Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2012) 213–218
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

The reordered BiCGStab method for distributed memory computer systems

Boris Krasnopolsky[1,*]

Institute of Mechanics, Lomonosov Moscow State University,
119192, Michurinsky pr. 1, Moscow, Russian Federation

Abstract

A new reordered formulation of the preconditioned BiCGStab iterative method for the system of linear equations with large sparse nonsymmetric matrix is presented. The algorithm is reformulated in order to improve the efficiency on distributed memory computer systems. It allows to avoid all global synchronization points of the inner product operations. The order of computations permits to overlap the communication time of the inner products by the preconditioning computations. The efficiency of the implemented method with the algebraic multigrid preconditioner is demonstrated by the scalability results for the MPI and the hybrid (MPI+SHM) programming models.

Keywords: Krylov subspace iterative methods, BiCGStab, preconditioning, distributed memory computer systems
2000 MSC: 65F10, 65N22

1. Introduction

The solution of systems of linear algebraic equations is one of the prevalent computational problems. The majority of scientific and engineering problems related to the modeling of physical processes is reduced to the solving of systems of equations with large sparse nonsymmetric matrices. The choice of the optimal method depends on the type of the governing equation. The elliptic equations are among of the most complicated. For example, the main difficulties in the computational fluid dynamics modelling arises at solution of the Poisson pressure equation, and takes an overwhelming part of the total computational time (usually more than 90%). The fast direct methods for the elliptic equations were developed (e.g., [1]) for simple computational domains with orthogonal grids and geometries, that may be mapped by coordinate transformations to rectangular areas. However, these methods appear to be inapplicable for any complex computational domains or non-orthogonal grids. Nowadays the Krylov subspace iterative methods, e.g., BiCGStab [2] or GMRES [3], are widely used for arbitrary computational domains. Incomplete LU factorization [4] with various fill-in levels for convergence acceleration of these methods had been used for a long time. However, incomplete LU factorization has two disadvantages: (i) preconditioning quality degradation with increasing the matrix size and (ii) high sensitivity to the matrix reordering. The latter is crucial for parallel computations, since the effective

* *Email address:* krasnopolsky@imec.msu.ru (Boris Krasnopolsky)
[1] Corresponding author

1877-0509
doi:10.1016/j.procs.2010.04.024

1. x_0 = initial guess; $r_0 = b - Ax_0$
2. $\rho_0 = (r_0, r_0)$
3. $p_0 = r_0$
   for $j = 0, 1, \ldots$
4. $\quad v = Ap_j$
5. $\quad \alpha_j = \frac{\rho_j}{(v, r_0)}$
6. $\quad s_j = r_j - \alpha_j v$
7. $\quad t = As_j$
8. $\quad \omega_j = \frac{(t, s_j)}{(t, t)}$
9. $\quad x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j$; check for convergence
10. $\quad r_{j+1} = s_j - \omega_j t$
11. $\quad \rho_{j+1} = (r_{j+1}, r_0)$
12. $\quad \beta_j = \frac{\rho_{j+1}}{\rho_j} \frac{\alpha_j}{\omega_j}$
13. $\quad p_{j+1} = r_{j+1} + \beta_j (p_j - \omega_j v)$

    end

Figure 1: The classical BiCGStab method.

matrix distribution over the computational processes also requires matrix reordering. By these reasons, multigrid preconditioners [5], whose quality is essentially independent on matrix reordering, become popular for large sparse matrices.

2. The classical BiCGStab and its parallel modifications

The classical BiCGStab method is presented in fig. 1 (in the preconditioned BiCGStab each matrix-vector multiplication $y = Ax$ is substituted by the set of operations: $\hat{y} = K^{-1}x$, $y = A\hat{y}$, where K is the preconditioning matrix). Each iteration of the classical method includes two matrix-vector multiplications (and two preconditioning operations for the preconditioned one), four inner products and twelve vector updates. The traditional scheme of the block row-wise matrix distribution allows to perform efficiently vector updates and matrix-vector multiplications: the vector updates can be done locally while the time for matrix-vector multiplication communications can be overlapped by the computations. The main difficulties arise when calculating the inner products: after computation of the local inner products the global communications are needed. These communications can be implemented as a collective operation or a set of point-to-point operations. The latter may be preferable if the inner products can be reordered in such a way that point-to-point communications are overlapped by some other computations. Otherwise, the inner product operation become a global synchronization point for all computational processes, that is undesirable.

Inspecting the classical BiCGStab algorithm (fig. 1) one can see that the order of computations in this method is strongly sequential, which means that calculations of all previous i-1 rows must be finished to calculate the i-th row of this algorithm. The existing inner products cannot be rearranged and time for communications be overlapped by the computations. These inner products lead to three synchronization points per iteration and as a result, to low scalability of the given algorithm.

To the best of our knowledge there are some publications [6, 7] attempting to reformulate the classical algorithm in order to decrease the total number of the inner products demanding the global synchronization of computational processes. The Modified BiCGStab method with two synchronization points was proposed in [6], and later the Improved BiCGStab method with only one global synchronization point was suggested in [7]. However, no one was able to propose a variant with no global synchronization of computational processes. In this work, based on the approach, discussed in [6, 7], a new modification of the BiCGStab method with no global synchronization points was formulated. In contrast to the works referred above, in the present work the preconditioned variant of the BiCGStab method is considered, which gives an extra “freedom” in modification of the algorithm.

1. x_0 = initial guess; $r_0 = b - Ax_0$
2. $\rho_0 = (r_0, r_0)$
3. $z = K^{-1} r_0$
4. $\hat{v} = z$

   for $j = 0, 1, \ldots$
5. $v = A\hat{v}$
6. $\delta_j = (v, r_0)$
7. $s = K^{-1} v$
8. $\alpha_j = \frac{\rho_j}{\delta_j}$
9. $\hat{t} = z - \alpha_j s$
10. $t = A\hat{t}$
11. $x_{j+1} = x_j + \alpha_j \hat{v}$; check for convergence
12. $r_{j+1} = r_j - \alpha_j v$
13. $\theta_j = (t, r_{j+1}); \phi_j = (t, t); \psi_j = (t, r_0)$
14. $z = K^{-1} t$
15. $\omega_j = \frac{\theta_j}{\phi_j}$
16. $x_{j+1} = x_{j+1} + \omega_j \hat{t}$; check for convergence
17. $r_{j+1} = r_{j+1} - \omega_j t$
18. $\rho_{j+1} = -\omega_j \psi_j$
19. $z = \hat{t} - \omega_j z$
20. $\beta_j = \frac{\rho_{j+1}}{\rho_j} \frac{\alpha_j}{\omega_j}$
21. $\hat{v} = z + \beta_j (\hat{v} - \omega_j s)$

    end

Figure 2: The reordered preconditioned BiCGStab method.

3. The reordered BiCGStab method

The main idea of the method reordering is to split the preconditioning and the matrix-vector multiplication operations in order to overlap the inner products MPI-communications with the preconditioning operation computations. The proposed reordering is presented in fig. 2.

The reordered method has only four additional vector updates, while the IBiCGStab method has six additional vector updates and two inner products in comparison with the original algorithm. The convergence of the proposed formulation is identical to the convergence of the initial variant of the preconditioned BiCGStab method, since the modifications are only in the order of computations. The results of the computations after each iteration are equal up to machine accuracy to the original BiCGStab method computations results. The distinctive feature of this method is the absence of the global synchronization points. The inner products are grouped in the three blocks (rows 2, 6 and 13 in fig. 2). In each block, there are some calculations to be made before the results of these inner products are used. Thus, the results of the inner products in rows 2 and 6 are used only in row 8, while the results of row 13 are used in row 15. In all three cases, there is an ability to make the preconditioning calculations between the rows of inner products initialization and rows these results are used in. Taking into account that the preconditioning is usually the most computationally expensive operation, the inner products communications can be efficiently overlapped by these calculations. As a result, the efficiency of this modification is improved due to the absence of all global synchronization points.

4. Parallel implementation and testing

A hybrid parallel program with algebraic multigrid as a preconditioner was developed to test the reordered formulation. The inter-node communications were implemented via the MPI while the POSIX shared memory was used for

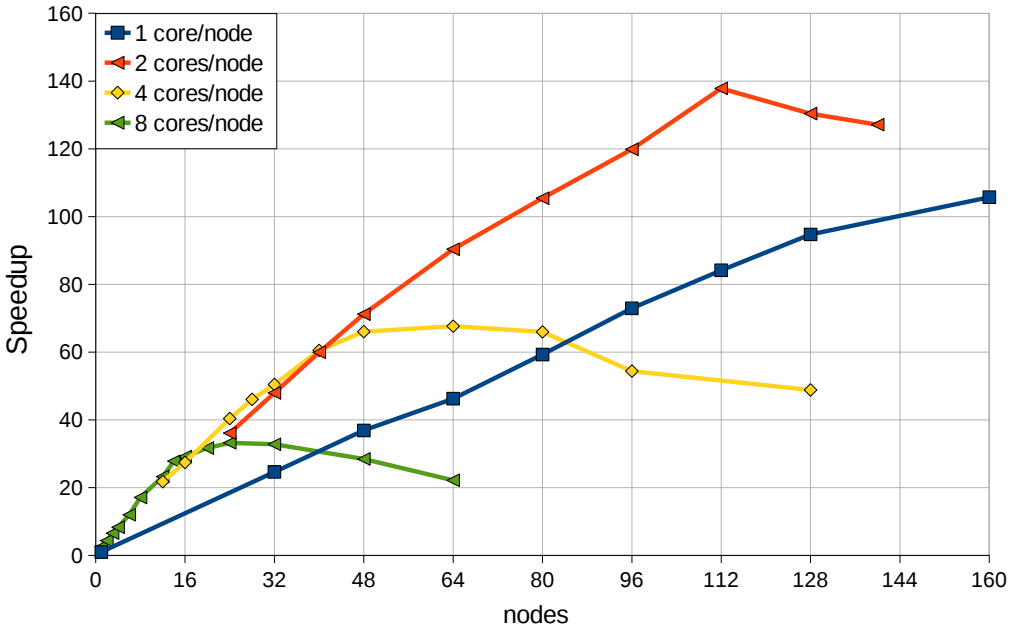

Figure 3: Scalability of the MPI-based reordered BiCGStab method implementation.

intra-node communications. The Hypre library [8] was used at the preconditioning setup phase to build the restriction-prolongation matrices and system matrices at different grid levels. The operational matrices were stored in CRS and DMSR formats and the ParMETIS library [9] was used to build the optimal initial matrix decomposition between the computational processes. The test matrices were chosen from the following CFD problems:

1. the discrete Poisson equation in a three-dimensional cubic cavity with a regular grid (7-point stencil, 8 million unknowns);
2. the discrete Poisson pressure equation in the computational domain of three-dimensional channel with the cubic obstacle at one of the walls (7-point stencil, 5.45 million unknowns). Grid structure for this test is equivalent to the grid used in direct numerical simulation of the incompressible turbulent flow over the cube [10].

As a criteria of convergence was used a condition

$$\frac{\|x_{j+1} - x_j\|}{\|x_j\|} < 10^{-7}.$$

At first, a parallel program with only MPI-communication interface over all computational processes was developed. The scalability of the code on the matrix related to the cubic computational domain (first test matrix), is presented[2] in fig. 3. The number of cores in the legend of this figure means the number of MPI-processes, allocated at each computational node.

According to the presented results, the top of the speedup occurs with about 200-250 computational processes that is equal to the local matrix of about 30-35 thousand rows for each process. The peak speedup of about 140 times is achieved on 112 nodes with utilization of only two cores per node (224 cores). Using of eight or four cores per node gives a slightly better results only for a few computational nodes: less than 16 and 40 respectively. Further

[2]These results and results, presented in fig. 4, are based on time measurements of a fixed number of iterations to avoid the fluctuations, caused by the small differences in the convergence process.

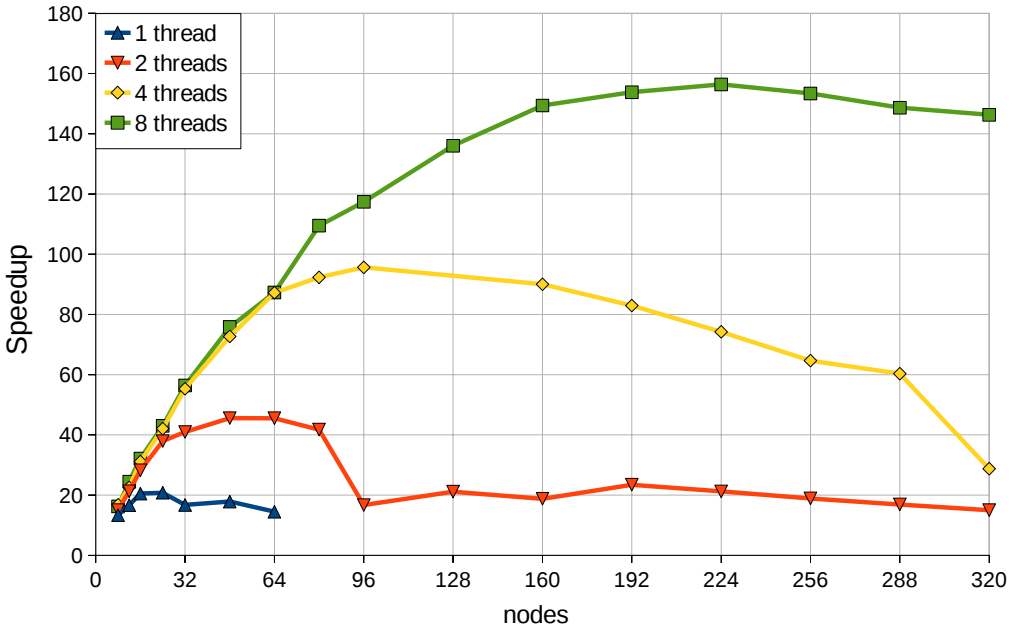

Figure 4: Scalability of the hybrid reordered BiCGStab method implementation.

increasing the number of computer nodes, causes the degradation of the scalability. The observed results for multiple MPI-processes per nodes can be explained by the essential growth of the interconnect network load. By these reasons, a hybrid programming model was implemented with only one MPI-communicating process per node and with intra-node communications over the processes inside the node via the POSIX shared memory. This code was tested using the second test matrix (matrix for DNS of the turbulent flow in the channel with the cubic obstacle). Corresponding results are presented in fig. 4. The number of threads in the legend of the figure means the number of processes, united via shared memory intra-communications. All available eight cores were utilized from the nodes in these runs, e.g., the line “4 threads” means that two blocks of four processes, united via the shared memory, were allocated at the nodes. The test regime of using the one thread per node is equal to the “8 cores/node” regime in the previous test case.

The scalability for this test matrix using of only the one thread become even lower in comparison with obtained for the first test matrix. The peak speedup for the one thread regime was obtained with about the same local matrix size of 28 thousand rows. However, the presented results shows all advantages of using the hybrid programming model approach. This model permits one to get the positive effect of using all available cores from the nodes. The peak speedup of 156 times was obtained at 224 computational nodes and 1792 cores with “8 threads” per node: 120.1 sec. for sequential run and 0.77 sec. for this run. The local matrix size for each process decreased to less than three thousand rows.

The results of the convergence and time measurements for the different matrix sizes on a fixed number of nodes are presented in tab. 1 (224 cores from 28 nodes were utilized for these tests). Here the matrices for the Poisson equation with 7-point discretization stencil in the cubic computational domain with different grid sizes were used. The presented results shows a weak dependence of the number of iterations to convergence from the matrix size. The increasing of the matrix size leads to close to linear growth of the solution time.

5. Conclusions

A new reordered formulation of the preconditioned BiCGStab method is proposed. It is optimized for the effective implementation on distributed memory computer systems. This method has no global synchronization points: the

Table 1: The number of iterations to convergence and solution time for different matrices (224 cores).

Grid size	Matrix size, mln. rows	Solution time, sec.	Iterations
465^3	100.5	17.2	9
500^3	125	25.2	10
550^3	166.4	31.7	10
600^3	216	44.7	11
650^3	274.6	61.4	11

preconditioning computations are used to overlap the inner products communications. The proposed method with the algebraic multigrid preconditioner was implemented with the MPI and hybrid (MPI+SHM) programming models. The efficiency of the reordered formulation is demonstrated by the experimental scalability results for the test matrices of 5.45 and 8 million rows. The presented results permits one to consider the reordered BiCGStab method with algebraic multigrid preconditioner as a well scalable solver for the systems of linear algebraic equations with large sparse nonsymmetric matrices.

6. Acknowledgements

The author thanks Dr. N. Nikitin for helpful remarks in the preparation of this paper.

Computational resources for this work were provided by the Research Computing Center of Lomonosov Moscow State University (cluster SKIF MSU "Chebyshev", 625 nodes: 2xIntel Quad core E5472 processors with IB DDR interconnect). This work was partially supported by Russian Foundation for Basic Research under the Grant No. 09-08-00390-a.

7. References

[1] P. N. Swarztrauber, A direct method for the discrete solution of separable elliptic equations, SIAM Journal on Numerical Analysis 11 (6) (1974) 1136–1150.
URL http://www.jstor.org/stable/2156231

[2] H. A. van der Vorst, BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 13 (2) (1992) 631–644. doi:10.1137/0913035.

[3] Y. Saad, M. H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869. doi:10.1137/0907058.

[4] Y. Saad, Iterative methods for sparse linear systems, 2nd edition, SIAM, Philadelpha, PA, 2003.

[5] U. Trottenberg, C. Oosterlee, A. Schuller, Multigrid, New York, Academic Press, 2001.

[6] T. Jacques, L. Nicolas, C. Vollaire, Electromagnetic scattering with the boundary integral method on MIMD systems, in: High-Performance Computing and Networking, Vol. 1593 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1999, pp. 1025–1031. doi:10.1007/BFb0100663.
URL http://www.springerlink.com/content/5g38713lh4244720/

[7] L. Yang, R. Brent, The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures, in: Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02). Proceedings., IEEE Computer Society, Los Alamitos, CA, USA, 2002, pp. 324–328. doi:10.1109/ICAPP.2002.1173595.

[8] Hypre: a library of high performance preconditioners.
URL https://computation.llnl.gov/casc/linear_solvers/sls_hypre.html

[9] G. Karypis, K. Schloegel, V. Kumar, ParMETIS: parallel graph partitioning and sparse matrix ordering library.
URL http://www-users.cs.umn.edu/~karypis/metis/parmetis/download.html

[10] A. Yakhot, T. Anor, H. Liu, N. Nikitin, Direct numerical simulation of turbulent flow around a wall-mounted cube: spatio-temporal evolution of large-scale vortices, Journal of Fluid Mechanics 566 (2006) 1–9. doi:10.1017/S0022112006002151.
URL http://journals.cambridge.org/article_S0022112006002151