Fast Multipole Method as a Matrix-Free Hierarchical Low-Rank Approximation

Rio Yokota, Huda Ibeid, David Keyes

Abstract There has been a large increase in the amount of work on hierarchical low-rank approximation methods, where the interest is shared by multiple communities that previously did not intersect. This objective of this article is two-fold; to provide a thorough review of the recent advancements in this field from both analytical and algebraic perspectives, and to present a comparative benchmark of two highly optimized implementations of contrasting methods for some simple yet representative test cases. The first half of this paper has the form of a survey paper, to achieve the former objective. We categorize the recent advances in this field from the perspective of compute-memory tradeoff, which has not been considered in much detail in this area. Benchmark tests reveal that there is a large difference in the memory consumption and performance between the different methods.

1 Introduction

The fast multipole method (FMM) was originally developed as an algorithm to bring down the $\mathscr{O}(N^2)$ complexity of the direct N-body problem to $\mathscr{O}(N)$ by approximating the hierarchically decomposed far field with multipole/local expansions. In its original form, the applicability of FMM is limited to problems that have a Green's function solution, for which the multipole/local expansions can be calculated analytically. Their function is also limited to matrix-vector multiplications, in contrast to the algebraic variants that can perform matrix-matrix multiplication and factorizations. However, these restrictions no longer apply to the FMM since the kernel

Rio Yokota
Tokyo Institute of Technology, 2-12-1 O-okayama Meguro-ku, Tokyo, Japan, e-mail: rioyokota@gsic.titech.ac.jp

Huda Ibeid, David Keyes
King Abdullah University of Science and Technology, 4700 KAUST, Thuwal, Saudi Arabia, e-mail: huda.ibeid@kaust.edu.sa,david.keyes@kaust.edu.sa

independent FMM [103] does not require a Green's function, and inverse FMM [2] can be used as the inverse operator instead of the forward mat-vec. Therefore the FMM can be used for a wide range of scientific applications, which can be broadly classified into elliptic partial differential equations (PDE) and kernel summation. Integral form of elliptic PDEs can be further categorized into boundary integrals for homogeneous problems, discrete volume integrals, and continuous volume integrals.

Scientific applications of FMM for boundary integrals include acoustics [97, 59], biomolecular electrostatics [105], electromagnetics [34, 42], fluid dynamics for Euler [96] and Stokes [88] flows, geomechanics [92], and seismology [22, 95]. Application areas of FMM for discrete volume integrals are astrophysics [14], Brownian dynamics [75], classical molecular dynamics [84], density functional theory [90], vortex dynamics [106], and force directed graph layout[107]. FMM for continuous volume integrals have been used to solve Schrödinger [108] and Stokes [79] equations. More generalized forms of FMM can be used as fast kernel summation for Bayesian inversion [3], Kalman filtering [74], Machine learning [49, 72], and radial basis function interpolation [54].

All of these applications have in common the key feature that they are global problems where the calculation at every location depends on the values everywhere else. Elliptic PDEs that represent a state of equilibrium, many iterations with global inner products for their solution, dense matrices in boundary integral problems, all-to-all interaction in N-body problems, and kernel summations with global support are all different manifestations of the same source of global data dependency. Due to this global data dependency, their concurrent execution on future computer architectures with heterogeneous and deep memory hierarchy is one of the main challenges of exascale computing. For global problems that require uniform resolution, FFT is often the method of choice, despite its suboptimal communication costs. The methods we describe here have an advantage for global problems that require non-uniform resolution. For such non-uniform global problems multigrid methods are known to do quite well. Whether the reduced synchronization and increased arithmetic intensity of the FMM will become advantageous compared to multigrid on future architectures is something that is yet to be determined.

Many of the original FMM researchers have now moved on to develop algebraic variants of FMM, such as $\mathscr{H}$-matrix [55], $\mathscr{H}^2$-matrix [56], hierarchically semiseprable (HSS) [25], hierarchically block-separable (HBS) [82], and hierarchically off-diagonal low-rank (HODLR) [1] matrices. The differences between these methods are concisely summarized by Ambikasaran & Darve [2]. These algebraic generalizations of the FMM can perform addition, multiplication, and even factorization of dense matrices with near linear complexity. This transition from analytic to algebraic did not happen suddenly, and semi-analytic variants were developed along the way [103, 39]. Optimization techniques for the FMM such as compressed translation operators and their precomputation, also fall somewhere between the analytic and algebraic extremes.

The spectrum that spans purely analytic and purely algebraic forms of these hierarchical low-rank approximation methods, represents the tradeoff between computation (Flops) and memory (Bytes). The purely analytic FMM is a matrix-free $\mathscr{H}^2$-matrix-vector product, and due to its matrix-free nature it has very high arithmetic intensity (Flop/Byte) [9]. On the other end we have the purely algebraic methods, which precompute and store the entire hierarchical matrix. This results in more storage and more data movement, both vertically and horizontally in the memory hierarchy. When the cost of data movement increases faster than arithmetic operations on future architectures, the methods that compute more to store/move less will become advantageous. Therefore, it is important to consider the whole spectrum of hierarchical low-rank approximation methods, and choose the appropriate method for a given pair of application and architecture.

There have been few attempts to quantitatively investigate the tradeoff between the analytic and algebraic hierarchical low-rank approximation methods. Previously, the applicability of the analytic variants were limited to problems with Green's functions, and could only be used for matrix-vector products but not to solve the matrix. With the advent of the kernel-independent FMM (KIFMM) [103] and inverse FMM (IFMM) [2], these restrictions no longer apply to the analytic variants. Furthermore, the common argument for using the algebraic variants because they can operate directly on the matrix without the need to pass geometric information is not very convincing. Major libraries like PETSc offer interfaces to insert ones own matrix free preconditioner as a function, and passing geometric information is something that users are willing to do if the result is increased performance. Therefore, there is no strong reason from the users perspective to be monolithically inclined to use the algebraic variants. It is rather a matter of choosing the method with the right balance between its analytic (Flops) and algebraic (Bytes) features.

The topic of investigating the tradeoff between analytic and algebraic hierarchical low-rank approximation methods is too broad to cover in a page-constrained article. In the present work, we limit our investigation to the compute-memory tradeoff in a comparison between FMM and HSS for Laplace and Helmholtz kernels. We also investigate the use of FMM as a preconditioner for iterative solutions to the Laplace and Helmholtz problems with finite elements, for which we compare with geometric and algebraic multigrid methods.

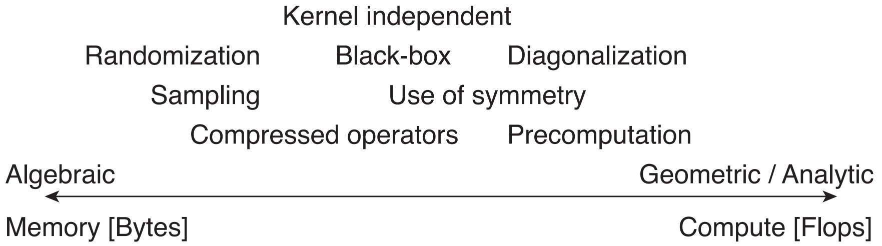

Fig. 1 The compute-memory tradeoff between the analytic and algebraic hierarchical low-rank approximation methods. Various techniques lie between the analytic and algebraic extremes.

2 Hierarchical Low-Rank Approximation: Analytic or Algebraic?

In this section we review the full spectrum of hierarchical low-rank approximations starting from the analytic side and proceeding to the algebraic side. The spectrum is depicted in Fig. 1, where various techniques like between the analytic and algebraic extremes. One can choose the appropriate method for a given architecture to achieve the best performance.

2.1 Analytic Low-Rank Approximation

On the analytic end of the spectrum, we have classical methods such as the Treecode [10], FMM [8, 51], and panel clustering methods [57]. These methods have extremely high arithmetic intensity (Flop/Byte) due to their matrix-free nature, and are compute-bound on most modern architectures. One important fact is that these are not brute force methods that do unnecessary Flops, but are (near) linear complexity methods that are only doing useful Flops, but they are still able to remain compute-bound. This is very different from achieving high Flops counts on dense matrix-matrix multiplication or LU decomposition that have $\mathscr{O}(N^3)$ complexity. The methods we describe in this section can approximate the same dense linear algebra calculation in $\mathscr{O}(N)$ or $\mathscr{O}(N\log N)$ time.

As an example of the absolute performance of the analytic variants, we refer to the Treecode implementation – `Bonsai`, which scales to the full node of Titan using 18,600 GPUs achieving 24.77 PFlops [14]. Bonsai's performance comes not only from its matrix-free nature, but also from domain specific optimizations for hardcoded quadrupoles and an assumption that all charges are positive. Therefore, this kind of performance cannot be transferred to other applications that require higher accuracy. However, viewing these methods as a preconditioner instead of a direct solver significantly reduces the accuracy requirements [67, 6].

2.2 Fast Translation Operators

A large part of the calculation time of FMM is spent on the translation of multipole expansions to local expansions (or their equivalent charges). Therefore, much work has focused on developing fast translation operators to accelerate this part of the FMM. Rotation of spherical harmonics [94], Block FFT [37], Planewaves [52] are analytic options for fast translation operators.

These translation operators are applied to a pair of boxes in the FMM tree structure that satisfy a certain proximity threshold. This proximity is usually defined as the parent's neighbors' children that are non-neighbors. This produces a list of boxes that are far enough that the multipole/local expansion converges, but are close enough that the expansion does not converge for the their parents. Such an interaction list can contain up to $6^3 - 3^3 = 189$ source boxes for each target box. Out of these 189 boxes, the ones that are further from the target box can perform the translation operation using their parent box as the source without loss of accuracy. There are a few variants for these techniques that reduce the interaction list size such as the level-skip M2L method [93] and 8,4,2-box method [95]. There are also methods that use the dual tree traversal along with the multipole acceptance criterion to construct optimal interaction lists [35], which automates the process of finding the optimal interaction list size.

Another technique to accelerate the translation operators is the use of variable expansion order, as proposed in the very fast multipole method (VFMM) [87], Gaussian VFMM [21], optimal parameter FMM [29], and error controlled FMM [32]. There are two main reasons why spatially varying the expansion order in the translation operators is beneficial. One is because not all boxes in the interaction list are of equal distance, and the boxes that are further from each other can afford to use lower expansion order, while retaining the accuracy. The other reason is because some parts of the domain may have smaller values, and the contribution from that part can afford to use lower expansion order without sacrificing the overall accuracy.

The translation operators can be stored as matrices that operate on the vector of expansion coefficients. Therefore, singular value decomposition (SVD) can be used to compress this matrix [43] and BLAS can be used to maximize the cache utilization [40]. Some methods use a combination of these techniques like Chebychev with SVD [39] and planewave with adaptive cross approximation (ACA) and SVD [61]. The use of SVD is a systematic and optimal way of achieving what the variable expansion order techniques in the previous paragraph were trying to do manually. Precomputing these translation matrices and storing them is a typical optimization technique in many FMM implementations [78].

One important connection to make here is that these matrices for the translation operators are precisely what $\mathscr{H}^2$-matrices and *HSS* matrices store in the off-diagonal blocks after compression. One can think of FMM as a method that has the analytical form to generate these small matrices in the off-diagonal blocks, without relying on numerical low-rank approximation methods. To complete this analogy, we point out that the dense diagonal blocks in $\mathscr{H}^2$-matrices and *HSS* matrices are simply storing the direct operator (Green's function) in FMM. Noticing this equivalence leads to many possibilities of hybridization among the analytic and algebraic variants. Possibly the most profound is the following. Those that are familiar with FMM know that translation operators for boxes with the same relative positioning are identical. This suggests that many of the entries in the off-diagonal blocks of $\mathscr{H}^2$-matrices and *HSS* matrices are identical. For matrices that are generated from a mesh that has a regular structure even the diagonal blocks would be identical, which is what happens in FMMs for continuous volume integrals [78]. This leads to $\mathscr{O}(1)$ storage for the matrix entries at every level of the hierarchy, so the total storage cost of these hierarchical matrices could be reduced to $\mathscr{O}(\log N)$ if the identical entires are not stored redundantly. This aspect is currently underutilized in the algebraic variants, but seems obvious from the analytic side. By making use of the translational invariance and rotational symmetry of the interaction list one can reduce the amount of storage even further [31, 33, 91]. This also results in blocking techniques for better cache utilization.

2.3 Semi-analytical FMM

The methods described in the previous subsection all require the existance of an analytical form of the multipole/local translation operator, which is kernel dependent. There are a class of methods that remove this restriction by using equivalent charges instead of multipole expansions [7, 15, 77]. A well known implementation of this method is the kernel independent FMM (KIFMM) code [103]. There are also variants that use Chebychev polynomials [36], and a representative implementation of this is the Black-box FMM [39]. As the name of these codes suggest, these variants of the FMM have reduced requirements for the information that has to be provided by the user. The translation operators are kernel-independent, which frees the user from the most difficult task of having to provide an analytical form of the translation operators. For example, if one wants to calculate the Matérn function for covariance martices, or multiquadrics for radial basis function interpolation, one simply needs to provide these functions and the location of the points and the FMM will handle the rest. It is important to note that these methods are not entirely kernel independent or black-box because the user still needs to provide the kernel dependent analytic form of the original equation they wish to calculate. Using the vocabulary of the algebraic variants, one could say that these analytical expressions for the hierarchical matrices are kernel independent only for the off-diagonal blocks, and for the diagonal blocks the analytical form is kernel dependent.

FMM for continuous volume integrals [38] also has important features when considering the analytic-algebraic tradeoff. The volume integrals are often combined with boundary integrals, as well [104]. One can think of these methods as an FMM that includes the discretization process [70]. Unlike the FMM for discrete particles, these methods have the ability to impose regular underlying geometry. This enables the use of precomputation of the direct interaction matrix in the analytic variants [78], and reduces the storage requirements of the dense diagonal blocks in the algebraic variants.

2.4 Algebraic Low-Rank Approximation

There are many variants of algebraic low-rank approximation methods. They can be categorized based on whether they are hierarchical, whether they use weak admissibility, or if the basis is nested, as shown in Table 1. For the definition of admissibility see [45]. Starting from the top, $\mathscr{H}$-matrices [55, 12] are hierarchical, usually use standard or strong admissibility, and no nested basis. The analytic counterpart of the $\mathscr{H}$-matrix is the Treecode. The $\mathscr{H}^2$-matrices [56, 16] are also hierarchical and use standard or strong admissibility, but unlike $\mathscr{H}$-matrices use a nested basis. This brings the complexity down from $\mathscr{O}(NlogN)$ to $\mathscr{O}(N)$. The analytic counterpart of the $\mathscr{H}^2$-matrix is the FMM. The next three entries in Table 1 do not have analytic counterparts because analytic low-rank approximations do not converge under weak admissibility conditions. Hierarchical off-diagonal low-rank (HODLR) matrices [1, 5], are basically $\mathscr{H}$-matrices with weak admissibility conditions. Similarly, hierarchically semi-seperable (HSS) [25, 101], and hierarchically block-seperable (HBS) [82] matrices are $\mathscr{H}^2$-matrices with weak admissibility conditions. The block low-rank (BLR) matrices [4] are a non-hierarchical version of the HODLR, with just the bottom level. A summary of implementations and their characteristics are presented in [89].

For methods that do not have weak admissibility, it is common to use geometrical information to calculate the standard/strong admissibility condition. This dependence on the geometry of the algebraic variants is not ideal. There have been various proposals for algebraic clustering methods [71, 85, 46]. This problem requires even more advanced solutions for high dimension problems [80]. Stronger admissibility is also problem for parallelization since it results in more communication. There have been studies on how to partition hierarchical matrices on distributed memory [68]. There are also methods to reduce the amount of memory consumption during the construction of HSS matrices[73].

The categorization in Table 1 is for the hierarchical matrix structure, and any low-rank approximation method can be used with each of them during the compression phase. The singular value decomposition is the most naïve and expensive way to calculate a low-rank approximation. QR or LU decompositions can be used to find the numerical rank by using appropriate pivoting. Rank-revealing QR [24] has been proposed along with efficient pivoting strategies [64, 27, 53]. Rank-revealing LU [23] also requires efficient pivoting strategies [66, 65, 83]. Rank-revealing LU is typically faster than rank-revealing QR [86]. There are other methods like the pseudo-skeletal method [44] and adaptive cross approximation (ACA) [11, 13], which do not yield the optimal low-rank factorizations but have a much lower cost. ACA has a better pivoting strategy than pseudo-skeletal methods, but can still fail because of bad pivots [18]. The hybrid cross approximation (HCA) [17] has the same proven convergence as standard interpolation but also the same efficiency as ACA. Yet another

Table 1 Categorization of algebraic low-rank approximation methods.

Method	Hierarchical	Weak admissibility	Nested basis
$\mathscr{H}$-matrix [55]	yes	maybe	no
$\mathscr{H}^2$-matrix [56]	yes	maybe	yes
HODLR [1]	yes	yes	no
HSS [25] / HBS [82]	yes	yes	yes
BLR [4]	no	yes	no

class of low-rank approximation is the interpolative decomposition (ID) [28, 82], where a few of its columns are used to form a well-conditioned basis for the remaining columns. ID can be combined with randomized methods [76], which has much lower complexity. For a nice review on these randomized methods see [58].

3 Low-Rank Approximation for Factorization

3.1 Sparse Matrix Factorization

Hierarchical low-rank approximation methods can be used as direct solvers with controllable accuracy. This makes them useful as preconditioners within a Krylov subspace method, which in turn reduces the accuracy requirements of the low-rank approximation. High accuracy and completely algebraic methods are demanding in terms of memory consumption and amount of communication, so they are unlikely to be the optimal choice unless they are the only solution to that problem.

There are two ways to use hierarchical low-rank approximations for factorization of a sparse matrix. The first way is to perform the LU decomposition on the sparse matrix, and use hierarchical low-rank approximations for the dense blocks that appear during the process [100, 101, 98]. The other way is to represent the sparse matrix with a hierarchical low-rank approximation and perform an LU decomposition on it [46, 47, 48]. The main difference is whether you view the base method as the nested dissection and the additional component as HLRA or vice versa. The former has the advantage of being able to leverage the existing highly optimized sparse direct solvers, whereas the latter has the advantage of handling both sparse and dense matrices with the same infrastructure.

There are various ways to minimize the fill-in and compress the dense blocks during factorization. These dense blocks (Schur complements) are an algebraic form of the Green's function [99], and have the same low-rank properties [26] stemming from the fact that some of the boundary points in the underlying geometry are distant from each other. Formulating a boundary integral equation is the analytical way of arriving to the same dense matrix. From an algebraic point of view, the sparse matrix for the volume turns into a dense matrix for the boundary, through the process of trying to minimize fill-in. Considering the minimization of fill-in and the compression of the dense matrices in separate phases leads to methods like HSS + multifrontal [100, 101, 98].

Ultimately, minimizing fill-in and minimizing off-diagonal rank should not be conflicting objectives. The former depends on the connectivity and the latter depends on the distance in the underlying geometry. In most applications, the closer points are connected (or interact) more densely, so reordering according to the distance should produce near optimal ordering for the connectivity as well. The same can be said about minimizing communication for the parallel implementation of these methods. Mapping the 3-D connectivity/distance to a 1-D locality in the memory space (or matrix column/row) is what we are ultimately trying to achieve.

3.2 Dense Matrix Factorization

The methods in the previous subsection are direct solvers/preconditioners for sparse matrices. As we have mentioned, there is an analogy between minimizing fill-in in sparse matrices by looking at the connectivity, and minimizing the rank of off-diagonal blocks of dense matrices by looking at the distance. Using this analogy, the same concept as nested dissection for sparse matrices can be applied to dense matrices. This leads to methods like the recursive skeletonization [62], or hierarchical Poincare-Steklov (HPS) [81, 41]. HPS is like a bottom-up version of what nested dissection and recursive skeletonization do top-down. For high contrast coefficient problems, it makes sense to construct the domain dissection bottom-up, to align the bisectors with the coefficient jumps. There are also other methods that rely on a similar concept [102, 50, 69, 19]. Furthermore, since many of these methods use weak admissibility with growing ranks for 3-D problems, it is useful to have nested hierarchical decompositions, which is like a nested dimension reduction. In this respect, the recursive skeletonization has been extended to hierarchical interpolative factorization (HIF) [63], the HSS has been extended to HSS2D [99]. There is also a combination of HSS and Skeletonization [30]. There are methods that use this nested dimension reduction concept without the low-rank approximation [60] in the context of domain decomposition for incomplete LU factorization. One method that does not use weak admissibility is the inverse FMM [2], which makes it applicable to 3-D problems in $\mathscr{O}(N)$ without nested dimension reduction.

4 Experimental Results

4.1 FMM vs. HSS

There have been few comparisons between the analytic and algebraic hierarchical low-rank approximation methods[20]. From a high performance computing perspective, the practical performance of highly optimized implementations of these various methods is of great interest. There have been many efforts to develop new methods in this area, which has resulted in a large amount of similar methods with different names without a clear overall picture of their relative performance on modern HPC architectures. The trend in architecture where arithmetic operations are becoming cheap compared to data movement, is something that must be considered carefully when predicting which method will perform better on computers of the future.

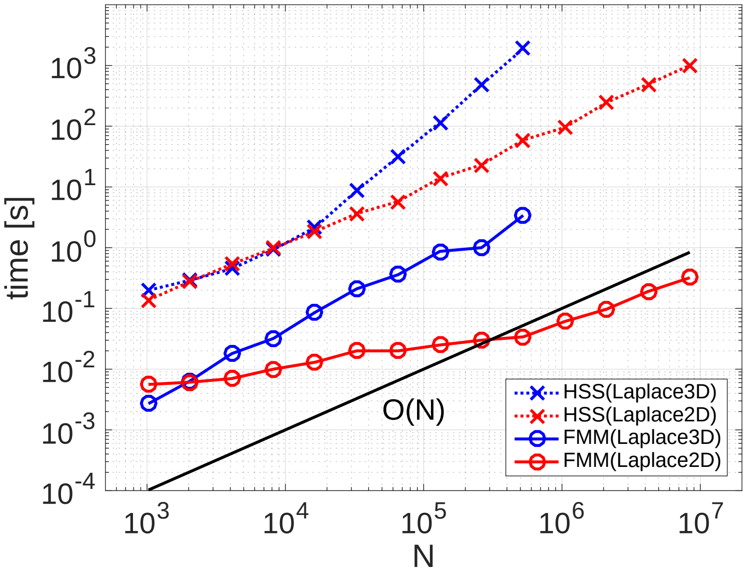

Fig. 2 Elapsed time for the matrix-vector multiplication using FMM and HSS for different problem sizes.

We acknowledge that the comparisons we present here are far from complete, and much more comparisons between all the different methods are needed in order to acheive our long term objective. The limitation actually comes from the lack of highly optimized implementations of these methods that are openly available to us at the moment.

In the present work we start by comparing exaFMM – a highly optimized implementation of FMM, with STRUMPACK – a highly optimized implementation of HSS. We select the 2D and 3D Laplace equation on uniform lattices as test cases. For HSS we directly construct the compressed matrix by calling the Green's function in the randomized low-rank approximation routine. We perform the matrix-vector multiplication using the FMM and HSS, and measure the time for the compression/precalculation and application of the matrix-vector multiplication. We also measure the peak memory consumption of both methods.

The elapsed time for the FMM and HSS for different problem sizes is shown in Fig. 2. In order to isolate the effect of the thread scalability of the two methods, these runs are performed on a single core of a 12-core Ivy Bridge (E5-2695 v2). For the 2D Laplace equation, the FMM shows some overhead for small N, but is about 3 orders of magnitude faster than HSS for larger problems. For the 3D Laplace equation, the FMM is about 2 orders of magnitude faster than HSS for smaller N, but HSS exhibits non-optimal behavior for large N because the rank keeps growing.

The large difference in the computational time is actually coming from the heavy computation in the sampling phase and compression phase of the HSS. In Fig. 3, we show the percentage of the computation time of HSS for different problem sizes

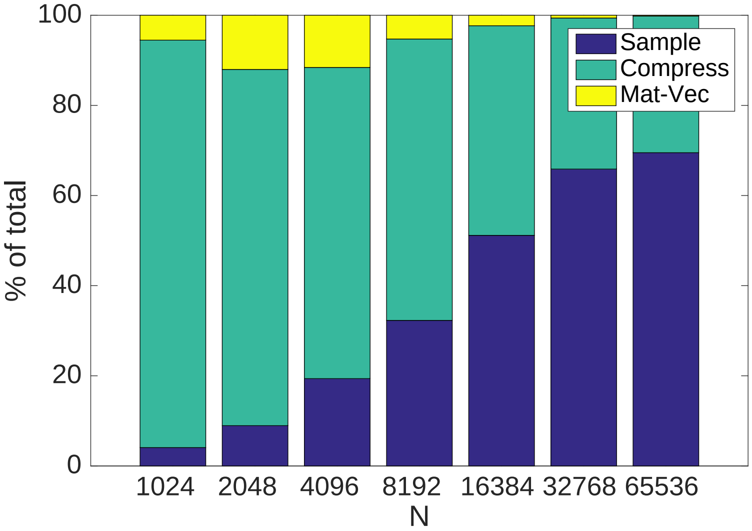

Fig. 3 Percentage of the computation time of HSS for different problem sizes.

N. "Sample" is the sampling time, "Compress" is the compression time, and "Mat-Vec" is the matrix-vector multiplication time. We can see that the sampling is taking longer and longer as the problem size increases. This is because the rank k increases with the problem size N, and both sampling and compression time increase with the k and N.

The peak memory usage of FMM and HSS is shown in Fig. 4 for the 3D Laplace equation. We see that the FMM has strictly $\mathscr{O}(N)$ storage requirements, but since the rank in the HSS grows for 3D kernels it does not show the ideal $\mathscr{O}(N\log N)$ behavior. The disadvantage of HSS is two-fold. First of all, its algebraic nature requires it to store the compressed matrix, where as the FMM is analytic and therefore matrix-free. Secondly, the weak admissibility causes the rank to grow for 3D problems, and with that the memory consumption grows at a suboptimal complexity.

4.2 FMM vs. Multigrid

If we are to use the FMM as a matrix-free $\mathscr{O}(N)$ preconditioner based on hierarchical low-rank approximation, the natural question to ask is "How does it compare against multigrid?", which is a much more popular matrix-free $\mathscr{O}(N)$ preconditioner for solving elliptic PDEs. We perform a benchmark test similar to the one in the previous subsection, for the Laplace equation and Helmholtz equation on a 3D cubic lattice $[-1,1]^3$, but for this case we impose Dirichlet boundary conditions at the faces of the domain. The preconditioners are used inside a Krylov subspace solver. The runs were performed on Matlab using a finite element package IFISS. Our fast

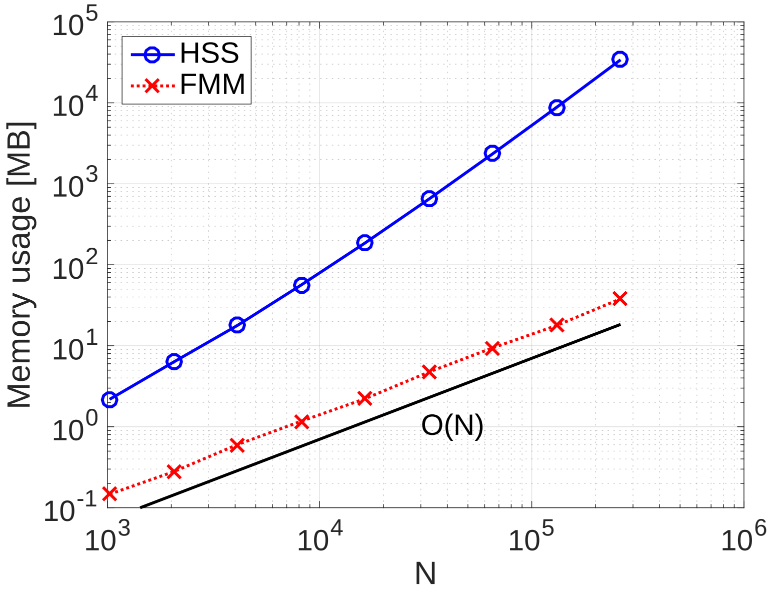

Fig. 4 Peak memory usage of FMM and HSS for the 3D Laplace equation.

multipole preconditioner is compared with the incomplete Cholesky (IC) factorization with zero fill implemented in Matlab and the algebraic multigrid (AMG) and geometric multigrid (GMG) methods in IFISS. The FMM code is written in C and called as a MEX function.

The convergence rate of the FMM and Multigrid preconditioners for the Laplace equation is shown in Fig. 5, for a grid spacing of $h = 2^{-5}$. "AMG" is algebraic multigrid, "GMG" is geometric multigrid, "Inc Chol" is incomplete Cholesky. The ε value represents the accuracy of the FMM. We see that the FMM preconditioner has comparable convergence to the algebraic and geometric multigrid method. Even for a very low-accuracy FMM with $\varepsilon = 10^{-2}$, the convergence rate is much better than the incomplete Cholesky. We refer to the work by Ibeid *et al.* [67] for more detailed comparisons between FMM and Multigrid.

A similar plot is shown for the Helmholtz equation with grid spacing of $h = 2^{-5}$ and wave number $\kappa = 7$ in Fig. 6. The nomenclature of the legend is identical to that of Fig. 5. In this case, we see a larger difference between the convergence rate of FMM and Multigrid. Even the FMM with the worst accuracy does better than the multigrid. We have also confirmed that the FMM preconditioner has a convergence rate that is independent of the problem size, up to moderate wave numbers of κ.

The strong scaling of FMM and AMG are shown in Figure 7, which includes the setup phase and all iterations it took to converge. All calculations were performed on the TACC Stampede system without using the coprocessors. Stampede has 6400 nodes, each with two Xeon E5-2680 processors and one Intel Xeon Phi SE10P coprocessor and 32GB of memory. We used the Intel compiler (version

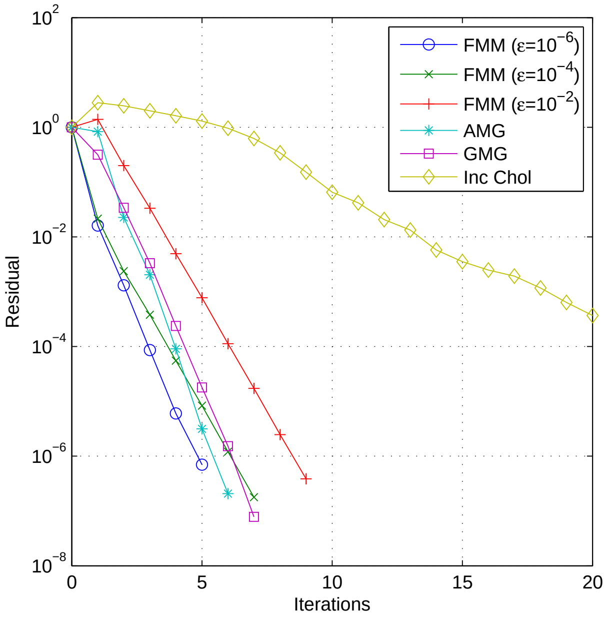

Fig. 5 Convergence rate of the FMM and Multigrid preconditioners for the Laplace equation on a $[-1,1]^3$ lattice with spacing $h = 2^{-5}$.

13.1.0.146) and configured PETSc with "`COPTFLAGS=-O3 FOPTFLAGS=-O3`
`--with-clanguage=cxx`
`--download-f-blas-lapack --download-hypre`
`--download-metis --download-parmetis`
`--download-superlu_dist --with-debugging=0`". For BoomerAMG we compared different relaxation, coarsening, and interpolation methods and found that
"`-pc_hypre_boomeramg_relax_type_all`
`backward-SOR/Jacobi`
`-pc_hypre_boomeramg_coarsen_type`
`modifiedRuge-Stueben`
`-pc_hypre_bommeramg_interp_type classical`" gives the best performance. We use a grid size of $N = 4096^2$ and run from 1 to 1024 cores using up to 16 cores per node on Stampede. For this particular Poisson problem on this particular machine using this particular FMM code we see an advantage over BoomerAMG past 512 cores.

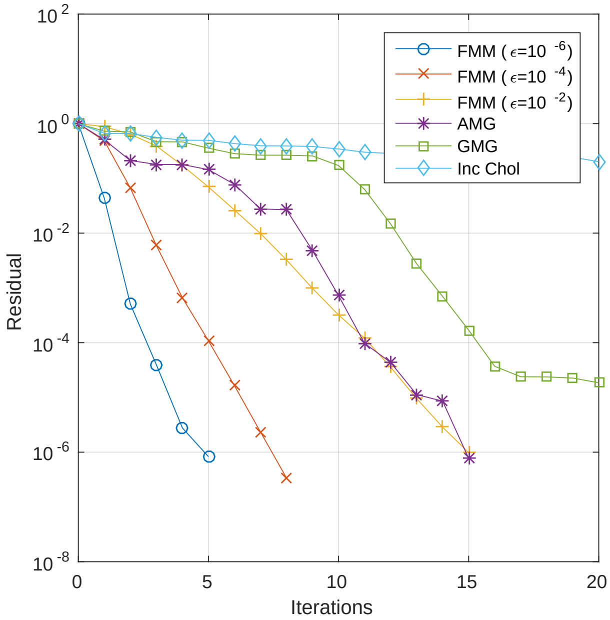

Fig. 6 Convergence rate of the FMM and Multigrid preconditioners for the Helmholtz equation on a $[-1,1]^3$ lattice with spacing $h = 2^{-5}$ and wave number $\kappa = 7$.

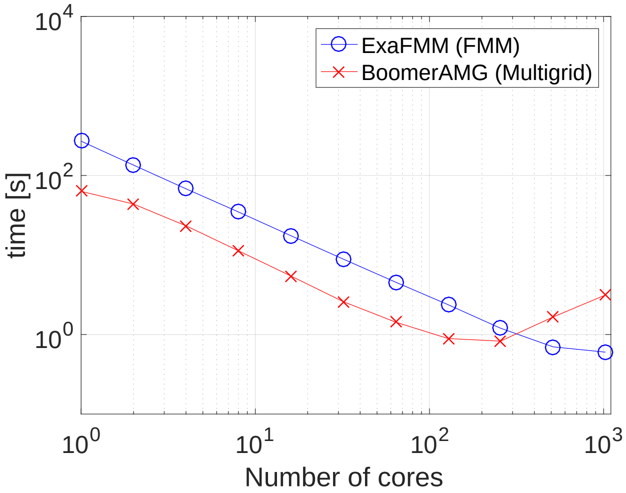

Fig. 7 Strong scaling of the 2-D FMM and AMG preconditioners.

5 Conclusions and Outlook

We have shown the contrast between the analytical and algebraic hierarchical low-rank approximations, by reviewing the contributions over the years and placing them along the analytical-algebraic spectrum. The relation between Treecode, FMM, KIFMM, black-box FMM, $\mathscr{H}$-matrix, $\mathscr{H}^2$-matrix, HODLR, HSS, HBS, and BLR were explained from the perspective of compute-memory tradeoff. This birds-eye view of the entire hierarchical low-rank approximation landscape from analytical to algebraic, allows us to place ideas like precomputation of FMM translation matrices and relate that to storage reduction techniques for the algebraic variants.

Some important findings from this cross-disciplinary literature review are:

- Translational invariance of the FMM operators suggest that $\mathscr{H}^2$-matrices (and the like) have mostly duplicate entries, which many are redundantly storing at the moment.
- The analytical variants can now perform factorization and are kernel independent, so the decision to use the algebraic variants at the cost of consuming more memory should be made carefully.
- The kernel-independent variants of FMM can be used as a matrix-free $\mathscr{O}(N)$ compression technique.
- The use of SVD to compress the FMM translation matrices, makes the work on variable expansion order and its error optimized variants redundant.
- The hierarchical compression should not be applied directly to the inverse or factorizations of sparse matrices just because they fill-in. One must first try to minimize fill-in, and then compress only the dense blocks that cannot be avoided.

The comparison benchmarks between FMM and HSS are still preliminary tests for a very simple case. However, they clearly demonstrate the magnitude of the difference that lies between the various hierarchical low-rank approximation methods. The comparison between FMM and multigrid is also a very simple test case, but it reveals the previously unquantified convergence properties of low-accuracy FMM as a preconditioner. Of course, for such simple problems the FMM can give the exact solution in finite arithmetic and therefore solve the problem in a single iteration. The interesting point here is not the fact that it can be used as a preconditioner, but the practical performance of the low-accuracy FMM being significantly faster than the high accuracy FMM, even if it requires a few iterations.

There is much more that can be done if all of these complicated hierarchical low-rank approximation methods could somehow be made easier to code. We believe a modular view of these methods will help the developers though separation of concerns. Instead of everyone coding a slightly different version of the whole thing, we could each choose a module to focus on that fits our research interests, and contribute to a larger and more sustainable ecosystem. A few ideas to facilitate the transition to such a community effort are:

1. Create a common benchmark (mini app) for each of the modules.
2. Gradually propagate standards in the community, starting from the major codes.
3. Develop a common interface between the hierarchical structure and inner kernels.
4. Do not try to unify code, just have a standard with a common API (like MPI).

Acknowledgements We thank François-Henry Rouet, Pieter Ghysels, and Xiaoye, S. Li for providing the STRUMPACK interface for our comparisons between FMM and HSS. This work was supported by JSPS KAKENHI Grant-in-Aid for Research Activity Start-up Grant Number 15H06196. This publication was based on work supported in part by Award No KUK-C1-013-04, made by King Abdullah University of Science and Technology (KAUST). This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

References

1. S. Ambikasaran and E. Darve. An O(NlogN) fast direct solver for partial hierarchically semi-separable matrices. *Journal of Scientific Computing*, 57:477–501, 2013.
2. S. Ambikasaran and E. Darve. The inverse fast multipole method. *arXiv:1407.1572v1*, 2014.
3. S. Ambikasaran, J.-Y. Li, P. K. Kitanidis, and E. Darve. Large-scale stochastic linear inversion using hierarchical matrices. *Computational Geosciences*, 17(6):913–927, 2013.
4. P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker. Improving multifrontal methods by means of block low-rank representations. *SIAM Journal on Scientific Computing*, 37(3):A1451–A1474, 2015.
5. A. Aminfar, S. Ambikasaran, and E. Darve. A fast block low-rank dense solver with applications to finite-element matrices. *Journal of Computational Physics*, 304:170–188, 2016.
6. A. Aminfar and E. Darve. A fast, memory efficient and robust sparse preconditioner based on a multifrontal approach with applications to finite-element matrices. *International Journal for Numerical Methods in Engineering*, accepted, 2016.
7. C. R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific and Statistical Computing*, 13(4):923–947, 1992.
8. A. W. Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6(1):85–103, 1985.
9. L. A. Barba and R. Yokota. How will the fast multipole method fare in the exascale era? *SIAM News*, 46(6):1–3, 2013.
10. J. Barnes and P. Hut. O(NlogN) force-calculation algorithm. *Nature*, 324:446–449, 1986.
11. M. Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86:565–589, 2000.
12. M. Bebendorf. *Hierarchical Matrices*, volume 63 of *Lecture Notes in Computational Science and Engineering*. Springer, 2008.
13. M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.
14. J. Bédorf, E. Gaburov, M. S. Fujii, K. Nitadori, T. Ishiyama, and S. Portegies Zwart. 24.77 Pflops on a gravitational tree-code to simulate the milky way galaxy with 18600 GPUs. In *Proceedings of the 2014 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2014.
15. C. L. Berman. Grid-multipole calculations. *SIAM Journal on Scientific Computing*, 16(5):1082–1091, 1995.
16. S. Börm. Construction of data-sparse h^2-matrices by hierarchical compression. *SIAM Journal on Scientific Computing*, 31(3):1820–1839, 2009.
17. S. Börm and L. Grasedyck. Hybrid cross approximation of integral operators. *Numerische Mathematik*, 101:221–249, 2005.
18. S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *Engineering Analysis with Boundary Elements*, 27:405–422, 2003.
19. J. Bremer. A fast direct solver for the integral equations of scattering theory on planar curves with corners. *Journal of Computational Physics*, 231:1879–1899, 2012.
20. D. Brunner, M. Junge, P. Rapp, M. Bebendorf, and L. Gaul. Comparison of the Fast Multipole Method with Hierarchical Matrices for the Helmholtz-BEM. *Computer Modeling in Engineering & Sciences*, 58(2):131–160, 2010.

21. J. C. Burant, M. C. Strain, G. E. Scuseria, and M. J. Frisch. Analytic energy gradients for the Gaussian very fast multipole method (GvFMM). *Chemical Physics Letters*, 248:43–49, 1996.
22. S. Chaillat, M. Bonnet, and J.-F. Semblat. A multi-level fast multipole BEM for 3-D elastodynamics in the frequency domain. *Computer Methods in Applied Mechanics and Engineering*, 197:4233–4249, 2008.
23. T. F. Chan. On the existence and computation of LU-factorizations with small pivots. *Mathematics of Computation*, 42(166):535–547, 1984.
24. T. F. Chan. Rank revealing QR factorizations. *Linear Algebra and its Applications*, 88/89:67–82, 1987.
25. S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals. A fast solver for HSS representations via sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):67–81, 2006.
26. S. Chandrasekaran, P. Dewilde, M. Gu, and N. Somasunderam. On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2261–2290, 2010.
27. S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorizations. *SIAM Journal on Matrix Analysis and Applications*, 15(2):592–622, 1994.
28. H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin. On the compression of low rank matrices. *SIAM Journal on Scientific Computing*, 26(4):1389–1404, 2005.
29. C. H. Choi, K. Ruedenberg, and M. S. Gordon. New parallel optimal-parameter fast multipole method (OPFMM). *Journal of Computational Chemistry*, 22(13):1484–1501, 2001.
30. E. Corona, P. G. Martinsson, and D. Zorin. An O(N) direct solver for integral equations on the plane. *Applied and Computational Harmonic Analysis*, 38:284–317, 2015.
31. O. Coulaud, P. Fortin, and J. Roman. High performance BLAS formulation of the multipole-to-local operator in the fast multipole method. *Journal of Computational Physics*, 227:1836–1862, 2008.
32. H. Dachsel. Corrected article: “an error-controlled fast multipole method”. *The Journal of Chemical Physics*, 132:119901, 2010.
33. E. Darve, C. Cecka, and T. Takahashi. The fast multipole method on parallel clusters, multicore processors, and graphics processing units. *Comptes Rendus Mecanique*, 339:185–193, 2011.
34. E. Darve and P. Havé. A fast multipole method for Maxwell equations stable at all frequencies. *Philosophical Transactions of the Royal Society of London A*, 362:603–628, 2004.
35. W. Dehnen. A hierarchical O(N) force calculation algorithm. *Journal of Computational Physics*, 179(1):27–42, 2002.
36. A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration, and differntiation. *SIAM Journal on Numerical Analysis*, 33(5):1689–1711, 1996.
37. W. D. Elliott and J. A. Board. Fast Fourier transform accelerated fast multipole algorithm. *SIAM Journal on Scientific Computing*, 17(2):398–415, 1996.
38. F. Ethridge and L. Greengard. A new fast-multipole accelerated Poisson solver in two dimensions. *SIAM Journal on Scientific Computing*, 23(3):741–760, 2001.
39. W. Fong and E. Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228:8712–8725, 2009.
40. P. Fortin. Multipole-to-local operator in the fast multipole method: Comparison of FFT, rotations and BLAS improvements. Technical Report RR-5752, Rapports de recherche, et theses de l’Inria, 2005.
41. A. Gillman, A. Barnett, and P. G. Martinsson. A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media. *BIT Numerical Mathematics*, 55:141–170, 2015.
42. Z. Gimbutas and L. Greengard. Fast multi-particle scattering: A hybrid solver for the Maxwell equations in microstructured materials. *Journal of Computational Physics*, 232:22–32, 2013.
43. Z. Gimbutas and V. Rokhlin. A generalized fast multipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing*, 24(3):796–817, 2002.

44. S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*, 261(1-3):1–21, 1997.
45. L. Grasedyck and W. Hackbusch. Construction and arithmetics of H-matrices. *Computing*, 70:295–334, 2003.
46. L. Grasedyck, R. Kriemann, and S. Le Borne. Parallel black box H-LU preconditioning for elliptic boundary value problems. *Computing and Visualization in Science*, 11:273–291, 2008.
47. L. Grasedyck, W. Hackbusch, and R. Kriemann Performance of H-LU preconditioning for sparse matrices. *Computational Methods in Applied Mathematics*, 8(4):336–349, 2008.
48. L. Grasedyck, R. Kriemann, and S. Le Borne. Domain decomposition based H-LU preconditioning. *Numerische Mathematik*, 112:565–600, 2009.
49. A. G. Gray and A. W. Moore. N-body problems in statistical learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 521—527. MIT Press, 2001.
50. L. Greengard, D. Gueyffier, P. G. Martinsson, and V. Rokhlin. Fast direct solvers for integral equations in complex three dimensional domains. *Acta Numerica*, 18:243–275, 2009.
51. L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.
52. L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997.
53. M. Gu and S. C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM Journal on Scientific Computing*, 17(4):848–869, 1996.
54. N. A. Gumerov and R. Duraiswami. Fast radial basis function interpolation via preconditioned Krylov iteration. *SIAM Journal on Scientific Computing*, 29(5):1876–1899, 2007.
55. W. Hackbusch. A sparse matrix arithmetic based on H-matrices, part I: Introduction to H-matrices. *Computing*, 62:89–108, 1999.
56. W. Hackbusch, B. Khoromskij, and S. A. Sauter. On h^2-matrices. In H. Bungartz, R. Hoppe, and C. Zenger, editors, *Lectures on Applied Mathematics*. Springer-Verlag, 2000.
57. W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik*, 54:463–491, 1989.
58. N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.
59. S. Hao, P. G. Martinsson, and P. Young. An efficient and highly accurate solver for multibody acoustic scattering problems involving rotationally symmetric scatterers. *Computers and Mathematics with Applications*, 69:304–318, 2015.
60. P. Hénon and Y. Saad. A parallel multistage ILU factorization based on a hierarchical graph decomposition. *SIAM Journal on Scientific Computing*, 28(6):2266–2293, 2006.
61. A. J. Hesford and R. C. Waag. Reduced-rank approximations to the far-field transform in the gridded fast multipole method. *Journal of Computational Physics*, 230:3656–3667, 2011.
62. K. L. Ho and L. Greengard. A fast direct solver for structured linear systems by recursive skeletonization. *SIAM Journal on Scientific Computing*, 34(5):A2507–A2532, 2012.
63. K. L. Ho and L. Ying. Hierarchical interpolative factorization for elliptic operators: Integral equations. *arXiv:1307.2666*, 2015.
64. Y. P. Hong and C. T. Pan. Rank-revealing QR factorizations and the singular value decomposition. *Mathematics of Computation*, 58(197):213–232, 1992.
65. T.-M. Hwang, W.-W. Lin, and D. Pierce. Improved bound for rank revealing LU factorizations. *Linear Algebra and its Applications*, 261(1):173–186, 1997.
66. T.-M. Hwang, W.-W. Lin, and E. K. Yang. Rank revealing LU factorizations. *Linear Algebra and its Applications*, 175:115–141, 1992.
67. H. Ibeid, R. Yokota, J. Pestana, and D. Keyes. Fast multipole preconditioners for sparse matrices arising from elliptic equations. *arXiv:1308.3339*, 2016.
68. M. Izadi. *Hierarchical Matrix Techniques on Massively Parallel Computers*. PhD thesis, Universitat Leipzig, 2012.

69. W. Y. Kong, J. Bremer, and V. Rokhlin. An adaptive fast direct solver for boundary integral equations in two dimensions. *Applied and Computational Harmonic Analysis*, 31:346–369, 2011.
70. H. Langston, L. Greengard, and D. Zorin. A free-space adaptive FMM-based PDE solver in three dimensions. *Communications in Applied Mathematics and Computational Science*, 6(1):79–122, 2011.
71. S. Le Borne. Multilevel hierarchical matrices. *SIAM Journal on Matrix Analysis and Applications*, 28(3):871–889, 2006.
72. D. Lee, R. Vuduc, and A. G. Gray. A distributed kernel summation framework for general-dimension machine learning. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, 2012.
73. K. Lessel, M. Hartman, and S. Chandrasekaran. A fast memory efficient construction algorithm for hierarchically semi-separable representations. *http://scg.ece.ucsb.edu/publications/MemoryEfficientHSS.pdf*, 2015.
74. J.-Y. Li, S. Ambikasaran, E. F. Darve, and P. K. Kitanidis. A Kalman filter powered by h^2-matrices for quasi-continuous data assimilation problems. *Water Resources Research*, 50:3734–3749, 2014.
75. Z. Liang, Z. Gimbutas, L. Greengard, J. Huang, and S. Jiang. A fast multipole method for the Rotne-Prager-Yamakawa tensor and its applications. *Journal of Computational Physics*, 234:133–139, 2013.
76. E. Liberty, F. Woolfe, P. G. Martinsson, V. Rokhlin, and M. Tygert. Randomized algorithms for the low-rank approximation of matrices. *PNAS*, 104(51):20167–20172, 2007.
77. J. Makino. Yet another fast multipole method without multipoles – Pseudoparticle multipole method. *Journal of Computational Physics*, 151(2):910–920, 1999.
78. D. Malhotra and G. Biros. PVFMM: A parallel kernel independent FMM for particle and volume potentials. *Communications in Computational Physics*, 18(3):808–830, 2015.
79. D. Malhotra, A. Gholami, and G. Biros. A volume integral equation stokes solver for problems with variable coefficients. In *Proceedings of the 2014 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2014.
80. W. B. March, B. Xiao, and G. Biros. ASKIT: Approximate skeletonization kernel-independent treecode in high dimensions. *SIAM Journal on Scientific Computing*, 37(2):A1089–A1110, 2015.
81. P. G. Martinsson. The hierarchical Poincaré-Steklov (HPS) solver for elliptic PDEs: A tutorial. *arXiv:1506.01308*, 2015.
82. P. G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *Journal of Computational Physics*, 205:1–23, 2005.
83. L. Miranian and M. Gu. Strong rank revealing LU factorizations. *Linear Algebra and its Applications*, 367:1–16, 2003.
84. Y. Ohno, R. Yokota, H. Koyama, G. Morimoto, A. Hasegawa, G. Masumoto, N. Okimoto, Y. Hirano, H. Ibeid, T. Narumi, and M. Taiji. Petascale molecular dynamics simulation using the fast multipole method on k computer. *Computer Physics Communications*, 185:2575–2585, 2014.
85. S. Oliveira and Y. F. An algebraic approcah for H-matrix preconditioners. *Computing*, 80:169–188, 2007.
86. C. T. Pan. On the existence and computation of rank-revealing LU factorizations. *Linear Algebra and its Applications*, 316:199–222, 2000.
87. H. G. Petersen, D. Soelvason, J. W. Perram, and E. R. Smith. The very fast multipole method. *The Journal of Chemical Physics*, 101(10):8870–8876, 1994.
88. A. Rahimian, I. Lashuk, K. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, D. Zorin, and G. Biros. Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, 2010.

89. F.-H. Rouet, X.-S. Li, P. Ghysels, and A. Napov. A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. *arXiv:1503.05464*, 2015.
90. Y. Shao, C. A. White, and M. Head-Gordon. Efficient evaluation of the Coulomb force in density-functional theory calculations. *The Journal of Chemical Physics*, 114(15):6572–6577, 2001.
91. T. Takahashi, C. Cecka, W. Fong, and E. Darve. Optimizing the multipole-to-local operator in the fast multipole method for graphical processing units. *International Journal for Numerical Methods in Engineering*, 89:105–133, 2012.
92. A. Verde and A. Ghassemi. Fast multipole displacement discontinuity method (FM-DDM) for geomechanics reservoir simulations. *International Journal for Numerical and Analytical Methods in Geomechanics*, 39(18):1953–1974, 2015.
93. Y. Wang, Q. Wang, X. Deng, Z. Xia, J. Yan, and H. Xu. Graphics processing unit (GPU) accelerated fast multipole BEM with level-skip M2L for 3D elasticity problems. *Advances in Engineering Software*, 82:105–118, 2015.
94. C. A. White and M. Head-Gordon. Rotating around the quartic angular momentum barrier in fast multipole method calculations. *The Journal of Chemical Physics*, 105(12):5061–5067, 1996.
95. D. R. Wilkes and A. J. Duncan. A low frequency elastodynamic fast multipole boundary element method in three dimensions. *Computational Mechanics*, 56:829–848, 2015.
96. D. Willis, J. Peraire, and J. White. FastAero – a precorrected FFT-fast multipole tree steady and unsteady potential flow solver. *http://hdl.handle.net/1721.1/7378*, 2005.
97. W. R. Wolf and S. K. Lele. Aeroacoustic integrals accelerated by fast multipole method. *AIAA Journal*, 49(7):1466–1477, 2011.
98. J. Xia. Randomized sparse direct solvers. *SIAM Journal on Matrix Analysis and Applications*, 34(1):197–227, 2013.
99. J. Xia. O(N) complexity randomized 3D direct solver with HSS2D structure. Proceedings of the Project Review, Geo-Mathematical Imaging Group 317–325, Purdue University, 2014.
100. J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1382–1411, 2009.
101. J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseperable matrices. *Numerical Linear Algebra with Applications*, 17:953–976, 2010.
102. N. Yarvin and V. Rokhlin. An improved fast multipole algorithm for potential fields on the line. *SIAM Journal on Numerical Analysis*, 36(2):629–666, 1999.
103. L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, 196(2):591–626, 2004.
104. L. Ying, G. Biros, and D. Zorin. A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains. *Journal of Computational Physics*, 219:247–275, 2006.
105. R. Yokota, J. P. Bardhan, M. G. Knepley, L. A. Barba, and T. Hamada. Biomolecular electrostatics using a fast multipole BEM on up to 512 GPUs and a billion unknowns. *Computer Physics Communications*, 182:1272–1283, 2011.
106. R. Yokota, T. Narumi, K. Yasuoka, and L. A. Barba. Petascale turbulence simulation using a highly parallel fast multipole method on GPUs. *Computer Physics Communications*, 184:445–455, 2013.
107. E. Yunis, R. Yokota, and A. Ahmadia. Scalable force directed graph layout algorithms using fast multipole methods. In *The 11th International Symposium on Parallel and Distributed Computing*, Munich, Germany, June 2012.
108. Z. Zhao, N. Kovvali, W. Lin, C.-H. Ahn, L. Couchman, and L. Carin. Volumetric fast multipole method for modeling Schrödinger's equation. *Journal of Computational Physics*, 224:941–955, 2007.