Commun. Comput. Phys.
doi: 10.4208/cicp.150215.260615sw

Vol. **18**, No. 2, pp. 516-528
August 2015

COMPUTATIONAL SOFTWARE

Simple FMM Libraries for Electrostatics, Slow Viscous Flow, and Frequency-Domain Wave Propagation

Zydrunas Gimbutas[1,*] and Leslie Greengard[2,3]

[1] *Information Technology Laboratory, National Institute of Standards and Technology, 325 Broadway, Mail Stop 891.01, Boulder, CO 80305-3328, USA.*
[2] *Simons Center for Data Analysis, Simons Foundation, 160 Fifth Avenue, NY, NY 10010, USA.*
[3] *Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012-1110, USA.*

Received 15 February 2015; Accepted (in revised version) 26 June 2015

Abstract. We have developed easy to use fast multipole method (FMM) libraries for the Laplace, low-frequency Helmholtz, and Stokes equations in two and three dimensions. The codes are based on a new method for applying translation operators and provide reasonable performance on either single core processors, or small multi-core systems using OpenMP.

AMS subject classifications: 52B10, 65D18, 68U05, 68U07

Key words: Fast multipole method, Laplace equation, Helmholtz equation, Stokes equation, layer potentials, projection-based point and shoot translation operators.

Program Summary

Program title: FMMLIB

Nature of problem: Fast multipole method

Software licence: GPL 2.0

CiCP scientific software URL:

Distribution format: tar.gz, .zip

*Corresponding author. *Email addresses:* `zydrunas.gimbutas@nist.gov` (Z. Gimbutas), `greengard@cims.nyu.edu` (L. Greengard)

Programming language(s): Fortran, Matlab

Computer platform: Any

Operating system: Any

Compilers: GNU Fortran, Intel Fortran Compiler

RAM:

External routines/libraries: FFTPACK (included)

Running time:

Restrictions:

Supplementary material and references:

Additional Comments:

1 Introduction

The FMMLIB package provides fully adaptive implementations of the fast multipole method (FMM) for the Laplace, Helmholtz and Stokes equations. It is not highly optimized, intended rather to be accessible and modifiable with only modest effort. In particular, the translation operators used in the three-dimensional libraries are based on rotation and projection – a new, unified, and simple framework discussed briefly in Section 3.1. For optimal performance, plane wave-based translation operators should be used [3, 14]. This, however, would add significant complexity to the code, and would make the algorithm less transparent to the user and more difficult to modify. Instead, we provide a fully parallelized code which is reasonably well optimized for performance on small multi-core systems using OpenMP. Further acceleration could be obtained by precomputation and storage in matrix form of many of the modules in the library (formation of expansions from sources, evaluation of multipole and local expansions, etc.). In addition to increasing the memory costs, this would generally require a two-pass procedure and would, again, make the software itself less accessible.

The fast multipole method computes N-body interactions and evaluates layer potentials in $\mathcal{O}(N\log N)$ time for non-pathological particle distributions. Typically, $\mathcal{O}(N\log N)$ work with a small constant is needed to build the adaptive tree data structure on which the method relies and $\mathcal{O}(N)$ work with a larger constant is then required for the computation itself. In the case of the Helmholtz equation, we assume that the entire computational domain (the support of the scatterers) is a modest number of wavelengths in size. This is the "low frequency" regime from the point of view of either scattering theory or FMM implementation. The high-frequency version of the FMM is a more complex algorithm, and has not been incorporated into this software release. We do, however, provide a subroutine which is able to evaluate the scattered field at an arbitrary distance from the scatterers in terms of a single multipole expansion about the center of the computational domain.

2 Laplace and Helmholtz point FMM in R^2

FMMLIB2D computes sums of the form:

$$\phi(\mathbf{y}_i)=\sum_{j=1}^{N} q_j G_k(\mathbf{y}_i-\mathbf{x}_j)+p_j\mathbf{n}_j\cdot\nabla_{\mathbf{x}_j}G_k(\mathbf{y}_i-\mathbf{x}_j), \tag{2.1}$$

for $i=1,\cdots,M$, as well as up to second derivatives of ϕ, where

$$G_k(\mathbf{x})=\frac{i}{4}H_0^{(1)}(k\|\mathbf{x}\|) \quad \text{for } k\neq 0, \quad G_0(\mathbf{x})=\log(\|\mathbf{x}\|).$$

$\mathbf{x}_j$ are the source locations, $\mathbf{y}_i$ are the target locations, q_j is referred to as the charge strength and p_j as the dipole strength. $\mathbf{n}=(n_1,n_2)$ is a vector whose direction determines the dipole orientation (if present). If all target locations coincide with the source locations, i.e., $N=M$ and $y_i=x_i$ for all $i=1,\cdots,N$, then the sums (2.1) are evaluated ignoring the self-interactions. For $k\neq 0$, we assume that k is in the upper half of the complex plane. It is designed for scattering calculations, and there are scaling issues that would need to be incorporated to handle the modified Helmholtz (Yukawa) regime where k is both large and near the imaginary axis (the code may perform poorly or fail in that regime). Note also that when $k=0$, we omit the $-\frac{1}{2\pi}$ scaling of $\log(\|\mathbf{x}\|)$ that defines the true Green's function. There are, in fact, several different routines available when $k=0$. Subroutines with the prefix `lfmm2d` compute complex-valued sums of the form:

$$\phi(\mathbf{y}_i)=\sum_{j=1}^{N} q_j\log(\|\mathbf{y}_i-\mathbf{x}_j\|)+p_j\mathbf{n}_j\cdot\nabla_{\mathbf{x}_j}\left(\log(\|\mathbf{y}_i-\mathbf{x}_j\|)\right). \tag{2.2}$$

Subroutines with the prefix `zfmm2d` compute complex-valued sums of the form:

$$\phi(\xi_i)=\sum_{j=1}^{N} p_j\frac{1}{\xi_i-z_j}. \tag{2.3}$$

Subroutines with the prefix `cfmm2d` compute complex-valued sums of the form:

$$\phi(\xi_i)=\sum_{j=1}^{N} q_j\log(\xi_i-z_j)+p_j\frac{1}{\xi_i-z_j}. \tag{2.4}$$

The `cfmm2d` routines are not intended for novice users, since the complex valued logarithm is a multi-valued function. As a result, the sums (2.4) have to be interpreted carefully and the routines are intended for advanced users only. To be more precise, the sums (2.4) are intended to be evaluated with strictly real-valued charges q_j yielding a well-defined real part of (2.4). Internally, we use two calls to cfmm2d to evaluate the sums (2.2).

3 Laplace and Helmholtz point FMM in R^3

FMMLIB3D computes sums of the form:

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} q_j G_k(\mathbf{y}_i - \mathbf{x}_j) + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j} G_k(\mathbf{y}_i - \mathbf{x}_j), \tag{3.1}$$

for $i = 1, \cdots, M$, as well as derivatives of ϕ, where

$$G_k(\mathbf{x}) = \frac{e^{ik\|\mathbf{x}\|}}{\|\mathbf{x}\|},$$

$\mathbf{x}_j$ are the source locations, $\mathbf{y}_i$ are the target locations, q_j is referred to as the charge strength, and p_j as the dipole strength. $\mathbf{n} = (n_1, n_2, n_3)$ is a vector whose direction determines the dipole orientation (if present). As in the 2D case, the sums (3.1) are evaluated ignoring the self-interactions, if all target locations coincide with the source locations, and we omit the scaling factor $\frac{1}{4\pi}$ which is present in the true free-space Green's function. For $k \neq 0$, we assume that k is in the upper half of the complex plane with $\Re(k) \geq \Im(k)/10$. It is designed for scattering calculations, and there are scaling issues that would need to be incorporated to handle the modified Helmholtz (Yukawa) regime where k is nearer the imaginary axis.

Important note: The charge and dipole strengths are assumed to be **complex** double precision numbers for both the Laplace and Helmholtz libraries.

3.1 Translation via projection

The workhorse of classical FMM codes involves the movement of data between boxes – either from child boxes to their parent (so-called "multipole-to-multipole" translation operators), from parent boxes to their children (so-called "local-to-local" translation operators), or from a given box B to other boxes at the same level of the spatial hierarchy that are in B's "interaction list". We refer the reader to [3,4,14] for a more detailed discussion. It is well-known that naive translation operators for the FMM require $\mathcal{O}(p^4)$ work in three dimensions, where p is the order of the spherical harmonic expansion. Since this is a dominant factor in the net computational cost of the FMM, there has been a substantial amount of work in reducing the complexity of this step. Plane-wave based schemes, as in [3, 4, 7, 14], require $\mathcal{O}(p^2)$ work and are essentially optimal, but require a much more complicated implementation. We have chosen to use "point-and-shoot" translation operators in FMMLIB3D, which require $\mathcal{O}(p^3)$ work, but are more straightforward to understand. The basic idea is that when translating from a box B to a box C, one first rotates

the coordinate system so that the z-axis is oriented in the direction from the center of B to the center of C. This requires $\mathcal{O}(p^3)$ work. Second, translation is carried out along the z-axis, again at a cost of $\mathcal{O}(p^3)$ work. Finally the coordinate system is restored to the original one through a second rotation. The relevant formulas for these steps are available in the literature (see, for example, [3–5, 13–15, 19, 20] and the references therein). For FMMLIB3D, we actually designed a new variant of the point-and-shoot method, which we refer to as "translation via projection" that also requires $\mathcal{O}(p^3)$ work [12] but bypasses the need for complicated analytic expressions for the z-translation (or rotation) steps. For instance, the incoming multipole expansion for the box C is given by

$$\begin{aligned} U^{in} &= \sum_{n=0}^{p} \sum_{m=-n}^{n} B_n^m j_n(kr) P_n^m(\cos\theta) e^{im\phi} \\ &= \sum_{m=-n}^{n} e^{im\phi} \sum_{n=0}^{p} B_n^m j_n(kr) P_n^m(\cos\theta), \end{aligned} \tag{3.2}$$

where, in accordance with conventions in [4], (r,θ,ϕ) are the local spherical coordinates of the box C, j_n are the spherical Bessel functions of the first kind and order n, and P_n^m are the normalized associated Legendre functions. For a fixed box radius R, the factors $B_n^m j_n(kR)$ can be easily obtained at a cost of $\mathcal{O}(p^3)$ by using the spherical harmonic transform algorithm [2]. Unfortunately, the multipole expansion coefficients B_n^m, in general, can not be recovered from these factors due to possible zeros of $j_n(kR)$. To address this issue, we form the multipole expansion for the radial derivative of the potential U^{in}

$$\begin{aligned} \frac{\partial U^{in}}{\partial r} &= \sum_{n=0}^{p} \sum_{m=-n}^{n} k B_n^m j_n'(kr) P_n^m(\cos\theta) e^{im\phi} \\ &= \sum_{m=-n}^{n} e^{im\phi} \sum_{n=0}^{p} k B_n^m j_n'(kr) P_n^m(\cos\theta), \end{aligned} \tag{3.3}$$

and observe that $j_n(kR)$ and $kj_n'(kR)$ can not vanish simultaneously. If (3.2) and (3.3) are linearly combined, then the incoming multipole coefficients will be recovered from the factors $B_n^m(j_n(kR)+\alpha k j_n'(kR))$, where α is the weighting given to (3.3).

4 Laplace and Helmholtz layer FMM in R^3

FMMLIB3D also provides subroutines designed to evaluate layer potentials

$$\phi(\mathbf{y}) = \int_T \sigma(\mathbf{x}) \frac{e^{ik\|\mathbf{y}-\mathbf{x}\|}}{\|\mathbf{y}-\mathbf{x}\|} + \mu(\mathbf{x})\mathbf{n}(\mathbf{x}) \cdot \nabla_{\mathbf{x}} \left(\frac{e^{ik\|\mathbf{y}-\mathbf{x}\|}}{\|\mathbf{y}-\mathbf{x}\|} \right) d\mathbf{x}, \tag{4.1}$$

where T is a closed surface in R^3 and $\mathbf{y}$ is a target point either on or off surface. In the present release, the layer potential routines assume the discretization of T consists of N

flat triangles and that $\sigma(\mathbf{x})$ and $\mu(\mathbf{x})$ are simply constants: σ_j and μ_j on the jth triangle, respectively.

Both single and double layer potentials can be computed as well as their derivatives, including on-surface principal value and hypersingular integrals. Thus, FMMLIB3D permits the computation of

$$\phi(\mathbf{y}_i)=\sum_{j=1}^{N}\int_{T_j}\sigma_j\frac{e^{ik\|\mathbf{y}_i-\mathbf{x}\|}}{\|\mathbf{y}_i-\mathbf{x}\|}+\mu_j\mathbf{n}_j\cdot\nabla_{\mathbf{x}}\left(\frac{e^{ik\|\mathbf{y}_i-\mathbf{x}\|}}{\|\mathbf{y}_i-\mathbf{x}\|}\right)d\mathbf{x}, \tag{4.2}$$

for $i=1,\cdots,N_t$, where N_t denotes the number of targets, as well as $\nabla\phi(\mathbf{y}_j)$ if desired. While straightforward to use, it is important to note that the quadrature is only first-order accurate.

5 Stokes point FMM in R^3

The flow of an incompressible Newtonian fluid at small values of Reynolds number Re is governed by the Stokes equation

$$\mu\Delta\mathbf{u}=\nabla p,\quad \operatorname{div}\mathbf{u}=0, \tag{5.1}$$

where $\mathbf{u}$ is the velocity of the fluid, p is the pressure, and μ is the dynamic viscosity. Without loss of generality, we can set the value of viscosity $\mu=1$ and consider the normalized version of the Stokes equation

$$\Delta\mathbf{u}=\nabla p,\quad \operatorname{div}\mathbf{u}=0. \tag{5.2}$$

The standard fundamental solutions (Green's functions) for the Stokes flow are usually referred to as the stokeslet $\mathbf{S}$ and stresslet $\mathbf{T}$, respectively, which correspond to singular point forces and dipoles embedded in the flow, [8,16,17]. The fast multipole method computes such Stokes N-body interactions in approximately linear time for non-pathological particle distributions. STFMMLIB3D computes sums of the form

$$\mathbf{u}(\mathbf{y}^m)=4\pi\sum_{n=1}^{N}\mathbf{S}(\mathbf{y}^m-\mathbf{x}^n)\mathbf{f}^n+\mathbf{T}(\mathbf{y}^m-\mathbf{x}^n)\boldsymbol{\nu}^n\mathbf{g}^n,$$
$$p(\mathbf{y}^m)=4\pi\sum_{n=1}^{N}\mathbf{P}(\mathbf{y}^m-\mathbf{x}^n)\mathbf{f}^n+\boldsymbol{\Pi}(\mathbf{y}^m-\mathbf{x}^n)\boldsymbol{\nu}^n\mathbf{g}^n,$$

for $m=1,\cdots,M$, where

$$S_{ij}(\mathbf{x})=\frac{1}{8\pi}\left(\delta_{ij}\frac{1}{\|\mathbf{x}\|}+\frac{\mathbf{x}_i\mathbf{x}_j}{||\mathbf{x}||^3}\right),\quad T_{ijk}(\mathbf{x})=\frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \tag{5.3}$$

$$P_j=\frac{2}{8\pi}\frac{\mathbf{x}_j}{||\mathbf{x}||^3},\quad \Pi_{jk}=\frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3}+3\frac{\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \tag{5.4}$$

$\mathbf{f}^n$ are referred to as the charge strengths and $\mathbf{g}^n$ as the dipole strengths, and $\boldsymbol{\nu}^n$ are the vectors whose directions determine the dipole orientations (if present). More precisely, the sums computed by STFMMLIB3D are of the form

$$u_i(\mathbf{y}^m) = 4\pi \sum_{n=1}^{N} \sum_{j=1}^{3} S_{ij}(\mathbf{y}^m - \mathbf{x}^n) f_j^m + 4\pi \sum_{n=1}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} T_{ijk}(\mathbf{y}^m - \mathbf{x}^n) \nu_k^n g_j^n,$$
$$p(\mathbf{y}^m) = 4\pi \sum_{n=1}^{N} \sum_{j=1}^{3} P_j(\mathbf{y}^m - \mathbf{x}^n) f_j^n + 4\pi \sum_{n=1}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} \Pi_{jk}(\mathbf{y}^m - \mathbf{x}^n) \nu_k^n g_j^n.$$

In some situations, it is more convenient to consider modifications of the standard stresslet $\mathbf{T}$ and the corresponding pressure tensor $\mathbf{\Pi}$:

$$T_{ijk}^{(2)}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi_{jk}^{(2)} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \tag{5.5}$$
$$T_{ijk}^{(3)}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi_{jk}^{(3)} = 0, \tag{5.6}$$
$$T_{ijk}^{(4)}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi_{jk}^{(4)} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \tag{5.7}$$

that are referred in the literature to as the symmetric part of the Stokes doublet, the rotlet, and the Stokes doublet, respectively. The STFMMLIB3D routines are able to replace the standard stresslet $\mathbf{T}$ with one of these kernels with a help of a properly set flag.

STFMMLIB3D is also able to evaluate gradients of the velocities $\mathbf{u}$. Formulas for the strain $\boldsymbol{\varepsilon}$ and stress $\boldsymbol{\sigma}$ tensors, can then be obtained from partial derivatives of the preceding formulas for velocities with respect to each component x_i:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{5.8}$$
$$\sigma_{ij} = -\delta_{ij}p + \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{5.9}$$

and the vorticity $\boldsymbol{\omega}$ and the stress $\mathbf{t}$ vectors are

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}, \quad \mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n}, \tag{5.10}$$

where $\mathbf{n}$ is a unit direction vector.

In addition to the standard fundamental solutions for the Stokes equations, it is sometimes convenient to be able to evaluate second derivatives of harmonic functions. Calculations of this type arise, for example, in evaluating pressure gradients ($\Delta\mathbf{S} = \nabla\mathbf{P}$), in hydrodynamic interactions for spheres (Rotne-Prager-Yamakawa tensor), in evaluating

stresslets, rotlets, and Stokes doublets, etc. The additional harmonic (Laplace) routines available in the STFMMLIB3D library compute sums of the form:

$$\phi(\mathbf{y}^m)=\sum_{n=1}^{N} q^n G(\mathbf{y}^m-\mathbf{x}^n)+p^n \boldsymbol{\nu}^n\cdot\nabla_{x^n} G(\mathbf{y}^m-\mathbf{x}^n)+h^n \boldsymbol{\eta}^n\cdot\nabla_{x^n}\nabla_{x^n} G(\mathbf{y}^m-\mathbf{x}^n),$$

for $m=1,\cdots,M$, where $G(\mathbf{x})=1/||\mathbf{x}||$, q^n are referred to as the charge strengths, p^n as the dipole strengths, h^n as the quadrupole strengths, and $\boldsymbol{\nu}^n$, $\boldsymbol{\eta}^n$ are the vectors whose directions determine the dipole and quadrupole orientations, respectively (if present). We use a non-standard way to represent the quadrupole orientation tensor, and take advantage of the symmetry to express it as a six-dimensional vector. More precisely, the sums computed are of the form

$$\begin{aligned}\phi(\mathbf{y}^m)=&\sum_{n=1}^{N} q^n G(\mathbf{y}^m-\mathbf{x}^n)+p^n(\nu_1^n\cdot\partial_{x_1^n}+\nu_2^n\cdot\partial_{x_2^n}+\nu_3^n\cdot\partial_{x_3^n})G(\mathbf{y}^m-\mathbf{x}^n)\\&+h^n(\eta_1^n\cdot\partial^2_{x_{11}^n}+\eta_2^n\cdot\partial^2_{x_{22}^n}+\eta_3^n\cdot\partial^2_{x_{33}^n}+\eta_4^n\cdot\partial^2_{x_{12}^n}+\eta_5^n\cdot\partial^2_{x_{13}^n}+\eta_6^n\cdot\partial^2_{x_{23}^n})G(\mathbf{y}^m-\mathbf{x}^n).\end{aligned}$$

5.1 Stokes FMM via harmonic potentials

STFMMLIB3D makes use of the fact that the Stokeslet and Stresslet summation formulas can be written in terms of four harmonic functions, so that the harmonic FMMLIB3D routines can be used in "black-box" fashion, [17, 18].

6 Software installation and numerical examples

Software implementing the various point and layer potential FMMs discussed above is available at the Courant Mathematics and Computing Laboratory website [9–11]. All timing tests were performed on a 16-core Intel(R) Xeon(R) CPU E5-2687W workstation at 3.1 GHz with 128GB RAM, using Intel Fortran compiler 14.0.2.

The results of numerical experiments are summarized in Tables 1-13. In all tables, the first column represent the number of sources (and, if requested, the targets) for which all calculations are done. The second, third, and forth columns contain the CPU times required by the algorithm using OpenMP on one, four, or fifteen cores, respectively. The last columns contain the average relative L_2 error obtained at the source and target locations. As it is customary in the FMM literature [1,3], due to excessive computation times, we used the direct algorithm to evaluate the potentials at a subset of 100 source (target) locations, and used the resulting data to estimate the relative errors. Similarly, due to large computational constants associated with the layer potentials, we used a smaller number of sources and targets in Tables 9-12. As expected, the CPU time for the FMM algorithm grows linearly with the number of elements N and the use of OpenMP typically accelerates the scheme by an additional factor of about 8-10 while using 15 cores.

6.1 Sample drivers for FMMLIB2D

In the FMM2D/examples directory, the file `lfmm2dpart_driver.f` contains a sample driver for `lfmm2dparttarg`. It creates a random distribution of source points on the unit circle centered at the origin and a random distribution of target points on a distinct unit circle, centered at $(3,0)$. The code then computes the potential, gradient, and second

Table 1: Laplace particle FMM in R^2, lfmm2dpart, self and target evaluation; CPU times (in seconds) for computing the Laplace potential, gradient, and second derivatives on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; complex-valued charges and dipoles randomly distributed on a unit circle, the precision parameter iprec $=$ 4 (12 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
10000	0.16393E+00	0.71320E-01	0.83323E-01	0.46637E-14	0.19465E-14
100000	0.12463E+01	0.45248E+00	0.26063E+00	0.78264E-14	0.86886E-14
1000000	0.12538E+02	0.43906E+01	0.19226E+01	0.20800E-13	0.20692E-13

Table 2: Laplace particle FMM in R^2, lfmm2dpart, self evaluation; CPU times (in seconds) for computing the Laplace potential, gradient, and second derivatives on 1, 4, and 15 cores, respectively; the relative L_2 errors for self potential; complex-valued charges and dipoles randomly distributed on a unit circle, the precision parameter iprec $=$ 4 (12 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$
10000	0.13305E+00	0.53702E-01	0.64474E-01	0.46691E-14
100000	0.10162E+01	0.35395E+00	0.17123E+00	0.80473E-14
1000000	0.10048E+02	0.32991E+01	0.12728E+01	0.20576E-13

Table 3: Helmholtz particle FMM in R^2, hfmm2dpart, self and target evaluation; CPU times (in seconds) for computing the Helmholtz potential, gradient, and second derivatives on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; complex-valued charges and dipoles randomly distributed on a unit circle, $k=20$, the precision parameter iprec $=$ 4 (12 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
10000	0.30586E+00	0.10494E+00	0.88889E-01	0.52505E-14	0.71748E-14
100000	0.26139E+01	0.84250E+00	0.44690E+00	0.12803E-13	0.86788E-14
1000000	0.25685E+02	0.82447E+01	0.38303E+01	0.16638E-13	0.21756E-13

Table 4: Helmholtz particle FMM in R^2, hfmm2dpart, self evaluation; CPU times (in seconds) for computing the Helmholtz potential, gradient, and second derivatives on 1, 4, and 15 cores, respectively; the relative L_2 errors for self potential; complex-valued charges and dipoles randomly distributed on a unit circle, $k=20$, the precision parameter iprec $=$ 4 (12 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$
10000	0.25523E+00	0.80805E-01	0.72309E-01	0.53536E-14
100000	0.21407E+01	0.64547E+00	0.31610E+00	0.13693E-13
1000000	0.21107E+02	0.62151E+01	0.26586E+01	0.16785E-13

derivatives at all source and target points. The file `hfmm2dpart_driver.f` contains a sample driver for `hfmm2dparttarg`. It creates the same distribution of sources and targets, and sets the Helmholtz parameter to $k=20$. Sample test and timing drivers for the MATLAB routines can be found in the `matlab` directory in files: `test_lfmm2dpart_direct.m`, `test_hfmm2dpart_direct.m`, and `timings_lfmm2dpart.m`, `timings_hfmm2dpart.m`, respectively.

6.2 Sample drivers for FMMLIB3D

In the FMM3D/examples directory, the file `lfmm3dpart_driver.f` contains a sample driver for `lfmm3dparttarg`. It creates a random distribution of source points on the unit sphere centered at the origin and a random distribution of target points on a distinct unit sphere, centered at $(0,0,2)$. The code then computes the potential and field at all source and target points. The file `hfmm3dpart_driver.f` contains a sample driver for `hfmm3dparttarg`. It creates the same distribution of sources and targets, and sets the

Table 5: Laplace particle FMM in R^3, lfmm3dpart, self and target evaluation; CPU times (in seconds) for computing the Laplace potential and field on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; complex-valued charges and dipoles randomly distributed on a unit sphere, the precision parameter iprec = 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
10000	0.61539E+00	0.20607E+00	0.12182E+00	0.37863E-06	0.10149E-05
100000	0.57365E+01	0.16259E+01	0.65051E+00	0.65374E-06	0.94945E-06
1000000	0.55084E+02	0.15367E+02	0.58204E+01	0.70074E-06	0.11390E-05

Table 6: Helmholtz particle FMM in R^3, hfmm3dpart, self and target evaluation; CPU times (in seconds) for computing the Helmholtz potential and field on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; complex-valued charges and dipoles randomly distributed on a unit sphere, $k=1+.1i$, the precision parameter iprec = 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
10000	0.17493E+01	0.51668E+00	0.26894E+00	0.18719E-06	0.12647E-05
100000	0.16354E+02	0.48285E+01	0.19766E+01	0.28700E-05	0.80072E-06
1000000	0.17412E+03	0.50511E+02	0.18975E+02	0.23781E-07	0.10640E-05

Table 7: Laplace particle FMM in R^3, lfmm3dpart, self and target evaluation; CPU times (in seconds) for computing the Laplace potential on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; complex-valued charges randomly distributed on a unit sphere, the precision parameter iprec = 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
10000	0.35813E+00	0.12353E+00	0.80815E-01	0.67558E-07	0.25475E-06
100000	0.32408E+01	0.94223E+00	0.38952E+00	0.69363E-07	0.19598E-06
1000000	0.31128E+02	0.88007E+01	0.33091E+01	0.70526E-07	0.27832E-06

Table 8: Helmholtz particle FMM in R^3, hfmm3dpart, self and target evaluation; CPU times (in seconds) for computing the Helmholtz potential on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; complex-valued charges randomly distributed on a unit sphere, $k=1+.1i$, the precision parameter iprec $=$ 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
10000	0.68473E+00	0.22920E+00	0.15869E+00	0.17307E-06	0.17746E-06
100000	0.70423E+01	0.21784E+01	0.10148E+01	0.17364E-06	0.12472E-06
1000000	0.71567E+02	0.21520E+02	0.90982E+01	0.17417E-06	0.15986E-06

Table 9: Laplace layer FMM in R^3, lfmm3dtria, self and target evaluation; CPU times (in seconds) for computing the Laplace potential and field on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; flat triangulation of a unit sphere, complex-valued single and double layer, the precision parameter iprec $=$ 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
2880	0.15278E+01	0.44817E+00	0.20770E+00	0.60037E-06	0.21845E-04
11520	0.64024E+01	0.18415E+01	0.79300E+00	0.72089E-06	0.38454E-05

Table 10: Helmholtz layer FMM in R^3, hfmm3dtria, self and target evaluation; CPU times (in seconds) for computing the Helmholtz potential and field on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; flat triangulation of a unit sphere, complex-valued single and double layer, $k=1+.1i$, the precision parameter iprec $=$ 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
2880	0.29545E+01	0.81850E+00	0.34847E+00	0.27091E-05	0.14158E-05
11520	0.11593E+02	0.31844E+01	0.11267E+01	0.67729E-05	0.81338E-07

Table 11: Laplace layer FMM in R^3, lfmm3dtria, self and target evaluation; CPU times (in seconds) for computing the Laplace potential on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; flat triangulation of a unit sphere, complex-valued single layer, the precision parameter iprec $=$ 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
2880	0.89785E+00	0.27776E+00	0.16465E+00	0.24975E-07	0.50104E-05
11520	0.36576E+01	0.10650E+01	0.45428E+00	0.21279E-07	0.16857E-05

Helmholtz parameter to $k=1+0.1i$. The file `lfmm3dtria_driver.f` contains a sample driver for `lfmm3dtriatarg`. The file `hfmm3dtria_driver.f` contains a sample driver for `hfmm3dtriatarg` and `hfmm3dtriampftarg`.

6.3 Sample drivers for STFMMLIB3D

In the STFMM3D/examples directory, the file `stfmm3dpart_dr.f` contains a sample driver for `stfmm3dparttarg`. It creates a random distribution of source points on the unit sphere centered at the origin and a random distribution of target points on a distinct

Table 12: Helmholtz layer FMM in R^3, hfmm3dtria, self and target evaluation; CPU times (in seconds) for computing the Helmholtz potential on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; flat triangulation of a unit sphere, complex-valued single layer, $k=1+.1i$, the precision parameter iprec $=$ 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
2880	0.17459E+01	0.49066E+00	0.25870E+00	0.56620E-07	0.39769E-07
11520	0.67373E+01	0.18582E+01	0.70807E+00	0.66320E-07	0.34791E-07

Table 13: Stokes particle FMM in R^3, stfmm3dpart, self and target evaluation; CPU times (in seconds) for computing the Stokes potential (i.e., the velocity field and pressure), and the velocity gradient on 1, 4, and 15 cores, respectively; the relative L_2 errors for self and target potentials; real-valued single forces randomly distributed on a unit sphere, the precision parameter iprec $=$ 1 (3 digits).

N	$T_{(1\text{ core})}$	$T_{(4\text{ cores})}$	$T_{(15\text{ cores})}$	$E_{(\text{pot})}$	$E_{(\text{pottarg})}$
10000	0.22757E+01	0.90660E+00	0.65632E+00	0.61481E-06	0.94692E-05
100000	0.17700E+02	0.70828E+01	0.51069E+01	0.14095E-05	0.71472E-05
1000000	0.20701E+03	0.85321E+02	0.60980E+02	0.14330E-05	0.39816E-05

unit sphere, centered at $(1,0,-2)$. The code then computes the velocity field, pressure and velocity gradient at all source and target points.

Acknowledgments

The work of Zydrunas Gimbutas was supported in part by the Office of the Assistant Secretary of Defense for Research and Engineering, AFOSR under NSSEFF Program Award FA9550-10-1-0180, by the National Science Foundation under grant DMS-0934733, and by a research grant from Meyer Sound Laboratories. Contributions by staff of NIST, an agency of the U.S. Government, are not subject to copyright within the United States. Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST.

The work of Leslie Greengard was supported in part by the Office of the Assistant Secretary of Defense for Research and Engineering and AFOSR under NSSEFF Program Award FA9550-10-1-0180, by the National Science Foundation under grant DMS-0934733, and by the Applied Mathematical Sciences Program of the U.S. Department of Energy under Contract DEFGO288ER25053.

References

[1] J. Carrier, L. Greengard, and V. Rokhlin, A Fast Adaptive Multipole Algorithm for Particle Simulations, SIAM J. Sci. and Stat. Comput., 9 (1988), 669–686.

[2] J. B. Drake, P. Worley and E. D'Azevedo, Spherical Harmonic Transform Algorithms, ACM Trans. Math. Softw., 35 (2008), 111-126.

[3] H. Cheng, L. Greengard and V. Rokhlin, A Fast Adaptive Multipole Algorithm in Three Dimensions, J. Comput. Phys., 155 (1999), 468–498.
[4] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. Greengard, F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao, A wideband Fast Multipole Method for the Helmholtz equation in three dimensions, J. Comput. Phys., 216 (2006), 300–325.
[5] W. C. Chew, Recurrence relations for three-dimensional scalar addition theorem, J. Electromagnet. Waves Appl. 6 (1992), 133–142.
[6] W. Crutchfield, Z. Gimbutas, L. Greengard, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao, Remarks on the implementation of the wideband FMM for the Helmholtz equation in two dimensions, Contemporary Mathematics, 408 (2006), 99-110.
[7] E. Darve and P. Havé, Efficient Fast Multipole Method for Low-Frequency Scattering, J. Comput. Phys., 197 (2004), 341–363.
[8] J. Happel and H. Brenner (1973). Low Reynolds Number Hydrodynamics, 2nd ed., Noordhoff International Publishing, Leyden, Netherlands.
[9] Z. Gimbutas and L. Greengard (2012). FMMLIB2 - Fast Multipole Method (FMM) library for the evaluation of potential fields governed by the Laplace and Helmholtz equations in R^2, `http://www.cims.nyu.edu/cmcl/fmm2dlib/fmm2dlib.html`.
[10] Z. Gimbutas and L. Greengard (2012). FMMLIB3 - Fast Multipole Method (FMM) library for the evaluation of potential fields governed by the Laplace and Helmholtz equations in R^3, `http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html`.
[11] Z. Gimbutas and L. Greengard (2012). STFMMLIB3 - Fast Multipole Method (FMM) library for the evaluation of potential fields governed by the Stokes equations in R^3, `http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html`.
[12] Z. Gimbutas and L. Greengard, Translation of multipole expansions via projection, in preparation.
[13] L. Greengard and J. Huang, A New Version of the Fast Multipole Method for Screened Coulomb Interactions in Three Dimensions, J. Comput. Phys., 180 (2002), 642–658.
[14] L. Greengard and V. Rokhlin, A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions, Acta Numerica (1997), 229–269.
[15] N. A. Gumerov and R. Duraiswami, Recursions for the computation of multipole translation and rotation coefficients for the 3-D Helmholtz equation, SIAM J. Sci. Comput., 25 (2003), 1344–1381.
[16] C. Pozrikidis (1992). Boundary integral and singularity methods for linearized viscous flow. Cambridge University Press, Cambridge.
[17] A.-K. Tornberg and L. Greengard, A fast multipole method for the three-dimensional Stokes equations. J. Comput. Phys., 227 (2008), 1613–1619.
[18] H. Wang, T. Lei, J. Li, J. Huang, Z. Yao, A parallel fast multipole accelerated integral equation scheme for 3D Stokes equations. Int. J. Num. Meth. Eng., 70 (2007), 812–839.
[19] C. A. White and M. Head-Gordon, Rotating around the quartic angular momentum barrier in fast multipole method calculations, J. Chem. Phys. 105 (1996), 5061–5067.
[20] Y. Xu, Electromagnetic scattering by an aggregate of spheres, Appl. Opt., 34 (1995), 4573–4588.