Journal of Computational and Applied Mathematics 271 (2014) 285–294

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

An improved generalized conjugate residual squared algorithm suitable for distributed parallel computing☆

Xian-Yu Zuo [a,*], Li-Tao Zhang [b], Tong-Xiang Gu [c]

[a] Institute of Data and Knowledge Engineering, School of Computer and Information Engineering, Henan University, Kaifeng, Henan, 475004, PR China

[b] Department of Mathematics and Physics, Zhengzhou Institute of Aeronautical Industry Management, Zhengzhou, Henan, 450015, PR China

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

ARTICLE INFO

Article history:
Received 5 September 2013
Received in revised form 16 December 2013

MSC:
65F10
65F15

Keywords:
Sparse nonsymmetric linear systems
IGCRS algorithm
Krylov subspace methods
Global communication

ABSTRACT

In this paper, based on GCRS algorithm in Zhang and Zhao (2010) and the ideas in Gu et al. (2007), we present an improved generalized conjugate residual squared (IGCRS) algorithm that is designed for distributed parallel environments. The new improved algorithm reduces two global synchronization points to one by changing the computation sequence in the GCRS algorithm in such a way that all inner products per iteration are independent so that communication time required for inner products can be overlapped with useful computation. Theoretical analysis and numerical comparison of isoefficiency analysis show that the IGCRS method has better parallelism and scalability than the GCRS method, and the parallel performance can be improved by a factor of about 2. Finally, some numerical experiments clearly show that the IGCRS method can achieve better parallel performance with a higher scalability than the GCRS method and the improvement percentage of communication is up to 52.19% averagely, which meets our theoretical analysis.

1. Introduction

One of fundamental tasks of numerical computation is to solve linear systems. These systems arise frequently in scientific computing, for example, from finite difference or finite element discretization of partial differential equations, as intermediate steps in finding the solution of nonlinear problems or as subproblems in linear and nonlinear programming. Usually, these systems are large, sparse and nonsymmetric and solved by iterative methods [1].

Among the iterative methods for large sparse systems, the Krylov subspace methods are the most powerful. For example, conjugate gradient (CG) method for solving symmetric positive definite linear systems, the GMRES method, BiCG method [1], QMR method [2], BiCGStab method [3] and BiCR [4], CRS [5,6], GCRS [7] methods for solving nonsymmetric linear systems and so on. However, the Krylov subspace methods enforce bottleneck, i.e., the global communication induced by inner product computations, when used in large scale parallel computing.

☆ This research of this author is supported by the NSFC Tianyuan Mathematics Youth Fund (11226337), NSFC (61202098, 61203179, 61170309, 61033009, 91130024 and 11171039), Aeronautical Science Foundation of China (2013ZD55006), Project of Youth Backbone Teachers of Colleges and Universities in Henan Province (2013GGJS-142), Major project of development foundation of science and technology of CAEP (2012A0202008), Scientific and Technological Key Project of Education Department of Henan Province (12B110028, 13B430355), Basic and Advanced Technological Research Project of Henan Province (132300410373) and the School Youth Fund (2012113004).

* Corresponding author. Tel.: +86 18637866093.
E-mail addresses: xianyu_zuo@163.com (X.-Y. Zuo), litaozhang@163.com (L.-T. Zhang).

http://dx.doi.org/10.1016/j.cam.2014.04.009
0377-0427/

The basic time-consuming computational kernels of all the Krylov subspace methods are usually [1]: inner products, vector updates and matrix–vector multiplications. Vector updates are naturally parallel and, for large sparse matrices, matrix–vector multiplications can be implemented with communication between only 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 parallel processors is increased and thus they have the potential to affect the scalability of the algorithm in a negative way. The detailed discussion on the communication problem on distributed memory systems can be found in [8].

Three remedies can be used to solve the bottleneck leading to performance degeneration. The first is to eliminate data dependency, resulting several inner products can be computed and passed at the same time. The second is reconstructing algorithm, resulting communication and computation can be overlapped efficiently. The last is replacement of computation involving global communications by another computation without global communications. Of course, the three strategies can be applied in combination. The remedy, used in this paper, belongs to the first, i.e., reducing the global communication times or number of global synchronization points.

Bücker et al. [9] and Yang et al. [10] proposed a new parallel quasi-minimal residual (QMR) method based on the coupled two-term recurrences Lanczos process. Sturler et al. [8] showed how to reduce affection of the global communication in GMRES(m) and CG methods. Yang et al. [11–13] propose the improved CGS, BiCG and BiCGStab methods respectively. Liu et al. gave an improved CR algorithm [14]. Liu and Gu et al. [15] proposed the parallel QMRCGSTAB method. Zhang et al. [5,6] proposed the improved conjugate residual squared method. All of these methods are based on the first two strategies mentioned in the former paragraph. Gu, Liu and Mo [16] proposed a CG-type method without global inner products, i.e., multiple search direction conjugate gradient (MSD-CG) method. Based on domain discretion, MSD-CG method replaced the inner products computation in CG method by small size linear systems. Therefore, it eliminates global inner products completely, and belongs to the last remedy.

In this paper, based on GCRS algorithm for large sparse nonsymmetric linear systems in Zhang et al. [7], we present an improved generalized conjugate residual squared (IGCRS) algorithm, which is designed for distributed parallel environments. The improved GCRS (IGCRS) method is reorganized without changing the numerical stability and all inner products per iteration are 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 a little increased computation. Theoretical analysis and numerical comparison of isoefficiency analysis show that the IGCRS method has better parallelism and scalability than the GCRS method and the parallel performance can be improved by a factor of about 2.

This paper is organized as follows. First of all, we give the improved GCRS method. Then, theoretical analysis and isoefficiency analysis of two methods are presented in Sections 3 and 4. Numerical experiments are presented in Section 5. Finally, we make some concluding remarks in Section 6.

2. The improved GCRS algorithm

Consider solving a large sparse nonsymmetric linear system

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

on a parallel distributed memory machine, where $A \in R^{N\times N}$, $x, b \in R^N$. Denote x_0 any initial guess for the solution and $r_0 = b - Ax_0$ the initial residual. Then, Zhang et al. [7] proposed the generalized CRS algorithms by using products of two nearby BiCR polynomials and formal orthogonal polynomial. The GCRS and GCRS2 algorithms described in [7] are written as follows:

Algorithm 1 (Generalized Conjugate Residual Squared Method) (GCRS) (see [7])

1) Compute $r_0 = b - Ax_0$, r_0^* is arbitrary
   Set $p_0 = u_0 = t_0 = r_0$
2) For $n = 0, 1, \ldots$, until convergence. Do:
3) $\alpha_n = \frac{(r_n, A^T r_0^*)}{(Ap_n, A^T r_0^*)}$
4) choose $\tilde{\alpha}_n$
5) $s_n = t_n - \alpha_n A p_n$
6) $q_n = u_n - \tilde{\alpha}_n A p_n$
7) $\beta_n = -\frac{(As_n, A^T r_0^*)}{(Ap_n, A^T r_0^*)}$
8) choose $\tilde{\beta}_n$
9) $u_{n+1} = r_{n+1} + \beta_n q_n$
10) $t_{n+1} = r_{n+1} + \tilde{\beta}_n s_n$
11) $p_{n+1} = t_{n+1} + \beta_n(q_n + \tilde{\beta}_n p_n)$
12) $r_{n+1} = r_n - A(\alpha_n u_n + \tilde{\alpha}_n s_n)$
13) $x_{n+1} = x_n + \alpha_n u_n + \tilde{\alpha}_n s_n$
14) EndDo