Hybrid Parallel Linear System Solvers

Ahmed H. Sameh and Vivek Sarin
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA
Tel: (765)494-7801, FAX: (765)494-0739
{sameh,sarin}@cs.purdue.edu

Abstract

This paper presents a new approach to the solution of nonsymmetric linear systems that uses hybrid techniques based on both direct and iterative methods. An implicitly preconditioned modified system is obtained by applying projections onto block rows of the original system. Our technique provides the flexibility of using either direct or iterative methods for the solution of the preconditioned system. The resulting algorithms are robust, and can be implemented with high efficiency on a variety of parallel architectures. The algorithms are used to solve linear systems arising from the discretization of convection-diffusion equations as well as those systems that arise from the simulation of particulate flows. Experiments are presented to illustrate the robustness and parallel efficiency of these methods.

1 Introduction

The past few decades have seen significant advances in the development of iterative methods for solving nonsymmetric linear systems. Coupled with advances in parallel architectures and algorithms, such methods have provided the only means for large scale scientific simulations in several disciplines. The lack of robustness of these iterative solvers, as well as the lack of effective parallel preconditioners, however, have prevented these iterative solvers from being as dependable as direct methods. The reader is referred to [1, 9] for an overview of iterative methods.

In this paper, we propose hybrid techniques for the solution of large sparse linear systems

$$Ax = b, \tag{1}$$

in which A is assumed to be a nonsymmetric matrix. Our approach uses projections onto subspaces of block rows to obtain a modified system that is often favorably preconditioned as well. In contrast to existing iterative methods, the proposed algorithms combine the use of

direct and iterative methods to obtain hybrid strategies that can be implemented efficiently on parallel computers.

Before we provide details of our projection-based hybrid schemes, we give an overview of Krylov subspace methods in Section 2 along with issues relating to preconditioning, parallel computation, and their lack of robustness. In Section 3 we discuss the origins of the linear systems arising from particulate fluids, followed by an outline of projection-based hybrid algorithms and their parallel implementation. Conclusions are presented in Section 4.

2 Krylov Subspace Methods

Iterative methods in this class of algorithms construct an approximate solution to problem (1) in the so-called Krylov subspace. The Krylov subspace is defined as

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\},$$

where $r_0 = b - Ax_0$ is the residual for an arbitrary initial vector x_0. The approximate solution x_m lies in the shifted space $x_0 + \mathcal{K}_m$, and the residual is made to satisfy certain conditions. Depending on the condition enforced on the residual, we obtain one of the many types of Krylov subspace methods. One such approach requires the residual r_m to be orthogonal to the Krylov subspace, i.e., $b - Ax_m \perp \mathcal{K}_m(A, r_0)$. This leads to well known methods including the Conjugate Gradient (CG) and Lanczos methods. When the residual is minimized over $\mathcal{K}_m(A, r_0)$, we get methods like GMRES, MINRES, and ORTHODIR. Another approach is to orthogonalize r_m against a subspace such as $\mathcal{K}_m(A^T, r_0)$; methods such as Bi-CG and QMR belong to this class.

The CG algorithm is one of the most successful solution methods for symmetric positive definite linear systems. For nonsymmetric systems, the choice of a robust and practical iterative method is unclear. GMRES, which is the most popular Krylov subspace method for nonsymmetric systems, requires storage proportional to the number of iterations. Practical implementations of this method include restarted or truncated GMRES. A number of variants such as "flexible" GMRES, and GMRESR have been proposed recently. Relatively inexpensive alternatives to GMRES can be found in Bi-CG and QMR algorithms. Some of the shortcomings of these methods have been addressed in variants such as Bi-CGSTAB, CGS, TFQMR, etc. In addition, block variants of all these methods have been proposed that may be of advantage in certain situations.

In spite of several years of research, there are significant lacunae in our understanding of these methods. In particular, important issues related to the convergence of Krylov subspace methods for nonsymmetric systems are not well understood. In addition, these algorithms lack robustness leading to breakdown of the solution process. As an example, one may consider practical implementations of GMRES that display unsatisfactory behavior due to arbitrary parameters dictating the restart or truncation of the algorithm. In general, in this class of iterative methods, it is difficult to assure robustness for most nonsymmetric linear systems that arise from practical applications.

2.1 Preconditioning

In order to improve the convergence of iterative methods, some form of preconditioning must be applied. A preconditioner is a matrix M whose inverse is assumed to be a close approximation to the inverse of A. Instead of (1), we now solve the preconditioned linear system $M^{-1}A = M^{-1}b$. The iterative algorithms of the previous section can be reformulated so that each iteration requires the solution of the system: $My = d$.

The most popular class of preconditioners are the incomplete LU factorizations that approximate the matrix M by constructing sparse approximations of the LU factors of A. Attempts to improve such preconditioners rely on increasing the non-zeros of the approximate factors, specifying drop-tolerance for retaining nonzero elements, enforcing rowsum constraints, and ordering the rows and columns prior to computing the factors. Block variants of these schemes have also been proposed. While such techniques are useful for certain classes of problems, they fail quite often leading to non-convergence of the iterative Krylov subspace scheme.

Another useful preconditioning technique relies on domain decomposition methods for solving PDEs. Solution of the PDE on subdomains is used as an approximation to the global solution. When the subproblems are also solved iteratively, one obtains an inner-outer iterative scheme. Another promising alternative computes the sparse approximate inverse M^{-1} directly. This has the advantage of using the matrix-vector product $y = M^{-1}d$ in each iteration. Unlike typical ILU preconditioners that require triangular solves, these algorithms can be quite effective on parallel computers.

It must be pointed out that the main requirements of a good preconditioner are the following: (i) the product $M^{-1}A$ should have a favorable spectrum that improves convergence – the closer it is to the identity, the better, (ii) M should be easy to invert, and (iii) M (or M^{-1}) should have a sparse representation. Furthermore, it is vital that operations involving the construction and application of M or M^{-1} be implemented with high efficiency on parallel computers. Alternatively, one may conceive a strategy to modify the problem in order to obtain an implicitly preconditioned linear system such that matrix-vector products with this modified system are efficiently parallelized. Computing robust preconditioners that satisfy these properties has largely been a matter of experimentation so far.

2.2 Parallelism

Large scale realistic simulations are feasible only with efficient preconditioned iterative solution techniques implemented on fast parallel computers. The iterative methods described above use the following types of operations: vector operations (scaled additions and dot products), matrix-vector products, preconditioner computation, and preconditioner application (e.g., triangular solves). The operations on the preconditioner are frequently the bottleneck in parallel implementations. In addition, dot products tend to become expensive for methods such as GMRES that require orthogonalization with a set of vectors. These observations again underscore the need for effective parallel preconditioning techniques for iterative methods.

3 Hybrid Parallel Solvers

The general structure of commonly used preconditioned Krylov subspace methods is the following: a preconditioner is computed prior to the start of the iterative process, then at each iteration, a system with the preconditioner matrix is solved via a direct or an iterative method. The hybrid solution techniques presented in the following sections differ significantly in philosophy from the Krylov subspace methods described previously.

The iterative algorithms discussed here are hybrid techniques derived from the application of direct and iterative algorithms to the overall solution process. The balance scheme discussed in Section 3.2 provides the flexibility of using a direct or iterative method to construct a reduced system that must be solved only once to obtain the solution. The reduced system can, again, be solved either by direct or iterative methods. Solution of the reduced system with a direct method yields a technique with the opposite structure to that of the preconditioned Krylov subspace methods. The implicit block Jacobi preconditioning approach presented in Section 3.3 is an alternate hybrid iterative algorithm for obtaining a reduced system. This is applicable to systems that can be partitioned into nonsingular diagonal blocks, and may be used as a robust parallel preconditioner.

3.1 Particulate Flow Problem

In this paper, we consider the motion of large number of particles in liquids under the action of the hydrodynamic forces and torques exerted by the suspending fluid. Such simulations are aimed at Newtonian fluids that are governed by the Navier-Stokes equations as well as several popular models of viscoelastic fluids. Particulate flow simulations are applicable to a number of industrial problems such as sedimentation, fluidization and slurry transport of solid particles.

For our experiments, we consider an incompressible fluid flowing a two-dimensional periodic channel with a pressure gradient set against gravity (Fig. 1), and N_p solid particles moving freely in the fluid. The equations for fluid and particle motion are:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\cdot\nabla\mathbf{u} = \rho\mathbf{g} - \nabla p + \nabla\cdot\tau, \tag{2}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{3}$$

$$\mathbf{M}\frac{d\mathbf{U}}{dt} = \mathbf{F}, \tag{4}$$

$$\frac{d\mathbf{X}}{dt} = \mathbf{U}, \tag{5}$$

where ρ is the fluid density, $\mathbf{g}$ is gravity, p is the pressure, $\mathbf{u}$ is the fluid velocity, τ is the extra-stress tensor (for Newtonian fluids, $\tau = \mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$), $\mathbf{X}$ and $\mathbf{U}$ are the generalized position and velocity vectors of particles, respectively, and $\mathbf{F}$ constitutes the forces and torques acting on the particles. $\mathbf{M}$ denotes the generalized mass matrix of the particles. A no-slip condition for the fluid velocity on the particle surface gives the additional equation

$$\mathbf{u} = \mathbf{U} + \mathbf{r}\times\Omega, \tag{6}$$

in which $\mathbf{r}$ is the position vector from the center of the particle to a point on its surface, and Ω is the angular velocity of the particle.

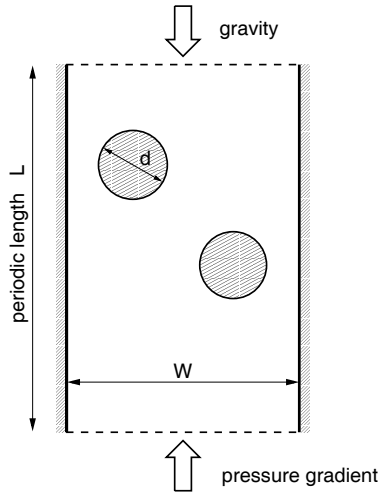

Figure 1: Two solid particles moving in a periodic channel.

The physical system is evolved from an initial state by the backward Euler method. At each time step, a system of nonlinear algebraic equations is solved using Newton's method, which further requires the solution of a series of linear systems involving the Jacobian of the nonlinear equations. A mixed finite elements approximation with P2/P1 pair of elements is used for these equations.

The most time-consuming aspect of the computation is the solution of linear systems of the following form:

$$\begin{pmatrix} A & B^T & G \\ B & 0 & 0 \\ E & F & M \end{pmatrix} \begin{pmatrix} u \\ p \\ U \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ h \end{pmatrix}. \tag{7}$$

For typical simulations, the velocity unknowns for the particles are a very small fraction of the velocity and pressure unknowns, often less than 0.01% of the total number of unknowns. This suggests the following two approaches to deal with the particle unknowns. The first one factors the matrix in (7) as shown below:

$$\begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \tilde{E} & \tilde{F} & I \end{pmatrix} \begin{pmatrix} A & B^T & G \\ B & 0 & 0 \\ 0 & 0 & \tilde{M} \end{pmatrix} \begin{pmatrix} u \\ p \\ U \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ h \end{pmatrix},$$

where

$$(\tilde{E}, \tilde{F}) = (E, F) \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}^{-1}, \qquad \tilde{M} = M - (E, F) \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}^{-1} \begin{pmatrix} G \\ 0 \end{pmatrix}.$$

It is easy to see that one doesn't need to compute the submatrix $(\tilde{E}, \tilde{F})$ at all. On the other hand, the matrix $\tilde{M}$ is computed explicitly by solving the saddle-point problem

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} = \begin{pmatrix} G \\ 0 \end{pmatrix}, \tag{8}$$

with multiple right-hand sides. Once the particle velocities U have been determined, one can recover the remaining unknowns u and p by solving another saddle-point problem of a similar form:

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ 0 \end{pmatrix}. \tag{9}$$

The convergence of an iterative method is known to be faster for multiple right-hand sides. This approach is especially effective when the number of particle unknowns is relatively small.

For larger number of particles, one may use a different factorization of the matrix in (7):

$$\begin{pmatrix} I & 0 & \tilde{G} \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \tilde{A} & \tilde{B}^T & 0 \\ B & 0 & 0 \\ E & F & M \end{pmatrix} \begin{pmatrix} u \\ p \\ U \end{pmatrix} = \begin{pmatrix} f \\ 0 \\ h \end{pmatrix},$$

where $\tilde{G} = GM^{-1}$, $\tilde{A} = A - GM^{-1}E$, and $\tilde{B}^T = B^T - GM^{-1}F$. The main computation now consists of solving the system

$$\begin{pmatrix} \tilde{A} & \tilde{B}^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} \hat{f} \\ 0 \end{pmatrix}. \tag{10}$$

The sparsity structure of $\tilde{A}$ and $\tilde{B}$ is similar to A and B, except for the additional fill resulting from the coupling between velocity and pressure unknowns on the surface of a particle.

One can further reduce the systems in (8), (9) and (10) to the following form:

$$BA^{-1}C^T p = g, \tag{11}$$

which is then solved by an iterative method such as GMRES. The conjugate gradients method (CG) is preferred when A is symmetric positive definite, and $C = B$. This approach is feasible only if one has a fast method to compute solutions of the type $Ax = b$ which occur at each step of the iterative process.

The choice of an inner iterative method for the system $Ax = b$ depends heavily on the nature of the matrix A, which is sensitive to simulation parameters such as the time step, viscosity, and fluid velocity. The time step is adaptively chosen during the simulation. Furthermore, since fluid velocity may change substantially over the entire simulation, no particular solution method is suitable throughout the computation. For instance, when the time step is very small, A approaches the velocity mass matrix, and GMRES with diagonal preconditioner work quite well. For larger time steps, although A is no longer diagonally dominant, it may be real positive, especially for viscous fluids; in such cases, GMRES with approximate factorization preconditioners may suffice. However, it must be noted that these

preconditioners are not parallelizable and susceptible to breakdown. The implicit block Jacobi preconditioner described in Section 3.3 has been shown to be a robust and parallel alternative for systems in which the diagonal blocks are nonsingular; this is recommended for real positive matrices.

With increasing fluid velocity, use of a large time step often results in A which is nonsymmetric and indefinite. In this situation, approximate factorization based preconditioners and solvers fail to converge to the solution. The next section describes an alternative approach, the balance scheme, that can be used to solve these systems without breakdown or loss of parallel performance.

3.2 The Balance Scheme

In this section, we present a hybrid parallel algorithm for the solution of nonsymmetric sparse linear systems such as those arising in particulate flows. Our technique is based on a new approach based on projections onto subspaces spanned by block rows of the linear system. (See e.g., [5, 8, 7, 12] for more details.) This approach is most suitable for banded linear systems with bandwidth that is about 1%–15% of the total number of unknowns. In this method a reduced system, defined only on unknowns common to consecutive block rows, is computed and solved by a direct or iterative method. We have also provided extensions of our algorithm [5] for the case where the reduced system is available only implicitly through matrix-vector products.

Consider a banded system that can be represented in the following block form

$$\begin{pmatrix} A_1 & B_1 & & & & & & \\ & C_2 & A_2 & B_2 & & & & \\ & & \ddots & \ddots & & & & \\ & & & C_{p-1} & A_{p-1} & B_{p-1} & & \\ & & & & & C_p & A_p \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \xi_1 \\ \mathbf{x}_2 \\ \vdots \\ \xi_{p-1} \\ \mathbf{x}_p \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \vdots \\ \mathbf{b}_p \end{pmatrix}. \tag{12}$$

The rows of A have been partitioned into p blocks, and the unknowns ξ_i are common to consecutive blocks of the matrix. The ith block row is given as $E_i = (C_i, A_i, B_i)$, and the ith system is given as

$$\begin{pmatrix} C_i & A_i & B_i \end{pmatrix} \begin{pmatrix} \tilde{\xi}_{i-1} \\ \mathbf{x}_i \\ \xi_i \end{pmatrix} = \mathbf{b}_i. \tag{13}$$

The general solution of the ith system is $\mathbf{x} = \mathbf{p}_i + Q_i \mathbf{y}_i$, where $\mathbf{p}_i$ is a particular solution, Q_i is a basis for $\mathcal{N}(E_i)$, and $\mathbf{y}_i$ is an arbitrary vector. The common unknown vector $\xi_i(\mathbf{y}_i)$ is determined from the ith system as well as the $(i+1)$th system. Let us designate $\tilde{\xi}_i(\mathbf{y}_{i+1})$ to be the value of $\xi_i(\mathbf{y}_i)$ determined from the $(i+1)$th system. Enforcing the constraints $\xi_i(\mathbf{y}_i) = \tilde{\xi}_i(\mathbf{y}_{i+1})$, $i = 1, \ldots, p-1$, we obtain the reduced system

$$M\mathbf{y} = \mathbf{g}, \tag{14}$$

where

$$M = \begin{pmatrix} Q_{1,2} & -Q_{2,1} & & & \\ & Q_{2,3} & -Q_{3,1} & & \\ & & \ddots & \ddots & \\ & & & Q_{p-1,3} & -Q_{p,1} \end{pmatrix}, \qquad \mathbf{g} = \begin{pmatrix} \mathbf{p}_{2,1} - \mathbf{p}_{1,2} \\ \mathbf{p}_{3,1} - \mathbf{p}_{2,3} \\ \vdots \\ \mathbf{p}_{p,1} - \mathbf{p}_{p-1,3} \end{pmatrix}. \tag{15}$$

The linear system (12) may be solved by computing the solutions of each of the underdetermined systems (13), followed by the solution of the reduced system (14). The algorithm is thus given by,

---

Algorithm *Balance_Scheme*

1. Solve the underdetermined systems of block rows to obtain $\mathbf{p}_i$ and Q_i, $(i = 1, \ldots, p)$.
2. Solve the reduced system $M\mathbf{y} = \mathbf{g}$.
3. Back-substitute $\mathbf{y}$ to determine all $\mathbf{x}_i$ and ξ_i.

It is not essential to explicitly form the reduced system. For such a situation, the algorithm has been extended to allow the computation of matrix-vector products directly from the projections (see, e.g., [5]). The normal form of the reduced system, i.e., $MM^T\mathbf{z} = \mathbf{g}$, must now be solved by an iterative algorithm. If the projections are also computed iteratively, one obtains an inner-outer iteration method.

The product of a vector with the reduced system matrix can be computed efficiently on a multiprocessor by concurrent projections onto subspaces of block rows. Our experiments in Section 3.2.1 indicate that the balance scheme is a robust and competitive alternative to existing Krylov subspace schemes, particularly for problems with strong indefinite symmetric part.

The balance scheme is a typical example of hybrid parallel solvers. The structure of the algorithm does not restrict the use of direct or iterative methods to any particular aspect of the solution process. When it is relatively inexpensive expensive to compute the matrix Q_i which is an orthogonal basis for $\mathcal{N}(E_i)$, the reduced system can be computed explicitly and solved by a direct method. Alternatively, an iterative method may be used to solve the reduced system that is implicitly available through matrix-vector products. One may also obtain the reduced system (or its approximation) through a series of matrix-vector operations, and solve it using a direct method. Finally, it is worth mentioning that a similar scheme may be developed for systems that are not banded.

3.2.1 Balance Scheme for Particulate Flows

The balance scheme was used to solve several instances of the linear systems (8) and (9) for simulations with a single particle. Since these systems were quite large, it was not feasible

to compute QR factorizations of each block. Therefore, we used the variant of the reduced system in which matrix-vector products with the reduced system are computed directly without explicitly generating the reduced system.

Table 1 illustrates the performance of restarted GMRES with Krylov subspace size of 40 and the balance scheme for the system $Ax = b$, where A is nonsymmetric indefinite on account of large time step coupled with large fluid velocity. The two problems instances

		GMRES(40)		Balance Scheme	
Time Step	Newton Iter.	Iter.	Time (sec)	Iter.	Time (sec)
t=0.15 sec,	1	280	9	37	25
Δt=0.08 sec	2	774	24	53	38
	3	failure	–	53	38
	4	failure	–	52	37
t=0.1875 sec,	1	677	20	78	44
Δt=0.1 sec	2	failure	–	74	41
	3	failure	–	74	41
	4	failure	–	72	40

Table 1: Comparison of the balance scheme with restarted GMRES (failure indicates non-convergence in 800 iterations).

were obtained at time 0.15 seconds and 0.1875 seconds, respectively, from the start of the simulation. The first system had 15,969 unknowns, 339,906 nonzeros, a bandwidth of 992, and yielded a reduced system of size 2894. The second system had 16,017 unknowns, 340,992 nonzeros, a bandwidth of 991, and yielded a reduced system of size 3032. In each case, the rows were partitioned into 8 blocks.

As the nonlinear iterations progress, these systems becomes increasingly difficult to solve, and finally result in failure of GMRES. The breakdown of GMRES is particularly severe for the system with larger time step Δt. In contrast, the balance scheme demonstrates consistent convergence behavior that is relatively independent of the indefiniteness of the coefficient matrix. In these experiments, GMRES was preconditioned with the diagonal of A. (Preconditioners based on block ILUT with suitable reordering to increase parallelism in the triangular system solves have been largely unsuccessful for such systems.)

The next set of experiments highlight the parallel performance of the balance scheme. The saddle-point problem (11) was solved in which A was symmetric positive definite, and $C = B$. The system was preconditioned with incomplete Cholesky factorization of $BD^{-1}B^T$, where $D = diag(A)$, and solved using the CG algorithm. At each iteration, the balance scheme was used to solve linear systems of the form $Ax = b$. The projections were computed in parallel using preconditioned CG algorithm as well.

Table 2 illustrates the performance of the balance scheme for the solution of the inner system as well as the saddle-point problem (11). The matrix A is of size 17,025 with bandwidth

System		P = 1	P = 2	P = 4	P = 8	P =16
$Ax = b$	Time (sec)	9.1	4.0	2.0	1.2	0.9
	Speed Improv.	1.0	2.3	4.6	7.6	10.1
	Efficiency	1.0	1.1	1.1	0.9	0.6
$B^T A^{-1} Bp = g$	Time (sec)	109.4	55.4	28.9	15.1	12.3
	Speed Improv.	1.0	1.9	3.8	7.3	8.9
	Efficiency	1.0	1.0	0.9	0.9	0.6

Table 2: Performance of the balance scheme on SGI Origin 2000.

1,450 and 362,641 nonzeros whereas the matrix $B^T A^{-1} B$ is of size 2199. For experiments utilizing up to 8 processors, the rows of A were partitioned into 8 blocks resulting in a reduced system of size 3680. These experiments show almost linear speed improvement, with isolated superlinear performance due to cache effects. A straightforward extension of the algorithm to 16 processors requires partitioning of rows into 16 blocks resulting in a larger reduced system ($m = 5508$), and hence, decreased performance. Since additional parallelism is available in computing individual projections, parallel speedup can be improved substantially by assigning multiple processors to each block row. In other words, one may use 16 processors for a matrix partitioned into 8 block rows by assigning two processors to each block. This is clearly more advantageous for cluster-based multiprocessors.

The balance scheme can be used for three-dimensional simulations as well. The linear systems in particulate flow problems are much larger in size and the bandwidth can also be considerable. In such a situation, the balance scheme can be used to precondition an iterative method. One may choose a submatrix of the linear system as a preconditioner, such that it has a much smaller bandwidth, and can be solved efficiently using the balance scheme. Such a matrix can be obtained by carefully dropping relatively small off-diagonal values in order to reduce the bandwidth of the original matrix. This must be accompanied by a compensation to the matrix such that the modification resembles an update of the matrix by appropriate element stiffness matrices. It must be noted, however, that the ratio of bandwidth to the order of the linear system is still favorable, since it generally depends upon the aspect ratio of the simulation domain. Therefore, the balance scheme is applicable to such systems without much change.

3.3 An Implicit Block Jacobi Preconditioner

The particulate flow problem often yields a real positive matrix A in which each diagonal submatrix is nonsingular. Such a situation arises when using a relatively small time-step for slow moving fluids. In this section, we present a preconditioning approach for GMRES that is ideally suited for efficient parallel solution of these problems. The experiments in Section 3.3.1 demonstrate the strengths of our technique over other preconditioners.

Consider a banded sparse linear system with rows partitioned into p blocks. The right-

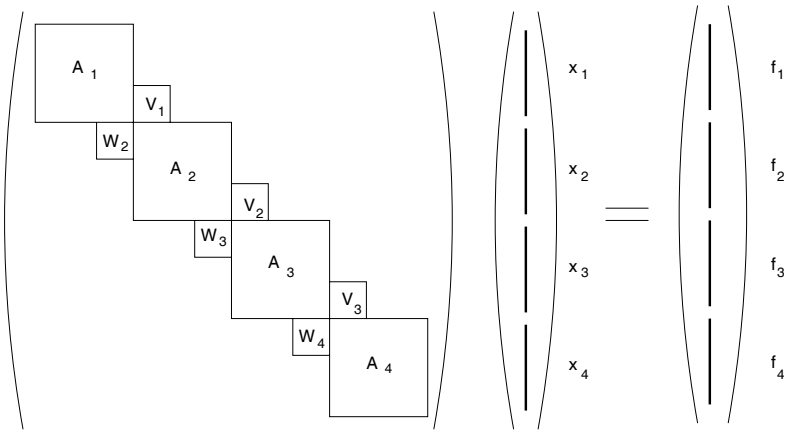

Figure 2: Almost–block diagonal structure of a banded matrix with $p = 4$ block rows.

hand side and the solution vector have been partitioned in the same manner (see, e.g., Fig. 2), where each diagonal block A_i, $i = 1, \ldots, p$ is nonsingular with n_i rows. Since the A_i's are nonsingular, we can find B_i and C_i, both of size $n_i \times m$, such that

$$A_i B_i = \begin{pmatrix} W_i \\ 0 \end{pmatrix}, \qquad A_i C_i = \begin{pmatrix} 0 \\ V_i \end{pmatrix},$$

respectively. Also, let $h_i = A_i^{-1} f_i$. Considering the block diagonal matrix

$$D = diag(A_1, A_2, \ldots, A_p),$$

as an outer preconditioner, we get the preconditioned system

$$D^{-1} A x = h,$$

where $h = D^{-1} f$. This preconditioned system is of the form

$$\begin{pmatrix} I & \tilde{C}_1 & & & \\ \tilde{B}_2 & I & \tilde{C}_2 & & \\ & \tilde{B}_3 & I & \tilde{C}_3 & \\ & & \ldots & \ldots & \ldots \\ & & & & \tilde{C}_{p-1} \\ & & & \tilde{B}_p & I \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \ldots \\ x_{p-1} \\ x_p \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ \ldots \\ h_{p-1} \\ h_p \end{pmatrix},$$

where I is the identity matrix of dimension n_i, and

$$\tilde{B}_i = (0, B_i), \qquad \tilde{C}_i = (C_i, 0)$$

with the B_i and C_i's being, generally, dense $n_i \times m$ matrices. In order to obtain the reduced system, we need to partition the "spikes" B_i and C_i, as well as x_i and h_i. Let $B_t^{(i)}$ and $B_b^{(i)}$

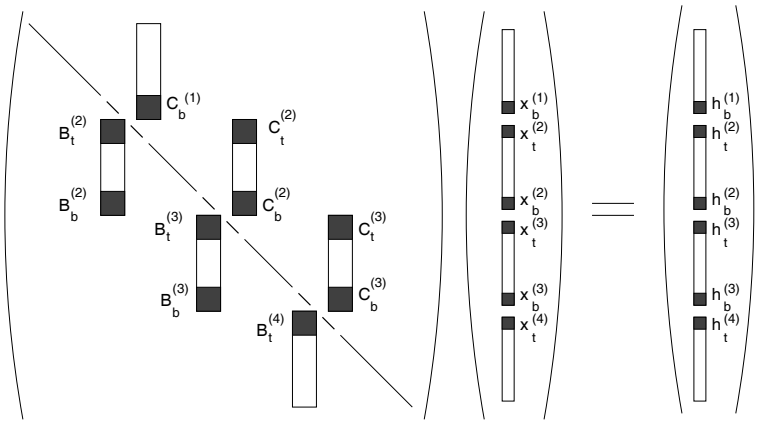

Figure 3: Forming the reduced system from $D^{-1}Ax = h$.

be the top and bottom m rows of B_i, respectively. Similarly, we can define $C_t^{(i)}$ and $C_b^{(i)}$, $x_t^{(i)}$ and $x_b^{(i)}$, and $h_t^{(i)}$ and $h_b^{(i)}$.

In Fig. 3, we illustrate how these partitions are formed. It can be observed that those small portions, described above (shaded areas in the figure), actually form an independent subsystem of equations which involves only the $x_t^{(i)}$ and $x_b^{(i)}$'s. It can be shown that there exists a permutation matrix P, such that

$$P(D^{-1}A)P^T = \begin{pmatrix} I & T \\ 0 & \tilde{A}_r \end{pmatrix}, \qquad Px = \begin{pmatrix} y \\ x_r \end{pmatrix}, \qquad Ph = \begin{pmatrix} s \\ h_r \end{pmatrix},$$

where

$$\tilde{A}_r x_r = h_r, \tag{16}$$

is called the reduced system. The reduced system has the following structure:

$$\tilde{A}_r = \begin{pmatrix} I_m & C_b^{(1)} & & & & & & \\ B_t^{(2)} & I_m & O_m & C_t^{(2)} & & & & \\ B_b^{(2)} & O_m & I_m & C_b^{(2)} & & & & \\ & & B_t^{(3)} & I_m & O_m & C_t^{(3)} & & \\ & & B_b^{(3)} & O_m & I_m & C_b^{(3)} & & \\ & & & & \cdots & & & \\ & & & & B_t^{(p-1)} & I_m & O_m & C_t^{(p-1)} \\ & & & & B_b^{(p-1)} & O_m & I_m & C_b^{(p-1)} \\ & & & & & & B_t^{(p)} & I_m \end{pmatrix},$$

$$x_r = \begin{pmatrix} x_b^{(1)} \\ x_t^{(2)} \\ x_b^{(2)} \\ x_t^{(3)} \\ x_b^{(3)} \\ \cdots \\ x_t^{(p-1)} \\ x_b^{(p-1)} \\ x_t^{(p)} \end{pmatrix}, \qquad h_r = \begin{pmatrix} h_b^{(1)} \\ h_t^{(2)} \\ h_b^{(2)} \\ h_t^{(3)} \\ h_b^{(3)} \\ \cdots \\ h_t^{(p-1)} \\ h_b^{(p-1)} \\ h_t^{(p)} \end{pmatrix}.$$

In general, the reduced system is of order $n_r = 2m(p-1)$.

Solving the reduced system yields x_r, and the solution x of the original system may be retrieved by:

$$\begin{aligned} x_1 &= g_1 - C_1 x_t^{(2)}, \\ x_i &= g_i - B_i x_b^{(i-1)} - C_i x_t^{(i+1)}, \qquad \text{for } i = 2, \ldots, p-1, \\ x_p &= g_p - B_p x_b^{(p-1)}. \end{aligned}$$

In practice, if n_i is still large, it is not advisable to form the matrix A_r explicitly. Rather, the reduced system can be solved by an iterative scheme such as GMRES, for example, where the matrix-vector multiplication involves solving independent linear systems of the form $A_i u_i = b_i$. These systems may be solved by an iterative scheme again, or by a direct algorithm such as the sequential block Gaussian elimination scheme, parallel variants of block cyclic reduction ([3]), or the SPIKE algorithm ([2, 10, 11]). The choice of the algorithms used in this case depends upon the degree of parallelism available. If direct solvers are adopted in this inner matrix-vector multiplication, and if the reduced system is diagonally dominant, the reduced system (16) may be preconditioned by its block-diagonal part (i.e., by dropping the submatrices $C_t^{(i)}$'s and $B_b^{(i)}$'s). In the experiments presented in Section 3.3.1, we solve the linear systems $A_i u_i = b_i$ using a direct linear solver. In addition, in order to compute $w = C_t^{(2)} v$, where w and v are vectors of size m, we solve the system

$$A_2 w = \begin{pmatrix} 0 \\ V_2 \end{pmatrix} v.$$

In this manner, the matrix-vector product $y_r = \tilde{A}_r x_r$ is carried out in parallel via solving several independent linear systems.

Once the reduced system (16) is solved by an iterative method, the right-hand side is updated and the remaining components of x are obtained again by solving independent linear systems involving the diagonal blocks A_i. The following is an outline of the algorithm.

Algorithm *Implicit_Block_Jacobi*

1. Factor diagonal blocks A_i's.
2. Update the right-hand side: $h_i = A_i^{-1} f_i$.
3. Solve the reduced system, iteratively, by solving systems involving the A_i's.
4. Update the right-hand side.
5. Solve $A_i x_i = \bar{f}_i$.

3.3.1 Performance of Implicit Block Jacobi Preconditioner

The experiments in this section were conducted to illustrate the robustness of the implicit block Jacobi preconditioner for indefinite problems. In order to precisely control the spectral properties of the linear system A, we restrict our attention to the three-dimensional model convection-diffusion equation:

$$\begin{aligned} -\Delta u + \mathbf{a} \cdot \nabla u - cu &= f \qquad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \partial\Omega, \end{aligned}$$

where $c > 0$ is a constant. A standard five-point finite difference scheme is used for discretization, and the rows of the resulting matrix are partitioned such that that each A_i has order $n_i = 4096$. In addition, the off-diagonal blocks V_i's and W_i's are each of size $m = 64$. The linear systems with A_i are solved in parallel using GMRES with ILUT preconditioner.

Tables 3 and 4 compare the performance of the implicit block Jacobi with block Jacobi parallel preconditioner on two distinct linear systems. In each case, the linear systems with diagonal blocks were solved in parallel using GMRES with an ILUT preconditioner. We use GMRES(k) to denote restarted GMRES with Krylov subspace of size k. It is clear from these experiments that block Jacobi is ineffective as a parallel preconditioner for these problems.

	Block Jacobi		Implicit Block Jacobi	
Algorithm	Iter.	Time (sec)	Iter.	Time (sec)
GMRES(5)	failure	–	19	9
GMRES(20)	failure	–	14	7
Gaussian Elim.				30

Table 3: Comparison of implicit block Jacobi with block Jacobi parallel preconditioner for a problem with 65,536 unknowns on SGI Challenge with 16 processors.

	Block Jacobi		Implicit Block Jacobi	
Algorithm	Iter.	Time (sec)	Iter.	Time (sec)
GMRES(5)	failure	–	38	5.2
GMRES(20)	failure	–	28	4.4
Gaussian Elim.				6.7

Table 4: Comparison of implicit block Jacobi with block Jacobi parallel preconditioner for a problem with 147,456 unknowns on Cray C90 with 8 processors.

Table 5 illustrates the behavior of GMRES preconditioned with ILUT on a single processor. These experiments were performed on the same matrix as the one reported in Table 3.

Algorithm	Preconditioner	Iter.	Time (sec)
GMRES(30)	ILUT(2)	35	37
GMRES(15)	ILUT(4)	14	22
GMRES(10)	ILUT(8)	9	28
GMRES(5)	ILUT(16)	failure	–

Table 5: Convergence of incomplete factorization based preconditioners for the problem in Table 3.

These experiments clearly demonstrate that the implicit block Jacobi preconditioner is a robust and parallel preconditioning alternative to the commonly used ILUT preconditioners. In contrast, GMRES preconditioned with the ILUT preconditioner is not a robust iterative algorithm; in fact, its effectiveness depends on fortunate choices of the size of Krylov subspace, the number of fill-in elements used in the approximate factorization, and the threshold used to determine the numerical dropping of elements. Attempts to improve the ILUT preconditioner with additional fill often result in failure as well. In addition to this, ILUT is not suitable as a parallel preconditioner.

4 Conclusions

In this paper, we have presented hybrid techniques for the solution of general sparse linear systems. A reduced system is obtained from the original linear system by the use of orthogonal projections onto subspaces derived from the block rows of the matrix. This reduced system is often favorably preconditioned and may be solved by either direct or iterative methods. Our experiments with linear systems arising from simulations of particles in fluids demonstrate the superiority of these algorithms over Krylov subspace methods on two important issues,

namely, robustness, and efficient implementation on parallel computers.

5 Acknowledgements

This work was supported in part by the NSF through the grants CCR-9619763 and ECS-9527123.

References

[1] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, Cambridge, 1994.

[2] M. Berry and A. Sameh. Multiprocessor schemes for solving block tridiagonal linear systems. *Intl. J. Supercomp. Appl.*, 3:37–57, 1988.

[3] S. Bondeli. Divide and conquer: A parallel algorithm for the solution of tridiagonal linear system of equations. *Parallel Computing*, 17:419–434, 1984.

[4] R. Bramley and A. H. Sameh. Row projection methods for large nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:168–193, 1992.

[5] G. H. Golub, A. H. Sameh, and V. Sarin. A parallel balanced method for sparse linear systems. *(submitted to Numerical Linear Algebra with Applications)*, 1997.

[6] C. Kamath and A. H. Sameh. A projection method for solving nonsymmetric linear systems on multiprocessors. *Parallel Computing*, 9:291–312, 1988/89.

[7] F. Lou and A. Sameh. A parallel splitting method for solving linear systems. Technical Report 92-125, AHPCRC, Univ. of Minnesota, Minneapolis, 1992.

[8] M. Oettli and A. Sameh. A partitioning scheme for the parallel solution of banded linear systems. Technical report, Computer Science Dept., Univ. of Minnesota, Minneapolis, 1997.

[9] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, 1996.

[10] A. Sameh. On two numerical allgorithms for multiprocessors. In *Proc. of NATO Adv. Res. Workshop on High-Speed Comp.*, volume 7, pages 311–328, 1983.

[11] A. Sameh and D. Kuck. On stable parallel linear system solvers. *JACM*, 25:81–91, 1978.

[12] V. Sarin. Spectral analysis of balanced CG methods. Technical report, Computer Science Dept., Univ. of Minnesota, Minneapolis, 1993.