Impact of Far-Field Interactions on Performance of Multipole-Based Preconditioners for Sparse Linear Systems*

Ananth Y. Grama
Department of Computer Science
Purdue University
West Lafayette, IN 47907
ayg@cs.purdue.edu

Vivek Sarin
Department of Computer Science
Texas A&M University
College Station, TX 77843
sarin@cs.tamu.edu

ABSTRACT

Dense operators for preconditioning sparse linear systems have traditionally been considered infeasible due to their excessive computational and memory requirements. With the emergence of techniques such as block low-rank approximations and hierarchical multipole approximations, the cost of computing and storing these preconditioners has reduced dramatically. In our prior work [15], we have demonstrated the use of multipole-based techniques as effective parallel preconditioners for sparse linear systems. At one extreme, multipole-based preconditioners behave as dense (bounded interaction) matrices (multipole degree 0), while at the other extreme, they are represented entirely as series expansions. In this paper, we show that: (i) merely truncating the kernel of the integral operator generating the preconditioner leads to poor convergence properties; (ii) far-field interactions, in the form of multipoles, are critical for rapid convergence; (iii) the importance and required accuracy of far-field interactions varies with the complexity of the problem; and (iv) the preconditioner resulting from a judicious mix of near and far-field interactions yields excellent convergence and parallelization properties. Our experimental results are illustrated on the Poisson problem and the generalized Stokes problem arising in incompressible fluid flow simulations.

Categories and Subject Descriptors: G.4 Mathematical Software — *Algorithm design and analysis, parallel and vector implementations*; G.1.3 Numerical Linear Algebra — *Linear systems (direct and iterative methods)*

General Terms: Algorithms.

Keywords: Multipole methods, Preconditioning, Iterative methods, Stokes problem.

*This research was supported by the NSF grants NSF-CCR 9984400 and NSF-CCR 9972533.

ICS'04, June 26–July 1, 2004, Saint-Malo, France.

1. INTRODUCTION

Conventional preconditioning techiques for sparse linear systems have focused on representations with limited fill that have sparse non-zero structure. This is critical for restricting the storage and computational complexity of the preconditioning step. In general, it is not necessary to have a sparse non-zero structure to restrict the cost associated with the preconditioner. Examples of such matrices include low-rank matrices, Toeplitz matrices, and, more interestingly, hierarchical operators based on analytical approximations such as multipole representations.

Multipole methods were first proposed in the context of particle dynamics methods and boundary element solvers. The intuition behind hierarchical multipole methods is that the influence of a set of entities coupled by the underlying Green's functions can be approximated by a bounded degree multipole series. For example, the gravitational potential due to a set of objects can be represented by their center of mass, provided the evaluation point is sufficiently far away. The coupling Green's function in this case is determined by the Laplace operator, and is $1/r$ in three dimensions, where r is the distance between source and observation points. With this simple intuition, the entire domain can be hierarchically decomposed and potentials computed at all n source/observation points in $O(n)$ time. This family of hierarchical multipole methods includes the Fast Multipole Method [10] and the method of Barnes and Hut [1].

Hierarchical multipole methods can be adapted to solve integral equations for which a closed-form Green's operator and the associated multipole expansions are known. This has been used to solve scattering problems and problems in electrostatics with considerable success [3, 4, 5]. Since the multipole expansions satisfy boundary conditions implicitly, these methods have an added advantage over finite element and finite difference methods for problems where boundary conditions are specified at infinity. One such example is the Sommerfeld radiation condition for scattering problems.

In many problems, however, a closed form Green's function is not available. Examples of this include scattering from a dielectric and scattering from a dielectric embedded in a conductor. Even in these cases, the functions' decaying nature can be exploited to construct a block low-rank approximation to the matrix of coupling Green's functions. This allows computation and storage of the matrix as well as its application to a vector in $O(n)$ or $O(n \log n)$ time (de-

pending on the specific method used). This principle forms the basis for dense solvers such as CDENSE [7].

It is easy to see that a multipole operator is the ideal preconditioner for the corresponding differential operator. For instance, a multipole operator with the Green's function $1/r$ may be used as a preconditioner for the Laplace operator to improve the rate of convergence of the iterative solver. The preconditioning step in each iteration requires $O(n)$ operations only since the cost of applying the multipole operator is $O(n)$.

In [15], we had discussed the application of a multipole-based preconditioner to the solution of the Poisson problem on a three dimensional domain. To further illustrate the use of the preconditioner for more challenging problems, we had considered the solution of the generalized Stokes problem which arises in the simulation of incompressible fluid flows. A parallel implementation of the algorithm achieved high efficiency on several platforms including the SGI Origin, IBM p690, and x86-based SMPs.

The experiments presented in this paper indicate that (i) merely truncating the kernel of the integral operator generating the preconditioner leads to poor convergence properties; (ii) far-field interactions, in the form of multipoles, are critical for rapid convergence; (iii) the importance and required accuracy of far-field interactions varies with the complexity of the problem; and (iv) the preconditioner resulting from a judicious mix of near and far-field interactions yields excellent convergence and parallelization properties.

The rest of the paper is organized as follows: Section 2 presents a brief overview of multipole methods and their use as preconditioners for the Poisson problem and the generalized Stokes problem; Section 3 discusses parallel formulations and performance of the preconditioned solver; Section 4 describes the results of varying accuracy parameters of multipole-based matrix-vector products (mat-vecs) on convergence and solution time; and conclusions are drawn in Section 5.

2. PRELIMINARIES

In this section, we provide an overview of multipole-based methods and their use as preconditioners. The reader should refer to [15] for details.

2.1 Hierarchical Multipole-Based Methods

We start with a brief overview of hierarchical multipole methods and their use in computing dense matrix-vector products in $O(n)$ time (or $O(n \log n)$ time) for an $n \times n$ matrix. Consider the problem of determining the charge on the surface of a conductor under given boundary conditions. The charge distribution is related to the potential by the following integral equation:

$$\psi(x) = \int_{\partial\Omega} G(x, x')\sigma(x')da', \tag{1}$$

where $\partial\Omega$ is the boundary of the domain Ω, $G(x, x')$ is the Green's function, $\psi(x)$ is the given potential at point x on the surface, $\sigma(x)$ is the unknown surface charge density, and da' represents the surface element. The Green's function $G(x, x')$ in three dimensions is given by $1/r$, where r is Euclidean distance between x and x'.

The surface charge density can be approximated using n known expansion functions:

$$\sigma(x) = \sum_{i=1}^{n} q_i r_i(x). \tag{2}$$

The coefficients of the expansion functions can be determined by using collocation conditions of the form

$$p = Cq, \tag{3}$$

where p is a vector of the known potentials at n panels, C is an $n \times n$ coupling matrix of Green's functions, and q is the vector of unknown coefficients. Since a large number of panels are required for acceptable accuracy, iterative techniques must be used to solve the system (3). At each iteration, the most time-consuming component is the matrix-vector product of the form $p' = Cq'$, which requires computing the potential at a set of n collocation points due to the charge density of q', as indicated in (2).

There are several ways of computing the required potentials. The first approximation is that the charge densities are uniform over a panel and that the charge is represented as a point charge at the centeroid of the panel. This results in a discrete potential estimation problem similar to the n-body problem. This approximation is valid only when the source and observation panels are far apart; i.e., $\|x - x'\|$ is much larger than size of the element over which the integration is performed. This is not the case when we are dealing with spatially proximate elements. In these cases, Gaussian quadratures with larger number of Gauss points must be used to compute the potential.

The required matrix-vector product can be computed using approximate hierarchical methods. The matrix C is never computed explicitly. Instead panels are aggregated and their impact on other panels is expressed in terms of multipole expansions. We first introduce approximate hierarchical methods for particle dynamics and then illustrate how they can be used to compute matrix-vector products.

A number of hierarchical approximation techniques have been proposed for particle dynamics. Of these, the methods of Barnes and Hut [1], and the Fast Multipole Method [10] have gained widespread popularity. In this paper, we use an augmented version of the Barnes-Hut method for computing the mat-vecs. The method works in two phases: the tree construction phase and the force computation phase. In the tree construction phase, a spatial tree representation of the domain is derived. At each step in this phase, if the domain contains more than s particles (for some preset constant s), it is recursively divided into eight equal parts. This process continues until each part has at most s elements. The resulting tree is an unstructured oct-tree. Each internal node in the tree computes and stores an approximate multipole series representation of the particles contained in that sub-tree. Once the tree has been constructed, the force or potential at each particle can be computed as follows: a 'multipole acceptance criterion' is applied to the root of the tree to determine if an interaction can be computed; if not, the node is expanded and the process is repeated for each of the eight children. The multipole acceptance criterion for the Barnes-Hut method computes the ratio of the distance of the point from the center of mass of the box to the dimension of the box. If this ratio is greater than $1/\alpha$, where α is a constant less than unity, an interaction can be computed. The Barnes-Hut method is illustrated in Fig. 1.

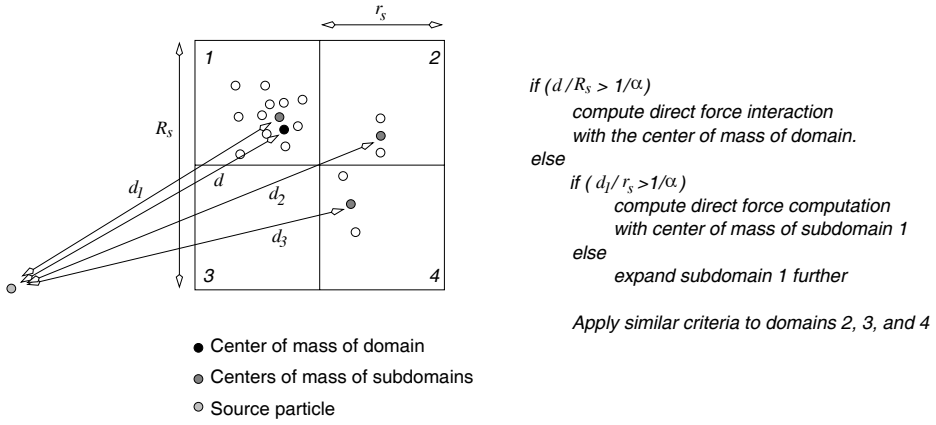

Figure 1: Illustration of the Barnes-Hut method.

For computing mat-vecs, particles correspond to Gauss points within elements and the force model between them corresponds to numerical integration. Computing a mat-vec in this manner involves the following steps:

1. Construct a hierarchical representation of the domain: Given a domain discretization into elements, element centers correspond to particle coordinates. An oct-tree is constructed based on these element centers. Each node in the tree stores the extremities along the x, y, and z dimensions of the subdomain corresponding to the node.

2. The number of particles in the tree is equal to the product of the number of elements and the number of Gauss points in the far field. In the case of a single Gauss point in the far field, the multipole expansions are computed with the center of the element as the particle coordinate.

3. For computing the mat-vec, we need to compute the potential at each of the n observation points. This is done using a variant of the Barnes-Hut method. The hierarchical tree is traversed for each of the elements. If an element falls within the near field of the observation element, integration is performed using direct Gaussian quadrature. The far-field contributions are computed using the multipole expansions. The α criterion of the Barnes-Hut method is slightly modified. The size of the subdomain is now defined by the extremities of all elements corresponding to the node in the tree. This is unlike the original Barnes-Hut method which uses the size of the box for computing the α criterion.

2.2 Multipole-Based Preconditioners

Multipole-based approximation schemes described in the preceding section can be directly used as solvers or preconditioners for the Laplace operator. Consider the following Laplace problem:

$$\Delta u = 0 \text{ in } \Omega, \qquad u = g \text{ on } \partial\Omega. \tag{4}$$

Physically, this problem is identical to one of estimating the potential in the interior of a perfectly conducting body, given the boundary potential. Since Δu corresponds to the charge and all charge resides on the boundary of a conductor, the correspondence is natural. The sparse linear system associated with this problem, $Ax = 0$, discretizes the Laplacian operator in the interior of the domain. To solve this sparse system, we can place m charges on the boundary of the conductor. We must now estimate these charges, subject to the constraint that the potential induced by these charges satisfies the boundary conditions. This reduces the problem to an equivalent dense linear system of the form $p = Cq$, discussed before. Here, C is the dense matrix of coupling Green's functions ($1/r$ in three dimensions). Iterative methods for solving this system require a mat-vec with matrix C, which can be computed using multipole methods. Once boundary charges have been estimated, the potential at interior points can be estimated easily using multipole methods as well. If a higher level of accuracy is desired, this multipole-based solver can be used as a preconditioner for an outer sparse solver.

Now, consider the problem of computing a function u on a domain Ω such that

$$\Delta u = f \text{ in } \Omega, \qquad u = g \text{ on } \partial\Omega, \tag{5}$$

where f and g are known functions. The associated linear system of equations has the form

$$Ax = b, \tag{6}$$

in which b and x can again be interpreted as vectors of charges and potentials, respectively, at n points in the interior of the domain. In contrast to the previous case, the potential at the boundary points is a result of interior as well as boundary charges. To reduce the problem to one defined on the boundary nodes, we must first cancel the effect on boundary potential of the internal nodes. To do this, we locate m points on the boundary where we estimate the potential due to internal nodes. We subtract this potential from the specified boundary conditions f to define a dense problem over the m boundary nodes. The charges at these boundary nodes are therefore determined by solving an $m \times m$ linear system of the form (3) with $p = g - g'$ where g' is the potential on the boundary due to interior charges. Once the boundary charges are known, x can again be computed as the potential due to the interior and boundary charges.

The linear system for the boundary charges is solved us-

ing an iterative method. The multipole-based approximations that are used to compute matrix-vector products with the dense coefficient matrix introduce errors in the solution. Nevertheless, one can use this approach as a preconditioning step for an outer iterative method which is used to solve (6). Since the linear system for the preconditioning step is defined only on the boundary, the inner iterative solve is not expensive.

2.2.1 Application to Incompressible Flows

The generalized Stokes problem arises in the solution of time-dependent Navier-Stokes equations for incompressible fluid flows. It consists of solving the following linear system:

$$\begin{bmatrix} \frac{1}{\Delta t}M + \nu L & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{7}$$

where u is the fluid velocity, p is the pressure, Δt is the time step, ν is the viscosity, and M, L, and B^T are the mass, Laplace, and divergence matrices, respectively. The linear system is large, sparse, and indefinite due to the incompressibility constraint $B^T u = 0$, which forces the velocity to lie in a discrete divergence-free subspace. While the primary challenge in developing robust preconditioners for this system is due to the indefiniteness of the matrix, parameters such as viscosity and time step also influence the effectiveness of the preconditioner.

The linear system in (7) can be transformed to the following *reduced system* by restricting velocity to the discrete divergence-free subspace:

$$P^T \left[\frac{1}{\Delta t}M + \nu L\right] Px = P^T f, \qquad u = Px. \tag{8}$$

Here, P is a basis for the discrete divergence-free subspace which is identical to the null space of B^T. While the symmetric positive definite character of the reduced system allows use of the CG method, it introduces additional complexity in constructing preconditioners.

In order to compute a discrete divergence-free basis P with modest computational and storage requirements, we observe that circulating flows or vortices can be used as a divergence-free basis in the continuum space. The discrete counterpart uses the edges of a mesh to define circulating flows that are divergence-free in the discretized domain. One can obtain these flows by computing the null space of submatrices of B^T defined over local regions of the mesh. Each such flow is represented as a vector, and the set of these vectors forms the columns of P (see, e.g., [14] for details). The matrix P is sparse due to the local support of the flows. In addition, matrix-vector products with P can be computed directly from the local flow vectors without assembling the matrix itself. Due to the unavailability of P, several commonly used preconditioning techniques such as those based on incomplete factorizations are no longer viable.

The similarity of this approach with the vorticity-velocity function formulation of the Navier-Stokes equations for incompressible flows can be exploited to develop a preconditioner for the reduced system. Observe that the mat-vecs Px and $P^T y$ compute the discrete curl of the functions represented by x and y, respectively. Furthermore, the mat-vec $P^T Px$ represents the application of the Laplace operator, say L_s, defined on the discrete divergence-free space. The reduced system may be approximated as shown below:

$$\frac{1}{\Delta t}P^T MP + \nu P^T LP \approx \frac{1}{\Delta t}M_s L_s + \nu L_s^2,$$

and preconditioned by the following matrix:

$$G = \left[\frac{1}{\Delta t}M_s + \nu L_s\right] L_s, \tag{9}$$

where M_s is the equivalent mass matrix for the flow vectors. Since the preconditioned system is spectrally equivalent to a symmetric positive definite matrix, one can use preconditioned CG to solve the reduced system (8). Experiments reported in [14] show that the preconditioner is very effective over a large range of values for Δt and ν.

To illustrate the effectiveness of the preconditioner, we consider the driven cavity problem on a three-dimensional cube. We use the Marker-and-Cell (MAC) scheme in which the domain is discretized by an $n \times n \times n$ uniform mesh. Pressure unknowns are defined at the nodes and velocity unknowns are defined at the mid-point of the edges. The x-component of velocity is defined on the edges along x-axis. Similarly, the y- and z-components of the velocity are defined on the edges along y-axis and z-axis, respectively. This gives a linear system with n^3 pressure unknowns and $3n^2(n-1)$ velocity unknowns.

The mesh is made up of an $(n-1) \times (n-1) \times (n-1)$ array of cubes. The local divergence-free flows are defined on the faces of these cubes. The normals to the faces are used to divide the flows into x, y, and z components. The preconditioning step requires computing

$$z = \left[\Delta t^{-1} I + \nu L_s\right]^{-1} L_s^{-1} r,$$

where L_s is a Laplace matrix with a block diagonal structure: $L_s = \text{diag}[L_x, L_y, L_z]$. The matrix L_x is the Laplace operator for the x-component of the divergence-free flows, and is defined on a mesh with nodes at the mid-points of the faces supporting these flows. The matrices L_y and L_z are defined in a similar manner. Dirichlet boundary conditions are assumed in each case.

The influence of the mesh width h, viscosity ν and time step Δt is indicated by the following condition number estimate:

$$\kappa\left(\frac{1}{\Delta t}M + \nu L\right) = \frac{\tau + 12}{\tau + h^2}, \qquad \tau = \frac{h^2}{\nu \Delta t}.$$

Tables 1 and 2 have been reproduced from [15] to illustrate the effectiveness of the preconditioner. Table 1 shows that the number of iterations required by the CG method to solve the reduced system in (8) is dramatically reduced by the use of the preconditioner in (9). The preconditioning step uses the multipole-based preconditioner for the Poisson problem and an inner CG method for the Helmholtz problem. The diagonal dominance of the Helmholtz problem assures rapid convergence without additional preconditioning. The outer CG iterations were terminated when the relative residual norm reduced below 10^{-4}.

Table 2 shows the number of iterations required by the preconditioned CG method for several instances of the generalized Stokes problem. Depending upon the value of τ, the condition number of the reduced system ranges from $O(h^{-2})$ to $O(h^{-4})$. The preconditioner ensures a stable convergence rate which is nearly independent of problem parameters.

Table 1: Effectiveness of the preconditioner for the generalized Stokes problem ($\tau = 10^{-3}$).

Mesh Size	Unknowns	Iterations	
		Unprec.	Prec.
$8 \times 8 \times 8$	1856	66	8
$16 \times 16 \times 16$	15616	208	12
$32 \times 32 \times 32$	128000	772	17

Table 2: Effectiveness of the preconditioner for various instances of the generalized Stokes problem.

Mesh Size	$\tau = h^2/(\nu \Delta t)$				
	10^{-3}	10^{-1}	10^{0}	10^{1}	10^{3}
$8 \times 8 \times 8$	8	8	6	5	5
$16 \times 16 \times 16$	12	10	8	6	6
$32 \times 32 \times 32$	17	13	10	7	7

3. PARALLEL PERFORMANCE

The multipole-based preconditioning step is the most expensive component of the iterative solver. The execution time in the experiments in Tables 1 and 2 is dominated by the hierarchical multipole-based matrix-vector products with the dense preconditioner, with over 90% of total time spent in the preconditioning step. The remaining components of the iterative solver include sparse matrix-vector products and vector operations, which can be parallelized easily once the mesh has been partitioned across processors. Conventional mesh partitioning packages such as METIS and CHACO can be used to obtain good partitions.

This section gives an outline of an efficient parallel formulations of the multipole-based preconditioning step (see [15] for details). As described in Section 2.2, the multipole-based preconditioner is posed as a boundary-enforced solve in which unknown boundary charges are computed to satisfy the boundary condition on potential. The interior and boundary charges are then used to evaluate potential inside the domain. The kernel operation in each case is a dense matrix-vector product, which is computed using a multipole-based hierarchical method. A single instance of this method requires a tree construction and a tree traversal. Since the tree construction phase is relatively inexpensive (requiring less than 2% of total time in our experiments) we focus on efficient parallelization of the tree traversal phase. It has been observed by us and others in the past [9, 18, 19] that an effective parallelization strategy can be derived from the observation that two spatially proximate particles are likely to interact with largely the same nodes in the tree. This leads to a partitioning strategy in which spatially proximate particles are aggregated into a single concurrent computational unit.

In our parallel formulation, we first sort the nodes in the mesh in a proximity preserving order such as a Peano-Hilbert ordering, and group a set of m nodes into a single thread. The parameter m should be chosen in such a way that the number of threads is greater than the number of processors p by a factor $\log p$. This is generally adequate for ensuring load balance in the tree traversal phase as well as amortizing the cost of remote communication.

This parallel formulation of multipole methods on SMP multiprocessors such as the SGI Origin and IBM p690 is observed to yield high performance and excellent scalability. The multipole operator is implemented in POSIX threads and is portable across a range of serial and parallel platforms. The performance of the operator is a function of a number of parameters – the degree of multipole series, the α parameter of the Barnes-Hut method, and the underlying architecture, in addition to the problem size.

Increasing the degree d of the multipole series increases the volume of data (as $O(d^2)$) that must be communicated from remote nodes. While the corresponding computation also increases (as $O(d^2)$), the associated per-word communication time exceeds per-degree computation time. Therefore the efficiency can be expected to decrease with increasing degree. (Note that this does not hold for translation of multipole operators since translation complexity varies as $O(d^4)$.) Decreasing the α parameter of the Barnes-Hut method increases the range of interaction, and therefore results in a lower efficiency for the corresponding parallel formulation.

The preconditioning step in the CG method consists of three different types of multipole-based dense matrix-vector products: (i) computation of boundary potential from interior charges, (ii) computation of boundary charges via an inner GMRES method that requires boundary potential computation from boundary charges at each iteration, and (iii) computation of interior potential from all the charges. Since the last step of the computation involves the greatest number of potential evaluations, it dominates the cost of the preconditioning step. The objective of this set of experiments is to demonstrate that it is possible to achieve high efficiency in an algorithm that requires each of these three steps (with associated data redistribution).

Table 3 provides a summary of the parallel performance of the algorithm (see [15] for details). On 8 processors of the IBM p690, the code exhibits parallel efficiency in excess of 80% in most cases. The parallel efficiency of the code is retained on x86 Solaris SMPs with up to 8 Pentium processors indicating that for multipole-based preconditioners, these SMPs can be an inexpensive alternative to high-end multiprocessors.

4. IMPACT OF MULTIPOLE EXPANSION PARAMETERS

To understand the impact of various parameters associated with the multipole preconditioner on the rate of convergence of the iterative methods, we run a number of test cases of a preconditioned Poisson solve. We present the results of these experiments in Table 4. Specifically, we attempt to understand the following: (i) role of far-field potential in convergence, (ii) interplay between accuracy of inner (preconditioning) solve and time to solution, (iii) required accuracy on far-field potential estimate, and (iv) behavior of iteration counts with number of unknowns.

The inner (preconditioning) solve is characterized by the following parameters – the stopping criteria, the restart associated with GMRES, and the accuracy parameters of the dense mat-vec. In this study, we focus on the accuracy parameters of the dense mat-vec, namely, the degree of the multipole series and the α criterion (which determines the cutoff between near-field and far-field). The other parameters have been the subject of numerous studies (see, for

Table 3: Runtime (in seconds) and efficiency of the parallel algorithm on IBM p690 and x86 Solaris shared-memory multiprocessors.

IBM p690

Mesh Size	$p = 1$	$p = 8$	Efficiency
$30 \times 30 \times 30$	19.96	3.25	0.77
$40 \times 40 \times 40$	31.18	4.80	0.81
$50 \times 50 \times 50$	68.88	9.69	0.81

4-processor SMP (550 MHz P3)

Mesh Size	$p = 1$	$p = 4$	Efficiency
$30 \times 30 \times 30$	41.39	13.19	0.78
$40 \times 40 \times 40$	67.37	20.76	0.81
$50 \times 50 \times 50$	129.87	39.49	0.82

8-processor SMP (750 MHz Xeon)

Mesh Size	$p = 1$	$p = 8$	Efficiency
$30 \times 30 \times 30$	28.78	5.97	0.60
$40 \times 40 \times 40$	47.23	7.19	0.82
$50 \times 50 \times 50$	115.29	16.70	0.86

instance analysis of FMGMRES). We examine the impact of the multipole degree and α criterion on problem sizes ranging from $20 \times 20 \times 20$ (8K unknowns) to $40 \times 40 \times 40$ (64K unknowns). Trends evident from our result show that the performance gains of our preconditioner increase sharply with increasing number of unknowns.

We present the results of these experiments in Table 4. The execution time gives solution time on a 2.4GHz P4 workstation with 1GB RAM running Solaris. The column marked "Dense" refers to the case when α was chosen to ensure that all interactions were near field interations thereby eliminating error in the dense mat-vec.

Table 4 illustrates several important facts. None of the instances converge to solution in 20 iterations if far-field is not accounted for (multipole degree 0). This demonstrates the need for incorporating far field potentials for convergence. In many cases, while a less accurate inner mat-vec resulted in larger number of outer iterations, the associated time was lower. However, if the accuracy of the mat-vec was very low, convergence is not guaranteed. For this reason, it is important to select the right accuracy parameters. Finally, in most cases, a low-order approximation of the far field is adequate, and ideal for fast convergence. Multipole series present an excellent means for achieving this approximation.

5. CONCLUSIONS

In this paper, we have examined the impact of various parameters associated with hierarchical multipole methods on the preconditioner's performance and associated runtime. In particular, we demonstrate the following results: (i) accounting for far-field is critical for convergence – specifically, convergence was not observed in any of the test cases when far-field was not taken into account, (ii) while convergence in terms of iteration count is a function of accuracy of inner iteration (in each case the dense inner mat-vec yields best iteration counts), minimum time is achieved at judiciously selected values of multipole degree and α which determines the cutoff between near-field and far-field, (iii) in general, a low-order approximation of the far-field is adequate for fast convergence, however, in some cases, convergence at this coarse approximation is not guaranteed, (iv) as the number of unknowns become large, the number of iterations in the preconditioned solve stays largely constant, resulting in considerable time savings compared to unpreconditioned solves, and (v) multipole-based preconditioners are extremely simple to parallelize and can yield excellent parallel performance.

Table 4: Impact of multipole expansion parameters on the rate of convergence and acceleration. Runtime (in seconds) and iterations (in paranthesis) are shown.

Mesh size: $20 \times 20 \times 20$

Degree	α=0.91	α=0.77	α=0.67	Dense
0	—	—	—	12.54 (3)
1	—	—	—	
2	3.23 (3)	4.03 (3)	5.19 (3)	
Unpreconditioned: 0.82 (799)				

Mesh size: $30 \times 30 \times 30$

Degree	α=0.91	α=0.77	α=0.67	Dense
0	—	—	—	93.36 (2)
1	8.42 (3)	11.28 (3)	15.30 (3)	
2	17.63 (3)	22.71 (3)	30.03 (3)	
Unpreconditioned: 13.35 (1895)				

Mesh size: $40 \times 40 \times 40$

Degree	α=0.91	α=0.77	α=0.67	Dense
0	—	—	—	454.28 (2)
1	20.82 (3)	28.83 (3)	40.00 (3)	
2	34.50 (3)	23.61 (2)	32.51 (2)	
Unpreconditioned: 59.87 (3451)				

— indicates no convergence in 20 iterations

6. REFERENCES

[1] J. Barnes and P. Hut. A hierarchical O(n log n) force calculation algorithm. *Nature*, Vol. 324, 1986.

[2] R. Bramley and V. Menkov. Parallel Preconditioners with Low Rank Off-Diagonal Blocks. Technical Report TR446, Indiana University, 1996.

[3] R. Coifman, V. Rokhlin, and S. Wandzura. The fast multipole algorithm for the wave equation: A pedestrian prescription. *IEEE Antennas and Propagation Magazine*, 35(3), 1993.

[4] B. Dembart and E. Yip. A 3D fast multipole method for electromagnetics with multiple levels. In *Proceedings of the 11th Annual Review of Progress in Applied Computational Electromagnetics*, Monterey, CA, 1995.

[5] N. Engheta, W. Murphy, V. Rokhlin, and M. Vassiliou. The fast multipole method for electromagnetic scattering problems. *IEEE Transactions on Antennas and Propagation*, 40(6), 1992.

[6] G. Karypis and V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. *SIAM Review*, 41(2):278–300, 1999.

[7] S. Goreinov, E. Tyrtyshnikov, and A. Yeremin. Matrix-free iterative solution strategies for large dense linear systems. Technical Report EM-RR 11/93, Elegant Mathematics, Inc., 1993.

[8] A. Grama, A. Gupta, G. Karypis, and V. Kumar, Introduction to parallel computing, Addison Wesley, 2003.

[9] A. Grama, V. Kumar, and A. Sameh. Parallel hierarchical solvers and preconditioners for boundary element methods. *SIAM Journal on Scientific Computing*, 20(1):337–358, 1998.

[10] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Physics*, Vol. 73:325–348, 1987.

[11] B. Hendrickson and R. Leland. The CHACO user's guide version 2.0, Technical Report SAND94-2692. Sandia National Laboratories, Albuquerque, NM, 1994.

[12] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, Vol. 49:409–436, 1952.

[13] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, Vol. 7:856–869, 1986.

[14] S. R. Sambavaram. High Performance Parallel Algorithms for Incompressible Flows. M. S. Thesis. Texas A&M University, 2002.

[15] S. R. Sambavaram, V. Sarin, A. H. Sameh, and A. Grama. Multipole-based preconditioners for large sparse linear systems. *Parallel Computing*, Vol. 29:1261–1273, 2003.

[16] V. Sarin and A. H. Sameh. Hierarchical divergence-free bases and their application to particulate flows. *Journal of Applied Mechanics*, Vol. 70:44–49, 2003.

[17] V. Sarin and A. H. Sameh. An efficient iterative method for the generalized Stokes problem. *SIAM Journal of Scientific Computing*, 19(1):206–226, 1998.

[18] J. P. Singh, C. Holt, J. L. Hennessy, and A. Gupta. A parallel adaptive fast multipole method. In *Proceedings of the Supercomputing '93 Conference*, 1993.

[19] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree N-body algorithm. In *Proceedings of the Supercomputing '93 Conference*, 1993.