outperforms the original one when higher accuracy is required (large multipole/local orders); however, it requires more storage due to the introduction of exponential expansions. For 5–6 digits accuracy, the new version of FMM with multipole/local/exponential order = 18 processes a matrix vector product in 25 s requesting 200 MB memory for a $36\,504 \times 36\,504$ matrix. This setting is used in the following calculations.

6.2. *Accuracy and performance of Stokes solvers*

Our current numerical solver uses flat triangular discretization with constant basis functions. In Figure 10 we compare the relative error in the total drag force as a function of the total number

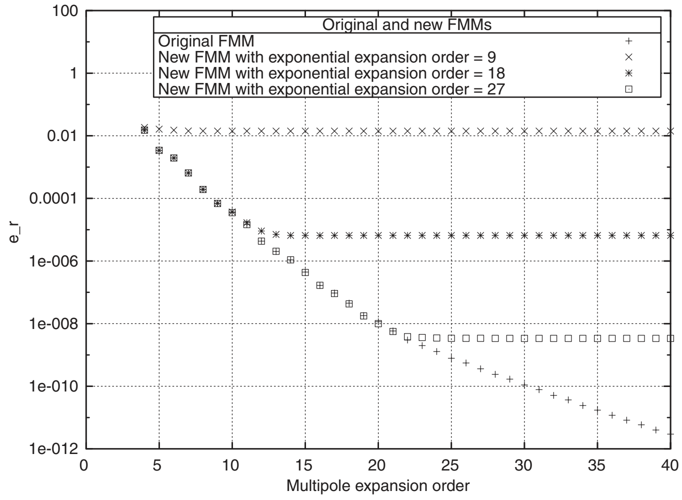

Figure 7. Multipole expansion order *vs* relative error.

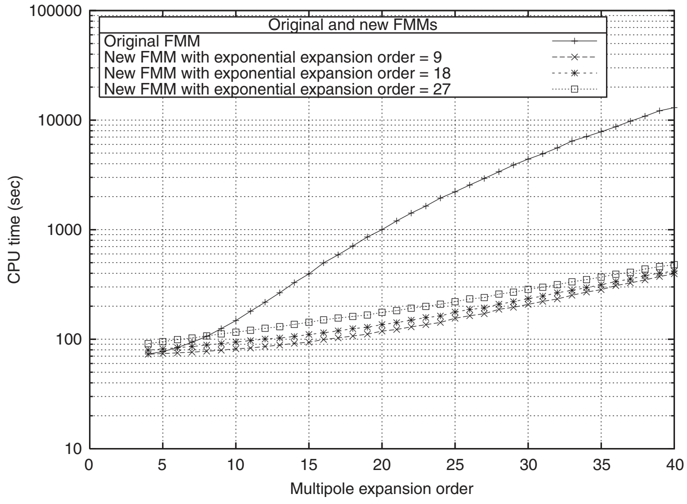

Figure 8. Multipole expansion order *vs* CPU time.

of elements. Clearly, the method is 1st order. Higher order basis functions are being considered and results will be reported in the future.

The computation time and memory requirement comparisons between the new version of FMM accelerated BIE and conventional BIE (using Lapack) on serial computer architecture are shown in Figures 11 and 12, respectively. The computation and storage complexity can be seen from the slope of each plotted line. Comparison of the parallelized FMM–BIE and conventional BIE (using

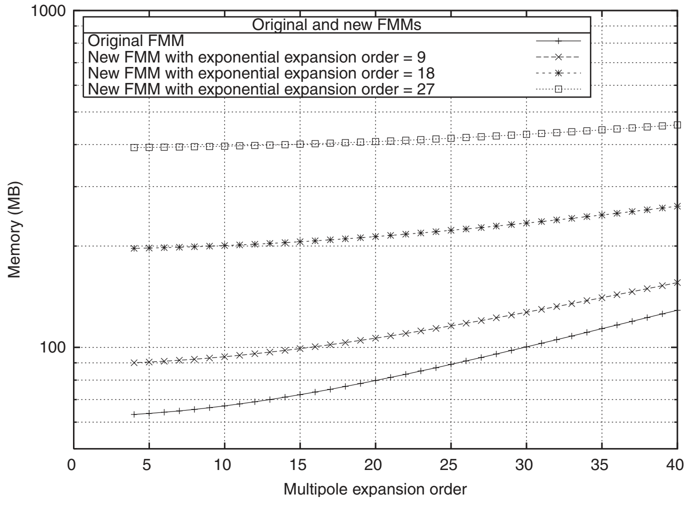

Figure 9. Multipole expansion order *vs* memory.

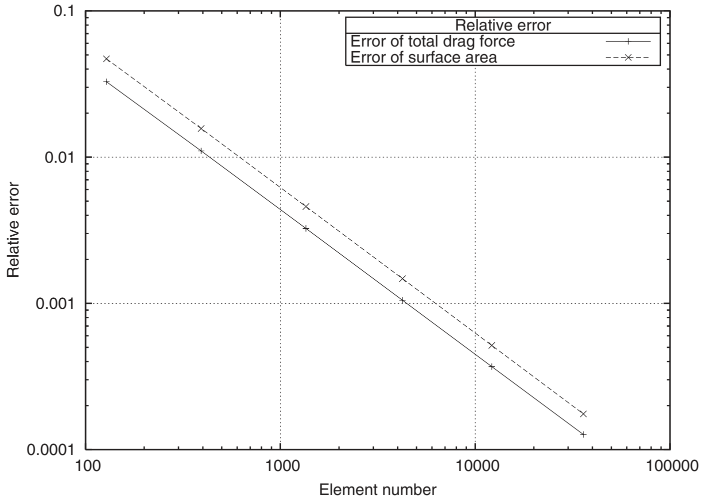

Figure 10. Relative error in total drag force *vs* element number.

ScaLAPACK [73]) is given in Figure 13. The FMM–BIE method inherits the virtue of parallel efficiency as an iterative solver.

6.3. Micro-fluidic device simulations

Our numerical solver is also being applied to the simulation of Stokes flow inside a micro-fluid device with complex geometry as shown in Figures 14 (3D view) and 15 (2D view). This device

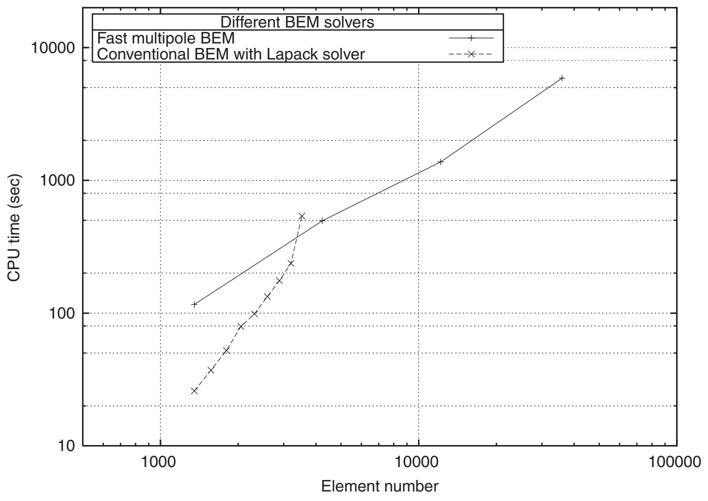

Figure 11. Computational time *vs* element number.

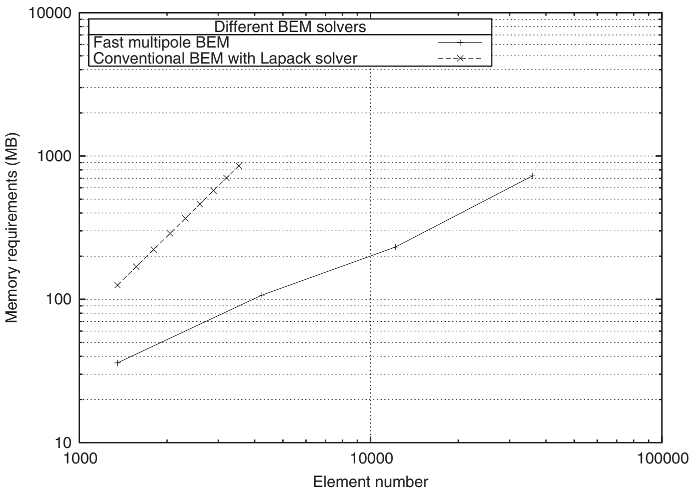

Figure 12. Memory requirement *vs* element number.

consists of two parallel circular plates, with 81 spouts placed on the top. The bottom plate is moving with axial symmetrical normal velocity as shown in Figure 16 with radius and surface normal vector defined in Figures 15 and 16. The inlet and spouts are specified by traction boundary conditions with $T_n = 0.101\,325\,\mathrm{N/mm^2}$ as shown in Figure 14. Zero-velocity boundary conditions are specified on other surfaces.

In our simulation, the micro-fluid device is discretized into 279 046 triangular elements. A detailed view of the meshes is shown in Figure 17. The new version of FMM with 18 terms in

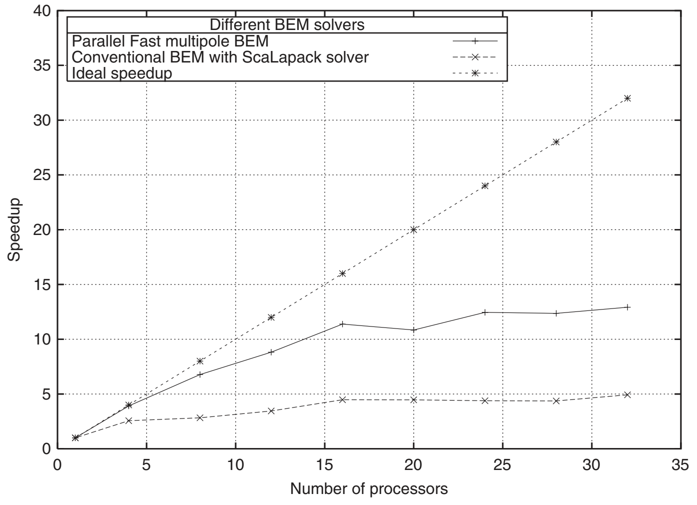

Figure 13. Parallel speed up *vs* number of processors.

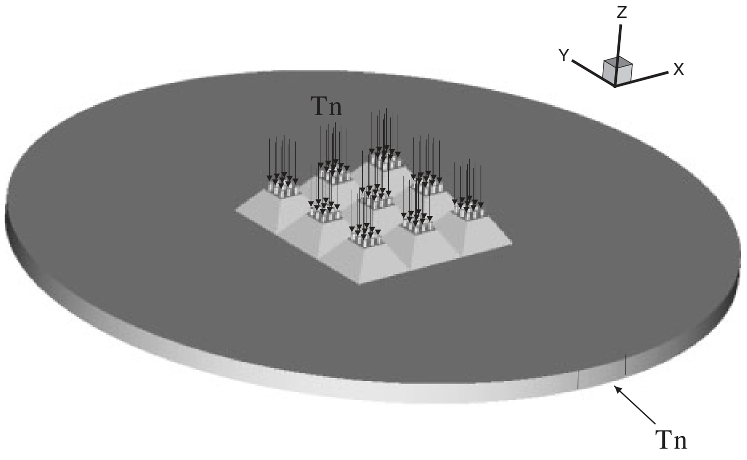

Figure 14. Geometry of the micro-fluid device.

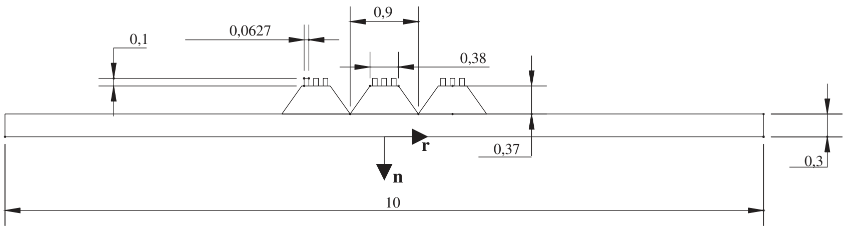

Figure 15. Dimensions of micro-fluid device (unit: mm).

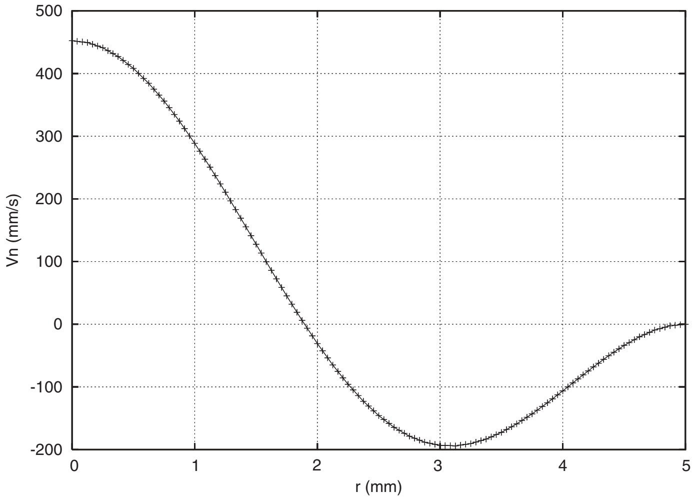

Figure 16. Normal velocity (V_n) at bottom plate as a function of plate radius (r).

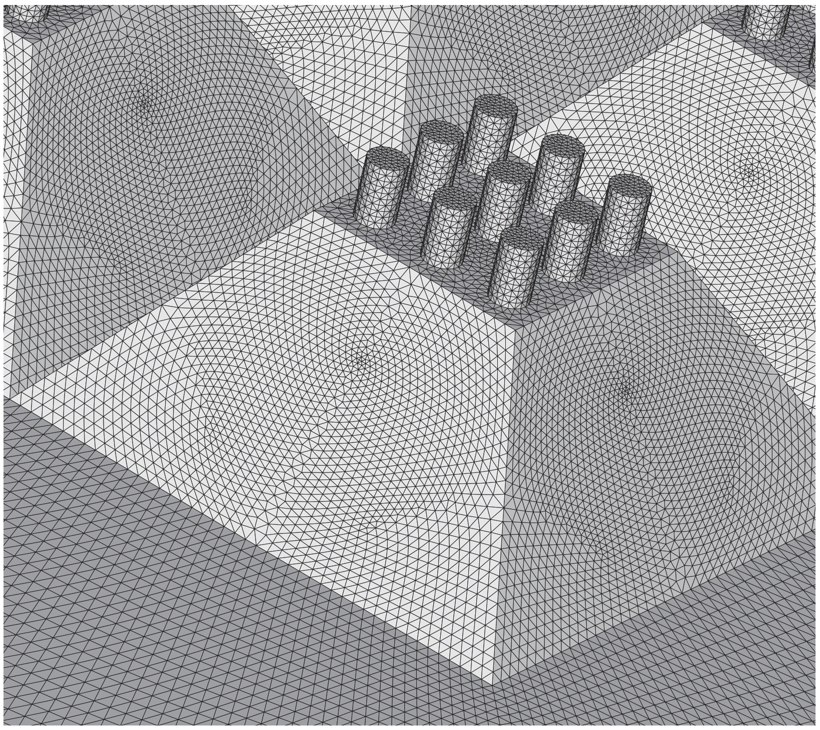

Figure 17. Zoomed view of the BEM meshes.

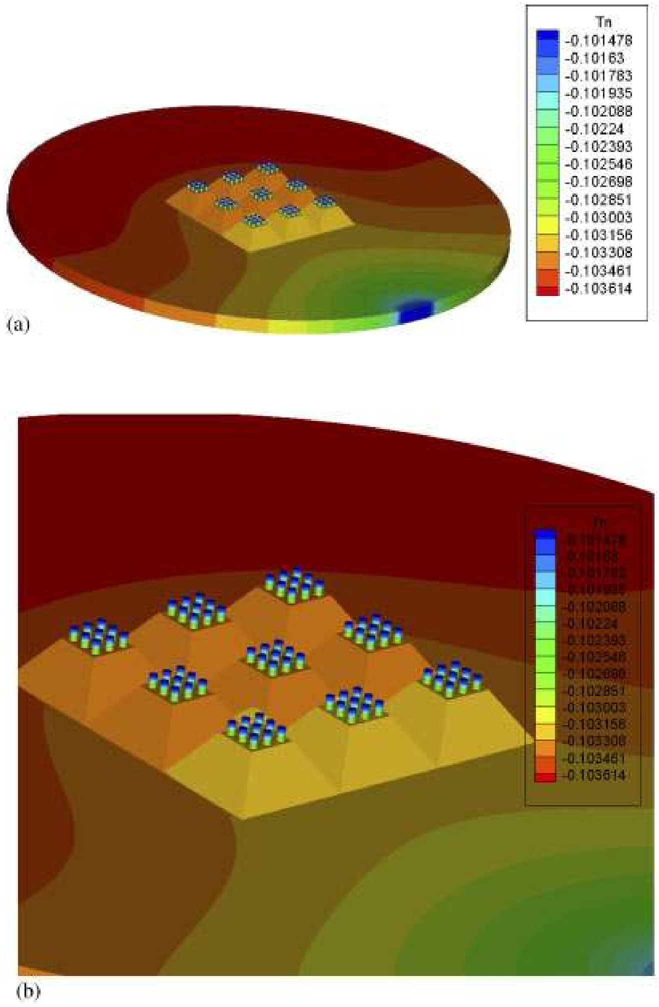

Figure 18. Surface normal stress of the micro-fluid device (unit: N/mm^2): (a) normal stress distribution.; and (b) zoomed view of the normal stress distribution.

the multipole, local and exponential expansions is adopted, and the convergence residual for the GMRES is set to 1.0×10^{-6}. The CPU time for a problem with DOF = 1 000 000 is about 3 h on our parallel cluster.

In Figures 18 and 19, the contour plots of the surface normal traction and velocity are shown respectively. One can see that both the inlet and spouts have positive normal velocity, which is approximately in agreement with observed experimental results. Currently, the device is being optimized by applying several other velocity boundary conditions using our fast Stokes solver.

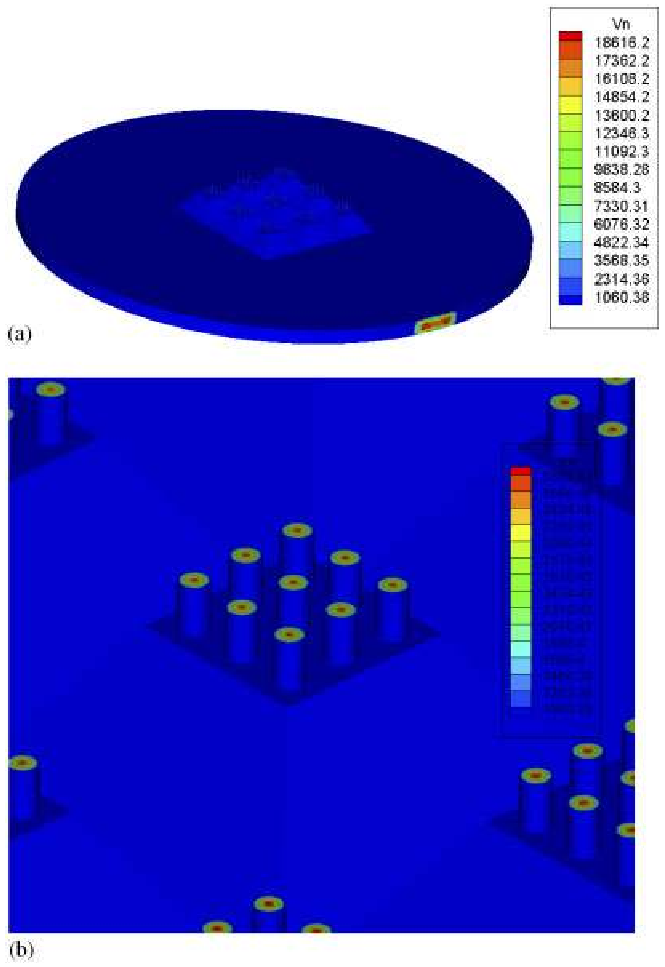

Figure 19. Contour plot of the surface normal velocity (unit: mm/s): (a) normal velocity distribution; and (b) zoomed view of the normal velocity distribution.

7. CONCLUSIONS

In this paper, we present an integral equation-based numerical method for the Stoke equations in three dimensions. The method uses Stokeslet and stresslet, and the dense matrix vector multiplications are calculated using the new version of FMM. The numerical solver allows for complex geometry and is parallelized for large-scale problems. It is currently being applied to bioMEMS design problems as well as simulation of fluids in porous medium.

Currently, the numerical method is being optimized for better performance. Efforts include the design of Fredholm second kind integral equation formulations, the choice of Krylov subspace methods, and better graphic user interface (GUI). In addition, we are trying to generalize the current code to the incompressible Navier–Stokes equations of low to moderate Reynold's numbers as well as to incorporate moving interfaces in fluid. Results along these directions will be reported later.

ACKNOWLEDGEMENTS

We want to express our gratitude to Prof. Houde Han of Tsinghua University for many helpful discussions. We also want to thank Prof. Xiaohao Wang for introducing the bioMEMS problem to us which leads to the development of our current numerical package. Finally, the work in this paper is supported by a research grant from Tsinghua University, and their support is thankfully acknowledged.

REFERENCES

1. Li Z, LeVeque R. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis* 1994; **31**:1019–1044.
2. Peskin CS. The Immersed boundary method. *Acta Numerica* 2002; **11**:479–517.
3. Bell JB, Berger MJ, Saltzman JS, Welcome ML. Three dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM Journal on Scientific Computing* 1994; **15**(1):127–138.
4. Berger MJ, Colella P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1989; **82**:64–84.
5. Berger MJ, Oliger J. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics* 1984; **53**:484–512.
6. Berger MJ, Rigoustsos I. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man and Cybernetics* 1991; **21**:1278–1286.
7. Darden T, York D, Pedersen L. Particle mesh Ewald—an $O(n \log n)$ method for Ewald sums in large systems. *Journal of Chemical Physics* 1993; **98**(12):10089–10092.
8. Hockney RW, Eastwood JW. Computer simulation using particles. Institute of Physics Publishing: 1988.
9. Phillips JR, White J. A precorrected-FFT method for capacitance extraction of complicated 3-D structures. *International Conference on Computer-Aided Design*, Santa Clara, California, 1994.
10. Appel AW. An efficient program for many-body simulations. *SIAM Journal on Scientific and Statistical Computing* 1985; **6**(1):85–103.
11. Barnes J, Hut P. A hierarchical $O(n \log n)$ force calculation algorithm. *Nature* 1986; **324**(4):446–449.
12. Greengard L, Rokhlin V. A fast algorithm for particle simulations. *Journal of Computational Physics* 1987; **73**(2):325–348.
13. Greengard L, Rokhlin V. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta numerica* 1997; **6**:229–269.
14. Kamon M, Tsuk MJ, White J. FastHENRY: a multipole-accelerated 3-D inductance extraction program. *IEEE Transactions on Microwave Theory and Techniques* 1994; **42**(9):1750–1758.
15. Kapur S, Long DE. IES3: a fast integral equation solver for efficient 3-dimensional extraction. *ICCAD* 1997; 448–455.
16. Kapur S, Long DE. Large-scale capacitance calculation. *DAC* 2000; 744–749.
17. Kapur S, Long DE. Large-scale full-wave simulation. *DAC* 2004; 806–809.
18. Liu Y, Nishimura N, Otani Y. A fast boundary element method for the analysis of fiber-reinforced composites based on a rigid-inclusion model. *Journal of Applied Mechanics* (ASME) 2005; **72**(1):115–128.
19. Michielssen E, Ergin A, Shanker B, Weile D. The multilevel plane wave time domain algorithm and its applications to the rapid solution of electromagnetic scattering problems: a review. *Mathematical and Numerical Aspects of Wave Propagation* (Santiago de Compostela 2000). SIAM: Philadelphia, PA, 2000; 24–33.
20. Nabors K, Kim S, White J. Fast capacitance extraction of general three-dimensional structures. *IEEE Transactions on Microwave Theory and Techniques* 1992; **40**(7):1496–1507.
21. Nabors K, Korsmeyer FT, Leighton FT, White J. Multipole accelerated preconditioned iterative methods for three-dimensional potential integral equations of the first kind. *SIAM Journal on Scientific and Statistical Computing* 1994; **15**(3):713–735.
22. Nabors K, White J. FASTCAP: a multipole-accelerated 3-D capacitance extraction program. *IEEE Transactions on Computer Aided Design* 1991; **10**(11):1447–1459.

23. Fu YH, Klimkowski KJ, Rodin GJ, Berger E, Browne JC, Singer JK, Van de Geijn RA, Vemaganti KS. A fast solution method for three-dimensional many-particle problems of linear elasticity. *International Journal for Numerical Methods in Engineering* 1998; **42**(7):1215–1229.
24. Greengard L, Helsing J. On the numerical evaluation of elastostatic fields in locally isotropic two-dimensional composites. *Journal of the Mechanics and Physics of Solids* 1998; **46**(8):1441–1462.
25. Helsing J. An integral equation method for electrostatics of anisotropic composites. *Proceedings of the Royal Society of London*, *Series A* 1995; **450**:343–350.
26. Helsing J, Peters G. Integral equation methods and numerical solutions of crack and inclusion problems in planar elastostatics. *SIAM Journal on Applied Mathematics* 1999; **59**(3):965–982.
27. Takahashi T, Nishimura N, Kobayashi S. A fast BIEM for three-dimensional elastodynamics in time domain. *Engineering Analysis with Boundary Elements* 2003; **27**(5):491–506.
28. Wang HT, Yao ZH. A new fast multipole boundary element method for large scale analysis of mechanical properties in 3D particle-reinforced composites. *Computer Modeling in Engineering and Sciences* 2005; **7**(1): 85–95.
29. Yoshida K. Applications of fast multipole method to boundary integral equation method. *Ph.D. Dissertation*, Department of Global Environment Engineering, Kyoto University, 2001.
30. Yoshida K, Nishimura N, Kobayashi S. Application of fast multipole Galerkin boundary integral equation method to elastostatic crack problems in 3D. *International Journal for Numerical Methods in Engineering* 2001; **50**(3):525–547.
31. Yoshida K, Nishimura N, Kobayashi S. Application of new fast multipole boundary integral equation method to crack problems in 3D. *Engineering Analysis with Boundary Elements* 2001; **25**:239–247.
32. Aluru NR, White J. A fast integral equation technique for analysis of microflow sensors based on drag force calculations. *International Conference on Modeling and Simulation of Microsystems*, *Semiconductors*, *Sensors and Actuators*, Santa Clara, 1998; 283–286.
33. Biros G, Ying L, Zorin D. A fast solver for the Stokes equations with distributed forces in complex geometries. *Journal of Computational Physics* 2004; **193**(1):317–348.
34. Greengard L. Potential flow in channels. *SIAM Journal on Scientific and Statistical Computing* 1990; **11**(4): 603–620.
35. Greengard L, Kropinski MC. An integral equation approach to the incompressible Navier–Stokes equations in two dimensions. *SIAM Journal on Scientific Computing* 1998; **20**(1):318–336.
36. Greengard L, Kropinski MC. Integral equation methods for Stokes flow in doubly-periodic domains. *Journal of Engineering Mathematics* 2004; **48**(2):157–170.
37. Greengard L, Kropinski MC, Mayo A. Integral equation methods for Stokes flow and isotropic elasticity in the plane. *Journal of Computational Physics* 1996; **125**(2):403–414.
38. Wang X, Mucha P, White J. Fast fluid analysis for multibody micromachined devices. *International Conference on Modeling and Simulation of Microsystems*, *Semiconductors*, *Sensors and Actuators*, Hilton Head, North Carolina, 2001.
39. Ye W, Kanapka J, Wang X, White J. Efficiency and accuracy improvements for FastStokes: a precorrected-FFT accelerated 3-D Stokes solver. *International Conference on Modeling and Simulation of Microsystems*, *Semiconductors*, *Sensors and Actuators*, San Juan, 1999.
40. Ye W, Kanapka J, White J. A fast 3-D solver for unsteady Stokes flow with application to micro-electro-mechanical systems. *International Conference on Modeling and Simulation of Microsystems*, *Semiconductors*, *Sensors and Actuators*, San Juan, 1999.
41. Ye W, Wang X, White J. A fast Stokes solver for generalized flow problems. *International Conference on Modeling and Simulation of Microsystems*, *Semiconductors*, *Sensors and Actuators*, San Diego, 2000.
42. Ying LX, Biros G, Zorin D. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics* 2004; **196**(2):591–626.
43. Grama A, Kumar V, Same A. Parallel hierarchical solvers and preconditioners for boundary element methods. *SIAM Journal on Scientific Computing* 1998; **20**:337–358.
44. Leathrum JF, Board JA. The parallel fast multipole algorithm in three dimensions. *Technical Report TR92-001*, Duke University, Department of Electrical Engineering, 1992
45. Mammoli AA, Ingber MS. Parallel multipole BEM simulation of two-dimensional suspension flows. *Engineering Analysis with Boundary Elements* 2000; **24**:65–73.
46. Rankin WT. Efficient parallel implementations of multipole based N-body algorithms. *Ph.D. Thesis*, Duke University, 1999.

47. Singer JK. Parallel implementation of the fast multipole method with periodic boundary conditions. *East-West Journal of Numerical Mathematics* 1995; **3**:199–216.
48. Warren MS, Salmon JK. A parallel hashed oct-tree N-body algorithm. *Proceedings of Supercomputing'93*, Portland, Oregon, US, 1993; 12–21.
49. Yuan Y, Banerjee P. A parallel implementation of a fast multipole-based 3-D capacitance extraction program on distributed memory multicomputers. *Journal of Parallel and Distributed Computing* 2001; **61**(12):1751–1774.
50. Pozrikidis C. *Boundary Integral and Singularity Methods for Linearized Viscous Flow*. Cambridge University Press: Cambridge, 1992.
51. Chen K, Iserles A, Ciarlet PG (eds). *Matrix Preconditioning Techniques and Applications*. Cambridge University Press: Cambridge, 2005.
52. Nishida T, Hayami K. Application of the fast multipole method to the 3-D BEM analysis of electron guns. In *Boundary Elements XIX*, Marchetti M *et al*. (eds.). Computational Mechanics: Southampton, NY 1997; 613–622.
53. Nishimura N, Yoshida K, Kobayashi S. A fast multipole boundary integral equation method for crack problems in 3D. *Engineering Analysis with Boundary Elements* 1999; **23**:97–105.
54. Wang HT, Yao ZH, Wang PB. On the preconditioners for fast multipole boundary element methods for 2D multi-domain elastostatics. *Engineering Analysis with Boundary Elements* 2005; **29**:673–688.
55. Carrier J, Greengard L, Rokhlin V. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing* 1988; **9**(4):669–686.
56. Cheng H, Greengard L, Rokhlin V. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics* 1999; **155**:468–498.
57. Contopanagos H, Dembart B, Epton M *et al*. Well-conditioned boundary integral equations for three-dimensional electromagnetic scattering. *IEEE Transactions on Antennas and Propagation* 2002; **50**(12):1824–1830.
58. Jiang SD, Rokhlin V. Second kind integral equations for the classical potential theory on open surfaces I: analytical apparatus. *Journal of Computational Physics* 2003; **191**(1):40–74.
59. Jiang SD, Rokhlin V. Second kind integral equations for the classical potential theory on open surfaces II. *Journal of Computational Physics* 2004; **195**(1):1–16.
60. Kolm P, Jiang SD, Rokhlin V. Quadruple and octuple layer potentials in two dimensions. I. analytical apparatus. *Applied and Computational Harmonic Analysis* 2003; **14**(1):47–74.
61. Barrett R *et al*. *Templates for the Solution of Linear Systems*: *Building Blocks for Iterative Methods* (2nd edn). SIAM: Philadelphia, PA, 1994.
62. Broyden CG, Vespucci MT. Krylov solvers for linear algebraic systems. *Studies in Computational Mathematics* vol. 11. Elsevier: Berlin, 2004.
63. Kelly CT. *Iterative Methods for Linear and Nonlinear Equations*. SIAM: Philadelphia, PA, 1995.
64. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**:856–869.
65. Vorst HA. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press: Cambridge, 2003.
66. Greengard L, Huang JF. A new version of the fast multipole method for screened Coulomb interactions in three dimensions. *Journal of Computational Physics* 2002; **180**(2):642–658.
67. Greengard L, Huang JF, Rokhlin V *et al*. Accelerating fast multipole methods for the Helmholtz equation at low frequencies. *IEEE Computational Science and Engineering* 1998; **5**(3):32–38.
68. Hrycak T, Rokhlin V. An improved fast multipole algorithm for potential fields. *SIAM Journal on Scientific Computing* 1998; **19**:1804–1826.
69. Abramowitz M, Stegun IA. *Handbook of Mathematical Functions*. Dover: New York, 1970.
70. Cheng H, Rokhlin V, Yarvin N. Nonlinear optimization, quadrature, and interpolation. Dedicated to John E. Dennis Jr. on his 60th birthday. *SIAM Journal on Optimization* 1999; **9**(4):901–923.
71. Gumerov NA, Duraiswami R. Fast multipole method for the biharmonic equation in three dimensions. *Journal of Computational Physics* 2006; **215**(1):363–383.
72. Sagan H. *Space-Filling Curves*. Springer: New York, 1994.
73. Demmel J, Heath M, van der Vorst H. Parallel numerical linear algebra. *Acta Numerica*, vol. 2. Cambridge University Press: Cambridge, 1993.