Computer Physics Communications 181 (2010) 2086–2090

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

A Wideband Fast Multipole Method for the two-dimensional complex Helmholtz equation ☆

Min Hyung Cho *, Wei Cai

Department of Mathematics and Statistics, The University of North Carolina at Charlotte, 9201 University City Blvd., Charlotte, NC 28223-0001, United States

ARTICLE INFO

Article history:
Received 8 March 2010
Received in revised form 1 September 2010
Accepted 15 September 2010

Keywords:
Wideband Fast Multipole Method
Helmholtz equation
Fast solver

ABSTRACT

A Wideband Fast Multipole Method (FMM) for the 2D Helmholtz equation is presented. It can evaluate the interactions between N particles governed by the fundamental solution of 2D complex Helmholtz equation in a fast manner for a wide range of complex wave number k, which was not easy with the original FMM due to the instability of the diagonalized conversion operator. This paper includes the description of theoretical backgrounds, the FMM algorithm, software structures, and some test runs.

Program summary

Program title: 2D-WFMM
Catalogue identifier: AEHI_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEHI_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 4636
No. of bytes in distributed program, including test data, etc.: 82 582
Distribution format: tar.gz
Programming language: C
Computer: Any
Operating system: Any operating system with gcc version 4.2 or newer
Has the code been vectorized or parallelized?: Multi-core processors with shared memory
RAM: Depending on the number of particles N and the wave number k
Classification: 4.8, 4.12
External routines: OpenMP (http://openmp.org/wp/)
Nature of problem: Evaluate interaction between N particles governed by the fundamental solution of 2D Helmholtz equation with complex k.
Solution method: Multilevel Fast Multipole Algorithm in a hierarchical quad-tree structure with cutoff level which combines low frequency method and high frequency method.
Running time: Depending on the number of particles N, wave number k, and number of cores in CPU. CPU time increases as N log N.

1. Introduction

The Fast Multipole Method (FMM) is a numerical method based on a hierarchical tree structure [1,2] for the fast evaluation of N particle interactions governed by the fundamental solutions of Laplace's equation [3–5], Helmholtz equation [6–9], and the modified Helmholtz equation [10–12]. From a numerical linear algebra point of view, FMM can accelerate the multiplication of a dense matrix and a vector from $O(N^2)$ to $O(N)$ for Laplace's and the modified Helmholtz equation, and to $O(N \log N)$ in the case of the Helmholtz equation. Therefore, it can be used as a building block of the Krylov subspace based iterative matrix solver [13] for numerical solutions of integral equations.

The original FMM for the 2D Helmholtz equation uses far field forms of partial wave expansions and their diagonalized translation and conversion operators [6]. However, because of the asymptotic behavior of the high order Hankel function for small arguments, the conversion process from multipole expansion to local expan-

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail addresses: mhcho@uncc.edu (M.H. Cho), wcai@uncc.edu (W. Cai).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2010.09.010

sion had an instability issue for small values of $k \cdot R$, where R is a size of box or cube and k is the wave number. As a result, the original FMM for the Helmholtz equation could not be used for a certain range of small k or high level of mesh refinement. Recently, a 2D and 3D Wideband FMM for the Helmholtz equation were introduced [8,14]. In this paper, a Wideband FMM for the 2D complex Helmholtz equation is briefly introduced, and the code for a multi-core processor with OpenMP [15] is provided.

The quad-tree structure is divided into the low frequency part and high frequency part based on the value of $k \cdot R$. In the low frequency part, the original partial wave expansions and non-diagonalized translation and conversion operators, which can be derived from the addition theorem for Bessel functions [16,17], are directly used. Above the cutoff level, which divides low and high frequency, partial wave expansions are converted into their far field forms, and the original FMM is used (see Fig. 1). In this paper, the original FMM will be referred to as the High Frequency FMM (HF-FMM) and the FMM with partial wave expansions and non-diagonalized operators will be denoted as the Low Frequency FMM (LF-FMM) for the obvious reason.

In the next section, brief theoretical backgrounds are explained. Then, in the following section, a description of the algorithm and program structures are presented. In Section 4, four test run results for the real and complex k with 90,000 and 490,000 sources are tabulated and discussed. Finally, the paper concludes with a summary and future works.

2. Theoretical background

Rigorous theoretical backgrounds and remarks can be found in Ref. [8]. Therefore, in this paper, a short mathematical introduction is presented. Assume N particles are distributed at $X_i = (x_i, y_i)$ with strength q_i, $i = 1, 2, \ldots, N$. Then, the interactions between all the particles governed by the fundamental solution of the 2D Helmholtz equation can be written as

$$\Phi(X_i) = \sum_{j=1, j\neq i}^{N} q_j H_0^{(1)}\big(k\|X_i - X_j\|\big), \quad i = 1, 2, \ldots, N, \tag{1}$$

where $H_0^{(1)}$ denotes the 0th order Hankel function and k represents the wave number. The FMM for the 2D Helmholtz equation is a numerical scheme, which calculates Eq. (1) in a fast manner with a hierarchical quad-tree data structure by utilizing two kinds of partial wave expansions,

$$\psi(X) = \sum_{m=-\infty}^{\infty} \beta_m H_m(k\rho) e^{im\theta} \tag{2}$$

and

$$\phi(X) = \sum_{m=-\infty}^{\infty} \alpha_m J_m(k\rho) e^{im\theta}, \tag{3}$$

where $X = (x, y)$, ρ is the distance between X and the center of expansion, θ is the angle between X and x-axis, and H_m and J_m denote the mth order Hankel and Bessel function, respectively. Eqs. (2) and (3) will be referred as the H- and J-expansion, and they are equivalent to multipole and local expansions for the case of Laplace's equation, respectively. The H- and J-expansions can be derived from the addition theorem for Bessel function [16]. The HF-FMM, which uses diagonalized translation and conversion operators with far field forms of wave expansions, fails to converge when k is very small or level of refinement is high due to a divergent property of Hankel function in the conversion operator.

In detail, far field forms of H- and J-expansions are defined by

$$F(\theta) = \sum_{m=-\infty}^{\infty} \beta_m e^{-i(m\pi/2)} e^{im\theta}, \tag{4}$$

$$G(\theta) = \sum_{m=-\infty}^{\infty} \alpha_m e^{-i(m\pi/2)} e^{im\theta}, \tag{5}$$

respectively, where $\{\beta_m\}_{m=-\infty}^{+\infty}$ and $\{\alpha_m\}_{m=-\infty}^{+\infty}$ are coefficients of the H- and J-expansions from Eqs. (2) and (3). Then, $F(\theta)$ are converted to $G(\theta)$ via a diagonalized operator [6]

$$G_{c_3}(\theta) = \nu_n(\theta) \cdot F_{c_1}(\theta), \tag{6}$$

where

$$\nu_n(\theta) = \sum_{m=-n}^{n} e^{im(\theta+\theta_{13}-\pi)} H_m(k\rho_{13}), \tag{7}$$

and a subscript in $F(\theta)$ and $G(\theta)$ denotes the center of the expansions, ρ_{13} is a distance between two centers defined by $|c_3 - c_1|$, and θ_{13} is angle between x-axis and ρ_{13}. However, the diagonalized conversion operator $\nu_n(\theta)$ diverges quickly when the order of the Hankel function is larger than its argument ($m > k\rho_{13}$) because of the asymptotic behavior of the Hankel function [16], namely,

$$\lim_{m\to\infty} Y_m(z)\left(\frac{ez}{2m}\right)^m \frac{\sqrt{\pi m}}{\sqrt{2}} = -1, \tag{8}$$

where $H_m(z) = J_m(z) + iY_m(z)$. In order to overcome the divergence problem for a small k, the tree is divided into two parts with a cutoff level based on the quantity of $k \cdot R$, where R is the size of a box in each level of the tree. In the tree level with $k \cdot R < 4/e = 1.471518$ (boxes below the cutoff level), LF-FMM, which uses the H- and J-expansions and non-diagonalized conversion operator based on the addition theorem [16],

$$\alpha_m = \sum_{j=-n}^{n} e^{-ij(\theta_{13}-\pi)} \beta_{m-j} H_j(k\rho_{13}), \tag{9}$$

are directly used to avoid divergence of the conversion operator. Then, at and above the cutoff level, coefficients of the H- and J-expansions are converted into far field forms using Eq. (4) and (5), and the regular HF-FMM is used for the boxes above the cutoff level. In the actual implementation, $k \cdot R = 1.5$ is used for the determination of the cutoff level to be safe.

3. Overview of the software structure

3.1. Algorithm

A good multilevel FMM algorithm description is presented in Ref. [4]. The code is written in terms of each step described in the reference as strictly as possible. Steps 1 and 2 are the upward pass of the algorithm. In step 1, the computational box is made by enclosing all the particles, and the tree-depth is determined depending on the number of particles. Then, the quad-tree is constructed by subdividing the computational box into four equal boxes recursively until the depth of the tree is reached. The cutoff level is now obtained by examining the size of box and the wave number k. In step 2, if the cutoff level is smaller than the depth of the tree, coefficients of H-expansions are constructed at the finest level of the tree, and these are translated up to the cutoff level. Then, at the cutoff level, all the coefficients of H-expansions are converted into their far field forms $F(\theta)$ via Eq. (4), and $F(\theta)$ are translated to their parents box until it reaches the top of the tree. Steps 3 and 4 are the downward pass of the algorithm. From the tree level 1,

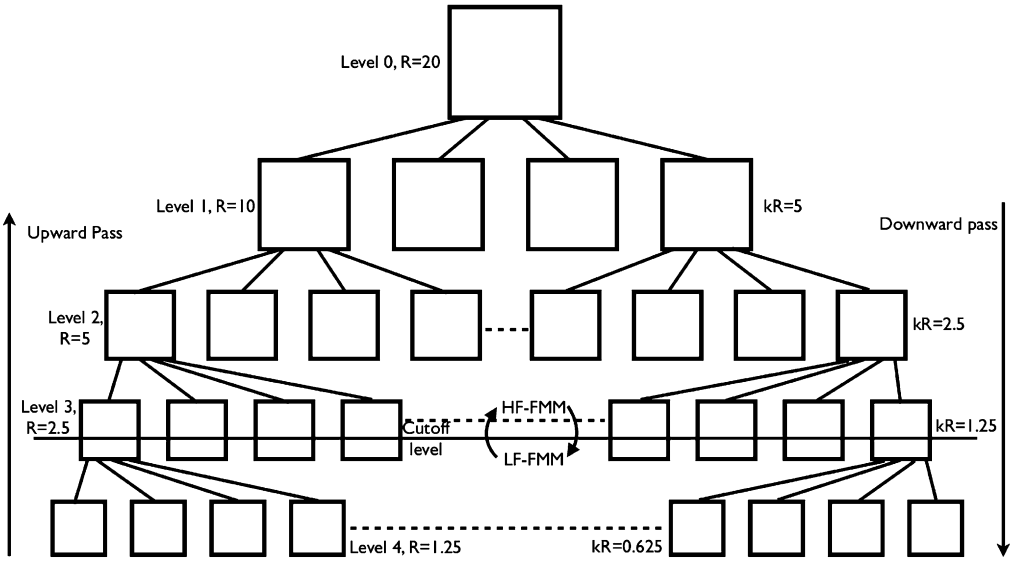

Fig. 1. Quad-tree structure for a numerical example with $N = 90{,}000$, tree depth $= 4$, and $k = 0.5$ in a $[-10, 10]^2$ box.

all the far field forms $F(\theta)$ in well-separated boxes, which are descendants of close neighbors of the current box's parent and do not share an edge or corner of the current box, are converted into far field forms $G(\theta)$ of J-expansion using Eq. (6). Then, $G(\theta)$ are shifted to their four children up to the cutoff level. Again, at the cutoff level, far field forms $G(\theta)$ are converted into coefficients of J-expansions by integrating Eq. (5). Now, in the tree level equal or higher than the cutoff level, coefficients of H-expansions in well-separated boxes are converted into coefficients of J-expansions until it reaches the bottom of the tree. Step 5 calculates the direct interactions between close neighbor boxes. In step 6, J-expansions are evaluated with Eq. (3) and added with direct interactions calculated from step 5. At last, data are sorted as an order of the input and return the approximated interaction between input particles.

If the cutoff level is higher than the depth of the tree, far field forms of H-expansions $F(\theta)$ are constructed at the finest level of the tree and translated up to the top of the tree, and $F(\theta)$ in well-separated boxes are converted into $G(\theta)$ in the process of the downward pass. Then, $G(\theta)$ is evaluated at each particle and added with direct interactions between close neighbor boxes.

3.2. Description of the individual software components

The *wfmm.zip* includes 15 files: *testrun.c*, *wfmm.c*, *complex.c*, *hankel.c*, *sort.c*, *fmmexact.c*, *fmmmesh.c*, *fmmqueue.c*, *fmmstep2.c*, *fmmstep3.c*, *fmmstep4.c*, *fmmstep5.c*, *fmmstep6.c*, *fmmoperator.c*, and *readme*. The *testrun.c* file includes a test run of wideband FMM, which can produce four tables in Section 4 by varying the wave number k and number of sources N. The *wfmm.c* file is the main FMM file and it takes a particle distribution and returns an output. The *complex.c* is a file that contains all the necessary complex operations for FMM. The *hankel.c* is a file for the special functions including integer order of the first kind Bessel function, the second kind Bessel function, and the Hankel function with real and complex argument [18]. All the Fortran codes in the textbook are converted into C and modified accordingly. Copyright of the Hankel function code remains with Ref. [18]. The diagonalized translation and conversion operators for HF-FMM are implemented in a *fmmoperator.c* file. The *fmmqueue.c* file has the queue data structure and operations for a tree searching algorithm. The *fmmmesh.c* is a file for generating a computational box and quad-tree. The *fmmstep2.c* file executes the upward pass in the FMM algorithm. The downward pass is implemented in the *fmmstep3.c* and *fmmstep4.c* files. In the *fmmstep5.c* file, direct interactions between close neighbors are evaluated. The *fmmstep6.c* evaluates fields at each point and sorts the data for a proper output order. The *readme* file describes how to compile and run a test case. As a note, the quick sort was good enough for sorting because the data in the quad-tree are partially ordered.

Finally, in order to take advantage of multi-core processors, OpenMP [15] is used to create multi-thread. OpenMP is a complier directive and is an easy way of creating multi-thread. Most loops in the code are parallelized. For example, either conversions from H-expansion in well-separated boxes to J-expansion or F in well-separated boxes to G, which are most time consuming part of the algorithm, are parallelized with *#pragma omp parallel for* complier directive by carefully identifying shared and private variables in each thread. The code is written to take advantage of eight cores in a CPU as a default. However, it can be easily modified by changing the number of thread parameters NT in the beginning of the code. Fig. 2(b) shows the speed up as a function of number of cores. As a final note on parallelization, OpenMP can be used in conjunction with the Message Passing Interface (MPI). Therefore, the code can be efficiently used in a cluster with multi-core processors.

3.3. Installation instructions

This code has all the necessary components, therefore, it does not require any external library or installation. Simply unzip the file named *wfmm.zip* and compile testrun.c with ">>gcc testrun.c -lm -O3 -fopenmp -otestwfmm.out". Then, it will produce the *testwfmm.out* file. Then, ">> .\testwfmm.out" will run one of the test cases given in the next section of the paper, and the outputs (k, depth of the tree, cutoff level, error, CPU time in seconds for FMM, and estimated direct calculation) will be saved in the *fmm_data.txt* file. If it is used in other code, this code can serve as a black box. Give the x and y coordinates, strength of particle q, and number of particle N, wave number k, and storage for outputs *solution* (x, y, q, N, k, and *solution*). Then, the depth of the tree and the cutoff level are automatically determined in the FMM, and it will return interactions between the given particle distribution in the *solution*. In the main file, include *wfmm.c* and call FMM as "wfmm(x, y, q, k, N, ind, solution, ex_solution)". In actual use of the code, the calculation of direct interactions is not practical. Thus, it should be commented out from the *wfmm.c* file.

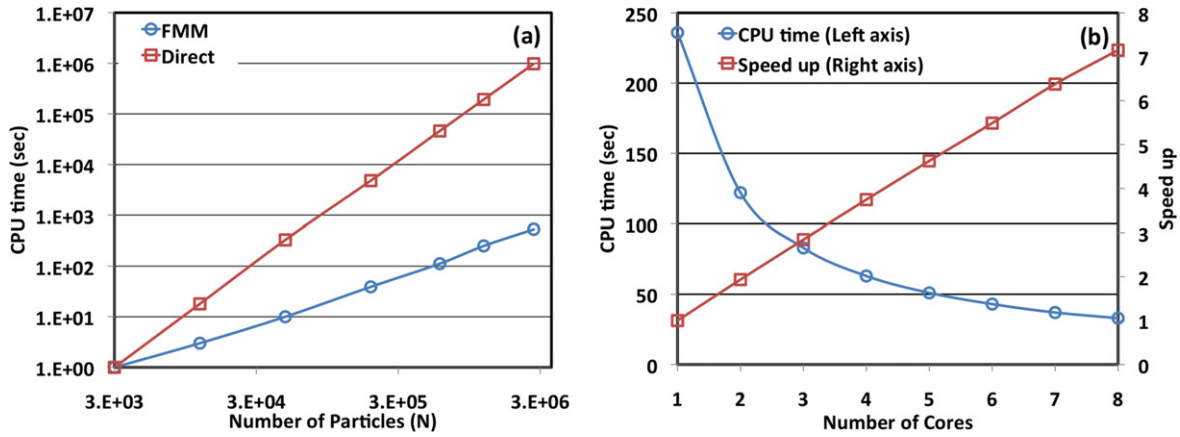

Fig. 2. (a) CPU time in seconds for FMM (○) and direct calculation (□) for $k = 1.0 + 0.1i$ as a function of N in the log–log scale. (b) CPU time (○) and speed up (□) as a function of number of cores for $k = 1.0 + 0.1i$ and $N = 90{,}000$.

Table 1
Wideband FMM and direct summation with real k.

$N = 300^2$ in a $[-10, 10]^2$ box, tree depth = 4				
k	Cutoff level	Error	FMM (sec)	Direct (sec)
10^{-10}	1	1.52E−07	9	270
10^{-5}	1	3.37E−07	10	270
0.1	1	5.75E−08	11	270
0.2	2	2.63E−07	11	270
0.3	2	7.08E−07	11	360
0.4	3	1.05E−06	11	270
0.5	3	3.71E−07	11	360
0.6	3	1.08E−06	11	360
0.7	4	1.06E−06	11	270
0.8	4	7.54E−07	10	270
0.9	4	1.07E−06	10	360
1.0	4	5.04E−07	11	270
5.0	4	2.28E−07	19	360
10.0	4	4.97E−08	31	270

Table 2
Wideband FMM and direct summation with complex k.

$N = 300^2$ in a $[-10, 10]^2$ box, tree depth = 4				
k	Cutoff level	Error	FMM (sec)	Direct (sec)
$10^{-10} + 0.1i$	1	1.30E−07	21	720
$10^{-5} + 0.1i$	1	1.30E−07	22	810
$0.1 + 0.1i$	1	7.46E−08	23	810
$0.2 + 0.1i$	2	5.57E−07	25	990
$0.3 + 0.1i$	3	3.34E−06	25	1170
$0.4 + 0.1i$	3	1.48E−06	28	1260
$0.5 + 0.1i$	3	6.88E−07	29	1440
$0.6 + 0.1i$	4	4.33E−06	28	1440
$0.7 + 0.1i$	4	1.54E−06	30	1440
$0.8 + 0.1i$	4	1.43E−07	31	1530
$0.9 + 0.1i$	4	1.32E−07	32	1530
$1.0 + 0.1i$	4	5.98E−07	34	1530
$5.0 + 0.1i$	4	2.02E−07	124	1710
$10.0 + 0.1i$	4	5.59E−08	244	1800

4. Test run description and discussions

Four tables are presented to demonstrate the performance of the Wideband FMM and the transition from LF-FMM to HF-FMM (cutoff level from 1 to the tree depth). First, the Wideband FMM is tested with $N = 300 \times 300 = 90{,}000$ uniform distribution in a $[-10, 10]^2$ box centered at the origin with real k (Table 1) and complex k (Table 2). In both cases, total level of refinement is determined as 4, and the cutoff level is varied from 1 to 4 depending on $k \cdot R$. Note that the wideband FMM becomes HF-FMM and LF-FMM when the cutoff level is set to the same as the tree depth and 1, respectively. In addition, $N = 700 \times 700 = 490{,}000$ in a $[-10, 10]^2$ box with real and complex k is used to test finer refinement of the computational box. In this case, the depth of the tree is set to 6 and computational results are displayed in Tables 3 and 4. The errors are calculated for the first 1000 sources with

Table 3
Wideband FMM and direct summation with real k and large N.

$N = 700^2$ in a $[-10, 10]^2$ box, tree depth = 6				
k	Cutoff level	Error	FMM (sec)	Direct (sec)
10^{-10}	1	1.71E−07	38	6860
10^{-5}	1	3.81E−07	38	6860
0.1	1	6.08E−08	62	6860
0.2	2	2.70E−07	62	7350
0.3	2	7.76E−07	68	8330
0.4	3	1.15E−06	63	8820
0.5	3	4.19E−07	68	8820
0.6	3	1.23E−06	68	9310
0.7	4	1.34E−06	63	9310
0.8	4	1.10E−06	62	9310
0.9	4	1.78E−06	63	9310
1.0	4	6.00E−07	68	9310
2.0	5	4.80E−07	68	9310
5.0	6	6.64E−07	65	9310
10.0	6	2.63E−07	101	9800
50.0	6	2.59E−08	1153	9800

Table 4
Wideband FMM and direct summation with complex k and large N.

$N = 700^2$ in a $[-10, 10]^2$ box, tree depth = 6				
k	Cutoff level	Error	FMM (sec)	Direct (sec)
$10^{-10} + 0.1i$	1	1.39E−07	76	23,520
$10^{-5} + 0.1i$	1	1.39E−07	78	23,030
$0.1 + 0.1i$	1	8.11E−08	85	24,990
$0.2 + 0.1i$	2	5.82E−07	87	29,890
$0.3 + 0.1i$	3	3.66E−06	71	34,790
$0.4 + 0.1i$	3	1.63E−06	92	38,710
$0.5 + 0.1i$	3	7.83E−07	101	41,650
$0.6 + 0.1i$	4	4.39E−06	78	43,120
$0.7 + 0.1i$	4	1.98E−06	99	44,100
$0.8 + 0.1i$	4	2.05E−06	101	44,100
$0.9 + 0.1i$	4	2.26E−06	104	45,570
$1.0 + 0.1i$	4	7.29E−07	112	46,060
$2.0 + 0.1i$	5	8.66E−07	129	46,060
$5.0 + 0.1i$	6	6.30E−07	221	47,040
$10.0 + 0.1i$	6	3.29E−07	597	48,020
$50.0 + 0.1i$	6	2.48E−08	8526	47,530

$$Error = \left(\frac{\sum_{j=1}^{1000} |\Phi_{exact}(X_j) - \Phi_{FMM}(X_j)|^2}{\sum_{j=1}^{1000} |\Phi_{exact}(X_j)|^2} \right)^{\frac{1}{2}} \tag{10}$$

in all computations. The CPU time for direct calculation is estimated based on the CPU time for the first 1000 calculations. All the test runs are conducted with gcc version 4.4.1 in a machine operated by the Fedora release 11 (two 3.00 GHz quad-core Xeon processors and 32 GB memory).

In the first two numerical examples with $N = 90{,}000$ in a $[-10, 10]^2$ box, the box size R in level 1, 2, 3, and 4 are 10.0, 5.0, 2.5, and 1.25, respectively. For $k = 0.1$, $k \cdot R$ in level 1, 2, 3, and 4 are 1.0, 0.5, 0.25, and 0.125, respectively. The cutoff level is set to 1 and the solutions are found to be convergent (LF-FMM). For $k = 0.5$, $k \cdot R$ in each level is 5.0 (level 1), 2.5 (level 2), 1.25 (level 3), and 0.625 (level 4), cutoff level 3 or smaller (level 3 has the box with $k \cdot R = 1.25 < 4/e = 1.471518$) gives a convergent solution.

In detail, when $k = 0.5$ (see Fig. 1), the cutoff level is 3 and the depth of the tree is 4. Since the depth of the tree is larger than the cutoff level, the H-expansions are formed at level 4 and translated into the parent box in level 3 using non-diagonalized translation operators and added with other H-expansions translated from other boxes in level 4 according to the algorithm presented in Section 3.1. Now, at level 3, all the H-expansions are converted into far field forms $F(\theta)$ of H-expansion and translated into the parent box in level 2 and 1 subsequently with diagonalized translation operators, and it completes the upward pass of FMM. For the downward pass, $F(\theta)$ in well-separated boxes in level 1 are converted into far field forms $G(\theta)$ of J-expansion and shifted into the center of its children boxes in level 2 with diagonalized translation operators. Similarly, $F(\theta)$ in well-separated boxes in level 2 are converted into $G(\theta)$ with diagonalized conversion operators and shifted into its children boxes in level 3. At level 3 or cutoff level, $G(\theta)$ are converted into coefficients of J-expansions. At the same time, H-expansions in well-separated boxes in level 3 are converted into J-expansions, and it is shifted into level 4. Finally, all the J-expansions at level 4 are ready. Now, all the solutions can be obtained by evaluating J-expansions at each data point and adding the direct interactions with sources in close neighbors.

The third and fourth numerical examples show performances of the real and complex wideband FMM for 490,000 sources in a $[-10, 10]^2$ box with a finer refinement of the tree in Tables 3 and 4.

A few remarks have to be made on some data in the tables. First, a large variation in the direct calculation time for complex k can be explained by the numerical algorithm used for the special function. The zeroth order complex Bessel function is implemented for small and large arguments with two different series expansions. In the actual implementation, the magnitude of the argument is divided at 12 [18]. As a result, the direct calculation time, which uses only the zeroth order Hankel function, shows some variations as k varies because of the transition between the two algorithms. But the computation time for FMM shows almost no effects because direct interactions are calculated with very close sources within the box itself or close neighbors. If a more effective numerical scheme for the special function is available, then it can replace the Hankel function subroutine supplied in this code. Secondly, the computation time for FMM is slightly decreased between $k = 0.6$ and 0.7 in Table 3 and $k = 0.5 + 0.1i$ and $0.6 + 0.1i$ in Table 4 because of cutoff level transition. Finally, the error is slightly increased near the transition of the cutoff level (for example, between $k = 0.3$ and 0.4 in Table 3 where the cutoff level is changed from 2 to 3). This is obvious from the definition of the cutoff level.

In Fig. 2(a), the CPU time in seconds for FMM (circle) and direct calculation (square) are plotted in the log–log scale as a function of number of particles ($N = 900$–$2{,}250{,}000$ in a $[-10, 10]^2$ box with a fixed $k = 1.0 + 0.1i$). Almost linear growth for FMM is observed and quadratic growth for the direct calculation is shown. Fig. 2(b) displays the CPU time in seconds (circle, left axis) and the speed up compared to the single core CPU time (square, right axis) as a function of number of cores from 1 to 8 for $N = 90{,}000$ in a $[-10, 10]^2$ box with $k = 1.0 + 0.1i$.

5. Conclusions

A Wideband FMM program for the 2D Helmholtz equation for the real and complex wave number k is introduced. The LF-FMM and HF-FMM are seamlessly combined, and it extends the applicability of FMM for low and high frequency at the same time. So it will have numerous real world applications in computational electromagnetics and can be used as a building block of other fast solvers. Therefore, it will benefit physicists and engineers [19,20]. As a final note, the plane wave expansions suggested for the acceleration of the LF-FMM [21] is not implemented in this code because the speed up is negligible and makes the code more complex. Also, the special function implementation can be further optimized for better efficiency.

Acknowledgements

This work is supported by the US Department of Energy (grant numbers: DEFG0205ER25678). Authors would like acknowledge Prof. Jingfang Huang at UNC Chapel Hill for valuable comments and Mr. Mark Hamrick for the computing support.

References

[1] A.W. Appel, SIAM J. Sci. Statist. Comput. 6 (1985) 85.
[2] J. Barnes, P. Hut, Nature 324 (1986) 446.
[3] L. Greengard, V. Rokhlin, J. Comput. Phys. 73 (1987) 325.
[4] L. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems, MIT Press, Cambridge, 1988.
[5] L. Greengard, V. Rokhlin, Acta Numer. (1997) 229.
[6] V. Rokhlin, J. Comput. Phys. 86 (1990) 414.
[7] V. Rokhlin, Appl. Comput. Harmonic Anal. 1 (1993) 82.
[8] W. Crutchfield, Z. Gimbutas, L. Greengard, J. Huang, V. Rokhlin, N. Yarvin, J. Zhao, Contemp. Math. 408 (2006) 99.
[9] B. Engquist, L. Ying, Commun. Math. Sci. 7 (2009) 327.
[10] L. Greengard, J. Huang, J. Comput. Phys. 180 (2002) 642–658.
[11] H. Chen, J. Huang, T.J. Leiterman, J. Comput. Phys. 211 (2006) 616–637.
[12] J. Huang, J. Jia, B. Zhang, Comput. Phys. Commun. 180 (2009) 2331.
[13] Yousef Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, 1996.
[14] H. Cheng, W.Y. Crutchfield, Z. Gimbutas, L. Greengard, J.F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, J. Zhao, J. Comput. Phys. 216 (2006) 300.
[15] OpenMP, http://openmp.org/wp/.
[16] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, Dover, New York, 1970.
[17] J.A. Stratton, Electromagnetic Theory, McGraw–Hill, New York, 1941.
[18] S. Zhang, J.M. Jin, Computation of Special Functions, John Wiley & Sons, New York, 1996.
[19] W. Cai, Adv. Comput. Math. 16 (2002) 157.
[20] T. Yu, W. Cai, Commun. Comput. Phys. 1 (2006) 229.
[21] L. Greengard, J. Huang, V. Rokhlin, S. Wandzura, IEEE Comput. Sci. Eng. (July–Sep. 1998) 32.