Preconditioning Parallel Multisplittings for Solving Linear Systems of Equations

Chiou-Ming Huang
Dianne P. O'Leary

Department of Computer Science
University of Maryland
College Park, MD 20742.

April 13, 1992

Abstract

We consider the practical implementation of Krylov subspace methods (conjugate gradients, GMRES, etc.) for parallel computers in the case where the preconditioning matrix is a multisplitting. 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 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.

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

ICS '92-7/92/D.C., USA

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

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 [15] provided a framework for studying multiple splittings of matrices. If a matrix can be partitioned in several ways,

$$A = M_i - N_i, \quad i = i, ..., p, \tag{2}$$

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

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

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 [9]. The Krylov subspace $\mathcal{K}(A,q,j)$ of dimension j is the span of the vectors $\{q, Aq, \ldots, A^{j-1}q\}$, where q is a vector in $\mathcal{R}^n$. 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 $\mathcal{K}(A,b,k)$. If preconditioning is used, then the subspace becomes $\mathcal{K}(M^{-1}A, M^{-1}b, k)$, and in the case of multisplittings this can be written as $\mathcal{K}(B, Gb, k)$.

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 alternate 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.

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 §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 §4.

2 Algorithm KMS: Krylov Multisplitting

In this section we first present the algorithm in a general way, without restrictions on the choice of multisplitting. Some implementation notes and specific examples of multisplittings follow.

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

Algorithm KMS: A parallel multisplitting-preconditioning of a Krylov subspace minimization

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

- Algorithm for multisplitting $Task_i$, $i = 1, ..., p$:

  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}, $i = 1, ..., p$. (See Note 1.)

endfor

- Algorithm for minimization $Task_0$.

  Initialize $r = b$ and S to be the null matrix.
  While $||r|| > \epsilon\ ||b||$,
    Receive any available new directions Δz from tasks $1, ..., p$ and use them to forms columns for the matrix S
    Set x to be the minimizer of the error function over the subspace $\mathcal{S}$ spanned by the columns of S, and set the residual $r = b - Ax$. (A more complete description is given in §2.1.)
  end
  Send halt signal to $Task_1$,...,$Task_p$.

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. An alternate algorithm sends just the "outer" iteration direction $\hat{x}^{k+1} - \hat{x}^k$ 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 processors. 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

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

$k = 0, 1, \ldots$. Two stage (or inner/outer) methods based on such splittings have been studied, for example, by Frommer and Szyld [7] and Golub and Overton [8]. □

Note 5. 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 take s to be the full number of iterations, and we allow extra basis vectors in the minimization.

Note 6. 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:

1. Our minimization subspace can be somewhat richer, including the subvectors for the multisplittings; they use only the vectors $\hat{x}^{k+1} - \hat{x}^k$.
2. 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.
3. 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 $||w||_A^2 = w^T Aw$ 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

$$S^T A^T AS\alpha = S^T A^T b.$$

Attempting to solve this system directly will 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 $n \times \ell$ matrix Q

are orthogonal (i.e., $Q^TQ = I$), R is an $\ell \times \ell$ upper triangular matrix ($n \geq \ell$), 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 $\hat{\ell} \leq \ell$ that corresponds to a well conditioned subset of basis vectors, and solve a reduced system. For the QR factorization we determine the first $\hat{\ell}$ components of α from the reduced system

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

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

The task of finding α in a numerically stable way can also be accomplished using a rank revealing URV decomposition [18]. Here, R is again a triangular matrix, and U and V are orthogonal. For the URV decomposition, the reduced system becomes

$$R_1\hat{\alpha} = U_1^T b. \tag{10}$$

where U_1 consists of the first $\hat{\ell}$ columns of U, and α is determined by multiplying V^T by the vector $\hat{\alpha}$ padded with zeroes.

Note that the QR or URV decompositions need not be recomputed every time a new column is received. Either factorization can be updated in a very efficient way using the factorization computed previously. In both cases it is possible to discard the rank deficient columns and work only with the well-conditioned submatrix. Details of updating a rank-revealing QR or URV decomposition that stores a rectangular Q or U will be given in [13].

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 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 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_aR_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.

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 $M_1z = d$ and $M_2z = d$ 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 [9]). Since we solve $M_1z_1 = d_1$ and $M_2z_2 = d_2$ 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 would set $m = 1$ and have the two tasks exchange information after each update. We show in §3 that the resulting subspace for minimization includes the Krylov subspace $\mathcal{K}(M^{-1}A, M^{-1}b, k)$. □

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

$$M = \begin{pmatrix} M_{11} & \\ & M_{22} \end{pmatrix} \tag{11}$$

$$= \begin{pmatrix} F_1 & \\ & F_2 \end{pmatrix} - \begin{pmatrix} G_1 & \\ & G_2 \end{pmatrix} \tag{12}$$

$$= F - G \tag{13}$$

where F is nonsingular, then the two tasks can iterate using the splittings $M_1 = \mathrm{diag}(F_1, 0)$, $M_2 = \mathrm{diag}(0, F_2)$. We use these two-stage methods to produce a vector

space for minimization. This raises the question about 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 linear system $Mz = d$ exactly we do not necessary increase the convergence rate. Hence the answer to this question really 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 $\hat{x}^0 = 0$.

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*

$$\hat{x}^{k+1} = B\hat{x}^k + \hat{b}, \tag{14}$$

where

$$B = \sum_{i=1}^{p} D_i (M_i^{-1} N_i)^m \tag{15}$$

and

$$\hat{b} = \sum_{i=1}^{p} D_i \sum_{j=0}^{m-1} (M_i^{-1} N_i)^j M_i^{-1} b \tag{16}$$

Thus, the directions

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

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

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*

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

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

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 $M_i^{-1} A\mathcal{K}(B, \hat{b}, K)$ in union with the span of $\{M_i^{-1} b\}$, $i = 1, ..., p$.*

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 $\mathcal{K}(H, \bar{b}, K)$, where*

$$H = \sum_{i=1}^{p} D_i (I - (F_i^{-1} G_i)^s) M_i^{-1} A \tag{17}$$

and

$$\bar{b} = \sum_{i=1}^{p} D_i (I - (F_i^{-1} G_i)^s) M_i^{-1} b. \tag{18}$$

3.2 Convergence Bounds

Now we have tools to analyze the rate of convergence of Algorithm KMS. 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 [9] 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 $\mathcal{K}(\hat{M}^{-1}A, \hat{M}^{-1}b, k)$, then the error is bounded as

$$||e_{(k+1)}||_A \ \leq \ 2(\frac{1 - \sqrt{\kappa^{-1}}}{1 + \sqrt{\kappa^{-1}}})^{k+1} ||e_0||_A \tag{19}$$

where κ is the condition number of $\hat{M}^{-1}A$, the ratio of its largest to smallest eigenvalue.

For the multisplitting algorithm,

$$\hat{M}^{-1} = \sum_{i=1}^{p} D_i \sum_{j=0}^{m-1} (M_i^{-1} N_i)^j M_i^{-1}, \tag{20}$$

and

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

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 $M_i - N_i$ is a convergent splitting of A, if $D_i = \alpha_i I$, where α_i is a positive scalar ($i = 1, ..., p$), and if we minimize $||x - x^*||_A$ over the Krylov subspace generated by the KMS algorithm then the bound (19) applies, where κ is the condition number of $I - B$.*

In the nonsymmetric case, we obtain bounds such as the standard *GMRES* result [6].

Theorem 3.3 *Suppose that we minimize the norm of the residual over the Krylov subspace generated by the KMS algorithm, and that the preconditioner B satisfies*

(i) $(v, ABv)_2 \geq \delta_1 ||v||_2^2$, *all v;*

(ii) $||ABv||_2 \leq \delta_2 ||v||_2$, *all v, for some positive constant δ_1, δ_2.*

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

$$||r_k||_2 \leq \sqrt{1 - (\delta_1/\delta_2)^2}\,||r_{k-1}||_2, \quad = 1, 2, \ldots . \quad (21)$$

4 Parallel Implementation

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 heterogenous parallel computer may be possible, using, for example, a hypercube for the multisplitting and a systolic array (or Connection Machine) for the minimization process.

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 [14,18].

Huang [10] has performed a detailed analysis of the KMS algorithm 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 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 $\frac{N^2}{2s^2}$ unknowns to each processor. The computation and communication time for each "outer" iteration (s inner iterations) is

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

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 $\mathcal{O}(\frac{kN^2+k^2}{t})$ on t processors.

5 Conclusions

We have presented a non-traditional implementation of Krylov subspace methods.

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. Additional vectors can be added to the subspace if the Krylov generators are not working fast enough to keep the minimization task busy.

2. Vectors can be dropped if they do not provide a sufficient decrease.

3. 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 zeroes. This might improve the convergence if the preconditioner is locally good but normalization between pieces of the vector is not so good.

4. 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.

5. There are natural extensions of these ideas to nonlinear problems [10].

6 Acknowledgements

In the special case of $p = 1$ (i.e., a single splitting), the idea 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.

Both authors were supported in part by the AFOSR under Grant 87-0158. The work of O'Leary was also supported 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.

References

[1] Loyce Adams and J. Ortega. A multi-color SOR method for parallel computation. In *Proc. 1982 Int. Conf. Parallel Processing*, pages 53–56, 1982.

[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, New Jersey, 1989.

[4] T.F. Chan. Rank-revealing QR factorization. *Lin. Alg. and Its Applics.*, 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 Computing*, 11:37–53, 1989.

[6] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. on Numer. Anal.*, 20:345–357, 1983.

[7] Andreas Frommer and Daniel B. Szyld. *H-Splittings and Two-Stage Iterative Methods.* Technical Report 91-71, Department of Mathematics, Temple University, Philadelphia, 1991.

[8] 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 Numer ical Analysis*, Springer-Verlag, University of Dundee, Scotland, June 1981.

[9] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, Maryland, 1983.

[10] Chiou-Ming Huang. *Preconditioning Parallel Multisplittings for Solving Systems of Equations.* Technical Report Ph.D. dissertation, Computer Science Department, University of Maryland, College Park, August, 1992.

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

[12] D. P. O'Leary. Ordering schemes for parallel processing of certain mesh problems. *SIAM J. Sci. Stat. Computing*, 5:620–632, 1984.

[13] D. P. O'Leary. *Updating a Range-Revealing QR or URV Decomposition.* Technical Report, To appear.

[14] D. P. O'Leary and P. Whitman. Parallel QR factorization by Householder and modified Gram-Schmidt algorithms. *Parallel Computing*, 16:99–112, 1990.

[15] Dianne P. O'Leary and R. E. White. Multi-splittings of matrices and parallel solution of linear systems. *SIAM J. Disc. Math.*, 6:630–640, 1985.

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

[17] R. Glowinski et al, editor. *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] G. W. Stewart. *An Updating Algorithm for Subspace Tracking.* Technical Report 2494, Department of Computer Science, University of Maryland, College Park, 1990.