A Krylov Multisplitting Algorithm for Solving Linear Systems of Equations*

Chiou-Ming Huang
Department of Computer Science
University of Maryland
College Park, Maryland 20742

and

Dianne P. O'Leary
Department of Computer Science
and Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742

Submitted by Robert J. Plemmons

ABSTRACT

We consider the practical implementation of Krylov subspace methods (conjugate gradients, GMRES, etc.) for parallel computers in the case where the preconditioning matrix arises from a multisplitting. We show that the algorithm can be efficiently implemented by dividing the work into tasks that generate search directions and a single task that minimizes over the resulting subspace. Each task is assigned to a subset of processors. It is not necessary for the minimization task to send frequent information to the direction generating tasks, and this leads to high utilization with a minimum of synchronization. We study the convergence properties of various forms of the algorithm and present results of numerical examples on a sequential computer.

*This work was supported by the AFOSR under grant 87-0158, by the General Research Board of the University of Maryland Graduate School, and by the Institute for Mathematics and Its Applications at the University of Minnesota.

1. INTRODUCTION

The preconditioned algorithms in the *Krylov subspace family* (conjugate gradients, GMRES, CGSTAB, etc.) are standard tools in the numerical solution of large systems of linear equations

$$Ax = b$$

on high performance computers. The choice of preconditioners is still a topic of research, since there is no general rule for choosing a preconditioner that will ensure rapid convergence.

In many cases the matrix A has a natural splitting as $A = M - N$, where linear systems involving the matrix M can be solved easily. A splitting of A induces an iterative method

$$Mx^{k+1} = Nx^k + b, \tag{1}$$

initialized by a choice of x^0, and the iteration is convergent for each choice of the initial guess x^0 if and only if the spectral radius ρ of the matrix $M^{-1}N$ satisfies $\rho(M^{-1}N) < 1$. Common splittings include M equal to the main diagonal of A (the Jacobi partition), M equal to the diagonal and lower triangular elements of A (the Gauss-Seidel partition), and block variants of these. In some cases the splitting is induced by the structure of the problem (e.g., domain decomposition methods for elliptic partial differential equations). In many instances, a matrix has several possible convergent splittings, each with advantages and disadvantages.

O'Leary and White [14] provided a framework for studying multiple splittings of matrices. If a matrix can be partitioned in several ways,

$$A = M_i - N_i, \qquad i = i, \dots, p, \tag{2}$$

then a *multisplitting* of A is defined by

$$B = \sum_{i=1}^{p} D_i M_i^{-1} N_i, \tag{3}$$

where the matrices D_i are diagonal matrices that sum to the identity matrix.

Multisplittings induce the iterative method

$$\hat{x}^{k+1} = \sum_{i=1}^{p} D_i \hat{x}_{(i)}^{k+1}, \tag{4}$$

where

$$M_i \hat{x}_{(i)}^{k+1} = N_i \hat{x}^k + b. \tag{5}$$

This can be written in the equivalent form

$$\hat{x}^{k+1} = B\hat{x}^k + Gb, \tag{6}$$

where

$$G = \sum_{i=1}^{p} D_i M_i^{-1}. \tag{7}$$

Thus the convergence of the multisplitting is dependent on the condition $\rho(B) < 1$.

Part of the motivation behind multisplittings is that the work for each individual splitting can be assigned to a subset of processors and communication is required only to combine the results using the diagonal weighting factors. If any element of D_i is zero, then the corresponding component of $\hat{x}_{(i)}^{k+1}$ need not be computed, so the multisplitting framework can be used to partition variables among processors.

The parallel implementation of multisplittings has been investigated in many papers (e.g., [16, 11]), and it is natural to try to accelerate the convergence of the iteration using Krylov subspace-based methods [8]. The Krylov subspace $\mathscr{K}(A, q, j)$ of dimension j is the span of the vectors $\{q, Aq, \dots, A^{j-1}q\}$, where q is a vector in $\mathscr{R}^n$. The Krylov subspace methods that we discuss form iterates y^k that minimize some error function (e.g., $\|Ax - b\|_2$) over the Krylov subspace $\mathscr{K}(A, b, k)$. If preconditioning is used, then the subspace becomes $\mathscr{K}(M^{-1}A, M^{-1}b, k)$, and in the case of multisplittings this can be written as $\mathscr{K}(B, Gb, k)$.

The natural parallel implementation of a multisplitting preconditioner for a Krylov subspace method is to divide the algorithm into $p + 1$ tasks and assign a set of processors to each task. Each of the p multisplitting tasks computes one of the vectors $\hat{x}_{(i)}$. The remaining task combines these vectors, generates a new basis vector to expand the Krylov subspace, minimizes the error function over that subspace, and sends the updated vector back to the multisplitting tasks. There is a great deal of parallelism within each task, but to improve parallel utilization, it is possible to let the multisplittings run m iterations at a time before generating the next Krylov vector.

Using this form of the algorithm, it can be difficult to achieve very high utilization, since the work is bimodal: either some of the p multisplitting tasks are active or the Krylov task is active, but not both. Synchronization can be a significant bottleneck: the Krylov minimization cannot be performed until each of the multisplitting tasks has reported, and the multisplitting tasks are idle while the minimization is being performed. The assignment of tasks to processors must be done quite carefully in order to balance the work in each phase and minimize waiting time.

In this paper we consider an alternative algorithm. Again there are $p + 1$ tasks, p for the multisplitting and one for the minimization, and each task is assigned to a subset of processors. But in this algorithm, the multisplittings report individually to the minimization task and do not need to wait for a response. This reduces waiting time at the cost of some additional complication in the minimization, and it will be shown that the subspace over which the error function is minimized equals the Krylov subspace used in the standard algorithm. Although no synchronization signals are sent from the minimization task to the multisplittings, the minimization task can be implemented in a way that makes the algorithm deterministic rather than chaotic. In practice, due to roundoff error, the minimization task must periodically reinitialize the multisplittings, but this can be done infrequently.

This approach provides additional flexibility as well. We can generate a larger dimensional space for minimization by using each of the multisplitting vectors in the minimization, and we can add other promising vectors to the subspace. This creates a subspace that includes the standard Krylov subspace.

In Section 2 we define the parallel tasks in the algorithm. Section 3 is devoted to determining the properties of the subspace over which the minimization is performed and establishing a convergence rate for the symmetric positive definite case. The parallel complexity is considered in Section 4, and results of numerical examples on sequential machines are presented in Section 5. A preliminary report on part of this work was given in [10].

2. ALGORITHM KMS: KRYLOV MULTISPLITTING

As noted in the introduction, Krylov subspace methods such as the conjugate gradient algorithm and GMRES form approximate solutions to a linear system $Ax = b$ by minimizing an error function over the subspace $\mathscr{K}(M^{-1}A, M^{-1}b, k)$, $k = 1, 2, \ldots$. Standard implementations use an orthonormal basis for this subspace in order to make the minimization more efficient. In this section we present the algorithm in a form that uses the natural basis rather than the orthogonal one in order to reduce communication among tasks, to allow for extra basis vectors, and to achieve better parallel utilization. We first present the algorithm in a general way. Some implementation notes and specific examples of multisplittings follow.

$Task_0$ is the minimization task, and the multisplitting tasks are denoted by $Task_i$, $i = 1, \ldots, p$.

ALGORITHM KMS (A parallel multisplitting-preconditioning of a Krylov subspace minimization)

```
Cobegin $Task_0, Task_1, \ldots, Task_p$. Send the initial guess $\hat{x}^0$ to each task
and the convergence tolerance $\varepsilon$ to $Task_0$. Coend.

• Algorithm for multisplitting $Task_i$, $i = 1, \ldots, p$:
   For $k = 0, 1, \ldots,$ until receiving a halt signal from $Task_0$,
      Receive the latest multisplitting iterate, $\hat{x}^k$ and call it $z^0$.
         For $j = 1, \ldots, m$
            Determine $z^j$ by solving $M_i z^j = N_i z^{j-1} + b$.
            Send the search direction $\Delta z_i^{k+1,j} = z^j - z^{j-1}$ to $Task_0$ for
            minimization.
         endfor
         Form $z_i^{k+1} = D_i z^m$, and participate with the other multisplitting
         tasks in forming $\hat{x}^{k+1}$ by summing the $z_i^{k+1}$, $i = 1, \ldots, p$. (See
         Note 1.)
   endfor

• Algorithm for minimization $Task_0$.
   Initialize $r = b$ and $S$ to be the null matrix.
   While $\|r\| > \varepsilon\|b\|$,
      Receive any available new directions $\Delta z$ from tasks $1, \ldots, p$, and
      use them to form columns for the matrix $S$.
      Set $x$ to be the minimizer of the error function (e.g., a norm of the
      $x$-error or of the residual) over the subspace $\mathscr{S}$ spanned by the
      columns of $S$, and set the residual $r = b - Ax$. (A more complete
      description is given in Section 2.1.)
   end
   Send halt signal to $Task_1, \ldots, Task_p$.
```

NOTE 1. The best algorithm to use for formation of the vector sum in the multisplitting tasks depends on the parallel architecture and on the particular multisplitting. If each element of the weighting matrices D_i is either 1 or 0, then the "summation" is simply a merging of subvectors and is performed by sending local values to all other tasks that depend on them and receiving relevant subvectors from other tasks. If the weighting matrices have elements strictly between 0 and 1 so that averaging is needed, then the summation is performed using standard algorithms [3] in logarithmic time using hypercube connections or linear time using a mesh-connected set of processors.

NOTE 2. The algorithm as written here generates a basis for a subspace including the Krylov subspace. An alternative algorithm sends just the "outer" iteration Krylov directions $\hat{x}^{k+1} - \hat{x}^k$ to $Task_0$ instead of sending all of the direction vectors from each "inner" iteration of each multisplitting task. Whether this is a good idea depends on the relative speed of computation vs. communication for $Task_0$. Ideally, we want to feed directions to $Task_0$ as fast as they can be processed. Since the amount of work in $Task_0$ grows with the number of columns in S, this may mean that $Task_0$ initially accepts every column it receives, but later may discard some early columns or accept only a subset of the new columns generated by the multisplitting tasks. In this way it may behave more like a restarted GMRES algorithm, for instance.

NOTE 3. The number of "inner" iterations m could be variable, determined adaptively depending on the convergence properties of the multisplitting and on the computing environment [2].

NOTE 4. An iterative method can be used to solve $M_i z^j = N_i z^{j-1} + b$. Given a splitting $M_i = F_i - G_i$, we perform s "inner" iterations with this splitting in order to approximate the solution. The resulting iterate is

$$z_{k+1} = (F^{-1}G)^s z_k + \sum_{j=0}^{s-1} (F^{-1}G)^j F^{-1}(Nz_k + b), \qquad k = 0, 1, \ldots . \quad (8)$$

Two-stage (or inner/outer) methods based on such splittings have been studied, for example, by Frommer and Szyld [6] and by Golub and Overton [7].

NOTE 5. Due to roundoff error, it is necessary to periodically reinitialize the multisplitting tasks so that they work with the most accurate computed solution, but this can be done relatively infrequently. Due to the growth of work in the minimization task, it is desirable to periodically reinitialize the subspace $\mathscr{S}$.

NOTE 6. The algorithm bears some relation to the s-step conjugate gradient method of Chronopoulos and Gear [5] that forms a series of matrix-vector products before orthogonalizing against the previous directions. In our case, we allow extra basis vectors in the minimization, and s is the total number of iterations between reinitialization of the multisplitting tasks.

NOTE 7. Our algorithm is also related to one given by Axelsson and Vassilevski [2], who propose using a variable preconditioner, perhaps changing the number of iterations m or the exact form of the operator from iteration to iteration. There are some important differences. Our minimization subspace can be somewhat richer, including the subvectors for the multisplittings; they use only the vectors $\hat{x}^{k+1} - \hat{x}^{k}$. In their algorithm, the direction-generating tasks must wait for an updated vector from the minimization task before proceeding. We will show in the next section that the same subspace can be generated without such waits. Their computation of the orthogonal basis for minimization is more direct, as is typical of most Krylov subspace algorithm implementations, and can be done with somewhat more simplicity.

2.1. *The Minimization Task*

To illustrate the implementation of the minimization algorithm, we will use the error function $\|Ax - b\|_2$ as an example. Other error functions (e.g., the conjugate gradient function $\|x - x^*\|_A$, where $\|w\|_A^2 = w^TAw$ and x^* is the true solution to the problem) or preconditioned forms of these functions yield similar procedures.

Mathematically, the minimizer of the error function $\|Ax - b\|_2$ over the subspace spanned by the l columns of a matrix S is $\tilde{x} = S\alpha$, where α is determined as the solution of the linear system

$$S^TA^TAS\alpha = S^TA^Tb.$$

Attempting to solve this system directly can lead to numerical difficulties, since the columns of S may contain some (near) linear dependencies, so a more reliable numerical approach is to factor the matrix AS using a rank-revealing QR factorization [4]. The matrix AS is factored as QR, where the columns of the $n \times l$ matrix Q are orthogonal (i.e., $Q^TQ = I$), R is an $l \times l$ upper triangular matrix ($n \geqslant l$), and the columns of AS have been rearranged in the course of the factorization so that the columns causing linear dependencies are pushed to the right. We pick a maximal leading principal submatrix R_1 of R of dimension $\hat{l} \leqslant l$ that corresponds to a well-conditioned subset of basis vectors, and solve a reduced system. For the QR factorization we determine the first $\hat{l}$ components of α from the reduced system

$$R_1 \alpha_1 = Q_1^T b, \tag{9}$$

where Q_1 consists of the first $\hat{l}$ columns of Q, and we set the last $l - \hat{l}$ components of α to zero.

The implementation of the minimization algorithm is somewhat dependent on the choice of splitting and computer architecture. As an illustration of the splitting dependence, if each element of the weighting matrices D_i is either 1 or 0, then the direction vectors are sparse, and savings can be achieved by taking advantage of this structure in forming the product of A with the vectors. There is a great deal of parallelism in the matrix updating and in the solution of the linear system, and this should be exploited on a given architecture.

The approach that we have just described keeps the size of the subspace small. However, it is desirable to use the most recent directions if the underlying iteration (i.e., the multisplitting iteration) is convergent. Therefore an alternate approach is to find an orthogonal basis for $\mathcal{S}$, form A times the orthogonal basis vectors, and solve the resulting system by QR. The orthogonalization of the basis vectors requires additional work, but it ensures that the matrix AS is at least as well conditioned as A is, and ensures that a new basis vector is generated for each direction vector (although it may be effectively random if the direction vectors are highly dependent). Therefore, the experiments discussed later are based on the following procedure. When a new vector v is received:

1. Update the QR decomposition of the vectors spanning the subspace $\mathcal{S}$ to include v. The last column q_k of Q is our new basis vector.
2. Form Aq_k.
3. Update the QR decomposition $Q_a R_a$ of AQ to include Aq_k. A rank-revealing QR algorithm is used, so that if necessary, the subspace can be truncated to maintain a well-conditioned matrix R_a.
4. Form the minimizer in the subspace by solving $R_a \alpha = Q_a^T b$.
5. Form the new iterate $x_k = Q\alpha$.

2.2. *Multisplitting Examples*

To better illustrate the division of labor in the multisplitting tasks, we give several specific examples. See also [21].

EXAMPLE 1. Discretization of elliptic partial differential equations may lead to symmetric positive definite matrices of the form

$$A = \begin{pmatrix} M_1 & F \\ F^T & M_2 \end{pmatrix},$$

where systems of the form $M_1 z = d$ and $M_2 z = d$ are easy to solve. Setting

$$M = \begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix}$$

leads to the very effective "block" Jacobi iteration (see [8]). Since we solve $M_1 z_1 = d_1$ and $M_2 z_2 = d_2$ independently, it is apparent that we can partition the work into two tasks, with each task updating a disjoint piece of the vector of unknowns. If we solve the linear systems involving M_1 and M_2 directly, then we set $m = 1$ and have the two tasks exchange information after each update. We show in Section 3 that the resulting subspace for minimization includes the Krylov subspace $\mathscr{K}(M^{-1}A, M^{-1}b, k)$.

EXAMPLE 2. The structure in Example 1 can be exploited in a different way: it is not necessary to solve $Mz_{k+1} = r_k$ exactly. Our goal is to efficiently provide a good subspace for minimization. Thus, if M has a splitting as

$$M = \begin{pmatrix} M_{11} & \\ & M_{22} \end{pmatrix} = \begin{pmatrix} F_1 & \\ & F_2 \end{pmatrix} - \begin{pmatrix} G_1 & \\ & G_2 \end{pmatrix} = F - G$$

where F is nonsingular, then the two tasks can iterate using the splittings $M_1 = \text{diag}(F_1, 0)$, $M_2 = \text{diag}(0, F_2)$. We use these two-stage methods to produce a vector space for minimization. This raises the question how many of these "inner" iterations we should perform. On the one hand, we should reduce the communication overhead between $Task_1$ and $Task_2$. On the other hand, even if we solve the linear system $Mz = d$ exactly, we do not necessary increase the convergence rate. Hence the answer to this question depends on the problem to be solved and the computing environment.

EXAMPLE 3. Splittings with more than $p = 2$ pieces arise naturally in discretizations of partial differential equations, using either domain decomposition or multicolorings.

In *domain decomposition* [17] the variables are partitioned into possibly overlapping subsets corresponding to subdomains for which solution is easy. The operators M_i correspond to a partial differential equation over the subdomain.

A similar partitioning can be used to handle problems in which the discretization has been *locally refined*. Variables can be partitioned into pieces of roughly equal size corresponding to "coarse" grid points and refined points, and a multisplitting can be constructed from this partitioning.

In *multicolorings*, the variables are partitioned into groups (colors) so that the matrices M_i are block diagonal. The simplest example is the well-known red-black ordering for the 5-point operator, but partitionings with $p > 2$ have also been developed [1, 12].

3. CONVERGENCE ANALYSIS

3.1. *Properties of the Multisplitting Subspace*

In this section, we characterize the subspace spanned by the vectors generated by the multisplitting tasks. We do this by deriving expressions for the iterates and their differences. Without loss of generality, we assume that $\hat{x}^0 = 0$.

LEMMA 3.1. *In Algorithm* KMS, *the directions* $\Delta z_i^{k+1,j}$, $j = 1, \dots, m$, *span the Krylov subspace* $\mathscr{K}(M_i^{-1}N_i, M_i^{-1}(b - A\hat{x}^k), m)$. *The iterates satisfy*

$$z_i^m = \left(M_i^{-1}N_i\right)^m \hat{x}^k + \sum_{j=0}^{m-1} \left(M_i^{-1}N_i\right)^j M_i^{-1} b. \tag{10}$$

Proof. *Task*$_i$ generates iterates

$$M_i z^j = N_i z^{j-1} + b,$$

where $z^0 = \hat{x}^k$, so the direction vectors $\Delta z^j \equiv z^j - z^{j-1}$ are

$$\Delta z^j = M_i^{-1} N_i \, \Delta z^{j-1}$$

for $j = 2, \dots, m$, with

$$\begin{aligned}\Delta z^1 &= z^1 - z^0 \\ &= M_i^{-1}N_i\hat{x}^k + M_i^{-1}b - \hat{x}^k \\ &= \left(M_i^{-1}N_i - I\right)\hat{x}^k + M_i^{-1}b \\ &= -M_i^{-1}A\hat{x}^k + M_i^{-1}b,\end{aligned}$$

and thus we generate the Krylov subspace $\mathscr{K}(M_i^{-1}N_i, -M_i^{-1}A\hat{x}^k + M_i^{-1}b, m)$.

The expression for the iterates is a standard result, easily verified by induction. ■

The following theorem shows that the subspace over which we minimize includes the standard Krylov subspace used by preconditioned algorithms. This is the key to applying standard convergence results.

THEOREM 3.1. *Given $\hat{x}^k$ ($\hat{x}^0 = 0$), the m steps of the multisplitting iteration generate iterates*

$$\hat{x}^{k+1} = B\hat{x}^k + \hat{b}, \tag{11}$$

where

$$B = \sum_{i=1}^{p} D_i\left(M_i^{-1}N_i\right)^m \tag{12}$$

and

$$\hat{b} = \sum_{i=1}^{p} D_i \sum_{j=0}^{m-1} \left(M_i^{-1}N_i\right)^j M_i^{-1}b. \tag{13}$$

Thus, the directions

$$\Delta\hat{x}^j \equiv \hat{x}^j - \hat{x}^{j-1}, \qquad j = 1, \dots, k,$$

span the Krylov subspace $\mathscr{K}(B, \hat{b}, k)$.

Proof. From (10) and (4) we see that for $k \geqslant 0$,

$$\hat{x}^{k+1} = \sum_{i=1}^{p} D_i \left(\left(M_i^{-1}N_i\right)^m \hat{x}^k + \sum_{j=0}^{m-1} \left(M_i^{-1}N_i\right)^j M_i^{-1}b \right) = B\hat{x}^k + \hat{b}$$

and

$$\hat{x}^1 = \Delta \hat{x}^1 = \sum_{i=1}^{p} D_i \sum_{j=0}^{m-1} \left(M_i^{-1}N_i\right)^j M_i^{-1}b = \hat{b}.$$

Thus, for $k > 0$ we have the expression

$$\Delta \hat{x}^{k+1} = B\,\Delta \hat{x}^k,$$

and the vectors $\Delta \hat{x}^j$, $j = 1, \dots, k$, span the subspace $\mathscr{K}(B, \Delta \hat{x}^1, k)$. ∎

The following corollary rewrites Lemma 3.1 in terms of the notation introduced in Theorem 3.1.

COROLLARY 3.1. *After K outer iterations with $\hat{x}^0 = 0$, the directions $\{\Delta z_i^{k,j}\}$ generated by the multisplitting tasks span the Krylov subspaces*

$$\mathscr{K}\left(M_i^{-1}N_i,\ M_i^{-1}\left(b - A\sum_{j=0}^{k-1} B^j \hat{b} \right), m \right)$$

for $i = 1, \dots, p$ and $k = 1, \dots, K$.

The next two corollaries specialize the theorem to the case of a single iteration between communication, and to two-stage methods.

COROLLARY 3.2. *If $m = 1$ (i.e., each multisplitting task performs a single iteration between communications), then the directions $\{\Delta z_i^{k,j}\}$ generated by the multisplitting span the space $M_i^{-1}A\mathscr{K}(B, \hat{b}, K)$ in union with the span of $\{M_i^{-1}b\}$, $i = 1, \dots, p$.*

COROLLARY 3.3. *If $m = 1$ and s steps of splittings $M_i = F_i - G_i$ are used as in (8), then after K outer iterations with $\hat{x}^0 = 0$, the directions $\{\Delta \hat{x}^k\}$ span the Krylov subspace $\mathscr{K}(H, \bar{b}, K)$, where*

$$H = \sum_{i=1}^{p} D_i\left[I - \left(F_i^{-1}G_i\right)^s \right] M_i^{-1}A \tag{14}$$

and

$$\bar{b} = \sum_{i=1}^{p} D_i\left[I - \left(F_i^{-1}G_i\right)^s\right]M_i^{-1}b. \tag{15}$$

Proof. From Algorithm KMS, the ith task generates its first iterate

$$\begin{aligned} z_i^1 &= \sum_{j=0}^{s-1} \left(F_i^{-1}G_i\right)^j F_i^{-1}b \\ &= \left[I - \left(F_i^{-1}G_i\right)^s\right]\left(I - F_i^{-1}G_i\right)^{-1}F_i^{-1}b \\ &= \left[I - \left(F_i^{-1}G_i\right)^s\right]M_i^{-1}b. \end{aligned}$$

Thus

$$\hat{x}^1 = \sum_{i=1}^{p} D_i\left[I - \left(F_i^{-1}G_i\right)^s\right]M_i^{-1}b = \bar{b}.$$

The $(k + 1)$st iterate is

$$\begin{aligned} z_i^{k+1} &= \left(F_i^{-1}G_i\right)^s\hat{x}^k + \sum_{j=0}^{s-1} \left(F_i^{-1}G_i\right)^j F_i^{-1}\left(N_i\hat{x}^k + b\right) \\ &= \left(F_i^{-1}G_i\right)^s\hat{x}^k + \left[I - \left(F_i^{-1}G_i\right)^s\right]M_i^{-1}N_i\hat{x}^k + \left[I - \left(F_i^{-1}G_i\right)^s\right]M_i^{-1}b. \end{aligned}$$

Hence

$$\begin{aligned} \hat{x}^{k+1} &= \sum_{i=1}^{p} D_i z_i^{k+1,1} \\ &= \sum_{i=1}^{p} D_i\left(F_i^{-1}G_i\right)^s\hat{x}^k + \sum_{i=1}^{p} D_i\left[I - \left(F_i^{-1}G_i\right)^s\right]M_i^{-1}N_i\hat{x}^k + \hat{x}^1. \end{aligned}$$

Then

$$\begin{aligned}
\hat{x}^k - \hat{x}^{k+1} + \hat{x}^1 &= \sum_{i=1}^{p} D_i \hat{x}^k - \sum_{i=1}^{p} D_i (F_i^{-1} G_i)^s \hat{x}^k \\
&\quad - \sum_{i=1}^{p} D_i \left[I - (F_i^{-1} G_i)^s \right] M_i^{-1} N_i \hat{x}^k \\
&= \sum_{i=1}^{p} D_i \left[I - (F_i^{-1} G_i)^s \right] \hat{x}^k \\
&\quad - \sum_{i=1}^{p} D_i \left[I - (F_i^{-1} G_i)^s \right] M_i^{-1} N_i \hat{x}^k \\
&= \sum_{i=1}^{p} D_i \left[I - (F_i^{-1} G_i)^s \right] M_i^{-1} A \hat{x}^k \\
&= H \hat{x}^k.
\end{aligned}$$

From this relation we see that the directions $\{\Delta \hat{x}^k\}$ span the Krylov subspace $\mathscr{K}(H, \bar{b}, K)$. ■

3.2. *Convergence Bounds*

Now we have tools to analyze the rate of convergence of Algorithm KMS. Theorem 3.1 guarantees that the subspace over which we minimize contains the standard Krylov subspace, and thus standard results for the convergence of algorithms such as preconditioned conjugate gradient or preconditioned GMRES [8] are applicable. For example, if $\hat{M}$ and A are symmetric and positive definite, and if we minimize the error function $\|x - x^*\|_A$ over the Krylov subspace $\mathscr{K}(\hat{M}^{-1}A, \hat{M}^{-1}b, k)$, then the error is bounded as

$$\|e_{(k+1)}\|_A \leqslant 2 \left(\frac{1 - \sqrt{\kappa^{-1}}}{1 + \sqrt{\kappa^{-1}}} \right)^{k+1} \|e_0\|_A, \tag{16}$$

where κ is the condition number of $\hat{M}^{-1}A$, the ratio of its largest to its smallest eigenvalue. (See, for example, [8].)

For the multisplitting algorithm,

$$\hat{M}^{-1} = \sum_{i=1}^{p} D_i \sum_{j=0}^{m-1} \left(M_i^{-1}N_i\right)^j M_i^{-1}, \tag{17}$$

and

$$\begin{aligned}\hat{M}^{-1}A &= \sum_{i=1}^{p} D_i \sum_{j=0}^{m-1} \left(M_i^{-1}N_i\right)^j \left(I - M_i^{-1}N_i\right)\\ &= \sum_{i=1}^{p} D_i\left[I - \left(M_i^{-1}N_i\right)^m\right]\\ &= I - B.\end{aligned}$$

Here are the conditions that naturally lead to symmetric positive definite preconditioners $\hat{M}$.

THEOREM 3.2. *If $m = 1$, if A and M_i are symmetric and positive definite, if $D_i = \alpha_i I$, where α_i is a positive scalar ($i = 1, \dots, p$), and if we minimize $\|x - x^*\|_A$ over the Krylov subspace generated by Algorithm* KMS, *then the bound* (16) *applies, where κ is the condition number of $I - B$.*

Proof. This follows from (17), since each term in the second sum is symmetric. ∎

In the nonsymmetric case, we obtain bounds like the standard GMRES result [18].

THEOREM 3.3. *Suppose that we minimize the norm of the residual over the Krylov subspace generated by Algorithm* KMS, *and that the preconditioner $\hat{M}^{-1}$ satisfies*

(i) $(v, A\hat{M}^{-1}v)_2 \geqslant \delta_1\|v\|_2^2$ *all* v;
(ii) $\|A\hat{M}^{-1}v\|_2 \leqslant \delta_2\|v\|_2$, *all v, for some positive constant δ_1, δ_2.*

Then the multisplitting algorithm converges monotonically and the residual norms satisfy the bound

$$\|r_k\|_2 \leqslant \sqrt{1 - (\delta_1/\delta_2)^2}\,\|r_{k-1}\|_2, \qquad k = 1, 2, \dots . \tag{18}$$

4. PARALLEL IMPLEMENTATION

In designing a parallel implementation of Algorithm KMS, our first observation is that the nature of the multisplitting tasks is fundamentally different from that of the minimization task. The multisplitting will generally be working with sparse matrices M_i and A, while the matrix AS for the minimization task may be dense. Thus, the use of a heterogeneous parallel computer may be possible, using, for example, a hypercube for the multisplitting and a systolic array (or Connection Machine) for the minimization task.

The implementation of the multisplitting is quite problem dependent, but the minimization task is somewhat easier to characterize. The main work is the updating of a matrix using a QR or URV decomposition. Parallel versions of these algorithms have been discussed in [13, 19].

Huang [9] has performed a detailed analysis of Algorithm KMS using a 2×2 block partitioning of the SSOR splitting on a model problem, Laplace's equation on a rectangle with Dirichlet boundary conditions. For convenience, assume that the matrix A is $N^2 \times N^2$ and that the two-dimensional mesh of multisplitting processors consists of $2s \times s$ processors. We assign $N^2/2s^2$ unknowns to each processor. The computation and communication time for each "outer" iteration (s inner iterations) is

$$s \cdot O\left(\frac{N^2}{s} + s\beta_M + N\tau_M\right) + \beta + \frac{N}{s}\tau + \frac{N}{s},$$

where the time for a typical floating point operation is 1, β_M is the startup time for communication within the multisplitting processors, τ_M is the per word transmission time for these processors, and β and τ are the corresponding times for communication between the multisplitting and the minimization task. After k directions are generated, the time for the minimization is $O((kN^2 + k^2)/t)$ on t processors.

5. NUMERICAL EXAMPLES

The purpose of this section is to illustrate the behavior of Algorithm KMS as compared with standard implementations of conjugate gradients or GMRES, and indicate some of the problem solving strategies that become possible when the minimization task is decoupled from the direction-generating tasks. Both the test problem and the preconditioning strategies are oversimplified, but they suffice for illustration.

Consider a convection-diffusion equation

$$-u_{xx} - u_{yy} + \sigma u_x + \tau u_y = f \quad \text{on } \Omega, \qquad u = g \quad \text{on } \partial\Omega.$$

Let Ω be a square, and use the second order accurate five-point finite difference method and upwind differencing of the first order terms to discretize the equation with m equally spaced interior mesh points in each direction. This gives the algebraic equation $Au = b$ where A is an $m^2 \times m^2$ matrix. Using the standard column by column ordering of mesh points, the matrix is block tridiagonal, with tridiagonal matrices $(-(1 + 2\gamma), 4 + 2(\delta + \gamma), -1)$ on the main diagonal, and $-I$ and $-(1 + 2\delta)I$ above and below, where $\gamma = \sigma h/2$, $\delta = \tau h/2$, and $h = 1/(m + 1)$.

Two test problems were used. Both used 50×50 meshes. The first was a symmetric problem ($\sigma = \tau = 0$), and the second was nonsymmetric ($\sigma = 1$, $\tau = 2$). The initial guess was taken to be all zeros, and the right hand side was a random vector with normally distributed entries and standard deviation 1. The tolerance for the rank determination in the QR decomposition was 10^{-5}.

The conjugate gradient (for the symmetric problem) and GMRES algorithms were run with block (column by column) SSOR preconditioning. These algorithms require three matrix-vector products per basis vector, although this complexity can be reduced somewhat by using the Eisenstat algorithm [15]. Algorithm KMS was used with three preconditionings:

1. KMS(BSSOR): Block SSOR preconditioning column by column. The resulting change in the iterate was used as a direction vector ($d = 1$ direction vector per iteration). Alternatively, we used two direction vectors per iteration ($d = 2$), one from the forward sweep and one from the backward sweep. There are three matrix-vector products per basis vector for the $d = 1$ method, and two for the $d = 2$ method.

2. KMS(ADBGS): Block Gauss-Seidel preconditioning. The first task runs the standard column by column ordering, but the second uses the row by row ordering. The two direction vectors are sent to the minimization task, and the average of the row and column iterates is used to initialize the next block Gauss-Seidel step. This requires two matrix-vector multiplications per direction vector.

3. KMS(PartBGS): $Task_1$ forms updates of the even grid columns by the block Gauss-Seidel iteration. At the same time, $Task_2$ forms updates of the even grid rows by block Gauss-Seidel. Values for the duplicated grid points (approximately $m^2/4$) are averaged, and values for the omitted grid points (approximately $m^2/4$) are computed by point Gauss-Seidel, using the updated neighboring values. This requires 2.25 matrix-vector products per direction vector.

The directions received by the minimization task were used to generate an orthonormal basis for the minimization subspace, and the product of the matrix A with these basis vectors (included in the counts above) was performed by the minimization task. Upon restart, the current residual was used as the right hand side.

Algorithm KMS was run in two modes: no communication from $Task_0$ to the other tasks, and communication every 20 iterations in order to have the splitting tasks work with the best solution vector found thus far. The maximal size for the minimization subspace in the KMS and GMRES algorithms was 20. The conjugate gradient algorithm, of course, was run without restarts.

Results of the experiments are given in Figures 1 and 2, in which the norms of the residuals for the various algorithms are plotted as a function of number of matrix-vector multiplications.

There are two curves for each KMS algorithm. The upper ones correspond to no communication from $Task_0$ to the other tasks and indicate that the algorithms stall and fail to converge. This is due to the generation of directions that are effectively random when Gram-Schmidt orthogonalization is applied to a long sequence of Krylov vectors. The same problem limits the

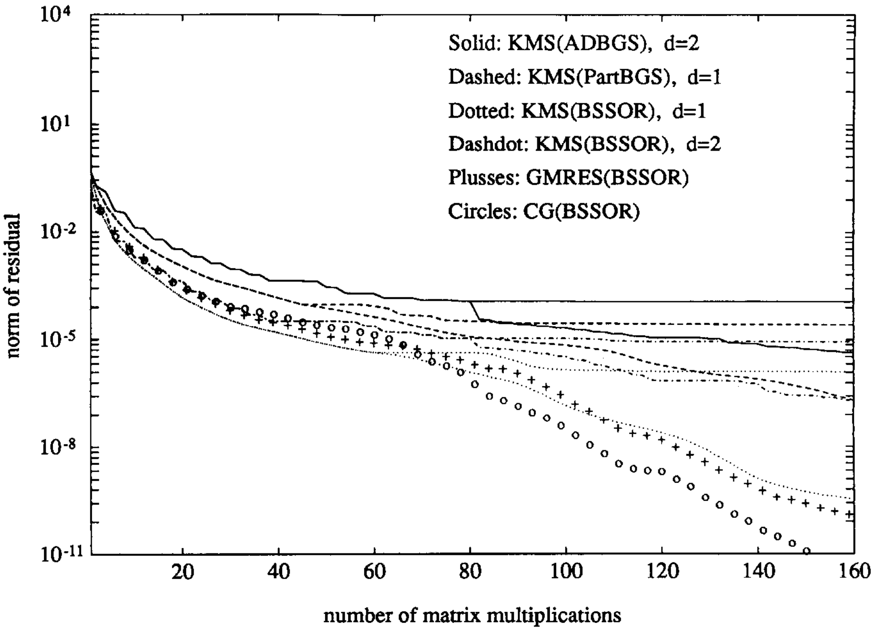

FIG. 1. 50 × 50 grid, Problem 1.

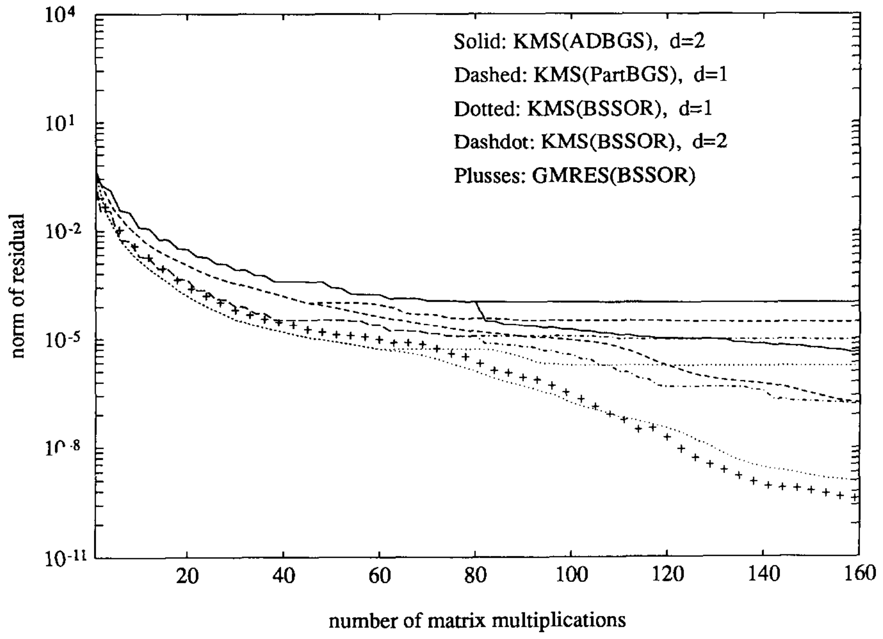

FIG. 2. 50 × 50 grid, Problem 2.

number of steps that can be performed in the s-step conjugate gradient algorithm of Chronopoulos and Gear, and can be somewhat alleviated by using Householder orthogonalization [20]. With feedback every 20 iterations, all KMS algorithms converged.

These experiments lead us to believe that Algorithm KMS is an effective alternative to standard implementations of Krylov subspace methods even on sequential computers, and the KMS formulation enables us to take advantage of multisplitting preconditioners. On parallel computers, the Algorithm KMS has the further advantage of not requiring frequent communication from $Task_0$ to the multisplitting tasks, and in balancing the number of matrix-vector multiplications among the tasks.

6. CONCLUSIONS

We have presented a nontraditional implementation of Krylov subspace methods that does not use an orthogonal basis to compute the subspace.

If only the vectors $\Delta\hat{x}$ are used for minimization, then this algorithm is equivalent to preconditioned conjugate gradients or GMRES; if additional vectors are used, then the minimization is performed over a larger subspace that includes the standard Krylov subspace. This algorithm has more flexibility than standard implementations and many advantages for parallel computation:

1. The number of synchronization points between the multisplitting tasks and the minimization task is greatly reduced.
2. Additional vectors can be added to the subspace if the Krylov generators are not working fast enough to keep the minimization task busy.
3. Vectors can be dropped if they do not provide a sufficient decrease.
4. We have the option of creating several vectors from any particular basis vector by partitioning it into subvectors and creating a vector from each of these padded with zeros. This might improve the convergence if the preconditioner is locally good but normalization between pieces of the vector is not so good.
5. The minimization task can reinitialize the direction generators at any time by sending the updated x vector. If the minimization has been performed using only the $\Delta\hat{x}$ vectors, then this has no effect on the computation, but if other directions have been added, then convergence can be accelerated without significant synchronization penalty.
6. There are natural extensions of these ideas to nonlinear problems [9].

Clearly, further work remains to be done in developing effective multisplittings and in implementing the algorithm on parallel machines.

In the special case of $p = 1$ (i.e., a single splitting), the idea behind Algorithm KMS *should be attributed to Gene Golub, who frequently asks the question, "But why do you need an orthogonal basis?"*

Bob Plemmons made useful comments on a draft of the manuscript.

REFERENCES

1 Loyce Adams and J. Ortega, A multi-color SOR method for parallel computation, in *Proceedings of the 1982 International Conference on Parallel Processing*, 1982, pp. 53–56.

2 O. Axelsson and P. S. Vassilevski, A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning, *SIAM J. Matrix Anal. Appl.* 12:625–644 (1991).

3 Dimitri P. Bertsekas and John N. Tsitsiklis, *Parallel and Distributed Computation Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J., 1989.

4 T. F. Chan, Rank-revealing *QR* factorization, *Linear Algebra Appl.* 88/89:67–82 (1987).

5 A. T. Chronopoulos and C. W. Gear, On the efficient implementation of

preconditioned s-step conjugate gradient methods on multiprocessors with memory hierarchy, *Parallel Comput.* 11:37–53 (1989).

6 Andreas Frommer and Daniel B. Szyld, H-Splittings and Two-Stage Iterative Methods, *Numer. Math.* 63:345–356 (1992).

7 G. H. Golub and M. Overton, Convergence of a two-stage Richardson iterative procedure for solving systems of linear equations, in *Proceedings of the Dundee Biennial Conference on Numerical Analysis*, Univ. of Dundee, Scotland, June 1981, Springer-Verlag.

8 Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, Johns Hopkins U.P., Baltimore, 1983.

9 Chiou-Ming Huang, Preconditioning Parallel Multisplittings for Solving Systems of Equations, Technical Report CS-TR-2948, Ph.D. Dissertation, Computer Science Dept., Univ. of Maryland, College Park, Aug. 1992.

10 Chiou-Ming Huang and Dianne P. O'Leary, Preconditioning parallel multisplittings for solving linear systems of equations, in *6th ACM International Conference on Supercomputing, Washington, DC*, July 1992.

11 M. Neumann and R. J. Plemmons, Convergence of parallel multisplitting iterative methods for M-matrices, *Linear Algebra Appl.* 88/89:559–573 (1987).

12 D. P. O'Leary, Ordering schemes for parallel processing of certain mesh problems, *SIAM J. Sci. Statist. Comput.* 5:620–632 (1984).

13 D. P. O'Leary and P. Whitman, Parallel QR factorization by Householder and modified Gram-Schmidt algorithms, *Parallel Comput.* 16:99–112 (1990).

14 Dianne P. O'Leary and R. E. White, Multi-splittings of matrices and parallel solution of linear systems, *SIAM J. Discrete Math.* 6:630–640 (1985).

15 J. M. Ortega, Efficient implementations of certain iterative methods, *SIAM J. Sci. Statist. Comput.* 9:882–891 (1988).

16 Theodore S. Papatheodorou and Yiannis G. Saridaki, Parallel algorithm and architectures for multisplitting iterative methods, *Parallel Comput.* 12:171–182 (1989).

17 R. Glowinski et al. (Eds.), *Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, des Ponts et Chaussees, Paris, France, January 7–9, 1987*, SIAM, Philadelphia, 1988.

18 Youcef Saad and Martin H. Schultz, GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* 7:856–869 (1986).

19 G. W. Stewart, An Updating Algorithm for Subspace Tracking, *IEEE Trans. on Signal Processing* 40:1535–1541 (1992).

20 Charles D. Swanson and Anthony T. Chronopoulos, Orthogonal s-Step Methods for Nonsymmetric Linear Systems of Equations, Technical Report UMSI91/341, Univ. of Minnesota Supercomputer Inst., Minneapolis, 1991.

21 R. E. White, Multisplitting with different weighting schemes, *SIAM J. Matrix Anal. Appl.* 10:481–493 (1989).

Received 10 June 1992; final manuscript accepted 2 September 1992