Journal of Computational and Applied Mathematics 226 (2009) 55–65

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An improved parallel hybrid bi-conjugate gradient method suitable for distributed parallel computing[☆]

Tong-Xiang Gu[a,*], Xian-Yu Zuo[b], Xing-Ping Liu[a], Pei-Lu Li[c]

[a] *Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, P.O.Box 8009, Beijing 100088, PR China*

[b] *Mathematics and Information Science College, Henan Normal University, Xinxiang 453007, PR China*

[c] *Department of Mathematics and Information Engineering, Puyang Vocational and Technical College, Puyang 457001, PR China*

ARTICLE INFO

Article history:
Received 3 July 2007
Received in revised form 15 December 2007

Keywords:
Sparse nonsymmetric linear system
IBiCGSTAB(2) method
Krylov subspace method
Distributed parallel environments
Global communication

ABSTRACT

An improved parallel hybrid bi-conjugate gradient method (IBiCGSTAB(2) method, in brief) for solving large sparse linear systems with nonsymmetric coefficient matrices is proposed for distributed parallel environments. The method reduces five global synchronization points to two by reconstructing the BiCGSTAB(2) method in [G.L.G. Sleijpen, H.A. van der Vorst, Hybrid bi-conjugate gradient methods for CFD problems, in: M. Hafez, K. Oshima (Eds.), Computational Fluid Dynamics Review 1995, John Wiley & Sons Ltd, Chichester, 1995, pp. 457–476] and the communication time required for the inner product can be efficiently overlapped with useful computation. The cost is only slightly increased computation time, which can be ignored, compared with the reduction of communication time. Performance and isoefficiency analysis shows that the IBiCGSTAB(2) method has better parallelism and scalability than the BiCGSTAB(2) method. Numerical experiments show that the scalability can be improved by a factor greater than 2.5 and the improvement in parallel communication performance approaches 60%.

1. Introduction

Among iterative methods for large sparse systems, Krylov subspace methods are the most powerful. The conjugate gradient (CG) method for solving symmetric positive definite linear systems, the GMRES method, BiCG method [9], QMR method [3], BiCGSTAB method [14], BiCR method [11] and BiCRSTAB method [7] for solving nonsymmetric linear systems and so on are all examples.

The basic time-consuming computational kernels of all Krylov subspace methods are usually [9]: inner products, vector updates and matrix-vector multiplications. In many situations, especially when matrix operations are well structured, these operations are suitable for implementation on vector and shared memory parallel computers. But for parallel distributed memory machines, the matrices and vectors are distributed over the processors, so that even when matrix operations can be implemented efficiently by parallel operation, we still cannot avoid the global communication required for inner product computations, i.e. accumulation of data from all the processors to one, and broadcasting the result to each processor. Vector updates are naturally parallel and, for large sparse matrices, matrix-vector multiplications can be implemented with communication only between nearby processors. The bottleneck is usually due to inner products enforcing global communication. These global communication costs become relatively more and more important when the number of

☆ The project is supported in part by National Natural Science Foundation of China(10571017), National Basic Research Program of China (2005CB221300) and Foundation of National Key Laboratory of Computational Physics.

* Corresponding author.
E-mail address: txgu@iapcm.ac.cn (T.-X. Gu).

0377-0427/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2008.05.017

parallel processors is increased and thus they have the potential to affect the scalability of the algorithm in a negative way. A detailed discussion on the communication problem on distributed memory systems can be found in [12,13].

Three remedies can be used to solve the bottleneck which leads to performance degeneration. The first remedy is to eliminate data relativity or reduce the number of global synchronization points, so that several inner products can be computed and passed at the same time. The second is reconstructing the algorithm so that communication and computation can be overlapped efficiently. The last is replacing the inner product computation by other computations which do not require global communications. Of course, they can be used concurrently.

Recently, Bücker et al. [1] and Yang et al. [15] proposed a new parallel Quasi-Minimal Residual(QMR) method based on a Lanczos process with coupled two-term recurrence. Sturler et al. [12] proposed how to reduce the effect of global communication in GMRES(m) and CG methods. Yang et al. [16–18] proposed the improved CGS, BiCG and BiCGSTAB methods respectively. Chi et al. gave an improved CR algorithm [2]. Gu, Zuo and Liu et al. [6–8] proposed parallel versions of BiCR, BiCRSTAB and QMRCGSTAB methods. All of these methods depend on the first two strategies. Gu, Liu and Mo [4,5] proposed a CG-type method without global inner products, i.e. multiple search direction conjugate gradient (MSD-CG) method. Based on domain decomposition, the MSD-CG method replaced the inner product computations in the CG method by small size linear systems. Therefore, it eliminates global inner products completely, which belongs to the last remedy.

In this paper, we give an improved parallel BiCGSTAB(2) method for distributed parallel environments based on the first two remedies mentioned above. The IBiCGSTAB(2) method is reorganized without changing numerical stability and all inner products per iteration are collected in two steps and independent (only one single global synchronization point), and subsequently communication time required for inner products can be overlapped efficiently with computation time. The cost is only slightly increased computation. Performance and isoefficiency analysis show that the IBiCGSTAB(2) method has better parallelism and scalability than the BiCGSTAB(2) method. Especially, the parallel performance can be improved by a factor greater than 2.5.

The paper is organized as follows. In Section 2, the design of the improved parallel BiCGSTAB(2) method is presented. Performance and isoefficiency analysis of the IBiCGSTAB(2) and BiCGSTAB(2) methods are presented in Sections 3 and 4. Numerical experiments carried out on a distributed memory parallel machine are reported in Section 5. Finally, we make some concluding remarks in Section 6.

2. Algorithm design of IBiCGSTAB(2) method

Consider solving a large sparse nonsymmetric linear system

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

on a parallel distributed memory machine, where $A \in R^{N\times N}$, $x, b \in R^N$.

Assume that the matrix and vectors are distributed according to row or domain decomposition to each processor of a distributed memory parallel processor, and have perfect load balance; the matrix is sparse and the matrix-vector multiplications can be implemented with communication only between nearby processors. The important bottleneck for Krylov subspace methods is usually due to inner products enforcing global communication.

For comparison, we give the BiCGSTAB(2) method for (1) discussed in [10], where x_0 and $r_0 = b - Ax_0$ is the initial guess and residual vector, respectively, such that $r_0^T r_0 \neq 0$.

Algorithm 1. The BiCGSTAB(2) Method, [10]

1) Compute $r_0 = b - Ax_0$, $\hat{r}_0 = r = r_0$, $p = v = w = 0$,
$\rho_1 = \alpha = \omega_1 = \omega_2 = 1$;
2) **For** $i = 0, 2, 4, 6, \ldots$, until convergence, **do**
3) $\rho_0 = -\omega_2\rho_1$;
even BiCG step

4) $\rho_1 = (r_i, \hat{r}_0)$; $\beta = \alpha\rho_1/\rho_0$; $\rho_0 = \rho_1$;
5) $p = r_i - \beta(p - \omega_1 v - \omega_2 w)$;
6) $v = Ap$;
7) $\gamma = (v, \hat{r}_0)$; $\alpha = \rho_0/\gamma$;
8) $r = r_i - \alpha v$;
9) $s = Ar$;
10) $x = x_i + \alpha p$;
odd BiCG step

11) $\rho_1 = (s, \hat{r}_0)$; $\beta = \alpha\rho_1/\rho_0$; $\rho_0 = \rho_1$;
12) $v = s - \beta v$;
13) $w = Av$;
14) $\gamma = (w, \hat{r}_0)$; $\alpha = \rho_0/\gamma$