The Improved BiCGStab Method for Large and Sparse Unsymmetric Linear Systems on Parallel Distributed Memory Architectures*

Laurence Tianruo Yang†‡, Richard P. Brent ‡
†Department of Computer Science, St. Francis Xavier University
P.O. Box 5000, Antigonish, B2G 2W5, NS, Canada
‡Computing Laboratory, Oxford University
Wolfson Building, Park Road, Oxford OX1 3QD, UK

Abstract

In this paper, an improved version of the BiCGStab (IBiCGStab) method for the solutions of large and sparse linear systems of equations with unsymmetric coefficient matrices is proposed. The method combines elements of numerical stability and parallel algorithm design without increasing the computational costs. The algorithm is derived such that all inner products of a single iteration step are independent and communication time required for inner product can be overlapped efficiently with computation time of vector updates. Therefore, the cost of global communication which represents the bottleneck of the parallel performance can be significantly reduced. The resulting IBiCGStab algorithm maintains the favorable properties of the original method while not increasing computational costs. Data distribution suitable for both irregularly and regularly structured matrices based on the analysis of the nonzero matrix elements is presented. Communication scheme is supported by overlapping execution of computation and communication to reduce waiting times. The efficiency of this method is demonstrated by numerical experimental results carried out on a massively parallel distributed memory system.

1 Introduction

One of the fundamental task of numerical computing is the ability to solve linear systems. These systems arise very frequently in scientific and engineering computing, for example from finite difference or finite element approximations to partial differential equations, as intermediate steps in computing the solution of nonlinear problems or as subproblems in linear and nonlinear programming.

For linear systems of small size, the standard approach is to use direct methods, such as LU decomposition which obtains the solution through a factorization of the coefficient matrix. In contrast with direct methods, iterative methods use successive approximations to obtain more accurate solutions to the linear systems at subsequent steps. Iterative methods are computationally more attractive than the direct methods particularly for large and sparse systems [9].

A powerful iterative method for the solution of large and sparse linear systems with unsymmetric positive definite coefficient matrices is the family of Krylov subspace methods [8, 11] involving the coefficient matrix only in the form of matrix-by-vector products. These methods basically consist of the generation of a suitable basis of a vector space called Krylov subspace and the choice of the actual iterate within that space. In this paper, We will be investigating the BiCGStab method [13], a fast and smoothly converging variant of BiConjugate Gradient method (BiCG) [8, 11] for the solutions of large and sparse linear systems with unsymmetric coefficient matrices.

The basic time-consuming computational kernels of BiCGStab are usually: inner products, vector updates, 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 [7]. But for parallel distributed memory machines, the matrices and vectors are distributed over the processors, so that even when the matrix operations can be implemented efficiently by parallel operations, we still can not avoid the global communication, i.e. accumulation of data from all to one processor, required for inner product computations. Vector updates are perfectly parallelizable 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. The detailed discussions on the communication

*The author's email is lyang@stfx.ca

problem on distributed memory systems can be found in [5, 6, 12, 16]. 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 [5, 6, 16].

How to design an efficient parallel BiCGStab method maintaining the favorable numerical properties while not increasing computational costs on the parallel distributed memory architectures is the main concern of the paper. Recently Jacques et al. [10] propose a new modified parallel version of the BiCGStab method (referred to the MBiCGStab method). They apply the new developed solver with preconditioner to linear systems from electromagnetic scattering modelling. The algorithm is reorganized without changing the numerical stability so that there is only two single global synchronization points per iteration reduced from the original three. Motivated by the same ideas, we would like to propose the new Improved BiCGStab method. The algorithm is reorganized without changing the numerical stability so that all inner products of a single iteration step are independent (only one single global synchronization point), and subsequently communication time required for inner product can be overlapped efficiently with computation time of vector updates. Therefore, the cost of global communication on parallel distributed memory computers can be significantly reduced. The resulting IBiCGStab algorithm maintains the favorable properties while not increasing computational costs. The efficient parallel implementation details, in particularly, data distribution and communication scheme, will be addressed as well. The efficiency of this method is demonstrated by numerical experimental results carried out on a massively parallel distributed memory computer.

The paper is organized as follows. In section 2, we will describe briefly the original and the new improved variant. The parallel implementation details including data distribution and communication scheme are presented in section 3. Finally numerical experiments carried out are reported and some comparisons are given with regards to numerical accuracy, parallel performance and efficiency.

2 The BiCGStab and IBiCGStab Methods

The BiCGStab method [13] for the solution of the linear equations

$$Ax = b, \quad \text{where} \quad A \in \Re^{n\times n} \quad x, b \in \Re^n. \tag{1}$$

is described in the Algorithm 1. Here x_0 is any initial guess for the solution and $r_0 = b - Ax_0$ is the initial residual such that $r_0^T r_0 \neq 0$.

Algorithm 1 The Classical BiCGStab Method

1: $r_0 = b - Ax_0$;
2: $\rho_0 = \alpha_0 = \omega_0 = 1$;
3: $v_0 = p_0 = 0$;
4: **for** n = 1, 2, 3, ... **do**
5: $\rho_n = r_0^T r_{n-1}$;
6: $\beta = \frac{\rho_n}{\rho_{n-1}} \frac{\alpha_n}{\omega_{n-1}}$;
7: $p_n = r_{n-1} + \beta_n(p_{n-1} - \omega_{n-1} v_{n-1})$;
8: $v_n = Ap_n$;
9: $\alpha_n = \frac{\rho_n}{r_0^T v_n}$;
10: $s_n = r_{n-1} - \alpha_n v_n$;
11: $t_n = As_n$;
12: $\omega_n = \frac{t_n^T s_n}{t_n^T t_n}$;
13: $x_n = x_{n-1} + \alpha_n p_n + \omega_n s_n$;
14: **if** x_n is accurate enough **then**
15: STOP
16: **end if**
17: $r_n = s_n - \omega_n t_n$;
18: **end for**

Recently Jacques et al. [10] propose a new modified parallel version of the BiCGStab method (we refer to the MBiCGStab method). They apply the new developed solver with preconditioner to linear systems from electromagnetic scattering modelling. The algorithm is reorganized without changing the numerical stability so that there is only two single global synchronization points per iteration reduced from the original three. Motivated by the same ideas, we would like to propose the new Improved BiCGStab method. The basic idea in reformulating the BiCGStab algorithm is to merge the main computational kernels such as vector updates, matrix-vector multiplications and inner products as much as possible so that they can be executed in parallel per iteration of the algorithm. More importantly, there is only one single global synchronization points per iteration.

Based on the classical BiCGStab method, if we denote the vectors $q_n = Av_n$, $u_n = Ar_n$ and $\sigma_n = r_0^T u_n, \pi_n = r_0^T q_n$, we can get the following forms for vectors v_n and t_n:

$$\begin{aligned} v_n &= Ap_n = A(r_{n-1} + \beta_n(p_{n-1} - \omega_{n-1} v_{n-1})) \\ &= u_{n-1} + \beta_n v_{n-1} - \beta_n \omega_{n-1} q_{n-1}, \end{aligned} \tag{2}$$

and

$$t_n = As_n = A(r_{n-1} - \alpha_n v_n) = u_{n-1} - \alpha_n q_n. \tag{3}$$

If we define the following inner products $\phi_n = r_0^T s_n$ and $\tau_n = r_0^T v_n$, then the inner product $r_0^T t_n$ can be

expressed as follows:

$$r_0^T t_n = r_0^T(u_{n-1} - \alpha_n q_n) = \sigma_{n-1} - \alpha_n \pi_n. \quad (4)$$

If we substitute v_n derived in (2), we have

$$\begin{aligned} \tau_n &= r_0^T v_n = r_0^T(u_{n-1} + \beta_n v_{n-1} - \beta_n \omega_{n-1} q_{n-1}) \\ &= r_0^T u_{n-1} + \beta_n r_0^T v_{n-1} - \beta_n \omega_{n-1} r_0^T q_{n-1} \\ &= \sigma_{n-1} + \beta_n \tau_{n-1} - \beta_n \omega_{n-1} \pi_{n-1}. \end{aligned} \quad (5)$$

Also by combining (4), we can get the formula for computing ρ_n based on the following relation:

$$\begin{aligned} \rho_n &= r_0^T r_{n-1} = r_0^T(s_{n-1} - \omega_{n-1} t_{n-1}) \\ &= r_0^T s_{n-1} - \omega_{n-1} r_0^T t_{n-1} \\ &= \phi_{n-1} - \omega_{n-1}(\sigma_{n-2} - \alpha_{n-1}\pi_{n-1}). \end{aligned} \quad (6)$$

Since we need the value of p_n, which has been eliminated, to update x_n, we have to find a update formula for $\alpha_n p_n$ denoted as z_n. The expression can be described as follows:

$$\begin{aligned} z_n &= \alpha_n p_n = \alpha_n r_{n-1} + \beta_n \alpha_n p_{n-1} - \beta_n \alpha_n \omega_{n-1} v_{n-1} \\ &= \alpha_n r_{n-1} + \beta_n z_{n-1} - \alpha_n \beta_n \omega_{n-1} v_{n-1}. \end{aligned} \quad (7)$$

If we define $f_0 = A^T r_0$, $\gamma_n = f_0^T s_n$ and $\eta_n = f_0^T t_n$, then σ_n can be expressed as follows:

$$\begin{aligned} \sigma_n &= r_0^T u_n = r_0^T(Ar_n) = r_0^T(As_n - \omega_n A t_n) \\ &= r_0^T A s_n - \omega_n r_0^T A t_n = s_n^T(A^T r_0) - \omega_n t_n^T(A^T r_0) \\ &= \gamma_n - \omega_n \eta_n. \end{aligned} \quad (8)$$

In order to make the expressions as simple as possible, we use an extra value δ_i as $\beta_i \omega_{n-1}$.

Based on the equations (2), (3), (5), (6), (7), (8) and complicated mathematical derivations, the IBiCGStab algorithm can be depicted in the Algorithm 2.

Under the assumptions, the Improved BiCGStab (IBiCGStab) method can be efficiently parallelized as follows:

- The inner products of a single iteration step (13), (14), (15), (16), (17) and (18) are independent.
- The vector updates (10) and (11) are independent.
- The vector updates (21) and (22) are independent.
- The communications required for the inner products (13), (14), (15), (16), (17) and (18) can be overlapped with the update for z_n in (12).

Also note that in the algorithm, compared with the BiCGStab and MBiCGStab methods, we only use several more vector updates (but they can be done in parallel), which do not require any communication at all and are perfectly parallelizable. But the global synchronization points per iteration has been reduced from 4 in the classical BiCGStab method and 2 in the MBiCGStab method to 1 in the proposed IBiCGStab method. Therefore, the cost of communication time on parallel distributed memory computers can be significantly reduced.

Algorithm 2 The IBiCGStab Method

1: $r_0 = b - Ax_0, u_0 = Ar_0, f_0 = A^T r_0, q_0 = v_0 = z_0 = 0$;
2: $\sigma_{-1} = \pi_0 = \phi_0 = \tau_0 = 0, \sigma_0 = r_0^T u_0, \rho_0 = \alpha_0 = \omega_0 = 1$;
3: **for** n = 1, 2, 3, ... **do**
4: $\rho_n = \phi_{n-1} - \omega_{n-1}\sigma_{n-2} + \omega_{n-1}\alpha_{n-1}\pi_{n-1}$;
5: $\delta_n = \frac{\rho_n}{\rho_{n-1}}\alpha_{n-1}, \beta_n = \frac{\delta_n}{\omega_{n-1}}$;
6: $\tau_n = \sigma_{n-1} + \beta_n \tau_{n-1} - \delta_n \pi_{n-1}$;
7: $\alpha_n = \frac{\rho_n}{\tau_n}$;
8: $v_n = u_{n-1} + \beta_n v_{n-1} - \delta_n q_{n-1}$;
9: $q_n = Av_n$;
10: $s_n = r_{n-1} - \alpha_n v_n$;
11: $t_n = u_{n-1} - \alpha_n q_n$;
12: $z_n = \alpha_n r_{n-1} + \beta_n z_{n-1} - \alpha_n \delta_n v_{n-1}$;
13: $\phi_n = r_0^T s_n$;
14: $\pi_n = r_0^T q_n$;
15: $\gamma_n = f_0^T s_n$;
16: $\eta_n = f_0^T t_n$;
17: $\theta_n = s_n^T t_n$;
18: $\kappa_n = t_n^T t_n$;
19: $\omega_n = \frac{\theta_n}{\kappa_n}$;
20: $\sigma_n = \gamma_n - \omega_n \eta_n$;
21: $r_n = s_n - \omega_n t_n$;
22: $x_n = x_{n-1} + z_n + \omega_n s_n$;
23: **if** x_n is accurate enough **then**
24: STOP
25: **end if**
26: $u_n = Ar_n$;
27: **end for**

3 Parallel Implementation

For large and sparse matrices, if you are working on different computer system architectures or dealing with different algorithms or data, the efficient storage schemes should be considered differently. In this paper, we decide to use one of the most common format called CRS format (compressed row storage). The main reason behind is that this type of storage scheme is very suitable for both regularly and irregularly structured large and sparse matrices. The detailed description can be found in the literature. Briefly speaking, the nonzeros of large and sparse matrix are stored in row-wise in three one-dimensional arrays. The values of the nonzeros are contained in array *value*. The corresponding column indices are contained in array *col_ind*. The elements of *row_ptr* point to the position of the beginning

of each row in *value* and *col_ind*.

In order to efficiently parallelize the IBiCGStab algorithm, in particularly, on a distributed memory architecture, we first need to decide the data distribution of matrix and vector arrays, hopefully optimally, to each processor and then determine an efficient communication scheme by taking into account different sparsity patterns, not only for matrix-vector multiplication but also for inner products, to minimize the overall execution time. In this paper, we will mainly follow the approach has been used in [14, 15] originally proposed in [1] for data distribution and communication scheme which do not require any knowledge about the matrix sparsity pattern. Also the communication scheme are automatically determined by the analysis of the indices of the non-zero matrix elements.

4 Numerical Experiments

In this section, the parallel variant of the improved BiCGStab (IBiCGStab) is compared with the modified version of BiCGStab (MBiCGStab) proposed by in [10] and the original BiCGStab on a massively distributed memory computer.

Here we mainly consider the partial differential equation taken from [2, 3, 4]

$$Lu = f, \qquad \text{on} \qquad \Omega = (0,1) \times (0,1),$$

with Dirichlet boundary condition $u = 0$ where

$$Lu = -\Delta u - 20(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y}),$$

and the right-hand side f is chosen so that the solution is

$$u(x,y) = \frac{1}{2}\sin(4\pi x)\sin(6\pi y).$$

Basically, we discretize the above differential equation using second order centered differences on a 400×400 with mesh size $1/441$, leading to a system of 193600 linear equations with a unsymmetric coefficient matrix of 966240 nonzero entries. Diagonal preconditioning is used. For our numerical tests, we choose $x_0 = 0$ as initial guess and $tol = 10^{-5}$ as stopping parameter.

Since the vectors are distributed over the processor grid, the inner products usually are computed in two steps. All processors start to compute in parallel the local inner products. After that, the local inner products are accumulated on one central processor and broadcasted. The communication time of an accumulation or a broadcast increases proportionally with the diameter of the processor grid. That means if the number of processors increases then the communication time for the inner products increases as well, and hence this is a potential threat to the scalability of the algorithm.

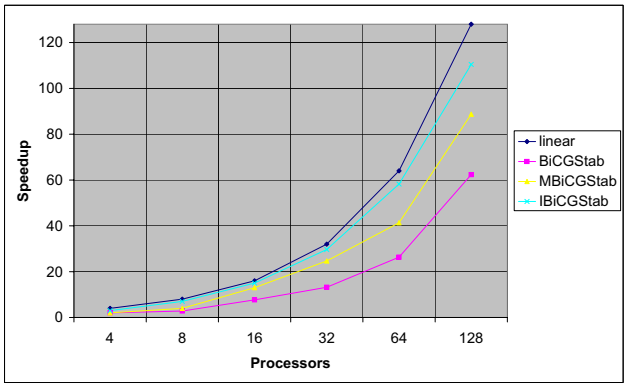

Figure 1. Experimental results of speed-up

The convergence of the proposed improved BiCGStab (IBiCGStab) is almost the same as the modified BiCGStab (MBiCGStab) suggested and original BiCGStab version where $\|r_n\|_2$ is computed recursively. A similar numerical behavior to these variants is observed. There is hardly any difference with respect to the true residual norm $\|b - Ax_n\|_2$ in those versions. The parallel performance are given in Figure 1 where linear is the theoretical linear speedup, IBiCGStab is the speedup of the improved BiCGStab method, MBiCGStab is the speedup of the modified BiCGStab method suggested in [10] and BiCGStab is the speedup of the original BiCGStab method. These results are based on timing measurements of a fixed number of iterations. Since we do not know exactly the implementation details of the modified BiCGStab method [10], we simply use the same implementation techniques as ours for the IBiCGStab method. We also make the preconditioner described in their approach as the identity matrix because we only want to compare the performance without any preconditioning technique acceleration. The speedup is computed as the ratio of the parallel execution time and the execution time using one processor. From the results, we can see clearly that the modified BiCGStab (MBiCGStab) suggested in [10] is faster than the original one. Meanwhile the new approach can achieve much better parallel performance with a higher scalability than the modified one. In comparison to the two other approaches, the reduction in execution time by the IBiCGStab increases with the number of processors. More precisely, the quantity is $1 - T_A(p)/T_B(p)$, where $T_A(p)$ and $T_B(p)$ are the execution times on p processors of approach A and B respectively. In Figure 2, first curve shows the percentage of reduction in execution time by our improved BiCGStab (IBiCGStab)

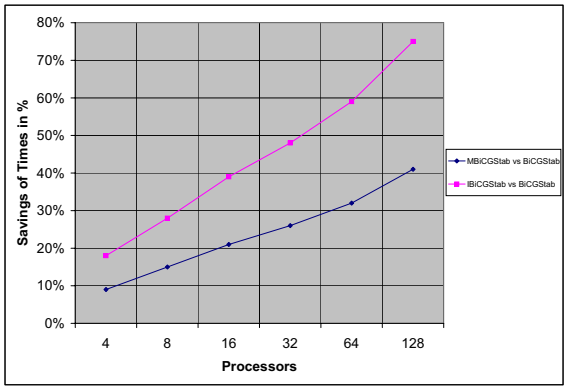

Figure 2. Execution time reduction

approach compared to the original one. Another curve shows the percentage of reduction of time for the modified BiCGStab (MBiCGStab) proposed in [10] compared to the original BiCGStab method.

5 Conclusions

In this paper, an improved version of the BiCGStab (IBiCGStab) method for the solutions of large and sparse linear systems of equations with unsymmetric coefficient matrices is proposed. The method combines elements of numerical stability and parallel algorithm design without increasing the computational costs. The algorithm is derived such that all inner products of a single iteration step are independent and communication time required for inner product can be overlapped efficiently with computation time of vector updates. Therefore, the cost of global communication which represents the bottleneck of the parallel performance can be significantly reduced. The resulting IBiCGStab algorithm maintains the favorable properties of the original method while not increasing computational costs. Data distribution suitable for both irregularly and regularly structured matrices based on the analysis of the non-zero matrix elements is presented. Communication scheme is supported by overlapping execution of computation and communication to reduce waiting times. The efficiency of this method is demonstrated by numerical experimental results carried out on a massively parallel distributed memory system.

References

1. A. Basermann, B. Reichel, and C. Schelthoff. Preconditioned CG methods for sparse matrices on massively parallel machines. *Parallel Computing*, 23(3):381–398, 1997.
2. H. M. Bücker and M. Sauren. A parallel version of the quasi-minimal residual method based on coupled two-term recurrences. In J. Waśniewski, J. Dongarra, K. Madsen, and D. Olesen, editors, *Proceedings of Workshop on Applied Parallel Computing in Industrial Problems and Optimization (Para96), LNCS184*, Lecture Notes in Computer Science, pages 157–165. Technical University of Denmark, Lyngby, Denmark, Springer-Verlag, August 1996.
3. H. M. Bücker and M. Sauren. A variant of the biconjugate gradient method suitable for massively parallel computing. In G. Bilardi, A. Ferreira, R. Lüling, and J. Rolim, editors, *Solving Irregularly Structured Problems in Parallel, Proceedings of the Fourth International Symposium, IRREGULAR'97, Paderborn, Germany, June 12–13, 1997*, volume 1253 of *Lecture Notes in Computer Science*, pages 72–79, Berlin, 1997. Springer.
4. H. M. Bücker and M. Sauren. Parallel biconjugate gradient methods for linear systems. In L. T. Yang, editor, *Parallel Numerical Computations with Applications*, number 51-70. Kluwer Academic Publishers, 1999.
5. E. de Sturler. A parallel variant of the GMRES(m). In *Proceedings of the 13th IMACS World Congress on Computational and Applied Mathematics*. IMACS, Criterion Press, 1991.
6. E. de Sturler and H. A. van der Vorst. Reducing the effect of the global communication in GMRES(m) and CG on parallel distributed memory computers. Technical Report 832, Mathematical Institute, University of Utrecht, Utrecht, The Netherland, 1994.
7. J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers*. SIAM, Philadelphia, PA, 1991.
8. R. Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Numerical Analysis Dundee 1975*, volume 506 of *Lecture Notes in Mathematics*, pages 73–89, Berlin, 1976. Springer.
9. R. W. Freund, G. H. Golub, and N. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, pages 57–100, 1991.
10. T. Jacques, L. Nicolas, and C. Vollaire. Electromagnetic scattering with the boundary integral method on MIMD systems. In L. T. Yang, editor, *Parallel Numerical Computations with Applications*, number 215-228. Kluwer Academic Publishers, 1999.
11. C. Lanczos. Solutions of systems of linear equations by minimized iterations. *Journal of Research of National Bureau of Standards*, 49(1):33–53, 1952.
12. C. Pommerell. *Solution of large unsymmetric systems of linear equations*. PhD thesis, ETH, 1992.
13. H. A. van der Vorst. Bi-CGStab: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13:631–644, 1992.
14. L. T. Yang and R. P. Brent. The improved conjugate gradient squared (ICGS) method on parallel distributed memory architectures. In *Workshop Proceedings of the 2001 International Conference on Parallel Processing (ICPP-HPSECA01)*, pages 161–165, Valencia, Spain, September 3-7, 2001.
15. L. T. Yang and R. P. Brent. The improved biconjugate gradient method on parallel distributed memory architectures. In *Workshop Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS-PDSECA02)*, pages 233–240, Fort Lauderdale, Florida, April 15-19, 2002.
16. L. T. Yang and R. P. Brent. Quantitative performance analysis of the improved quasi-minimal residual method on massively distributed memory computers. *Advances in Engineering Software*, 33:169–177, 2002.