Hierarchical Divergence-Free Bases and Their Application to Particulate Flows

V. Sarin[1]
Department of Computer Science,
Texas A&M University,
College Station, TX 77843
e-mail: sarin@cs.tamu.edu

A. H. Sameh
Department of Computer Science,
Purdue University,
West Lafayette, IN 47907
e-mail: sameh@cs.purdue.edu

The paper presents an algebraic scheme to construct hierarchical divergence-free basis for velocity in incompressible fluids. A reduced system of equations is solved in the corresponding subspace by an appropriate iterative method. The basis is constructed from the matrix representing the incompressibility constraints by computing algebraic decompositions of local constraint matrices. A recursive strategy leads to a hierarchical basis with desirable properties such as fast matrix-vector products, a well-conditioned reduced system, and efficient parallelization of the computation. The scheme has been extended to particulate flow problems in which the Navier-Stokes equations for fluid are coupled with equations of motion for rigid particles suspended in the fluid. Experimental results of particulate flow simulations have been reported for the SGI Origin 2000.
[DOI: 10.1115/1.1530633]

1 Introduction

The simulation of incompressible fluid flow is a computationally intensive application that has challenged high-performance computing technology for several decades. The ability to solve large, sparse linear systems arising from Navier-Stokes equations is critical to the success of such simulations. Linear systems of equations are typically solved by iterative methods that have the advantage of requiring storage proportional to the number of unknowns only. One can use the conjugate gradients method (CG), [1], for symmetric positive definite systems and the generalized minimum residual method (GMRES), [2], for nonsymmetric systems. Although these methods are memory-efficient in comparison to direct methods such as Gaussian elimination, the rate of convergence to the solution can be unacceptably slow. Often one needs to accelerate convergence by using some preconditioning strategy that computes an approximate solution at each step of the iterative method. It is well known that commonly used preconditioning schemes such as those based on incomplete factorization (see, e.g., [3]) may not be effective for indefinite linear systems with eigenvalues on both sides of the imaginary axis. Since the eigenvalue distribution of linear systems arising from the Navier-Stokes equations could produce such systems, it is a challenge to devise robust and effective preconditioners for incompressible flows.

The Navier-Stokes equations governing incompressible fluid are given as follows:

$$\rho \frac{\partial u}{\partial t} + \rho u \cdot \nabla u = \rho g - \nabla p + \nabla \cdot \tau, \tag{1}$$

$$\nabla \cdot u = 0, \tag{2}$$

where u denotes fluid velocity, p denotes pressure, ρ denotes fluid density, g represents gravity, and τ represents the extra-stress tensor. For Newtonian flows, τ takes the form

$$\tau = \mu(\nabla u + \nabla u^T). \tag{3}$$

After appropriate linearization and discretization, the following system must be solved:

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{4}$$

where u is the velocity vector, p is the pressure vector, B^T and B are discrete operators for divergence and gradient, respectively. The matrix A denotes the discrete operator on velocity in (1). This linear system is indefinite due to the incompressibility constraint on velocity which is enforced by $B^T u = 0$ in (4).

A convenient way to circumvent the indefiniteness of the linear system due to these constraints is to restrict the fluid velocity to divergence-free subspace. There are a number of techniques to construct divergence-free velocity functions. These include discretely divergence-free functions obtained from specially constructed finite element spaces, [4,5], as well as continuous functions derived from solenoidal functions such as those used in vortex methods. The problem is reduced to solving the momentum equation for divergence-free velocity functions without the need to include continuity constraints. In many cases, the resulting reduced linear systems are no longer indefinite. Furthermore, these reduced systems can be preconditioned to accelerate the convergence of iterative solvers.

The existing schemes for divergence-free functions are complicated and difficult to generalize to arbitrary discretizations. In this paper, we present an algebraic scheme to compute a basis for discretely divergence-free velocity. Our scheme constructs a basis for the null space of the matrix representing the linear constraints imposed on fluid velocity by (2). The algebraic nature of the scheme ensures applicability to a wide variety of methods including finite difference, finite volume, and finite elements methods. Since the choice of the basis preconditions the reduced linear system implicitly, it is possible to compute an optimal basis that leads to rapid convergence of the iterative solver. A more modest target is to compute a well-conditioned basis that preconditions the reduced system to some degree. The paper presents an algorithm to construct a hierarchical basis of divergence-free functions that is well conditioned too.

The paper is organized as follows: Section 2 presents the algorithm for hierarchical divergence-free basis and discusses computational aspects. Section 3 outlines the extension of the scheme to particulate flow problems. In Section 4, we describe the behavior of our scheme on benchmark problems in particulate flows. Conclusions are presented in Section 5.

[1]To whom correspondence should be addressed.

Contributed by the Applied Mechanics Division of THE AMERICAN SOCIETY OF MECHANICAL ENGINEERS for publication in the ASME JOURNAL OF APPLIED MECHANICS. Manuscript received by the ASME Applied Mechanics Division, July 2, 2001; final revision, Apr. 9, 2002. Associate Editor: T. E. Tezduyar. Discussion on the paper should be addressed to the Editor, Prof. Robert M. McMeeking, Department of Mechanical and Environmental Engineering University of California–Santa Barbara, Santa Barbara, CA 93106-5070, and will be accepted until four months after final publication of the paper itself in the ASME JOURNAL OF APPLIED MECHANICS.

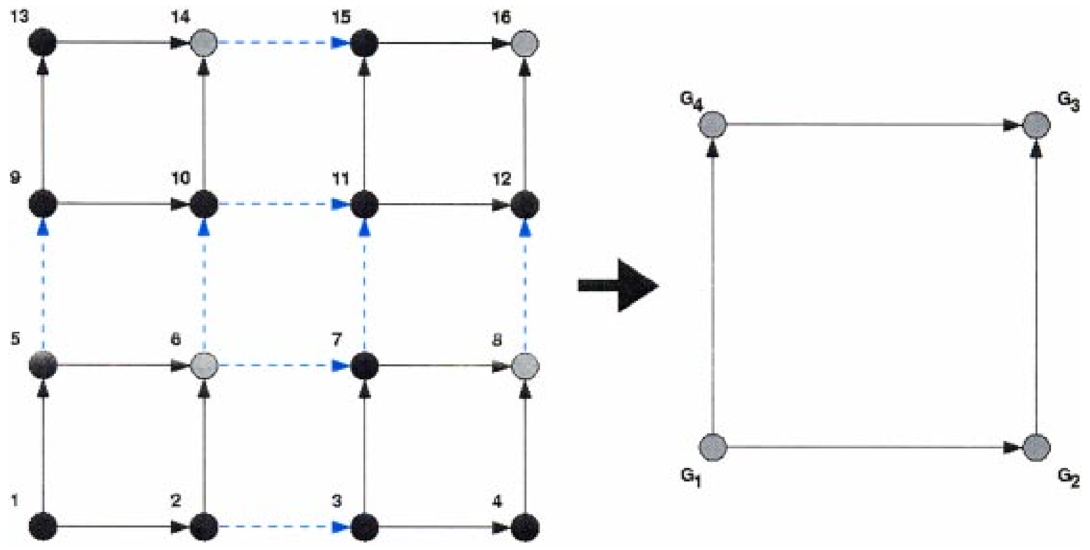

Fig. 1 The coarsening of a 4×4 mesh to a 2×2 mesh

2 Hierarchical Divergence-Free Basis

A straightforward way to construct discretely divergence-free bases is to compute the null space of the discrete divergence operator matrix B^T. This null space can be computed via full QR factorization or singular value decomposition (SVD) of B^T, [6]. For an m by n matrix $(m<n)$, the computation is proportional to m^2n while storage is proportional to mn. For the matrix B^T, the number of rows m corresponds to the number of pressure basis functions and the number of columns n corresponds to the number of velocity basis functions. Since B^T is large and sparse with nonzeros proportional to m, both QR factorization and SVD are unsuitable due to the prohibitive requirements of computation and storage.

The nonzero structure of B^T can be exploited to construct a null-space basis efficiently. The following outline of the algorithm to construct a hierarchical divergence-free basis follows the description in [7]. Suppose one can reorder the columns of B^T such that

$$B^T=[B_{\text{in}}^T \quad B_{\text{out}}^T], \tag{5}$$

where B_{in}^T is a block diagonal matrix with "small" nonzero blocks on the diagonal. Given the following singular value decomposition of B_{in}:

$$B_{\text{in}}=\text{USV}^T=[U_1 \quad U_2]\begin{bmatrix} S_1 & \\ & 0 \end{bmatrix}[V_1 \quad V_2]^T, \tag{6}$$

where S_1 is a nonzero diagonal matrix, B^T can be represented as follows:

$$B^T=VV^T[B_{\text{in}}^T \quad B_{\text{out}}^T]\begin{bmatrix} UU^T & \\ & I \end{bmatrix}=[V_1 \quad V_2]\left[\begin{array}{cc|c} S_1 & 0 & V_1^TB_{\text{out}}^T \\ 0 & 0 & V_2^TB_{\text{out}}^T \end{array}\right] \times\begin{bmatrix} U^T & \\ & I \end{bmatrix}. \tag{7}$$

Since

$$\left[\begin{array}{cc|c} S_1 & 0 & V_1^TB_{\text{out}}^T \\ 0 & 0 & V_2^TB_{\text{out}}^T \end{array}\right]\left[\begin{array}{cc} 0 & -S_1^{-1}V_1^TB_{\text{out}}^T \\ I & 0 \\ \hline 0 & I \end{array}\right]=\begin{bmatrix} 0 & 0 \\ 0 & V_2^TB_{\text{out}}^T \end{bmatrix}, \tag{8}$$

the null-space basis of B^T is given by

$$P=\begin{bmatrix} U & \\ & I \end{bmatrix}\begin{bmatrix} 0 & -S_1^{-1}V_1^TB_{\text{out}}^T \\ I & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & P^{(1)} \end{bmatrix}, \tag{9}$$

where $P^{(1)}$ is a null-space basis of the matrix $B^{(1)^T}=V_2^TB_{\text{out}}^T$. With this transformation, the problem of computing the null-space of the original matrix B^T is reduced to a problem of smaller size. By applying the same technique to compute the null-space of $B^{(1)^T}$, one gets a recursive strategy for constructing the null-space of B^T.

The preceding approach is viable only if the transformation is inexpensive and the reduced matrix $B^{(1)^T}$ is easy to compute and process subsequently. These criteria are met simultaneously by exploiting the relation of the nonzero structure of B^T with the discretization mesh. The pressure nodes in the mesh are clustered into groups of a few nodes each, and the velocity basis functions with support within a cluster are placed in B_{in} whereas those with support across clusters are placed in B_{out}. The resulting matrix B_{in} is block diagonal with small block sizes. Each diagonal block represents the divergence operator for the corresponding cluster of nodes. Due to the small size of the diagonal blocks, the SVD can be computed very efficiently.

To illustrate the technique, we reproduce an example of a 4×4 mesh from [8] (see Fig. 1). Pressure unknowns are defined at the nodes. The x-component of velocity is defined on the horizontal edges and y-component of velocity is defined on the vertical edges. The nodes are clustered into four groups: $G_1=\{1,2,5,6\}$, $G_2=\{3,4,7,8\}$, $G_3=\{9,10,13,14\}$, and $G_4=\{11,12,15,16\}$. The solid edges indicate velocity unknowns for B_{in} and the dashed edges indicate velocity unknowns for B_{out}. The associated matrices are

$$B_{\text{in}}=\begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & B_3 & \\ & & & B_4 \end{bmatrix},$$

$$B_{\text{out}}=\begin{bmatrix} -C_1 & C_2 & & \\ & & -C_1 & C_2 \\ -C_3 & & C_4 & \\ & -C_3 & & C_4 \end{bmatrix}, \tag{10}$$

in which

$$B_i=\begin{bmatrix}-1 & 1 & 0 & 0\\ 0 & 0 & -1 & 1\\ -1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\end{bmatrix}, \quad i=1,\ldots,4, \tag{11}$$

and

$$C_1=\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, C_2=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix},$$
$$C_3=\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}, C_4=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}. \tag{12}$$

The SVD of each block in $B_{\rm in}$ is given as

$$B_i=U_iS_iV_i^T=\begin{bmatrix}-1/2 & 1/2 & -1/2 & 1/2\\ 1/2 & 1/2 & -1/2 & -1/2\\ -1/2 & 1/2 & 1/2 & -1/2\\ 1/2 & 1/2 & 1/2 & 1/2\end{bmatrix}\begin{bmatrix}2 & & & \\ & \sqrt{2} & & \\ & & \sqrt{2} & \\ & & & 0\end{bmatrix}$$
$$\times\begin{bmatrix}1/2 & -\sqrt{2} & 0 & 1/2\\ -1/2 & 0 & -\sqrt{2} & 1/2\\ -1/2 & 0 & \sqrt{2} & 1/2\\ 1/2 & \sqrt{2} & 0 & 1/2\end{bmatrix}^T, \tag{13}$$

that yields

$$B^{(1)^T}=V_2^TB_{\rm out}^T$$
$$=\frac{1}{2}\begin{bmatrix}-1 & -1 & 0 & 0 & -1 & -1 & 0 & 0\\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1\\ 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1\end{bmatrix}. \tag{14}$$

Note that the rows of $B^{(1)^T}$ correspond to the nodes 6, 8, 14, and 16 of the original mesh, and the columns correspond to the cross-cluster edges. It is easy to see that $B^{(1)^T}$ is a divergence matrix for the coarse mesh shown in Fig. 1. Since columns $2j$-1 and $2j$ are identical for $j=1,\ldots 4$, the columns of $B^{(1)^T}$ can be reduced to four nonzero columns by multiplying with an orthogonal matrix from the right. The resulting matrix is the divergence matrix of a 2×2 mesh which has been scaled by $1/\sqrt{2}$.

In general, the nonzero structure of the matrix $B^{(1)^T}$ retains the structure of a coarse mesh obtained from grouping clusters into single nodes. Furthermore, $B^{(1)^T}$ may be considered equivalent to a divergence operator matrix for the coarse mesh. Thus, the recursive strategy can be applied in a straightforward manner. Since $B^{(1)^T}$ can also be computed efficiently from the SVD of $B_{\rm in}$, each step of the recursive algorithm is very efficient.

The recursive algorithm to construct the divergence-free basis gives rise to a hierarchical basis that consists of basis functions defined on each level of the mesh hierarchy. In the actual implementation of the algorithm, the null-space matrix is never computed explicitly. It is available only in the form of product of matrices constructed from the SVD matrices and the equivalent divergence operator matrix $B^{(*)^T}$ at each level of the mesh hierarchy.

The size of meshes in the hierarchy decreases geometrically from the finest mesh to the coarsest mesh. Since the size of $B^{(*)^T}$ is proportional to the mesh size at each level, the cost of computing and storing SVDs is also proportional to the mesh size at the corresponding level. Thus, the overall computation and storage is proportional to the size of B^T. This is a significant improvement over the QR and SVD algorithms. However, it should be noted that this reduction in computational complexity is gained at the expense of generating a basis that is not orthonormal. The reader is referred to [7] for more details of this method.

Once the divergence-free basis P has been constructed, the linear system in (4) is transformed to the following reduced system:

$$P^TAPx=P^Tf, \quad u=Px, \tag{15}$$

which is solved by GMRES to obtain x. When A is symmetric and positive definite, one can use CG instead of GMRES. Pressure can be computed correctly by solving the least-squares problem

$$Bp\approx f-Au, \tag{16}$$

which is consistent since $P^T(f-Au)=0$. At each iteration, one needs to compute matrix-vector products of the form $y=Px$ and $z=P^Tw$. The computation follows a recursive structure in which matrix-vector products are computed at each level of the mesh hierarchy. The computation proceeds from the coarsest mesh to the finest mesh for the product $y=Px$ and in the reverse direction for the product $z=P^Tw$. Since the computational complexity of each product is proportional to the size of B^T, the cost of computing the matrix-vector product for the reduced system in (15) is proportional to the number of velocity unknowns. Furthermore, the concurrency in the hierarchical structure of this algorithm can be exploited to develop high-performance software for incompressible flows. Details of an efficient parallel formulation are presented in [9].

3 Particulate Flows

Divergence-free velocity basis can be used to solve linear systems arising in solid-fluid mixtures that consist of rigid particles suspended in incompressible fluids. The solution of these linear systems is extremely computationally intensive and accounts for majority of the simulation time. The motion of particles is governed by Newton's equations whereas the fluid obeys Navier-Stokes equations. Assuming no-slip on the surface of the particle, the fluid velocity at any point on the particle surface is a function of the particle velocity. For the sake of simplicity, this discussion assumes spherical particles. For the ith particle, the position X_i and velocity U_i is obtained by solving the following equations:

$$M_i\frac{dU_i}{dt}=F_i, \tag{17}$$

$$\frac{dX_i}{dt}=U_i, \tag{18}$$

where U_i includes both translation and angular components of the particle, M_i represents the generalized mass matrix, and F_i represents the force and torque acting on the particles by the fluid as well as gravity. Fluid velocity at the surface of the particle is related to the particle velocity as follows:

$$u_j=U_{t,i}+r_j\times U_{r,i} \tag{19}$$

where $U_{t,i}$ and $U_{r,i}$ are the translation and angular velocity components, respectively, and r_j is the position vector of the jth point relative to the center of the particle.

A simple way to represent the linear system arising in particulate flows is as follows. The systems for fluid and particles are written independently along with the constraint in (19) that forces fluid velocity on a particle surface to be dependent on the particle velocity. Thus,

$$\begin{bmatrix}A & B & 0\\ B^T & 0 & 0\\ 0 & 0 & C\end{bmatrix}\begin{bmatrix}u\\ p\\ U\end{bmatrix}=\begin{bmatrix}b\\ 0\\ g\end{bmatrix}, \text{ where } u=\begin{bmatrix}u_f\\ u_p\end{bmatrix}=\begin{bmatrix}I & \\ & W\end{bmatrix}\begin{bmatrix}u_f\\ U\end{bmatrix}, \tag{20}$$

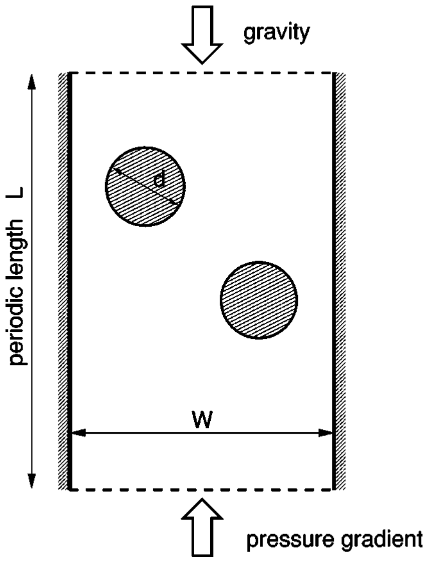

Fig. 2 Particles moving in a periodic channel

in which u_f is the fluid velocity in the interior of the fluid and W is the linear transformation from particle velocity U to fluid velocity u_p on particle surface given by (19). Using subscripts f and p to denote fluid interior and particle surface, respectively, the preceding system can be transformed to the following system:

$$\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & W^T & 0 & I \\ 0 & 0 & I & 0 \end{bmatrix} \begin{bmatrix} A_{ff} & A_{pf} & B_f & 0 \\ A_{fp} & A_{pp} & B_{fp} & 0 \\ B_f^T & B_{fp}^T & 0 & 0 \\ 0 & 0 & 0 & C \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & W & 0 \\ 0 & 0 & I \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} u_f \\ U \\ p \end{bmatrix} = \begin{bmatrix} b_f \\ g + W^T b_p \\ 0 \end{bmatrix}, \tag{21}$$

which can be simplified further to obtain the following system:

$$\begin{bmatrix} A_{ff} & A_{pf}W & B_f \\ W^T A_{fp} & W^T A_{pp} W + C & W^T B_{fp} \\ B_f^T & B_{fp}^T W & 0 \end{bmatrix} \begin{bmatrix} u_f \\ U \\ p \end{bmatrix} = \begin{bmatrix} b_f \\ g + W^T b_p \\ 0 \end{bmatrix}. \tag{22}$$

Note that this system has a form similar to the linear system in (4).

A hierarchical divergence-free basis can be computed for (22) without any difficulty. In this case, the null-space is computed for the constraint matrix $[B_f^T \; B_{fp}^T W]$. The basic algorithm remains unchanged although care has to be taken when coarsening the mesh with particles. The fluid nodes on the particle surface are absent from the mesh in this system. The presence of particles introduces a single node that is connected to all the fluid nodes that are adjacent to the particle surface. The algebraic scheme for computing the divergence-free basis ensures that the algorithm applies without any change to particulate flows as well.

4 Experiments

The hierarchical divergence-free basis method has been used to solve the linear systems arising in particulate flow simulations. The simulations involved incompressible fluid in a two-dimensional channel with a number of rigid particles moving freely under the action of gravitational force as well as force from the surrounding fluid (see Fig. 2).

The physical system is evolved from an initial state by the implicit backward Euler method. The first-order accuracy of this scheme was adequate because the time step was severely constrained by particle dynamics. At each time-step, a nonlinear system of equations was solved by an inexact Newton's method, [10]. At each iteration, a linear system of the form (22) was solved for the Jacobian of the nonlinear equations. In general, this Jacobian matrix is a saddle-point system with a nonsymmetric matrix A which tends to be real positive for a sufficiently small Reynolds number. The hierarchical divergence-free basis approach is used to transform the system in (22) to the reduced form shown in (15). The reduced system is solved by the GMRES method. The adaptive tolerance proposed in [11] was used as a stopping criteria.

The differential equations are approximated by the mixed finite elements method in which fluid velocity and pressure are represented by the P2/P1 pair of elements. The choice of quadratic velocity elements is necessary to capture the behavior of closely spaced particles. A nonuniform mesh is used to discretize the fluid domain resulting in a linear system that is large and sparse. The scheme proposed in [12] is used in an arbitrary Lagrange-Euler (ALE) framework to accommodate moving particles.

The parallel simulation code was developed using Petsc, [13]. Communication between processes was done by MPI, [14]. The mesh is generated using Triangle, [15] and partitioned using Parallel METIS, [16]. Further implementation details are available in [17].

Simulations were conducted for rigid particles falling in a 3.2 in. wide and 30 in. long two-dimensional vertical channel with a solid impenetrable bottom. The particles were assumed to be circular disks of diameter 0.25 in. and specific gravity 1.14. The initial position of particles was specified.

4.1 Single Particle Sedimentation. This benchmark simulates the sedimentation of a single particle from rest whose center is at a distance of 0.8 in. from the left wall and 30 in. from the bottom of the channel. At the first time-step, the computational mesh had 2461 elements and 1347 nodes. The number of unknowns in the unconstrained problem was 9418. Figure 3 shows

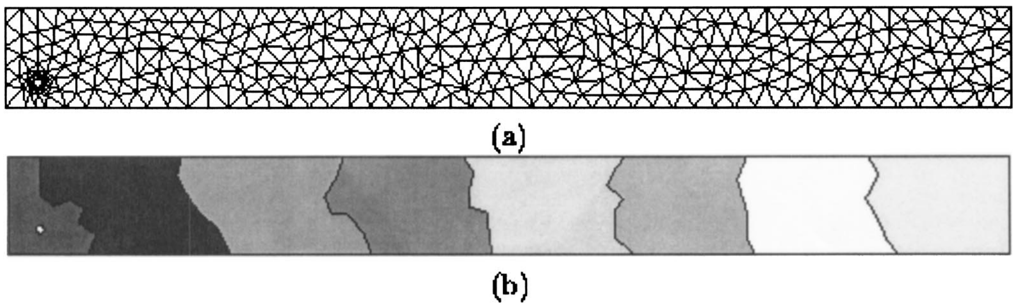

Fig. 3 Sedimentation of a single particle: (*a*) mesh with 2461 elements and 1347 nodes, (*b*) partitioning into eight domains. The gravitational force pulls the particles towards the right.

Table 1 Single particle sedimentation on the SGI origin 2000.

Processors	Time	Speedup	Efficiency
1	1819 s	1.0	1.00
2	822 s	2.2	1.11
4	502 s	3.6	0.91
8	334 s	5.3	0.66

Table 2 Multiple particle sedimentation on the SGI origin 2000

Processors	Time	Speedup	Efficiency
1	3066 s	1.0	1.00
2	1767 s	1.7	0.85
4	990 s	3.0	0.75
8	570 s	5.3	0.66

the initial mesh and the associated partitioning into eight subdomains.

To illustrate the numerical and parallel performance of the algorithm, the experiment was restricted to the first five time-steps starting with the particle and fluid at rest. Each time-step was 0.01 sec. Table 1 presents the performance of the algorithm on eight processors of the SGI Origin 2000 multiprocessor. The parallel efficiency is expected to be much higher for a larger problem. In this experiment, superlinear speedup is observed due to effective cache utilization when data on individual processors is small enough to fit within the cache.

4.2 Multiple Particle Sedimentation. The next benchmark simulates the sedimentation of 240 particles arranged in a stationary *crystal*. The crystal consists of an array of 240 particles in 20 rows and 12 columns. The centers of the particles coincide with the nodes of a uniform mesh with 20 rows and 12 columns. The centers of the particles are approximately 0.06154 in. apart in each direction. The distance between the walls and the nearest particles is also 0.06154 in. The top of the crystal is 30 in. above the channel bottom. Figure 4 shows the initial mesh and the associated partitioning into eight subdomains.

At the first time-step, the computational mesh had 8689 elements and 6849 nodes, giving rise to 43,408 unknowns in the unconstrained problem. The simulation was run for five time-steps starting with the particles and fluid at rest. Each time step was 0.01 sec. Table 2 presents the performance of the algorithm on eight processors of the SGI Origin 2000.

It is instructive to see the breakdown of the computational time into important steps. Table 3 presents the computational cost of critical steps. The nonlinear system solution time consists of the following main steps: calculation of the Jacobian matrix, application of the nonlinear operator, formation of the hierarchical divergence-free basis, and the solution of the linear system. The time to solve the linear system is dominated by matrix-vector multiplication with the Jacobian, application of the hierarchical basis, and orthogonalization of the Krylov subspace vectors in GMRES. The nonlinear solver takes most of the time, and its parallelization is critical to the overall performance.

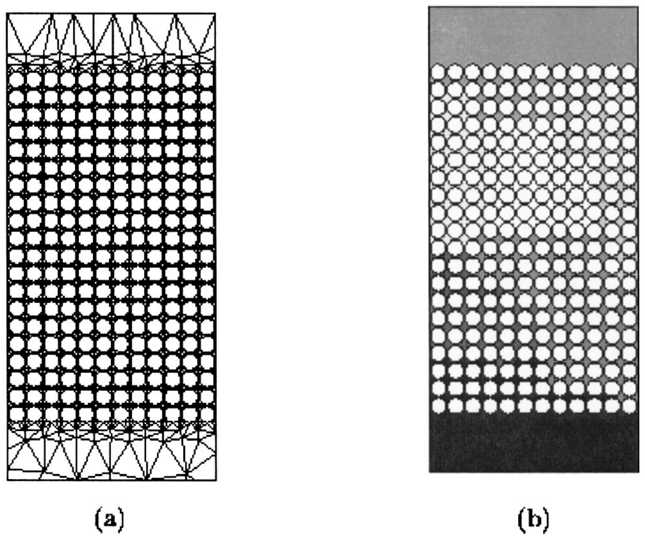

Fig. 4 Sedimentation of multiple particles: (*a*) mesh with 8689 elements and 6849 nodes, and (*b*) partitioning into eight domains. Only the region of interest is shown.

4.3 Additional Remarks. The parallel implementation of the algorithm demonstrates good parallel efficiency even for small-sized problems. The overall speedup of 5.3 on eight processors shown in Table 2 includes nonparallelizable components of the code as well as preconditioning effects that slowed the convergence of iterative solver on larger number of processors. The detailed view in Table 3 shows that the speedup in critical steps is 5.9 on eight processors. The computation of divergence-free velocity in the hierarchical basis is very efficient even on the small problem considered here. The relatively modest speedup in matrix-vector products is due to the structure of computation involving multiplication with the matrices of the hierarchical basis. As discussed in Section 2, this requires matrix-vector products with matrices defined on meshes whose size decreases geometrically from the finest to the coarsest level. In addition, it may be noted that parallel efficiency can be increased by replacing the orthogonalization step in GMRES with a variant that has a smaller serial component.

The preceding benchmark experiments define *speedup* as the improvement in speed over the *best* implementation of the algorithm on a uniprocessor. This implies that although the parallel algorithm demonstrates good *speed improvement* on multiple processors, the speedup may be modest. The code attempts to achieve high parallel performance by adopting an aggressive partitioning strategy which is aimed at good load balance in the overall computation. This particular implementation of the hierachical divergence-free basis algorithm computes a basis that changes with the number of processors. This has resulted in weaker preconditioning which has caused a growth in the number of iterations when the number of processors is increased. The deterioration in numerical efficiency of the algorithm can be eliminated by using the *same* basis on multiple processors. In this case, however, there is a marginal decrease in parallel efficiency which is offset by superior numerical convergence. The reader is referred to [9] for a scalable parallel implementation of this approach.

5 Conclusions

This paper describes an algorithm to compute discrete divergence-free velocity functions for incompressible fluid flow problems. The proposed scheme computes a basis for the null-space of the constraint matrix used to enforce incompressibility in the linearized Navier-Stokes equations. A multilevel recursive algebraic transformation of this constraint matrix yields a hierarchical basis for the required divergence-free functions. The algebraic nature of the scheme allows easy extension to particulate flow problems in which rigid particles are coupled with the surrounding fluid by no-slip condition on the particle surface. The paper outlines the extension of the hierarchical basis method for particulate flow problems. The effectiveness of the proposed scheme is demonstrated by a set of benchmark experiments with single and multiple sedimenting particles. The algorithm is designed to be parallelizable. The resulting implementation on the SGI Origin 2000 parallel computer demonstrates good parallel performance

Table 3 Parallel performance of important steps in the nonlinear solver for multiple particle sedimentation

	P=1		P=8		
Simulation Step	**Time**	**Percent**	**Time**	**Percent**	**Speedup**
Matrix assembly	224	11	33	10	6.8
Hierarchical Basis	1010	49	143	41	7.1
Matrix-vector multiplication	452	22	86	25	5.3
GMRES orthogonalization	360	18	83	24	4.3
Total	2046	100	345	100	5.9

even on small sized problems. For larger problems, the algorithm is expected to have significantly better parallel performance.

Acknowledgments

The authors wish to thank Dr. Knepley for providing the experimental data in Sections 4.1 and 4.2. This work was supported in part by the NSF through the grants ECS-9527123, CTS 9873236, CCR-9972533, and the infrastructure grant EIA 9871053. Sarin was also partially supported by the NSF grant CCR 9984400. Computational resources were provided by NCSA, University of Illinois at Urbana-Champaign.

References

[1] Hestenes, M. R., and Stiefel, E., 1952, ''Methods of Conjugate Gradients for Solving Linear Systems,'' J. Res. Natl. Bur. Stand., **49**, pp. 409–436.

[2] Saad, Y., and Schultz, M. H., 1986, ''GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems,'' SIAM (Soc. Ind. Appl. Math.) J. Sci. Stat. Comput., **7**, pp. 856–869.

[3] Meijerink, J. A., and Van der vorst, H. A., 1977, ''An Iterative Solution Method for Linear Equation Systems of Which the Coefficient Matrix is a Symmetric M-Matrix,'' Math. Comput., **31**, pp. 148–162.

[4] Gustafson, K., and Hartman, R., 1983, ''Divergence-Free Bases for Finite Element Schemes in Hydrodynamics,'' SIAM (Soc. Ind. Appl. Math.) J. Numer. Anal., **20**(4), pp. 697–721.

[5] Hall, C. A., Peterson, J. S., Porsching, T. A., and Sledge, F. R., 1985, ''The Dual Variable Method for Finite Element Discretizations of Navier-Stokes Equations,'' Int. J. Numer. Methods Eng., **21**, pp. 883–898.

[6] Golub, G. H., and Van Loan, C. F., 1996, *Matrix Computations* 3rd Ed., Johns Hopkins University Press Ltd., London.

[7] Sarin, V., and Sameh, A. H., 1998, ''An Efficient Iterative Method for the Generalized Stokes Problem,'' SIAM J. Sci. Comput. (USA), **19**(1), pp. 206–226.

[8] Sameh, A. H., and Sarin, V., 2002, ''Parallel Algorithms for Indefinite Linear Systems,'' Parallel Comput., **28**, pp. 285–299.

[9] Sarin, V., 1997, ''Efficient Iterative Methods for Saddle Point Problems,'' Ph.D. thesis, University of Illinois, Urbana-Champaign.

[10] Dembo, R. S., Eisenstat, S. C., and Steihaug, T., 1982, ''Inexact Newton Methods,'' SIAM (Soc. Ind. Appl. Math.) J. Numer. Anal., **19**(2), pp. 400–408.

[11] Eisenstat, S. C., and Walker, H. F., 1996, ''Choosing the Forcing Terms in an Inexact Newton Method,'' SIAM J. Sci. Comput. (USA), **17**(1), pp. 16–32.

[12] Hu, H., 1996, ''Direct Simulation of Flows of Solid-Liquid Mixtures,'' Int. J. Multiphase Flow, **22**.

[13] Smith, B. F., Gropp, W. D., McInnes, L. C., and Balay, S., 1995, ''Petsc 2.0 Users Manual,'' Technical Report TR ANL-95/11, Argonne National Laboratory.

[14] Gropp, W. D., Lusk, E., and Skjellum, A., 1994, *Using MPI: Portable Parallel Programming With the Message Passing Interface*, M.I.T. Press, Cambridge, MA.

[15] Shewchuk, J. R., 1996. ''Triangle: Engineering a 2D Quality Mesh Generator and Delaunary Triangulator,'' *First Workshop on Applied Computational Geometry*, ACM, Philadelphia, PA, pp. 124–133.

[16] Karypis, G., and Kumar, V., 1996. ''Parallel Multilevel *k*-Way Partitioning Scheme for Irregular Graphs,'' Technical Report 96-036, University of Minnesota.

[17] Knepley, M. G., Sarin, V., and Sameh, A. H., 1998. ''Parallel Simulation of Particulate Flows,'' *Proc. of the 5th Intl. Symp., IRREGULAR '98*, LNCS 1457, Springer, Berlin, pp. 226–237.