SIAM J. MATRIX ANAL. APPL.
Vol. 27, No. 4, pp. 1056–1068

A MULTIPRECONDITIONED CONJUGATE GRADIENT ALGORITHM*

ROBERT BRIDSON† AND CHEN GREIF†

Abstract. We propose a generalization of the conjugate gradient method that uses multiple preconditioners, combining them automatically in an optimal way. The algorithm may be useful for domain decomposition techniques and other problems in which the need for more than one preconditioner arises naturally. A short recurrence relation does not in general hold for this new method, but in at least one case such a relation is satisfied: for two symmetric positive definite preconditioners whose sum is the coefficient matrix of the linear system. A truncated version of the method works effectively for a variety of test problems. Similarities and differences between this algorithm and the standard and block conjugate gradient methods are discussed, and numerical examples are provided.

Key words. symmetric positive definite linear systems, conjugate gradients, preconditioning, Krylov subspace solvers

AMS subject classification. 65F10

DOI. 10.1137/040620047

1. Introduction.

The conjugate gradient (CG) method celebrated a few years ago a milestone 50th anniversary. In the years since its conception in 1952 [9], the preconditioned version of CG (PCG) has established itself as the method of choice for iteratively solving large sparse symmetric positive definite (SPD) linear systems, $Ax = b$. Throughout the years the method has seen many variants and generalizations, blossoming into a large family of Krylov subspace solvers.

This paper is concerned with a variation of the standard PCG method that employs multiple preconditioners. We call the algorithm MPCG: multipreconditioned conjugate gradient. The motivation is that for preconditioning certain problems there are several alternative approaches with different desirable properties, but it may be difficult to combine them into a single effective preconditioner. A multipreconditioned solver could automatically take advantage of all available preconditioners.

Our new iterative method obtains an energy norm minimization property, while maintaining A-conjugation and orthogonality properties similar to PCG, but with iterates constructed in a *generalized* Krylov space incorporating an arbitrary set of preconditioners.

As with flexible CG [10], our approach cannot generally maintain one of the most attractive features of standard PCG: the famous three-term recurrence relation. However, in practice a truncated version of the method works efficiently in many cases we have tested. Moreover, we are able to show analytically that for the case of two preconditioners whose sum is equal to the coefficient matrix, a short recurrence relation does hold.

Block versions of CG have been proposed in the literature, but none of them considers using multiple preconditioners. O'Leary [11] derived a block CG method

*Received by the editors December 2, 2004; accepted for publication (in revised form) by M. Benzi May 12, 2005; published electronically March 17, 2006. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.
http://www.siam.org/journals/simax/27-4/62004.html
†Department of Computer Science, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (rbridson@cs.ubc.ca, greif@cs.ubc.ca).

designed to handle multiple right-hand sides, which is also capable of accelerating the convergence of linear systems with a single right-hand side. The method uses a single preconditioner and can be classified as a block Krylov method.

Brezinski [3] proposes an effective adaptation of the block CG method for solving with a single right-hand side, partitioning the initial residual into multiple search directions (see also Bantegnies and Brezinski [1]). This partitioning approach makes it possible to group together blocks of components which might have distinct behavior, possibly due to representing different physical properties. A variety of multiparameter methods have been derived using this methodology; see [3] and references therein.

A block method designed for a parallel computing environment is proposed by Gu et al. in [7, 8]. The unknowns are partitioned into disjoint subdomains, taking into consideration the number of parallel processors available, but for any given search direction the conjugation property is weakened by zeroing out components of the search directions in other subdomains, eliminating global communication bottlenecks in a parallel environment at the price of slower convergence.

Related work that should also be mentioned is the family of flexible or inexact methods [5, 10, 12, 14]. In those methods the preconditioner changes throughout the iteration, whereas in the method that we propose, MPCG, the preconditioners are fixed and are determined a priori. (We note, however, that a flexible variant of MPCG can be applied in a straightforward fashion.)

The remainder of this paper is structured as follows. In section 2 we derive a multipreconditioned steepest descent method. In section 3, the main part of the paper, we derive the MPCG algorithm and discuss its properties. Section 4 is devoted to numerical experiments. Finally, in section 5 we draw some conclusions and point out possible directions for future research.

Throughout we will assume without loss of generality that our initial guess is $x_0 = 0$, with initial residual $r_0 = b$; generally quantities computed at the ith iteration of an algorithm will have the subscript i. The matrix A is symmetric positive definite, and the preconditioners are M_j, $j = 1, \ldots, k$, where $M_j^{-1} \approx A^{-1}$. This is the one case where the subscript does not indicate the iteration at which the quantity is computed: $M_1, \ldots, M_k$ are fixed throughout the algorithm.

2. Multipreconditioned steepest descent (MPSD). Since A is symmetric positive definite, it is possible to employ the notion of an energy norm: $\|e\|_A = \sqrt{e^T A e}$. The basic steepest descent (SD) algorithm for solving $Ax = b$ is to take the negative gradient of the energy norm of the error, i.e., the steepest descent direction, which also happens to be the current residual vector r_i, as the search direction for a step that minimizes the energy norm of the error associated with the new guess:

$$\begin{aligned} p_{i+1} &= r_i, \\ \alpha_{i+1} &= (p_{i+1}^T A p_{i+1})^{-1} (p_{i+1}^T r_i), \\ x_{i+1} &= x_i + \alpha_{i+1} p_{i+1}, \\ r_{i+1} &= r_i - \alpha_{i+1} A p_{i+1}. \end{aligned}$$

Of course, convergence is much faster if the search direction is closer to the actual error $A^{-1} r_i$, so it is natural to precondition this iteration by instead choosing $p_{i+1} = M^{-1} r_i$, where $M^{-1} \approx A^{-1}$, obtaining a *preconditioned* SD algorithm (PSD).

One possible approach to further improve the new residual is to enlarge the search space from one dimension to multiple dimensions: use a set of search directions $p^1, \ldots, p^k$. In particular, if multiple preconditioners $M_1, \ldots, M_k$ are available, use

$p_{i+1}^j = M_j^{-1} r_{i+1}$. Let $P_i = [p_i^1 | \dots | p_i^k]$. To get the same energy norm minimization, we derive an MPSD algorithm:

$$\begin{aligned} p_{i+1}^j &= M_j^{-1} r_i \quad \text{for} \quad j = 1, \dots, k, \\ \alpha_{i+1} &= (P_{i+1}^T A P_{i+1})^{-1} (P_{i+1}^T r_i), \\ x_{i+1} &= x_i + P_{i+1} \alpha_{i+1}, \\ r_{i+1} &= r_i - A P_{i+1} \alpha_{i+1}. \end{aligned}$$

Note that, as is the case for other block methods, α_{i+1} is now a *vector* of coefficients specifying the linear combination of search directions for updating the guess.

3. MPCG. Although the SD method converges, it is inefficient compared with the CG method. This section establishes the multipreconditioned analogue of CG in a similar fashion to the derivation of the standard PCG, whose first step is an iteration of PSD. The analogy to the derivation of the standard method with a single preconditioner allows us to make the reasonable assumption that MPCG will improve on MPSD in a way similar to the improvement obtained by using CG (or PCG) rather than SD (or PSD).

3.1. Derivation. One way of looking at CG and why it is so much faster than SD is to interpret it as a generalized SD with multiple search directions. At step $i+1$, the search directions are simply $p_1, \dots, p_{i+1}$, i.e., the new search direction plus all the previous ones. Thus we get a global energy norm minimum, not just a local greedy minimization. The clever part about CG is choosing the search directions to be A-conjugate, so that $P_{i+1}^T A P_{i+1}$ is just diagonal and trivial to invert. Furthermore, due to the global minimization, the previous search directions P_i are orthogonal to the most recent residual, and thus $P_{i+1}^T r_i$ is zero except for the last component, making the update even simpler.

We will want to preserve these features in generalizing PCG to have multiple search directions per step (generated from multiple preconditioners). That is, we want the property

$$P_i^T A P_j = 0 \quad \text{for} \quad i \neq j.$$

We begin with one step of MPSD:

$$\begin{aligned} p_1^j &= M_j^{-1} r_0 \quad \text{for} \quad j = 1, \dots, k, \\ \alpha_1 &= (P_1^T A P_1)^{-1} (P_1^T r_0), \\ x_1 &= x_0 + P_1 \alpha_1, \\ r_1 &= r_0 - A P_1 \alpha_1. \end{aligned}$$

Then we generate the preconditioned residuals to increase the dimension of the search space:

$$z_2^j = M_j^{-1} r_1 \quad \text{for} \quad j = 1, \dots, k.$$

Let $Z_i = [z_i^1 | \dots | z_i^k]$. We will want to get P_2 from Z_2 by making it A-conjugate to the previous directions:

$$P_2 = Z_2 - P_1 (P_1^T A P_1)^{-1} P_1^T A Z_2,$$

$p_1^j = z_1^j = M_j^{-1} r_0 \quad \text{for} \quad j = 1, \ldots, k,$
$P_1 = [p_1^1 | p_1^2 | \ldots | p_1^k],$
$\alpha_1 = (P_1^T A P_1)^{-1} (P_1^T r_0),$
$x_1 = x_0 + P_1 \alpha_1,$
$r_1 = r_0 - A P_1 \alpha_1,$
Repeat $i = 1, 2, \ldots$ until convergence
$\quad Z_{i+1} = [M_1^{-1} r_i | M_2^{-1} r_i | \ldots | M_k^{-1} r_i],$
$\quad P_{i+1} = Z_{i+1} - \sum_{j=1}^{i} P_j (P_j^T A P_j)^{-1} P_j^T A Z_{i+1},$
$\quad \alpha_{i+1} = (P_{i+1}^T A P_{i+1})^{-1} (P_{i+1}^T r_i),$
$\quad x_{i+1} = x_i + P_{i+1} \alpha_{i+1},$
$\quad r_{i+1} = r_i - A P_{i+1} \alpha_{i+1},$
end

FIG. 3.1. *The MPCG algorithm.*

or more generally,

$$P_{i+1} = Z_{i+1} - \sum_{j=1}^{i} P_j (P_j^T A P_j)^{-1} P_j^T A Z_{i+1}.$$

Now, with the new A-conjugate search direction, we can again seek a global minimum that simplifies to a local computation:

$$\begin{aligned} \alpha_{i+1} &= (P_{i+1}^T A P_{i+1})^{-1} (P_{i+1}^T r_i), \\ x_{i+1} &= x_i + P_{i+1} \alpha_{i+1}, \\ r_{i+1} &= r_i - A P_{i+1} \alpha_{i+1}. \end{aligned}$$

The algorithm is given in Figure 3.1. Each iteration involves k preconditioner solves. Vector operations in the original PCG are now replaced by operations on $n \times k$ matrices. The matrices $P_j^T A P_j$ are $k \times k$, hence their inversion (for computing P_{i+1} and α_{i+1}) is computationally negligible. We can say, then, that the cost of each MPCG iteration with k preconditioners is approximately k times more expensive than a single PCG iteration, provided that the computational cost of the preconditioner solves is similar.

3.2. Orthogonality properties. Since we do a global minimization of energy norm at the previous steps, we get the usual orthogonality property

$$P_i^T r_j = 0 \quad \text{for} \quad i \leq j. \tag{3.1}$$

From (3.1) it follows that

$$\begin{aligned} r_j^T Z_i &= r_j^T \left(P_i + P_{i-1} (P_{i-1}^T A P_{i-1})^{-1} P_{i-1}^T A Z_i \right) \\ &= r_j^T P_i + r_j^T P_{i-1} (\ldots) \\ &= 0 + 0 \quad \text{for} \quad i \leq j. \end{aligned} \tag{3.2}$$

Up to this point we have derived properties which are valid for any choice of the Z_i's, e.g., even with random nonsymmetric preconditioners chosen independently at

each step. However, let us now use the fact that our preconditioners are symmetric and do not change throughout the iteration (as opposed to a flexible method). Then

$$r_j^T z_i^s = r_j^T M_s^{-1} r_{i-1} = r_j^T (M_s^{-1})^T r_{i-1} = (M_s^{-1} r_j)^T r_{i-1} = (z_{j+1}^s)^T r_{i-1}.$$

In combination with (3.2), we conclude that

$$r_j^T Z_i = 0 \quad \text{for} \quad j \neq i-1. \tag{3.3}$$

3.3. Breakdown. The MPCG algorithm could break down if $P_j^T A P_j$ is singular for any j, which happens, for example, if two or more of the preconditioners are identical to each other. As previously mentioned, $P_j^T A P_j$ is k-by-k, where k is the number of preconditioners. It is thus a very small matrix whose singularity can be easily detected. Since preconditioners are typically selected so that they produce linearly independent search directions, a breakdown should rarely occur. In any case an automatic way to avoid breakdown may be, for example, to use the singular value decomposition of $P_j^T A P_j$ to eliminate redundant search directions.

3.4. A case where PCG and MPCG are equivalent. Consider the following case of polynomial preconditioning.

PROPOSITION 3.1. *If roundoff errors are ignored, the $2j$th iteration ($j = 1, 2, \ldots$) of PCG with a preconditioner M is identical to the jth iteration of MPCG with $M_1^{-1} \equiv M^{-1}$ and $M_2^{-1} \equiv M^{-1} A M^{-1}$.*

Proof. Assuming an initial guess of $x_0 = 0$ with initial residual $r_0 = b$, the standard PCG algorithm solves for x_{2j} as the vector from the Krylov subspace

$$\mathcal{K}^{2j}(M^{-1}A; M^{-1}r_0) = \text{span}\{M^{-1}r_0, (M^{-1}A)M^{-1}r_0, \ldots, (M^{-1}A)^{2j-1}M^{-1}r_0\}$$

that minimizes the energy norm of the error. Let us show that the same subspace is obtained with j iterations of MPCG with M_1 and M_2. Since $Z_1 = P_1 = [M^{-1}r_0 | M^{-1}AM^{-1}r_0]$ and $r_1 = r_0 - AP_1\alpha_1$, we have $r_1 \in \text{span}\{r_0, AM^{-1}r_0, AM^{-1}AM^{-1}r_0\}$. The corresponding subspace from which x is chosen is spanned by $M^{-1}r_0$ and $M^{-1}AM^{-1}r_0$ in the first iteration, and is extended by $M^{-1}(AM^{-1})^2 r_0$ and $M^{-1}(AM^{-1})^3 r_0$ in the second iteration. The rest of the proof readily follows by induction, for any given integer j, by repeating the same argument. Since MPCG finds the vector from this subspace that minimizes the energy norm of the error, just as PCG does, it must produce the same iterates as PCG. ∎

The result in Proposition 3.1 can be extended to more than two preconditioners: MPCG with the k preconditioning matrices $(M^{-1}A)^{j-1}M^{-1}$, $j = 1, \ldots, k$, is equivalent in the same fashion as above to PCG with a preconditioner M.

3.5. The recurrence relation. For regular PCG observe that

$$\begin{aligned} p_j^T A z_{i+1} &= (Ap_j)^T z_{i+1} \\ &= \left(\frac{r_{j-1} - r_j}{\alpha_j} \right)^T z_{i+1} \\ &= 0 \quad \text{for} \quad j < i. \end{aligned}$$

Thus the A-conjugation step may ignore all but the previous search direction, giving the short recurrence. Unfortunately in MPCG, α is a vector and cannot be inverted, so the above does not easily generalize. However, note that in MPCG, $AP_j\alpha_j = r_{j-1} - r_j$. Using (3.3), this means that for $j < i$

$$\alpha_j^T P_j^T A Z_{i+1} = 0, \tag{3.4}$$

and then after expanding α_j^T,

$$\begin{aligned}\left((P_j^T A P_j)^{-1}(P_j^T r_{j-1})\right)^T P_j^T A Z_{i+1} &= 0,\\ r_{j-1}^T \left(P_j (P_j^T A P_j)^{-1} P_j^T A Z_{i+1}\right) &= 0.\end{aligned}$$

We also know $r_s^T(P_j(P_j^T AP_j)^{-1}P_j^T AZ_{i+1}) = 0$ for $s \geq j$ since then $r_s^T P_j = 0$. Thus even if the update $P_j(P_j^T AP_j)^{-1}P_j^T AZ_{i+1}$ is not zero, it is orthogonal to all residuals from r_{j-1} up.

While this orthogonality condition is as close to a short recurrence relation as MPCG generally gets, there is an important case in which it is provably short and there is no error in the truncation: we now formulate and prove this result, beginning with a lemma.

LEMMA 3.2. *Suppose $A = B + C$, where B and C are used as the preconditioners for MPCG. Then the 2×2 matrix $Z_j^T AZ_{i+1}$ is diagonal for $j \neq i+1$.*

Proof. Write out the columns of each Z matrix and perform the multiplication explicitly:

$$\begin{aligned}Z_j^T AZ_{i+1} &= \left[B^{-1}r_{j-1}|C^{-1}r_{j-1}\right]^T (B+C)\left[B^{-1}r_i|C^{-1}r_i\right]\\ &= \begin{bmatrix} r_{j-1}^T B^{-1}(B+C)B^{-1}r_i & r_{j-1}^T B^{-1}(B+C)C^{-1}r_i \\ r_{j-1}^T C^{-1}(B+C)B^{-1}r_i & r_{j-1}^T C^{-1}(B+C)C^{-1}r_i \end{bmatrix}\\ &= \begin{bmatrix} (\ldots) & r_{j-1}^T(B^{-1}+C^{-1})r_i \\ r_{j-1}^T(B^{-1}+C^{-1})r_i & (\ldots) \end{bmatrix}\\ &= \begin{bmatrix} (\ldots) & r_{j-1}^T(Z_{i+1}^1 + Z_{i+1}^2) \\ r_{j-1}^T(Z_{i+1}^1 + Z_{i+1}^2) & (\ldots) \end{bmatrix}\\ &= \begin{bmatrix} (\ldots) & 0 \\ 0 & (\ldots) \end{bmatrix},\end{aligned}$$

where the last step uses (3.3). ∎

We now prove the short recurrence for this $A = B + C$ case, as follows.

THEOREM 3.3. *Suppose $A = B + C$, where B and C are SPD and are used as the preconditioners for MPCG. Then the search directions satisfy the short recurrence relation*

$$P_{i+1} = Z_{i+1} - P_i(P_i^T AP_i)^{-1}P_i^T AZ_{i+1}. \tag{3.5}$$

Proof. To show that the sum in the general A-conjugation formula for P_{i+1} collapses to just the one term as in (3.5), we will prove that $P_j^T AZ_{i+1} = 0$ for $j < i$. We begin our induction argument with the $j = 1$ case.

For $j = 1$, $P_1 = Z_1$. By Lemma 3.2, $P_1^T AZ_{i+1}$ is diagonal. Also recall from (3.4) that $\alpha_1^T(P_1^T AZ_{i+1}) = 0$. We argue that α_1 has all nonzero entries unless $r_0 = 0$: a zero entry would indicate that no energy norm improvement in the solution is possible along the corresponding search direction; i.e., for that column, say the ath column p_1^a of P_1, we have $r_0^T p_1^a = 0$. But $p_1^a = M_a^{-1} r_0$, so $r_0^T p_1^a$ cannot be zero, assuming that the preconditioners are positive definite. The only diagonal matrix that has a vector with all nonzero entries in its null-space is the zero matrix. Thus $P_1^T AZ_{i+1} = 0$.

Now assume that $P_s^T AZ_{i+1} = 0$ for all $s < j$, and let us work on the case for j.

Substituting the general summation formula for P_j gives

$$P_j^T AZ_{i+1} = \left(Z_j - \sum_{s=1}^{j-1} P_s (P_s^T AP_s)^{-1} P_s^T AZ_j \right)^T AZ_{i+1}$$
$$= Z_j^T AZ_{i+1} - \sum_{s=1}^{j-1} Z_j^T AP_s (P_s^T AP_s)^{-1} P_s^T AZ_{i+1}.$$

Since $s \leq j-1 < i$ in the above sum, the factor $P_s^T AZ_{i+1}$ in each term is zero by induction, so the sum is zero. We are left with $P_j^T AZ_{i+1} = Z_j^T AZ_{i+1}$. Just as in the base case, we know this is a diagonal matrix by Lemma 3.2, and that $\alpha_j^T (P_j^T AZ_{i+1}) = 0$ by (3.4). Also as in the base case note that, by the energy norm minimization property, if an entry of α_j were zero, then it would have to be that the corresponding column of P_j was orthogonal to r_{j-1}. Say that such a column is p_j^a and the corresponding column of Z_j is z_j^a. Using the definition of P_j, we can expand:

$$r_{j-1}^T p_j^a = r_{j-1}^T z_j^a - \sum_{s=1}^{j-1} r_{j-1}^T P_s (P_s^T AP_s)^{-1} P_s^T Az_j^a.$$

Since $s \leq j-1$, the factors $r_{j-1}^T P_s$ are zero by (3.1). We are left with $r_{j-1}^T z_j^a = r_{j-1}^T M_a^{-1} r_{j-1}$, which must be positive if the preconditioners are positive definite. Thus every entry of α_j is nonzero, and so, just as in the base case, we must have that $P_j^T AZ_{i+1}$ is the zero matrix. ∎

3.6. Truncated MPCG. Theorem 3.3 shows that in certain cases there is no need to use the full MPCG. Even when the recurrence relation is not short, numerical experiments indicate that it often is acceptable to truncate the A-conjugation step to the standard short recurrence (that is, only make P_{i+1} A-conjugate to P_i instead of all previous search directions). The terms we truncate are often very small, and convergence is often not significantly slowed down by the omission. Numerical results that demonstrate this are given in section 4.

While we do not have a full analytical justification for the expectation that a truncated version of MPCG will be effective, some insight may be provided by referring to the result that was obtained earlier in this section, namely that the update $P_j (P_j^T AP_j)^{-1} P_j^T AZ_{i+1}$ is orthogonal to all the residuals from r_{j-1} up.

We define MPCG(m) to be a truncated version of MPCG in which only the last m search directions are used in each iteration. Note that another parameter necessary for defining (full as well as truncated) MPCG is k, the number of preconditioners. However, to maintain simplicity of notation, we do not incorporate it into the definition.

4. Examples. We now present numerical examples and discuss different ideas for how to choose multiple preconditioners.

4.1. ADI examples. We start this section by considering two simple model problems with ADI preconditioning.

4.1.1. Two-dimensional Poisson with weak coupling in one direction. Consider

$$-u_{xx} - \varepsilon u_{yy} = f(x, y)$$

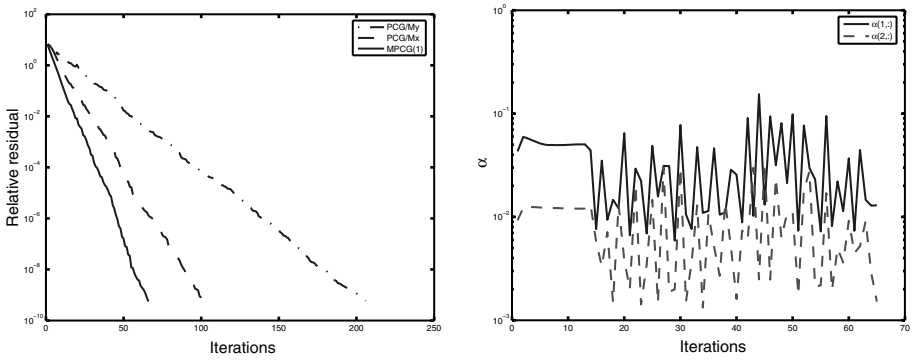

FIG. 4.1. *Convergence history for the example in section* 4.1.1*: Relative residuals and the components of* α *throughout the MPCG iteration for* $\varepsilon = 0.5$.

on $\Omega = (0,1) \times (0,1)$, with Dirichlet boundary conditions. We set the right-hand side and the boundary conditions so that $u(x,y) = \cos(\pi x)\cos(\pi y)$ is the exact solution.

Using the standard second order centered difference scheme with n grid points in each direction (that is, with mesh size $h = \frac{1}{n+1}$), the coefficient matrix is $n^2 \times n^2$, given by $A = I_n \otimes T_n + \varepsilon T_n \otimes I_n$, where $T_n = \text{tri}[-1, 2, -1]$. We select two preconditioners in an ADI fashion: $M_x = I_n \otimes T_n$ and $M_y = \varepsilon T_n \otimes I_n$. Thus, M_x is tridiagonal and corresponds to the discretized operator $-u_{xx}$, and M_y is a discrete operator corresponding to $-\varepsilon u_{yy}$. Since $A = M_x + M_y$ by construction, Theorem 3.3 holds and MPCG(1), which is based on short recurrences, produces the same iteration sequence as full MPCG (up to roundoff errors). Setting $\varepsilon = 1$ corresponds to the standard Poisson equation, for which the symmetry between x and y implies that MPCG with M_x and M_y as preconditioners is tied with the standard PCG with either M_x or M_y in terms of overall computational work. (Numerical experiments confirm this.) However, for smaller values of ε the equality of the roles played by x and y is lost, and differences between using M_x and M_y are expected. We take $\varepsilon = 0.5$ and compare the convergence of PCG and MPCG(1). Figure 4.1 illustrates the behavior of α and the convergence of MPCG(1). A 32×32 grid was used. As expected, M_x dominates the search space; the graphs for α confirm this.

Note that each iteration of MPCG(1) involves solving for two preconditioners and hence is more computationally expensive than a PCG iteration by a factor of nearly 2. The iteration counts that are presented in the graphs are 206 for PCG using M_y, 102 for PCG using M_x, and 66 for MPCG(1) using M_x and M_y. Thus, PCG with M_x outperforms MPCG(1), whereas PCG with M_y is inferior.

The point that we are making in this example is that while there might be a single preconditioner whose performance is better than a combination of preconditioners, the detection of the preconditioning for MPCG is done automatically, and does not rely on knowledge of the underlying continuous problem or properties of the matrix. MPCG could be useful in cases where one particular preconditioner clearly dominates but cannot be easily identified beforehand: a few iterations could be executed to determine what the most effective preconditioner is, and then one could switch to regular PCG with that choice.

4.1.2. ADI for three-dimensional Poisson. The three-dimensional Poisson equation, $-\nabla^2 u = f$ on the unit cube with Dirichlet boundary conditions, is discretized using standard centered finite differences and is solved using three precondi-

TABLE 4.1
The Poisson equation in three dimensions, using three preconditioners, in an ADI fashion. The right-hand-side vector in this case was random, with normal distribution.

n	n^3	MPCG (1)	Full MPCG
8	512	32	31
16	4096	61	60
24	13824	88	88

tioners in an ADI fashion: discrete operators that correspond to $-u_{xx}, -u_{yy}, -u_{zz}$. We used a random right-hand side and ran the program for several meshes. The short recurrence relation does not hold in this case; this was observed by keeping track of $P^T AZ$ throughout the iteration. However, numerical experiments indicate that MPCG(1) performs as well as full MPCG. In other words, the convergence behavior is practically unaffected by the truncation. Results are given in Table 4.1. We do not have an analytical explanation for this result.

4.2. Domain decomposition. One of the natural applications for MPCG is domain decomposition: each preconditioner corresponds to (approximately) solving a restriction of the PDE to a subdomain. MPCG will then automatically provide something akin to a coarse grid correction: the matrix equation for α is a Galerkin projection of the matrix onto a small subspace with one degree of freedom per subdomain. This allows for much greater scalability than the corresponding PCG method using just a fixed combination of subdomain solves. Note that the preconditioners for each subdomain will be singular.

For preliminary experiments we solved the standard 5-point Poisson problem on a square grid with Dirichlet boundary conditions. For preconditioners we partition the domain into disjoint rectangles, in each of which we exactly solve the restriction of the problem, i.e., inverting the submatrix of A corresponding to those unknowns. For regular PCG, we assemble these into a standard block diagonal preconditioner. For MPCG, we treat each subdomain solve as a separate preconditioner that can supply a unique search direction.

Our first observation is that if we have just two subdomains, then MPCG apparently (observed to roundoff error) preserves the short recurrence—though this situation is not covered by Theorem 3.3. In this case, MPCG is noticeably more efficient than PCG; e.g., to solve on a 100×100 grid (with 100×50 subdomains) to 10^{-10} relative residual reduction took PCG 49 iterations but MPCG just 37. The cost of these iterations is dominated by the subdomain solves, so the small amount of extra work that MPCG does per iteration is more than compensated for by the enhanced convergence rate. Figure 4.2 shows the residual norm histories.

For more subdomains, the short recurrence property is lost. However, we can see that the full (nontruncated) form of MPCG actually has significantly better scalability than standard PCG, at least in terms of iteration counts. If we keep the subdomain size constant as we increase the grid size, then the iteration count for PCG increases linearly with the side length of the grid. However, for full MPCG, the iteration count appears to increase only logarithmically; see Table 4.2 for the numbers from our numerical experiment.

Unfortunately with truncation—even keeping two or three previous iterations' search directions and not just one—the scalability is diminished and the results are noticeably slower. We conjecture that truncating to one search direction leads to a linear convergence (like PCG, but slower), but retaining more search directions

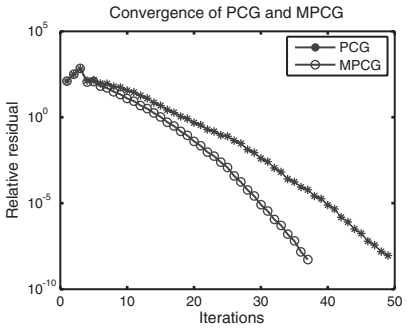

FIG. 4.2. *Residual norm history of PCG and MPCG for a two-dimensional Poisson problem on a* 100×100 *grid, preconditioned with two* 100×50 *subdomain solves.*

TABLE 4.2
Iteration counts for a two-dimensional domain decomposition scalability test, using approximately 8×8 *subdomain solves for the preconditioners.*

Grid size	PCG	Truncated MPCG (1)	Truncated MPCG (2)	Truncated MPCG (3)	Full MPCG
25	39	69	45	44	19
50	70	131	77	67	22
100	126	257	125	107	24

steadily improves the scalability, ultimately towards $O(\log n)$ for full MPCG. For example, MPCG(3) (keeping three previous search direction groups) appears from Table 4.2 to lead to $O(n^{2/3})$ iterations for an $n \times n$ grid.

4.3. A model bending problem. Our motivation for this example is plate and shell elasticity problems, or more generally PDEs where the matrix to solve is the sum of relatively easy to precondition parts (e.g., second order differential operators) and more challenging parts (e.g., fourth order differential operators). We used the standard centered finite difference discretization of

$$B\nabla^4 u - S\nabla^2 u + \frac{1}{\Delta t} u = f$$

with clamped boundary conditions ($u = \frac{\partial u}{\partial n} = 0$) on a unit square domain as a model for an implicit time step in a bending simulation.

The biharmonic term in this problem gives rise to a non-M-matrix and can cause standard incomplete Cholesky methods to break down, though modified incomplete Cholesky, for example, works very well for the other terms. A robust alternative that has been successfully applied to difficult shell problems is stabilized AINV [2]. We investigate using both SAINV (on the full matrix, permuted with a minimum degree ordering, with drop tolerance 0.1) and modified incomplete Cholesky (on all terms except the biharmonic operator, using the regular grid ordering, with level 0 fill) in MPCG.

Our test case uses a 100×100 grid, $B = 10^{-6}$, $\Delta t = 10^{-2}$, and various values for S. Our motivation for these specific choices is related to scaling of the operators. We present iteration counts for PCG with the two different preconditioners as well as their sum (i.e., giving them equal weight), for full MPCG and for truncated MPCG(1) and MPCG(2), in Table 4.3.

TABLE 4.3
Iteration counts (to reduce the residual by 10^{-10}) for a model bending problem. The parameter S is the coefficient that appears in the PDE.

S	SAINV PCG	MIC(0) PCG	Combined PCG	Full MPCG	Truncated MPCG(1)	Truncated MPCG(2)
10^{-3}	63	146	143	48	65	57
10^{-2}	49	126	117	40	47	47
10^{-1}	56	81	70	35	41	41
1	122	58	54	41	54	46
10	171	56	54	44	58	50
10^{2}	186	56	54	44	61	50

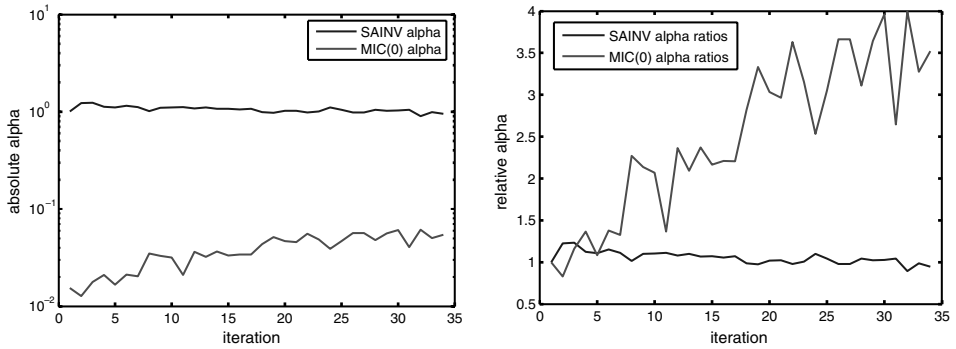

FIG. 4.3. *Plots of how the two components of α (one for the SAINV preconditioner and the other for the modified incomplete Cholesky preconditioner) evolve over the iterations for full MPCG applied to the $S = 0.1$ bending problem. The right-hand plot shows the relative change from the initial value of α.*

Observe that as the relative importance of the second order term changes, the effectiveness of PCG with a particular choice of preconditioners varies significantly. Meanwhile, full MPCG followed closely by truncated MPCG(2) robustly achieve the minimum iteration counts—though of course doing more work per iteration. For the more imbalanced problems (S very small or very large) it is almost certain that PCG with the appropriate preconditioner will be the clear winner in terms of actual time, but for the more interesting balanced cases—where it is unclear a priori what the appropriate preconditioner is—truncated MPCG could be a very competitive, robust choice.

To illustrate some of the dynamic behavior of MPCG, we plot the two components of α for full MPCG in the $S = 0.1$ problem in Figure 4.3. While the contribution from the SAINV preconditioner remains steady, the contribution from the modified incomplete Cholesky preconditioner steadily grows. We hypothesize this is due to SAINV being more effective overall, but MIC(0) doing a better job on low frequencies—which eventually are all that is left after SAINV deals with the rest of the spectrum. These steadily changing weights could not be duplicated by a fixed combination in regular PCG. Interestingly, we do not see the upwards trend in truncated MPCG: further investigation is required to understand this behavior.

We have observed variations on this problem where MPCG does not fare as well. From these experiments it appears that MPCG usually behaves in one of two ways (excepting the scalable domain decomposition results in the previous section, where we get the coarse grid correction effect). In some problems the multiple preconditioners

act synergistically, and the full and truncated forms of MPCG perform comparably: the additional search directions from the multiple preconditioners more than make up for the loss of global orthogonality and attendant loss of global optimality. For other problems truncated MPCG performs poorly, and while full MPCG necessarily converges in fewer iterations than simple PCG, it appears not to afford a major improvement: the extra search directions are nearly redundant, so PCG with its guarantee of global optimality is more efficient than truncated MPCG.

5. Conclusions. The MPCG method derived in this paper is an algorithm that incorporates multiple preconditioners automatically and obtains optimality in an energy minimization sense. The algorithm constructs a generalized Krylov space whose dimension is proportional to the number of preconditioners incorporated. Short recurrences cannot generally be preserved when more than one preconditioner is involved, but we were able to show analytically (Theorem 3.3) that for two preconditioners whose sum is equal to the coefficient matrix itself, a short recurrence relation holds, and the truncated algorithm MPCG(1) can be used without giving away anything. In addition, we have experimentally observed two situations which our analysis does not cover: in the three-dimensional Poisson equation with three ADI preconditioners, MPCG(1) converges as fast as full MPCG, even though the short recurrence relation does not hold. And in a nonoverlapping domain decomposition test problem the short recurrence holds for two subdomains. When more than two subdomains are applied, the short recurrence is lost, but scalability remains good, as is demonstrated in section 4.2.

In many complicated and large-scale problems, the choice of a preconditioner is not obvious, and if more than one candidate is available, a fixed combination of the preconditioners may not work well enough. This is one place where the mechanism of MPCG may come in handy, since it determines throughout the iteration how to combine the preconditioners. Even if MPCG ultimately is not faster than PCG with the right selection of preconditioner, a few iterations of MPCG may robustly identify what that selection should be.

Parallelism may be another aspect worth exploring. While we have not implemented our algorithm in a parallel environment, it is evident that the time-consuming step $Z_{i+1} = [M_1^{-1}r_i|M_2^{-1}r_i|\dots|M_k^{-1}r_i]$ can be straightforwardly parallelized. There may be useful parallel speed-up even from highly sequential preconditioners: for example, instead of running PCG with incomplete Cholesky, leaving half of a dual-processor workstation sitting idle, two variations on incomplete Cholesky could be run in parallel with MPCG.

Domain decomposition applications naturally lend themselves to an approach such as MPCG, particularly if the physics of one domain is significantly different from the physics of another domain (e.g., due to material interfaces). Singular preconditioners that practically affect only one particular subdomain could be used. Also, a flexible variant of MPCG might prove useful, allowing the preconditioners to vary throughout the iteration, for example, if they are applied to subproblems using PCG with a rough convergence tolerance.

Again recall that in the domain decomposition example our preconditioners were singular. Regular PCG of course cannot tolerate preconditioners whose null-spaces overlap the span of the right-hand side. It is tempting to ask whether we can push this further, with preconditioners that are even slightly indefinite (in different subspaces). A motivating case here is the sparse approximate inverse SPAI [6], whose definiteness is difficult to determine.

Another possible research direction is the derivation of multipreconditioned solvers for other classes of linear systems. In particular, since GMRES [13] does not possess a short recurrence relation, a multipreconditioned version of GMRES might be very competitive.

A MATLAB implementation of the MPCG method is available at [4]. The authors welcome comments and suggestions.

Acknowledgments. We thank Edmond Chow for his modified incomplete Cholesky code, which was used in the example in section 4.3, and an anonymous referee for helpful remarks.

REFERENCES

[1] F. BANTEGNIES AND C. BREZINSKI, *The Multiparameter Conjugate Gradient Algorithm*, Technical Report 429, Laboratoire d'Analyse Numérique et d'Optimisation, Université des Sciences et Technologies de Lille, Lille, France, 2001.

[2] M. BENZI, J. K. CULLUM, AND M. TUMA, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[3] C. BREZINSKI, *Multiparameter descent methods*, Linear Algebra Appl., 296 (1999), pp. 113–141.

[4] R. BRIDSON AND C. GREIF, *MPCG: Multi-preconditioned conjugate gradients implemented in* MATLAB, available online at http://www.cs.ubc.ca/∼rbridson/mpcg/.

[5] G. H. GOLUB AND Q. YE, *Inexact preconditioned conjugate gradient method with inner-outer iteration*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.

[6] J. M. GROTE AND T. HUCKLE, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[7] T. GU, X. LIU, Z. MO, AND X. CHI, *Multiple search direction conjugate gradient method* I: *Methods and their propositions*, Int. J. Comput. Math., 81 (2004), pp. 1133–1143.

[8] T. GU, X. LIU, Z. MO, AND X. CHI, *Multiple search direction conjugate gradient method* II: *Theory and numerical experiments*, Int. J. Comput. Math., 81 (2004), pp. 1289–1307.

[9] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[10] Y. NOTAY, *Flexible conjugate gradients*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460.

[11] D. P. O'LEARY, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[12] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.

[13] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[14] V. SIMONCINI AND D. B. SZYLD, *Flexible inner-outer Krylov subspace methods*, SIAM J. Numer. Anal., 40 (2003), pp. 2219–2239.