Multiple search direction conjugate gradient method I: methods and their propositions

Tongxiang Gu , Xingping Liu , Zeyao Mo & Xuebin Chi

International Journal of Computer Mathematics
Vol. 81, No. 9, September 2004, pp. 1133–1143

MULTIPLE SEARCH DIRECTION CONJUGATE GRADIENT METHOD I: METHODS AND THEIR PROPOSITIONS

TONGXIANG GU[a,b,*], XINGPING LIU[a], ZEYAO MO[a] and XUEBIN CHI[b]

[a]*Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, P.O. Box 8009, Beijing 100088, P.R. China;*
[b]*Supercomputing Center of Computer Network Information Center, Chinese Academy of Science, P.O. Box 349, Beijing 100080, P.R. China*

(Revised 24 September 2003; In final form 10 March 2004)

In this article, we proposed a new CG-type method based on domain decomposition method, which is called multiple search direction conjugate gradient (MSD-CG) method. In each iteration, it uses a search direction in each subdomain. Instead of making all search directions conjugate to each other, as in the block CG method [O'Leary, D. P. (1980). The block conjugate gradient algorithm and related methods. *Lin. Alg. Appl.*, **29**, 293–322.], we require that they are nonzero in corresponding subdomains only. The GIPF-CG method, an approximate version of the MSD-CG method, only requires communication between neighboring subdomains and eliminate global inner product entirely. This method is therefore well suited for massively parallel computation. We give some propositions and a preconditioned version of the MSD-CG method.

Keywords: Linear systems; Conjugate gradient-type method; Massively parallel computing; Inner product; Global communication

AMS subject classifications: 65F10; 65Y99

C.R. Categories: G1.3; G1.8

1 INTRODUCTION

A number of applications in science and engineering give rise to systems of linear equations. Such systems are extremely large and sparse, and require the use of supercomputers to compute the solution within reasonable time. A variety of iterative methods have been proposed to solve such linear systems. The main methods are Krylov subspace methods and their parallel implementation [see Refs. 15–18]. More and more researchers are paying their attention to preconditioning and parallel preconditioning techniques associated with Krylov subspace methods [see Refs. 1, 3, 11, 17].

Conjugate gradient (CG) method is an efficient Krylov subspace method for symmetric positive definite (SPD) problems. The difficulties in the parallelization of preconditioned conjugate gradient [PCG, see for instance in Ref. 17] method are the parallelization of preconditioner and

* Corresponding author. E-mail: gtx@iapcm.ac.cn

ISSN 0020-7160 print; ISSN 1029-0265 online © 2004 Taylor & Francis Ltd
DOI: 10.1080/00207160410001712305

the calculation of global inner products. The latter item is especially important on massively parallel computers. For inner product, we need global communication for both the reduction operation and for the broadcast of the assembled inner product, because all processors need to know the result. Inner products always act in parallel environment as synchronization points and require global communication. On distributed memory machines they form, apart from the preconditioning, a major bottleneck.

For a $p \times p$ processor grid ($P = p \times p$), global communication costs needed by inner product are proportional to p. This means that, for a constant length of vector segment per processor, these communication costs will dominate when p is large enough. This may be a severely limiting factor in achieving high speedups in a massively parallel environment, when n/P is only modest, say a few hundred. In the CG method, a new basis vector has to be made orthogonal to only the previous two basis vectors and this might lead one to be made that the inner products are less of a problem for this method. In Ref. [5], it is shown that, for matrices of order 90,000P with only five nonzero entries per row, the communication will dominate when the number of processor P is greater than 400 on a Parsytec GCel Computer. Note that the order of the matrices scales with the number of processors, but nevertheless the communication for the two inner products will eventually dominate. The main problem is that the communication for the two inner products cannot be combined, because the two inner products cannot be executed immediately after each other.

Several authors [3, 4, 6, 11, 12] have attempted to reduce the number of synchoronization points (and to improve the computation to memory reference ratio). The scheme of Meurant [11] is the prototype for these attempts: the two separated inner products can be replaced by three consecutive inner products. They can be computed in parallel and the communications can be combined.

In Refs. [2, 7, 8], another variants of CG were suggested, in which there is more possibility for overlapping all of the communication time with useful computations. This variant is nothing but a rescheduled version of the original CG scheme and is therefore equally stable. The key trick in this approach is to delay the updating of solution vector by one iteration step. This creates a possibility for overlap, since the update does not have to wait for the completion of the inner products.

All these methods improved the parallelism of the CG method, but they cannot eliminate the overhead from global inner products entirely. Alternatives for the PCG method on parallel computers are domain decomposition methods of additive Schwarz type [see Refs. 17, 18, for example], in which communication is only required between neighbouring subdomains. The classical example of these methods is the block Jacobi (BJ) method. The drawback of these methods is however that the convergence degrades if the number of subdomains is increased. This is mainly due to the lack of global information.

In this article, a method of multiple search directions conjugate gradient, MSD-CG in brief, is presented that is midway between the CG method and the BJ method. It is based on a notion of subdomains or partitioning of the unknowns. In each iteration there is one search direction per subdomain that is zero in the vector elements that are associated with other subdomains.

In this method, the global inner products have been replaced by the solution of a small linear system of equations. This small system has the same interaction structure as the subdomains that have been chosen. The iterative solution of this small system by a few iterations of some basic iterative method, such as Jacobi, leads to an approximate version of the MSD-CG method in which communication is only required between neighbouring subdomains. We call this approximate the MSD-CG method as global inner product free CG (GIPF-CG) method. It will be shown that the performance of the GIPF-CG method, in terms of operation counts, is only slightly worse than that of the CG method. Due to its better parallelization properties it can be a good alternative of the CG method on massively parallel computers.

2 MSD-CG METHOD AND PMSD-CG METHOD

Consider the solution of large sparse linear system

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

where $A \in \mathbb{R}^{n\times n}$, x and $b \in \mathbb{R}^n$. If A is SPD, one of the most efficient method for solving (1) is CG method:

ALGORITHM 2.1 (CG method)

```
Given $x^0 \in \mathbb{R}^n$, compute $r^0 = b - Ax^0$, set $p^0 = r^0$
For $k = 0, 1, 2, \dots$, until convergence, Do
    $\alpha^k = (p^k, r^k)/(Ap^k, p^k)$
    $x^{k+1} = x^k + \alpha^k p^k$
    $r^{k+1} = r^k - \alpha^k Ap^k$
    $\beta^k = -(Ap^k, r^{k+1})/(Ap^k, p^k)$
    $p^{k+1} = r^{k+1} + \beta^k p^k$
EndDo
```

In each iteration of Algorithm 2.1, it needs a matrix–vector product Ap^k which the number of operation denoted by mv(A), three inner products (p^k, r^k), (Ap^k, p^k) and (Ap^k, r^{k+1}), three vector updates and two divide. The number of operations per iteration is

$$O_{\mathrm{CG1}} = 12n + \mathrm{mv}(A) + 2 \tag{2}$$

From the properties of the CG method, parameter α^k and β^k in Algorithm 2.1 can be replaced by

$$\alpha^k = \frac{(r^k, r^k)}{(Ap^k, p^k)}, \quad \beta^k = \frac{(r^{k+1}, r^{k+1})}{(r^k, r^k)} \tag{3}$$

and this leads to the standards CG method, in which two inner products need per iteration. The number of operations per iteration is

$$O_{\mathrm{CG2}} = 10n + \mathrm{mv}(A) + 2 \tag{4}$$

On the basis of domain decomposition, we give a partitioning of unknowns. Assume that the index set $\{1, \dots, n\}$ is divided into L disjoint subsets or subdomains sub(l) for $l = 1, \dots, L$. The solution of original problem (1) is reduced to parallel solution of subsystems on subdomains, such as BJ method. The BJ method can be seen as a group of workers that cooperatively solve a problem. The workers are all responsible for a subdomain and treat the latest known values on the interfaces as given. It is often useful to solve the subdomain problems approximately, for instance by m iterations of the CG method. The method thus obtained is called in-exact BJ with CG(m) inner iteration or the IBJ-CG(m) method. The convergence of the BJ and the IBJ-CG(m) methods is often slow since the lack of global information.

We observe that in the IBJ-CG(m) method, the workers choose updates from a local minimization property. Furthermore, there is no attempt to take the updates of other subdomains into account, or to predict the effect of a local update on the error of other subdomains. The CG method for SPD problem makes energy normal of error minimum per iteration, but it needs global communication. We consider such a method that satisfies some minimal property, does not need global communication but does not lose the transportation of global information to ensure the faster convergence.

In order to do so, at each CG iteration k, we choose a search direction p_l^k associate with each sub(l) for $l = 1, \dots, L$. You can choose p_l^k under some rules. In this article, we restrict to taking the ith component of p_l^k equals zero for $i \notin$ sub(l). We call $p^k = \sum_{l=1}^{L} p_l^k$ the global search direction. On the other hand, let T_l be the operate which projects a vector onto sub(l), *i.e.*, it replaces all vector-elements not in sub(l) by zero. We get that p_l^k can be viewed as the project of the globe search direction onto sub(l), *i.e.*, $p_l^k = T_l(p^k)$. Let N_l be the number of element in sub(l), you can see that p_l^k is a n-vector, which has at most N_l nonzero components at the position associate with sub(l).

Denote P_k the $n \times L$ matrix which contains all search directions as its columns, *i.e.*, $P_k = [p_1^k, p_2^k, \dots, p_L^k]$. For example, in finite difference method, we can divide the unknowns into 16 disjoint subdomains (see Fig. 1).

If there are 16 nodes in each subdomain and we take natural ordering for subdomains and nodes therein, the structure of matrix P_k has form as follows

$$P_k = \begin{pmatrix} * & 0 & & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ * & 0 & & 0 & 0 \\ 0 & * & & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & * & & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & & * & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & * & 0 \\ 0 & 0 & & 0 & * \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & 0 & * \end{pmatrix}_{256\times 16}$$

Based on this domain decompostion, we can distribute matrix $A \in \mathbb{R}^{256\times 256}$ on a 4×4 parallel computer. Vectors of approximate solution, residual and multiple search directions are distributed correspondingly. Then, we obtain a parallel version of the CG method. This leads inner products (Ap^k, p^k), (p^k, r^k), (Ap^k, r^{k+1}) in CG turning to matrix–matrix and matrix–vector product $P_k^{\mathrm{T}} A P_k$, $P_k^{\mathrm{T}} r^k$, $P_k^{\mathrm{T}} A r^{k+1}$ and scalar α^k and β^k in CG to vectors of length L which each component associate with the corresponding subdomain. Vectors α^k and β^k are the solutions of small $L \times L$ linear systems (Eqs. (5) and (6) in following algorithm) which contain global information rather than divided of two inner products. What we want to get is some minimize property of the new method by choosing vector α^k appropriately. We call this method multiple search direction conjugate gradient method, MSD-CG in brief.

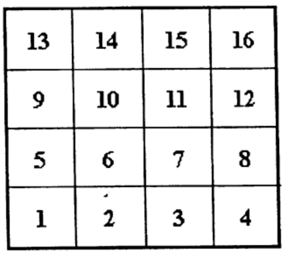

FIGURE 1 A 4×4 domain decomposition.

ALGORITHM 2.2 (MSD-CG method)

```
Given $x^0 \in \mathbb{R}^n$, compute $r^0 = b - Ax^0$,
Set
$p_l^0 = T_l(r^0)$ for $l = 1, \ldots, L$
For $k = 0, 1, 2, \ldots$, until convergence, Do
    $Q_k = AP_k$ , $C_k = Q_k^{\mathrm{T}} P_k = ((Ap_i^k , p_j^k ))$
    solve $C_k \alpha^k = P_k^{\mathrm{T}} r^k$        (5)
    $x^{k+1} = x^k + P_k \alpha^k$
    $r^{k+1} = r^k - Q_k \alpha^k$
    solve $C_k \beta^k = - Q_k^{\mathrm{T}} r^{k+1}$        (6)
    $p_l^{k+1} = T_l (r^{k+1}) + \beta_l^k p_l^k$ for $l = 1, \ldots , L$
EndDo
```

Note that if the number of subdomains L equals to 1, the MSD-CG method reduces to the CG method.

Now, we consider the storage requirement of the MSD-CG. It is roughly the same as that for CG. Both require an n-vector for x, r and p or P. The differences reside in the $L \times L$ matrix C in MSD-CG and the matrix Q. The nonzeros in q_l are the elements of sub(l) and the elements that are *adjacent* to sub(l):

DEFINITION 2.1 *An unknown i is called adjacent to sub(l) of the partitioning with respect to the system matrix A if $i \notin sub(l)$ and there exists a $j \in sub(l)$, such that $a_{ij} \neq 0$.*

All elements together that are adjacent to sub(l) are called the *interface points* of sub(l), and their number is denoted by IP(l). The total number of nonzeros in Q that must be stored is now $n + \sum_l \mathrm{IP}(l)$.

The number of operations per iteration of the MSD-CG method is

$$O_{\text{MSD-CG}} = 12n + 6\sum_{l=1}^{L} \mathrm{IP}(l) + \mathrm{mv}(A) + 2\mathrm{sv}(C) \tag{7}$$

where sv(C) stands for the number of operation for the solution of a system with coefficient matrix C. Obviously, $O_{\text{MSD-CG}}$ also depends on the structure of matrix A. Compare with O_{CG1} of Eq. (2), the increments of MSD-CG method is the second and the fourth items of Eq. (7). These increments are small when L is small.

Now, we give left preconditioned MSD-CG method as following. Assume that matrix A of original problem (1) and the left preconditioner M be SPD. Although A and M is symmetric cannot assures the symmetry of $M^{-1}A$, $\langle x, y\rangle = (x, My)$ definite a suitable inner product if M

is SPD. It is easy to show that $M^{-1}A$ is symmetric respect to this new inner product. Therefore, we can apply our the MSD-CG method to the preconditioned system

$$M^{-1}Ax = M^{-1}b \tag{8}$$

and we call this method PMSD-CG method.

ALGORITHM 2.3 (PMSD-CG method)

```
Given $x^0 \in \mathbb{R}^n$, compute $r^0 = b - Ax^0, z^0 = M^{-1}r^0$ and Set $p_l^0 = T_l(z^0)$ for $l = 1, \ldots, L$
For $k = 0, 1, 2, \ldots$, until convergence, Do
    $Q_k = M^{-1}AP_k, C_k = Q_k^{\mathrm{T}}P_k = P_k^{\mathrm{T}}AM^{-1}P_k$
    solve$C_k\alpha^k = P_k^{\mathrm{T}}z^k$                                    (9)
    $x^{k+1} = x^k + P_k\alpha^k$
    $r^{k+1} = r^k - Q_k\alpha^k$
    $z^{k+1} = M^{-1}r^{k+1}$
    solve $C_k\beta_k = -Q_k^{\mathrm{T}}z^{k+1}$                               (10)
    $p_l^{k+1} = T_l(z^{k+1}) + \beta_l^k p_l^k$ for $l = 1, \ldots, L$
EndDo
```

Note that if $L = 1$, PMSD-CG method degenerate the PCG method.

3 PROPOSITIONS

In this section, we deduce some basic properties of the MSD-CG and the PMSD-CG methods. We assume $A \in \mathbb{R}^{n\times n}$ is SPD except special illustration.

PROPOSITION 3.1 *If p_l^k in MSD-CG method is nonzero for each k and all l, then the matrix C_k is also SPD.*

Proof Since A is symmetric, then $C_k = (AP_k)^{\mathrm{T}}P_k = P_k^{\mathrm{T}}AP_k$ is symmetric too.

For any $0 \neq y \in \mathbb{R}^L$, from all vectors $p_l^k = T_l(r^k) \neq 0$, we obtain that $0 \neq \sum_{i=1}^L y_l p_l^k = P_k y \in \mathbb{R}^n$. Therefore,

$$y^{\mathrm{T}}C_k y = (P_k y)^{\mathrm{T}}AP_k y > 0$$

Hence C_k is positive definite. This completes the proof. ■

Remark From this proposition and the design of the MSD-CG method, we have that the structure and property of small linear systems (5) and (6) are determined by that of the original system (1) and the choice of multiple search directions.

PROPOSITION 3.2 *If p_l^k in the MSD-CG method is nonzero for each k and all l, then MSD-CG method is definite, i.e., there is no breakdown at each iteration.*

The proof is obvious.

PROPOSITION 3.3 *The choice of α^k in the MSD-CG method is such that $\|x^{k+1} - x^*\|_A$ is minimized over all $\alpha^k \in \mathbb{R}^L$. The energy normal of errors $e^{k+1} = \|x^{k+1} - x^*\|_A$ is nonincreasing, where x^* is the solution of $Ax = b$.*

Proof Let $F(\alpha^k) = \|x^{k+1} - x^*\|_A^2$, we have

$$\begin{aligned} F(\alpha^k) &= (x^{k+1} - x^*)^{\mathrm{T}} A (x^{k+1} - x^*) \\ &= (x^k - x^* + P_k\alpha^k)^{\mathrm{T}} A (x^k - x^* + P_k\alpha^k) \\ &= (x^k - x^*)^{\mathrm{T}} A (x^k - x^*) + 2\alpha^{k^T} P_k^{\mathrm{T}} A (x^k - x^*) + \alpha^{k^{\mathrm{T}}} P_k^{\mathrm{T}} A P_k \alpha^k \\ &= \left\|x^k - x^*\right\|_A^2 - 2\alpha^{k^{\mathrm{T}}} P_k^{\mathrm{T}} r^k + \alpha^{k^{\mathrm{T}}} C_k \alpha^k \end{aligned}$$

Therefore,

$$\nabla F = -2P_k^{\mathrm{T}} r^k + 2C_k\alpha^k = 0$$

Hence, the choice of α^k in the MSD-CG method is such that $\|x^{k+1} - x^*\|_A$ is minimized over all $\alpha^k \in \mathbb{R}^L$, and we also have

$$(e^{k+1})^2 = (e^k)^2 - \alpha^{k^{\mathrm{T}}} C_k \alpha^k \le (e^k)^2$$

If all vectors p_l^k are nonzero, the above inequality is strict. ■

Remark It says that the choice of α^k satisfies a minimal property (over affine space e^k+Range (P_k)) and the energy normal of global error of the method is nonincreasing.

PROPOSITION 3.4 *In the MSD-CG method, the global search direction $p^{k+1} = \sum_l p_l^{k+1}$ satisfies*

$$(p^{k+1}, q_j^k) = 0 \textit{ for } j = 1, \dots, L \tag{11}$$

Proof From the jth equation of Eq. (6), we obtain that $(P_k\beta^k, q_j^k) = -(q_j^k, r^{k+1})$.
Hence,

$$\begin{aligned} (p^{k+1}, q_j^k) &= \left(\sum_{l=1}^{L} T_l(r^{k+1}) + \sum_{l=1}^{L} \beta_l^k p_l^k, q_l^k\right) \\ &= (r^{k+1}, q_j^k) + (q_j^k, P_k\beta^k) \\ &= 0 \end{aligned}$$

■

Remark Since $q_j^k = Ap_j^k$, this proposition says that p^{k+1} is conjugate to all of the previous iteration. This describes how to construct new search directions and results in the fastest convergence in a comparison with several other strategies.

COROLLARY 3.1 *Globe search directions between two consecutive iteration of the MSD-CG method of conjugate, i.e., $(p^{k+1}, Ap^k) = 0$.*

PROPOSITION 3.5 *In the MSD-CG method, r^{k+1} is orthogonal to p_l^k for all $l = 1, \dots, L$ and hence $(r^{k+1}, p^k) = 0$.*

Proof From the lth equation of Eq. (5), we obtain that $(p_l^k, r^k) = (\sum_{j=1}^{L} \alpha_j^k q_j^k, p_l^k)$.

Hence,

$$\begin{aligned}(r^{k+1}, p_l^k) &= \left(r^k - \sum_{l=1}^{L} \alpha_j^k q_j^k, p_l^k\right)\\ &= (r^k, p_l^k) - \left(\sum_{l=1}^{L} \alpha_j^k q_j^k, p_l^k\right)\\ &= 0\end{aligned}$$

Remark The above propositions are similar to those of the CG method, but we do not find such multiple search directions that the MSD-CG method satisfies a finite termination property.

If matrix $A \in \mathbb{R}^{n\times n}$ is SPD and $p_l^k \neq 0$ for each k and all l, $C_k = P_k^{\mathrm{T}} A P_k$ can be shown to be SPD (see Proposition 3.1) and hence, nonsingular. From Eq. (5), we have

$$\begin{aligned}x^{k+1} &= x^k + P_k\alpha^k = x^k + P_k(P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}} r^k\\ &= (I - P_k(P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}} A)x^k + P_k(P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}} A x^* \qquad (12)\end{aligned}$$

Denote

$$S_k = P_k(P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}} A \qquad (13)$$

we can rewrite Eq. (12) as

$$x^{k+1} = (I - S_k)x^k + S_k x^* \qquad (14)$$

Denote $e^k = x^k - x^*$ the kth error and,

$$B_k = P_k(P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}}$$

we have error equation

$$e^{k+1} = (I - S_k)e^k \qquad (15)$$

and Eq. (12) can be rewritten as

$$x^{k+1} = x^k + B_k r^k = x^k + B_k(b - Ax^k) \qquad (16)$$

You can see from Eq. (16) that the MSD-CG method can be viewed as a more general non-stationary Richardson iterative method. We point that S_k in Eq. (13) is the matrix representation of projection that orthogonal to Range (AP_k) onto Range (P_k). By simple derivation, we have the following proposition.

PROPOSITION 3.6 *If p_l^k in the MSD-CG method is nonzero for each k and all l, then*

(a) $S_k^2 = S_k$;
(b) $(I - S_k)P_k = 0$;
(c) $(I - S_k^T)AP_k = 0$.

Proof We only give the proof of (a) and you can obtain (b) and (c) similarly.

$$S_k^2 = P_k(P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}} A P_k (P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}} A = P_k(P_k^{\mathrm{T}} A P_k)^{-1} P_k^{\mathrm{T}} A = S_k$$

■

Remark 1 From (b), $P_k^{\mathrm{T}}(I - S_k^{\mathrm{T}}) = 0$ and $(P_k^{\mathrm{T}}(I - S_k^{\mathrm{T}})r^k, w) = 0$ for any $w \in \mathbb{R}^L$. Hence, $(r^{k+1}, P_k^{\mathrm{T}} w) = 0$. If we take w as the unit base vectors in $\mathbb{R}^L$, we can get $(r^{k+1}, p_l^k) = 0$ for $l = 1, \ldots, L$. This is Proposition 3.4.

Remark 2 From (c), $P_k^{\mathrm{T}} A(I - S_k) = 0$ and $(A(I - S_k)e^k, P_k w) = 0$ for any $w \in \mathbb{R}^L$. If we take $w = \alpha^k$, then $(Ae^{k+1}, P_k\alpha^k) = 0$ from Eq. (15) and $\|e^{k+1}\|_A^2 = \|e^k\|_A^2 + \|P_k\alpha^k\|_A^2 \le \|e^k\|_A^2$. There is no increase of error energy norm of Proposition 3.3.

We can obtain the following corollary from 2 and 3 of Proposition 3.6.

COROLLARY 3.2 $(I - S_k)c = 0$ *for vector* $c \in$ *Range* (P_k)

$$(I - S_k^T)c = 0 \textit{ for vector } c \in \textit{Range } (AP_k).$$

The following propositions show the relationship between residual and global search direction and previous residual in the MSD-CG method.

PROPOSITION 3.7 *If* p_l^k *in Algorithum* 2.2 *is nonzero for each* k *and all* l, *we have the following relational expressions:*

(a) $r^{k+1} = (I - S_k^T)r^k$;
(b) $p^{k+1} = (I - S_k)(I - S_k^T)r^k$;
(c) $(r^{k+1}, r^{k+1}) = (r^k, p^{k+1})$.

Proof We only give the proof of (a) and you can obtain (b) and (c), similarly.

$$r^{k+1} = r^k - AP_k\alpha^k = r^k - AP_k(P_k^{\mathrm{T}} AP_k)^{-1} P_k^{\mathrm{T}} r^k = (I - S_k^{\mathrm{T}})r^k$$

PROPOSITION 3.8 *If* p_l^k *in Algorithm* 2.2 *is nonzero for each* k *and all* l, *we have the following equality:*

(a) $P_k\alpha^{k^T} = -S_k e^k$;
(b) $P_k\beta^k = -S_k r^{k+1}$;
(c) $Q_k\alpha^k = S_k^T r^k$;
(d) $Q_k\beta^k = -AS_k r^{k+1}$;
(e) $(p^{k+1}, Q_k\alpha^k) = 0$.

Proof We only give the proof of (e). From Proposition 3.4, we have

$$0 = (p^{k+1}, q_j^k) = (p^{k+1}, Ap_j^k) = (Ap^{k+1}, p_j^k), \quad \text{for } j = 1, \ldots, L$$

Hence,

$$(p^{k+1}, Q_k\alpha^k) = (p^{k+1}, AP_k\alpha^k) = \left(Ap^{k+1}, \sum_{j=1}^{L} \alpha_j^k p_j^k \right) = \sum_{j=1}^{L} \alpha_j^k (Ap^{k+1}, p_j^k) = 0$$

It is easy to validate the similar propositions for the PMSD-CG method. For example, Proposition 3.9.

PROPOSITION 3.9 *Let $A, M \in \mathbb{R}^{n\times n}$ are SPD and $p_l^k \neq 0$ for each k and all l, then matrix $P_k^T A M^{-1} P_k$ is SPD.*

PROPOSITION 3.10 *Let $A, M \in \mathbb{R}^{n\times n}$ are SPD and $p_l^k \neq 0$ for each k and all l, then PMSD-CG method is well definite, i.e., break down cannot occur at each step.*

We can get some analogous formulas like (12)–(16) for the PMSD-CG methods and some other similar propositions with those of the MSD-CG method.

Furthermore, we can give the preconditioned version of the GIPF-CG method, *i.e.*, solve small systems (9) and (10) in the PMSD-CG method (Algorithm 2.3) with iterative method.

4 SUMMARY

In this article, we proposed a new CG-type method based on domain decomposition method and we called it MSD-CG method. At each step of the method, we choose a search direction in each subdomain. The search direction only needs to be nonzero in the associated subdomain but not necessarily to be conjugated to each other as required by the block CG method. We have given a preconditioned version of the MSD-CG method and shown the basic propositions of the MSD-CG and PMSD-CG methods.

The MSD-CG method has much in common with the CG method. One of the essence difference is that the calculation of global inner products of the CG method has been replaced by the solution of small systems of equations. The structure and property of these small systems are determined by that of the original systems (1) and the choice of multiple search directions. The MSD-CG method has the local minimal property and the energy normal of global error of the MSD-CG method is non-increasing. Global information is exchanged by solving the small systems. Therefore, one can expect that the convergent rate of the MSD-CG method is faster than that of some additive Schwarz methods, such as the BJ and the IBJ-CG (m) method. In Ref. [9], we give some of the numerical experiments. From Eq. (16), the MSD-CG method can be viewed as a more general nonstationary Richardson iterative method and this will be used to prove the convergence and estimate the convergent rate of the MSD-CG method in Ref. [9].

The solves of the small systems also constitute a synchronization point in a parallel environment if the small systems are dense, and they require exchange of information between all processor. However, the possibilities for avoiding global communication in Algorithm 2.2 are obvious. We can replace the exact solution of the small systems by an iterative method such as Jacobi method. If the coefficient matrix A of original system has a special structure, such as block tridiagonal, block five diagonal, block seven diagonal or block nine diagonal, then C_k is tridiagonal, five diagonal, seven diagonal or nine diagonal. At this time, the solution of small systems Eqs. (5) and (6) requires only communication between neighbouring processors and eliminates global inner products calculation. In order to distinguish this method from the MSD-CG method in which the small systems are solved accurately (direct or many iteration), it is called GIPF-CG method in brief. It is very suitable for large parallel computing and is a good alternative method for the CG method on massive parallel processors. In Ref. [9], we show that the convergent rate of the MSD-CG method is very similar to that of the CG method and is at least faster than that of steepest descent method. Of course, the number of inner iteration for the small systems should not be too large. From the numerical experiments of Ref. [9], one can see that it can get a reasonable balance among convergence, computing time and communication time when we take 2–4 inner Jacobi iterations.

Acknowledgements

The project is supported partly by State Hi-Tech Research and Development Program (863) and Natural Science Fund of China (60373015) and the Foundation of National Key Laboratory of Computational Physics.

References

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

[2] Basermann, A. (1997). Conjugate gradient and Lanczos methods for sparse matrices on distributed memory multiprocessors. *J. Parallel Distr. Comput.*, **45**, 46–52.

[3] Basermann, A., Reichel, B. and Schelthoff, C. (1997). Preconditioned CG methods for sparse matrices on massively parallel machines. *Parallel Comput.*, **23**, 381–398.

[4] Chronopoulos, A. T. and Gear, C. W. (1989). s-Step iterative methods for symmetric linear systems. *J. Comp. Appl. Math.*, **25**, 153–168.

[5] Crone, L. and van der Vorst, H. A. (1993). Communication aspects of the conjugate gradient method on distributed-memory machines. *Supercomputer*, **X**(6), 4–9.

[6] D'Azevedo, E. F. and Romine, C. (1993). LAPACK working note 56: reducing communication costs in the conjugate gradient algorithm on distributed memory multiprocessors. *Technical Reports*. Computer Science Department, University of Knoxville, Knoxville, TN.

[7] da Sturler, E. (1996). A performance model for Krylov subspace methods on mesh-based parallel computers. *Parallel Comput.*, **22**, 57–74.

[8] Demmel, J. W., Health, M. T. and van der Vorst, H. A. (1993). *Parallel Numerical Linear Algebra*. In Acta Numerica 1993. Cambridge University Press, Cambridge.

[9] Gu, T.-X., Liu, X.-P., Mo, Z.-Y. and Chi, X.-B. (in same issue). Multiple search direction conjugate gradient method II: theory and numerical experiments. *Int. J. Comput. Math.*

[10] Hesrenses, M. R. and Stiefel, E. (1952). Method of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, **49**, 409–436.

[11] Meurant, G. (1984). The block preconditioned conjugate gradient method on vector computers. *BIT*, **24**, 623–633.

[12] Meurant, G. (1987). Multitasking the conjugate gradient method on the CRAY X-MP/48. *Parallel Comput.*, **5**, 267–280.

[13] O'Leary, D. P. (1980). The block conjugate gradient algorithm and related methods. *Lin. Alg. Appl.*, **29**, 293–322.

[14] Reid, J. K. (1971). On the method of conjugate gradients for the solution of large sparse systems of linear equation. In J. K. Reid (Ed.) *Large Sparse Sets of Linear Equations*. Academic Press, pp. 231–254.

[15] Saad, Y. (1981). Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comput.*, **155**, 105–126.

[16] Saad, Y. (1989). Krylov subspace methods on supercomputers. *SIAM J. Sci. Comput.*, **10**, 1200–1232.

[17] Saad, Y. (1996). *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston.

[18] Tang, W. P. (1992). Generalized Schwarz splittings. *SIAM J. Sci. Stat. Comp.*, **13**, 573–595.