ELSEVIER

Advances in Engineering Software 33 (2002) 169–177

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

Short Communication

Quantitative performance analysis of the improved quasi-minimal residual method on massively distributed memory computers

Laurence Tianruo Yang[a,b,*], Richard P. Brent[a]

[a]*Department of Computer Science, St Francis Xavier University, P.O. Box 5000, Antigonish, NS, Canada, B2G 2W5*
[b]*Computing Laboratory, Oxford University, Wolfson Building, Park Road, Oxford OX1 3QD, UK*

Received 1 July 1997; revised 9 March 2001; accepted 29 June 2001

Abstract

For the solutions of linear systems of equations with unsymmetric coefficient matrices, we have proposed an improved version of the quasi-minimal residual (IQMR) method [Proceedings of The International Conference on High Performance Computing and Networking (HPCN-97) (1997); IEICE Trans Inform Syst E80-D (9) (1997) 919] by using the Lanczos process as a major component combining elements of numerical stability and parallel algorithm design. For the Lanczos process, stability is obtained by a coupled two-term procedure that generates Lanczos vectors scaled to unit length. The algorithm is derived so that all inner products and matrix–vector multiplications of a single iteration step are independent and the communication time required for inner product can be overlapped efficiently with computation time. In this paper, a theoretical model of computation and communications phases is presented to allow us to give a quantitative analysis of the parallel performance with a two-dimensional grid topology. The efficiency, speed-up, and runtime are expressed as functions of the number of processors scaled by the number of processors that gives the minimal runtime for the given problem size. The model not only evaluates effectively the improvements in performance due to communication reduction by overlapping, but also provides useful insight into the scalability of the IQMR method. The theoretical results on the performance are demonstrated by experimental timing results carried out on a massively parallel distributed memory Parsytec system. © 2002 Published by Elsevier Science Ltd.

1. Introduction

One of the fundamental tasks of numerical computing is the requirement to solve linear systems with unsymmetric coefficient matrices. These systems arise very frequently in scientific 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.

One solution method, the quasi-minimal residual (QMR) algorithm [10], uses the Lanczos process [9] with look-ahead, a technique developed to prevent the process from breaking down in case of numerical instabilities, and in addition imposes a quasi-minimization principle. This combination leads to a quite efficient algorithm, among the most frequently and successfully used iterative methods. Such methods are widely used for very large and sparse problems, which are often solved on massively parallel computers.

The basic time-consuming computational kernels of the QMR method are usually: 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 [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 global communication, i.e. communication among all processors, required for inner product computations. Vector updates are perfectly parallelizable 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. Detailed discussions of the communication problem on distributed memory systems can be found in Refs. [4,6,16]. Global communication costs become relatively more important when the number of parallel processors increases and thus they have the potential to affect the scalability of the algorithm in a negative way [4,6].

Recently, we have proposed a new improved two-term recurrence Lanczos process [19,20] without look-ahead as the underlying process of the corresponding improved quasi-minimal residual method (IQMR). The idea is

* Corresponding author. Tel.: +1-902-867-5546; fax: +1-902-867-2448.
E-mail address: lyang@stfx.ca (L.T. Yang).

0965-9978/02/$ - see front matter © 2002 Published by Elsevier Science Ltd.
PII: S0965-9978(01)00055-2

motivated by the modified approach (we refer to as the MQMR method below) presented in Refs. [2,3], where there is only one global synchronization point each iteration. The IQMR algorithm is reorganized without changing its numerical stability. More importantly, compared with the MQMR method, the improved method is derived so that not only are all inner products independent (one global synchronization), but also matrix–vector multiplications are independent within a single iteration step. At the same time, communication required for inner products can be overlapped efficiently with computation of vector updates. Therefore, the cost of global communication on parallel distributed memory computers can be significantly reduced over that of the MQMR method. The resulting IQMR method maintains the favorable properties of the Lanczos process while not increasing computational costs. The detailed advantages over the original QMR and MQMR methods have been demonstrated by timing experiments in Refs. [19,20].

In this paper, a theoretical model of computation and communications phases is presented based on Refs. [5,18] to allow us to give a quantitative analysis of the parallel performance on a massively distributed memory computer with two-dimensional grid topology. The efficiency, speedup, and runtime are expressed as functions of the number of processors scaled by the number of processors that gives the minimal runtime for the given problem size. This provides a natural way to analyze the performance characteristics for the range of the number of processors that can be used effectively. The model not only shows clearly the influence of global communication on the performance, but also evaluates effectively the improvements in performance due to communication reductions by overlapping. The model also provides useful insight into the scalability of the IQMR method although it is limited by assumptions on the load balance and communication model. Further generalizations are being studied. The theoretical results on the performance are demonstrated by experimental timing results carried out on a massively parallel distributed memory computer Parsytec system.

The paper is organized as follows. In Section 2, we will describe briefly the classical unsymmetric Lanczos algorithm based on a three-term recurrence. A sketch of a new improved variant used as the underlying process is given in Section 3. The IQMR method is derived in detail in Section 4. In Section 5, the parallel performance model is presented, including the communication model and assumptions for computation time and communication costs. Finally, the effect of communication reduction and parallel performance are described in using both theoretical complexity analysis and experimental observations.

2. Lanczos process based on three-term recurrences

The classical unsymmetric Lanczos process [14] based on three-term recurrences reduces a matrix $A \in \Re^{N\times N}$ to tridiagonal form T using a similarity transformation which leads to the following three relations that serve to define the unsymmetric Lanczos process:

$$W^T V = I, \qquad AV = VT, \qquad A^T W = WT^T, \tag{1}$$

where I is the identity matrix and T is a tridiagonal matrix. More precisely, this process, starting with two vectors v_1 and w_1 satisfying $w_1^T v_1 = 1$, iteratively generates two finite sequences of vectors v_n and w_n such that, for $n = 1, 2, \ldots$

$$K_n(v_1, A) = \text{span}\{v_1, v_2, \ldots, v_n\},$$

$$K_n(w_1, A^T) = \text{span}\{w_1, w_2, \ldots, w_n\},$$

where $K_n(v_1, A) = \text{span}\{v_1, Av_1, \ldots, A^{n-1}v_1\}$, is the nth Krylov subspace with respect to v_1 and A and the two sets are biorthogonal as follows

$$w_m^T v_n = \begin{cases} 0 & \text{if } m \neq n, \\ 1 & \text{if } m = n. \end{cases}$$

where and in the sequel, we denote by $V = [v_1, v_2, \ldots, v_n]$ and $W = [w_1, w_2, \ldots, w_n]$, the matrices containing the Lanczos vectors v_n and w_n as columns.

This process leads to two inner products per iteration which require global communication on massively parallel distributed memory computers. Some improvements on the global communication required by inner product have been investigated in Ref. [13].

3. Lanczos process based on two-term recurrences

Although Lanczos used a similar technique built on coupled two-term recurrences in the early of 1950s the majority of papers deal with the three-term recurrences process. Recently, Freund et al. [11] reused Lanczo's idea to improve numerical stability. They claimed that, the two-term variant of the Lanczos process may be numerically more stable. This is why we consider this unsymmetric Lanczos process with two-recurrences as the underlying process of the QMR method.

Recently, Bücker et al. [2,3] proposed a new parallel modified version of the quasi-minimal residual method (MQMR) based on the coupled two-term recurrences Lanczos process without look-ahead strategy. The algorithm has the property that both generated sequences of Lanczos vectors are scalable and there is only a single global synchronization point per iteration. Based on a similar idea on their papers and further improvement, we will present a new improved two-term recurrence Lanczos process without look-ahead technique where, compared with the MQMR method, the improved Lanczos process has the property that not only are all inner products independent (one global synchronization), but also matrix–vector multiplications are independent within a single iteration step. Also, communication required for inner

product can be overlapped efficiently with computation of vector updates.

Here we suppose that the tridiagonal matrix T has an LU decomposition as

$$T = LU, \tag{2}$$

where the factors L and U are of lower and upper bidiagonal form, respectively. It is the bidiagonal structure of L and U that results in coupled two-term recurrences.

It has been pointed out by several authors [8,10,15,17] that in practice we should scale both sequences of Lanczos vectors to unit length in order to avoid over and underflow. This can only be achieved by giving up the bidiagonality $W^T V = I$ and setting $W^T V = D$ instead, where D is a diagonal matrix with entries $\delta_i \neq 0$ for $i = 1, 2, \ldots, N$.

The principal idea of the new approach suggested in Refs. [2,3] is to start from the scaling described above by using LU decomposition as well as introducing $P = VU^{-1}$ and $\tilde{Q} = WD^{-1}U^T$, which leads to

$$W^T V = D, \qquad V = PU, \tag{3}$$

and

$$\tilde{Q} = WD^{-1}U^T, \qquad AP = VL, \qquad A^T W = \tilde{Q}L^T D. \tag{4}$$

Suppose that the matrices introduced above have column vectors according to

$$P = [p_1, p_2, \ldots, p_n] \text{ and } \tilde{Q} = [\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_n].$$

Then, after a complicated derivation, we obtain

$$p_n = \frac{1}{\gamma_n}\tilde{v}_n - \mu_n p_{n-1}, \qquad \tilde{v}_{n+1} = u_n - \frac{\tau_n}{\gamma_n}\tilde{v}_n,$$

$$u_n = \frac{1}{\gamma_n}A\tilde{v}_n - \mu_n u_{n-1}, \qquad \tilde{w}_{n+1} = q_n - \frac{\tau_n}{\xi_n}\tilde{w}_n,$$

$$q_n = \frac{1}{\xi_n}A^T\tilde{w}_n - \frac{\gamma_n\mu_n}{\xi_n}q_{n-1}.$$

where the corresponding coefficients are

$$\gamma_{n+1} = (\tilde{v}_{n+1}, \tilde{v}_{n+1}), \qquad \xi_{n+1} = (\tilde{w}_{n+1}, \tilde{w}_{n+1}),$$

$$\rho_{n+1} = (w_{n+1}, \tilde{v}_{n+1}), \qquad \epsilon_{n+1}(A^T w_{n+1}, \tilde{v}_{n+1}),$$

$$\mu_{n+1} = \frac{\gamma_n\xi_n\rho_{n+1}}{\gamma_{n+1}\tau_n\rho_n}, \qquad \tau_{n+1} = \frac{\epsilon_{n+1}}{\rho_{n+1}} - \gamma_{n+1}\mu_{n+1}.$$

Algorithm 1: improved Lanczos process

1: $p_0 = q_0 = u_0 = 0$, $\gamma_1 = (\tilde{v}_1, \tilde{v}_1)$, $\mu_1 = 0$, $\xi_1 = (\tilde{w}_1, \tilde{w}_1)$,
2: $s_1 = A^T\tilde{w}_1$, $\rho_1 = (\tilde{w}_1, \tilde{v}_1)$, $\epsilon_1 = (s_1, \tilde{v}_1)$, $\tau_1 = \frac{\epsilon_1}{\rho_1}$;
3: **for** $n = 1, 2, \ldots$ **do**
4: $q_n = \frac{1}{\xi_n}s_n - \frac{\gamma_n\mu_n}{\xi_n}q_{n-1}$;
5: $\tilde{\omega}_{n+1} = q_n - \frac{\tau_n}{\xi_n}\tilde{w}_n$;
6: $s_{n+1} = A^T\tilde{w}_{n+1}$;
7: $t_n = A\tilde{v}_n$;
8: $u_n = \frac{1}{\gamma_n}t_n - \mu_n u_{n-1}$;
9: $\tilde{v}_{n+1} = u_n - \frac{\tau_n}{\gamma_n}\tilde{v}_n$;
10: $p_n = \frac{1}{\gamma_n}\tilde{v}_n - \mu_n p_{n-1}$;
11: $\gamma_{n+1} = (\tilde{v}_{n+1}, \tilde{v}_{n+1})$;
12: $\xi_{n+1} = (\tilde{w}_{n+1}, \tilde{w}_{n+1})$;
13: $\rho_{n+1} = (\tilde{w}_{n+1}, \tilde{v}_{n+1})$;
14: $\epsilon_{n+1} = (s_{n+1}, \tilde{v}_{n+1})$;
15: $\mu_{n+1} = \frac{\gamma_n\xi_n\rho_{n+1}}{\gamma_{n+1}\tau_n\rho_n}$;
16: $\tau_{n+1} = \frac{\epsilon_{n+1}}{\rho_{n+1}} - \gamma_{n+1}\mu_{n+1}$;
17: **end for**

Now we can reschedule the algorithm, without changing the numerical stability so that all inner products and matrix–vector multiplications of a single iteration step are independent and communication required for inner product can be overlapped efficiently with computation. The framework of this improved Lanczos process based on two-term recurrences is given as Algorithm 1. Under the assumptions, the improved Lanczos process can be efficiently parallelized as follows:

- The inner products of a single iteration step (11), (12), (13) and (14) are independent.
- The matrix–vector multiplications of a single iteration step (6) and (7) are independent.
- The communications required for the inner products (11), (12), (13) and (14) can be overlapped with the update for p_n in (10).

Therefore, the cost of communication on parallel distributed memory computers can be significantly reduced compared with the original QMR and the MQMR methods.

The biorthogonality relationships (3) and (4) are used to derive the algorithm. We can show that, in exact arithmetic, the vectors $\tilde{v}_i$ and $\tilde{w}_i$ generated by the above algorithm are biorthogonal.

Theorem 1. *Assuming no breakdown occurs, the vectors $\tilde{v}_i$ and $\tilde{w}_i$ generated by improved Lanczos process satisfy*

$$\tilde{w}_i^T\tilde{v}_j = \begin{cases} 0 & \text{if } i \neq j, \\ \rho_i \neq 0 & \text{if } i = j. \end{cases}$$

4. Improved quasi-minimal residual method

The improved Lanczos process now is used as a major component to a Krylov subspace method for solving a system of linear equations

$$Ax = b, \text{ where } A \in \Re^{N\times N}\ x, b \in \Re^N. \tag{5}$$

In each step, it produces approximation x_n to the exact solution of the form

$$x_n = x_0 + K_n(r_0, A), \qquad n = 1, 2, \ldots \tag{6}$$

Here x_0 is any initial guess for the solution of linear systems,

$r_0 = b - Ax_0$ is the initial residual, and $K_n(r_0, A) = \text{span}\{r_0, Ar_0, ..., A^{n-1}r_0\}$, is the nth Krylov subspace with respect to r_0 and A.

Given any initial guess x_0, the nth Improved QMR iterate is of the form

$$x_n = x_0 + V_n z_n, \tag{7}$$

where V_n is generated by the improved unsymmetric Lanczos process, and z_n is determined by a quasi-minimal residual property which will be described later.

For the improved Lanczos process, the nth iteration step generates

$$V_{n+1} = [v_1, v_2, ..., v_{n+1}] \text{ and } P_n = [p_1, p_2, ..., p_n],$$

satisfying

$$P_n = V_n U_n^{-1}, \qquad AP_n = V_{n+1} L_n, \tag{8}$$

where L_n and U_n are the leading principal $(n+1)\times n$ and $n \times n$ submatrices of the bidiagonal matrices L and U generated by the Lanczos algorithm. Note that L_n has full rank since we assume no breakdown occurs. The setting of $y_n = U_n z_n$ can be used to reformulate the improved QMR iterate in term of y_n instead of z_n giving

$$x_n = x_0 + P_n y_n. \tag{9}$$

The corresponding residual vector in term of y_n is obtained by the above scenario, namely

$$r_n = b - Ax_n = r_0 - V_{n+1} L_n y_n = V_{n+1}(\gamma_1 e_1^{(n+1)} - L_n y_n),$$

where the improved Lanczos process starts with $v_1 = (1/\|r_0\|_2) r_0$ and $e_1^{(n+1)} = (1, 0, ..., 0)^T$. Rather than minimizing $\|r_n\|_2$, generally and expensive task, the quasi-minimal residual property reduces costs by only minimizing the factor of the residual given in parentheses, i.e., y_n is the solution of least squares problem

$$\|\gamma_1 e_1^{(n+1)} - L_n y_n\|_2 = \min_y \|\gamma_1 e_1^{(n+1)} - L_n y\|_2.$$

Since L_n has full rank, the solution y_n is uniquely determined by the coupled iteration derived in Ref. [3] by avoiding the standard approach of computing a QR factorization of L_n by means of Givens rotations

$$y_n = \begin{pmatrix} y_{n-1} \\ 0 \end{pmatrix} + g_n, \qquad g_n = \theta_n \begin{pmatrix} g_{n-1} \\ 0 \end{pmatrix} + \kappa_n e_n^{(n)}, \tag{10}$$

where the scalars θ_n and κ_n are supplied from the following expressions

$$\theta_n = \frac{\tau_n^2(1-\lambda_n)}{\lambda_n \tau_n^2 + \gamma_{n+1}^2}, \quad \kappa_n = \frac{-\gamma_n \tau_n \kappa_{n-1}}{\lambda_n \tau_n^2 + \gamma_{n+1}^2},$$

$$\lambda_n = \frac{\lambda_{n-1}\tau_{n-1}^2}{\lambda_{n-1}\tau_{n-1}^2 + \gamma_n^2},$$

with $n \geq 2$, $\lambda_1 = 1$ and $\kappa_0 = 1$.

Inserting the first coupled recurrence relation yields

$$x_n = x_0 + P_{n-1} y_{n-1} + P_n g_n = x_{n-1} + d_n, \tag{11}$$

where $d_n = \theta_n P_n$ is introduced. Using the second coupled recurrence relation, the vector d_n is updated by

$$d_n = \theta_n P_{n-1} g_{n-1} + \kappa_n P_n e_n^{(n)} = \theta_n d_{n-1} + \kappa_n p_n, \tag{12}$$

where the vector is generated by the improved Lanczos process. Defining $f_n = Ad_n$, the residual vector is obtained by

$$r_n = r_{n-1} - f_n, \tag{13}$$

and the corresponding vector f_n is given by

$$f_n = \theta_n f_{n-1} + \kappa_n A p_n = \theta_n f_{n-1} + \kappa_n u_n. \tag{14}$$

The result is an improved QMR method (IQMR) based on coupled two-term recurrences with scaling of both sequences of Lanczos vectors. Numerical stability can be maintained and all inner products, matrix–vector multiplications of a single iteration step are independent and communication required for inner products can be overlapped efficiently with computation. The framework of this improved QMR method using the Lanczos algorithm based on two-term recurrences as underlying process is given in Algorithm 2.

Algorithm 2: improved quasi-minimal residual method

1: $\tilde{v}_1 = \tilde{w}_1 = r_0 = b - Ax_0$, $\lambda_1 = 1$, $\kappa_0 = 1$, $\mu_1 = 0$,
2: $p_0 = q_0 = u_0 = d_0 = f_0$, $\gamma_1 = (\tilde{v}_1, \tilde{v}_1)$, $\xi_1 = (\tilde{w}_1, \tilde{w}_1)$,
3: $s_1 = A^T \tilde{w}_1$, $\rho_1 = (\tilde{w}_1, \tilde{v}_1)$, $\epsilon_1 = (s_1, \tilde{v}_1)$, $\tau_1 = \frac{\epsilon_1}{\rho_1}$;
4: **for** n = 1, 2, ... **do**
5: $q_n = \frac{1}{\xi_n} s_n - \frac{\gamma_n \mu_n}{\xi_n} q_{n-1}$;
6: $\tilde{w}_{n+1} = q_n \frac{\tau_n}{\xi_n} \tilde{w}_n$;
7: $s_{n+1} = A^T \tilde{w}_{n+1}$;
8: $t_n = A\tilde{v}_n$;
9: $u_n = \frac{1}{\gamma_n} t_n - \mu_n u_{n-1}$;
10: $\tilde{v}_{n+1} = u_n - \frac{t_n}{\gamma_n} \tilde{v}_n$;
11: $p_n = \frac{1}{\gamma_n} \tilde{v}_n - \mu_n p_{n-1}$;
12: **if** $(r_{n-1}, r_{n-1}) <$ tol **then**
13: quit
14: **else**
15: $\gamma_{n+1} = (\tilde{v}_{n+1}, \tilde{v}_{n+1})$;
16: $\xi_{n+1} = (\tilde{w}_{n+1}, \tilde{w}_{n+1})$;
17: $\rho_{n+1} = (\tilde{w}_{n+1}, \tilde{v}_{n+1})$;
18: $\epsilon_{n+1} = (s_{n+1}, \tilde{v}_{n+1})$;
19: $\mu_{n+1} = \frac{\gamma_n \xi_n \rho_{n+1}}{\gamma_{n1} \tau_n \rho_n}$;
20: $\tau_{n+1} = \frac{\epsilon_{n+1}}{\rho_{n+1}} - \gamma_{n+1} \mu_{n+1}$;
21: $\theta_n = \frac{\tau_n^2(1-\lambda_n)}{\lambda_n \tau_n^2 + \gamma_{n+1}^2}$;

22: $\kappa_n = \frac{-\gamma_n \tau_n \kappa_{n-1}}{\lambda_n \tau_n^2 + \gamma_{n+1}^2}$;
23: $\lambda_n = \frac{\lambda_{n-1}\tau_{n-1}^2}{\lambda_{n-1}\tau_{n-1}^2 + \gamma_n^2}$;
24: $d_n = \theta_n d_{n-1} + \kappa_n p_n$;
25: $f_n = \theta_n f_{n-1} + \kappa_n u_n$;
26: $x_n = x_{n-1} + d_n$;
27: $r_n = r_{n-1} - f_n$;
28: **end if**
29: **end for**

Under the assumptions, the improved QMR method using Lanczos algorithm as underlying process can be efficiently parallelized as follows:

- The inner products of a single iteration step (12), (15), (16), (17) and (18) are independent.
- The matrix–vector multiplications of single iteration step (7) and (8) are independent.
- The communications required for the inner products (12), (15), (16), (17) and (18) can be overlapped with the update for p_n in (11).

Therefore, the cost of communication on parallel distributed memory computers can be significantly reduced compared with the original QMR and MQMR methods.

5. A theoretical performance model

We will make the following assumptions suggested in Refs. [5,6,18] for our communication model. First, the model assumes perfect load balance and the processors are configured as a 2D grid. However these restrictions are easily removed by changing the number of unknowns to the number of processors times the maximum number of unknowns over the processor grid and by changing the factor $\sqrt{P}$ to reflect the maximum distance over the processor graph more accurately. Apart from that, this model gives a lower bound on the performance of an efficient implementation. In this model, we assume that the preprocessing steps have been done in advance. Typical methods are reordering techniques like bandwidth reduction algorithms which derive from Cuthill and MacKee's algorithm, and the minimum degree algorithm [12] to improve the data locality. Therefore, each processor holds a sufficiently large number of successive rows of the matrix, and the corresponding sections of the vectors involved. That is, we assume that our problems have a strong data locality. Secondly, we can compute the inner products in two steps because the vectors are distributed over the processor grids. All processors start to compute the local inner product in parallel. After that, the local inner products are accumulated on a central processor and broadcasted. The communication time of an accumulation or a broadcast is of the order of the diameter of the processor grid. In other words, for an increasing number of processors, the communication time for the inner products increases as well, and as was pointed out [5,6], this is a potential threat to the scalability of the IQMR method. If the global communication for the inner products is not overlapped efficiently, it often becomes a bottleneck on large processor grids.

In this section, we will describe a simple performance model including the computation time and communication cost for the main kernels as we presented before based on our assumptions. The following terms are used in our paper as suggested in Ref. [6]:

- Communication cost: the term to indicate all the wall-clock time spent in communication, that is not overlapped with useful computation.
- Communication time: the term to refer to the wall-clock time of the whole communication.

In non-overlapped communication, the communication time and the communication cost are the same.

5.1. Computation time

The IQMR algorithm contains three distinct computational tasks per iteration

- Two simultaneous matrix–vector products, $A\tilde{v}_n$ and $A^T\tilde{w}_{n+1}$ whose computation time is given by $2t_{\text{fl}}N/P$.
- Five simultaneous inner products, $(\tilde{v}_{n+1}, \tilde{v}_{n+1})$, $(\tilde{w}_{n+1}, \tilde{w}_{n+1})$, $(\tilde{w}_{n+1}, \tilde{v}_{n+1})$, $(s_{n+1}, \tilde{v}_{n+1})$ and (r_{n-1}, r_{n-1}) whose computation time is given by $(2n_z - 1)t_{\text{fl}}N/P$.
- Nine vector updates, $q_n, \tilde{w}_{n+1}, u_n, \tilde{v}_{n+1}, p_n, d_n, f_n, x_n$ and r_n whose computation is given by $2t_{\text{fl}}N/P$.

Here N/P is the local number of unknown of a processor, t_{fl} is the average time for a double precision floating point operation and n_z is the average number of non-zero elements per row of the matrix.

The complete (local) computation time for the IQMR method is given approximately by the following equation:

$$T_{\text{comp}}^{\text{IQMR}} = (19 + 2n_z)\frac{N}{P}t_{\text{fl}} \tag{15}$$

5.2. Communication cost

Let the t_s denote communication start-up time and the transmission time from processor to processor associated with a single inner product computation be t_w and the diameter $p_d = \sqrt{P}$ for a processor grid with $P = p^2$ processors. Then the global accumulation and broadcast time for 1 inner product is taken as

$$T_{\text{comm}}^{\text{inner}} = 2p_d(t_s + t_w), \tag{16}$$

while the global accumulation and broadcast time for k simultaneous inner products take $2p_d(t_s + kt_w)$. For IQMR method the communication time is given as follows:

$$T_{\text{comm}}^{\text{IQMR}} = 2p_d(t_s + 5t_w). \tag{17}$$

6. Parallel performance

6.1. Theoretical complexity analysis

Before our discussion on timing experiments, we will focus on the theoretical analysis based on the model suggested in Refs. [5,18] on the performance of IQMR method. Firstly the efficiency, speed-up and runtime are expressed as functions of the number of processors scaled by the number of processors that gives the minimal runtime for the given problem size. Then we use this model to evaluate the impact in parallel performance due to communication reductions by overlapping.

The total runtime for IQMR is given by the following equation:

$$T_P^{\mathrm{IQMR}} = T_{\mathrm{comp}}^{\mathrm{IQMR}} + T_{\mathrm{comm}}^{\mathrm{IQMR}} = (19 + 2n_z)\frac{N}{P}t_{\mathrm{fl}} + 2\sqrt{P}(t_s + 5t_w). \tag{18}$$

This equation shows that for sufficiently large P the communication time will dominate the total runtime.

Let $P_{\max}$ denote as the number of processors that minimizes the total runtime for IQMR. Minimization of Eq. (18) gives

$$P_{\max} = \left(\frac{(19 + 2n_z)Nt_{\mathrm{fl}}}{t_s + 5t_w}\right)^{\frac{2}{3}}. \tag{19}$$

The percentage of the computation time in the whole runtime $E_P = T_1/PT_P$ for $P_{\max}$ processors is given by: $E_{P_{\max}} = 1/3$ where $T_1 = T_{\mathrm{comp}}^{IQMR}$. This means that $(1/3)P_{\max}$ is spent on communication.

The corresponding parallel speed-up $S_P = T_1/T_P$ for $P_{\max}$ processor is maximal, which is given by: $S_{P_{\max}} = (1/3)P_{\max}$.

The runtime for T_P for $P_{\max}$ is minimal and given as follows:

$$T_{P_{\max}} = 3((t_s + 5t_w)^2(19 + 2n_z)t_{\mathrm{fl}}N)^{1/3}. \tag{20}$$

For any P square processor grid, the corresponding parallel performance in terms of $P_{\max}$ can be stated as follows.

Theorem 2. *Let the runtime T_{P}, the speed-up S_{P}, and efficiency E_{P} be defined as before for any P square processor grids. Let α be the fraction $\alpha = P/P_{\max}$. The parallel efficiency is given by*

$$E_P = \frac{1}{1 + 2\alpha^{3/2}}, \tag{21}$$

the parallel speed-up S_{P} is given by

$$S_P = \frac{\alpha}{1 + 2\alpha^{3/2}}P_{\max}, \tag{22}$$

and the runtime T_{P} is given by

$$T_P = \frac{1 + 2\alpha^{3/2}}{\alpha}((t_s + 5t_w)^2(19 + 2n_z)t_{\mathrm{fl}}N)^{1/3}. \tag{23}$$

There are several interesting feaures in Theorem 2 which can be stated as follows:

- The number of processors that minimizes the runtime, $P_{\max}$, increases only as $\Theta(N^{2/3})$ and so the minimum runtime increases necessarily as $\Theta(N^{1/3})$ which means that for a sufficient increase in N the runtime increases as well no matter how many processors are being used. It seems that we cannot achieve perfect scalability, to be able to keep the runtime constant for the increasing N by increasing the number of processors P. However, this IQMR method scales well in the sense that the minimum runtime increases only slowly as a function of the problem size.
- For any number processors $P < P_{\max}$, we can improve the runtime as follows. While increasing N we simultaneously increase α so that the number of processors P increases faster than $P_{\max}$, thereby trading efficiency against performance.

6.2. The impact of reduction by overlapping

In this part, we will use our model to evaluate the impact in parallel performance due to the communication reductions by overlapping.

Since in the IQMR method, there is one possibility to overlap the computation time with communication time, we assume P_{ovl} as the number of processors for which all communication can be just overlapped with computation. In other words, the communication time for five simultaneous inner products is equal to the overlapping computation time of one vector update. The value for P_{ovl} follows from

$$\sqrt{P}(t_s + 5t_w) = t_{\mathrm{fl}}\frac{N}{P}.$$

It is easy to obtain the theoretical result:

$$P_{\mathrm{ovl}} = \left(\frac{t_{\mathrm{fl}}N}{t_s + 5t_w}\right)^{2/3}.$$

Based on these theoretical results, we will discuss three different situations:

- If $P < P_{\mathrm{ovl}}$, there is no significant communication visible.
- If $P > P_{\mathrm{ovl}}$, the overlap is no longer complete and the communication time is given by $2p_d(t_s + 5t_w) - 2t_{\mathrm{fl}}N/P$. For this case, we can see clearly that the efficiency decreases again because the communication time increases and the computation time in the overlap decreases.
- If $P = P_{\mathrm{ovl}}$, the communication time can just be

overlapped by the computation time. Then the corresponding runtime is given by $(19 + 2n_z)t_{\mathrm{fl}}N/P$. And accordingly we have linear speed-up and 100% parallel efficiency.

It is easy to show in general case $P_{\mathrm{ovl}} < P_{\max}$. For the general case, we assume that a fraction β of the computations in vector update can be used to overlap communication in inner products, the runtime of one iteration for P processors and N unknowns to be

$$\hat{T}_P = (19 + 2n_z - 2\beta)\frac{t_{\mathrm{fl}}N}{P} + \max\left(2\beta\frac{t_{\mathrm{fl}}N}{P}, 2(t_s + 5t_w)\sqrt{P}\right).$$

From the expression of the runtime we can easily drive the number of processors $\hat{P}_{\max}$ for which $\hat{T}_P$ is minimal, and hence for which $\hat{S}_P$ is maximal, is given by

$$\hat{P}_{\max} = \left(\frac{(19 + 2n_z - 2\beta)t_{\mathrm{fl}}N}{t_s + 5t_w}\right)^{2/3}. \tag{24}$$

The percentage of the computation time in the whole runtime $\hat{E}_P$ for $\hat{P}_{\max}$ processors is given:

$$\hat{E}_{\hat{P}_{\max}} = \frac{19 + 2n_z}{3(19 + 2n_z - 2\beta)}.$$

Since it is easy to show that $\hat{P}_{\max} > P_{\mathrm{ovl}}$, the overlapping factor $\beta = 1$. This leads to around $(2/3)\hat{T}_{\hat{P}_{\max}}$ spent in communication!

The corresponding parallel speed-up $\hat{S}_P$ for $\hat{P}_{\max}$ processor is maximal, which is given:

$$\hat{S}_{\hat{P}_{\max}} = \frac{(19 + 2n_z)\hat{P}_{\max}}{3(17 + 2n_z)} \approx \tfrac{1}{3}\hat{P}_{\max}.$$

The runtime for $\hat{T}_P$ for $\hat{P}_{\max}$ is minimal and given

$$\hat{T}_{\hat{P}_{\max}} = 3((t_s + 5t_w)^2(17 + 2n_z)t_{\mathrm{fl}}N)^{1/3}.$$

with regard to the runtime, efficiency, and speed-up for number of processors P with overlapped communication, the corresponding results can be stated in the following theorem:

Theorem 3. *Let the runtime, parallel speed-up and the parallel efficiency with overlapped communication denote as $\hat{T}_P$, $\hat{S}_P$ and $\hat{E}_P$ respectively. Let θ be the fraction as $\theta = P/\hat{P}_{\max}$, $\phi = (1/(19 + 2n_x - 2\beta))^{2/3}$ and $\psi = 19 + 2n_z - 2\beta$, then the parallel efficiency $\hat{E}_P$ is given by*

$$\hat{E}_P = \begin{cases} 1, & \theta \le \phi \\ \dfrac{\psi + 2\beta}{\psi(1 + 2\theta^{3/2})}, & \theta > \phi \end{cases}$$

the parallel speed-up $\hat{S}_P$ is given by

$$\hat{S}_P = \begin{cases} P, & \theta \le \phi \\ \dfrac{(\psi + 2\beta)P}{\psi(1 + 2\theta^{3/2})} & \theta > \phi \end{cases}$$

and the runtime $\hat{T}_P$ is given by

$$\hat{T}_P = \begin{cases} (\psi + 2\beta)\dfrac{t_{\mathrm{fl}}N}{P}, & \theta \le \phi \\ \psi(1 + 2\theta^{3/2})\dfrac{t_{\mathrm{fl}}N}{P}, & \theta > \phi \end{cases}$$

Some interesting remarks can be made as follows:

- The maximum speed-up in case of overlapped communication is reached for $\hat{P}_{\max}$. In general we can see clearly that $\hat{P}_{\max}$ is always smaller than $P_{\max}$. Furthermore, the speed-up for $\hat{P}_{\max}$ is always better.
- A direct comparison shows that the optimal performance for the IQMR method with overlapping is better than the approach without overlapping.
- With overlapping we can run faster on fewer processors, which, of course, gives a large improvement in efficiency. But the scalability of the IQMR method is not improved by overlapping the communication.
- The number of processors that minimizes the runtime, $\hat{P}_{\max}$, increases as $\Theta(N^{2/3})$, so that the runtime still increases as $\Theta(N^{1/3})$.

6.3. Numerical timing results

Now we compare the theoretical estimates from the model with measured performance by numerical experiments on the Parsytec massively parallel system. Since we are only interested in the delaying effects relative to the computational time, we will consider only one iteration.

Our problem is an electrostatic problem mathematically described by a linear partial differential equation of second order with Dirichlet boundary conditions and discretized using five-point centered finite difference approximations. The resulting system of linear equations has a unsymmetric coefficient matrix with $N = 10000$ whose nonzero entries are structured at most five nonzero entries per row according to the symmetric pattern by natural ordering of the unknowns described in Refs. [1]. The corresponding parameter values are:

$$n_z = 5,\ \ t_{\mathrm{fl}} = 3.00\ \mu\mathrm{s},\ \ t_s = 5.30\ \mu\mathrm{s},\ \ t_w = 4.80\ \mu\mathrm{s}.$$

The theoretical results for $P_{\max}$ and P_{ovl} of the IQMR method are $P_{\max} = 959$, $P_{\mathrm{ovl}} = 102$. For $P > P_{\mathrm{ovl}}$, $\hat{P}_{\max} = 910$. The theoretical and measured runtimes, efficiencies and speed-ups for the IQMR method without overlapping consideration are given in Table 1. Due to the physical limitations, we can measure only the results for the limited processors. In order to see the trend of the performance, we use the least squares method to estimate and predict the results when the critical point of parallel processor numbers is larger than the available number.

For the processor grids we used we have $P < P_{\max}$, so

Table 1
Measured/estimated runtime, efficiencies and speed-ups without overlapping consideration

Processor grid	Theoretical results				Measured/estimated results		
	α	T_p (ms)	E_p (%)	S_p	T_p (ms)	E_p (%)	S_p
6×6	0.04	24.52	98.57	35.48	24.78	98.42	35.54
10×10	0.10	9.29	93.69	93.69	9.32	93.66	93.68
14×14	0.20	5.26	84.40	164.43	5.31	84.48	164.33
18×18	0.34	3.74	71.80	232.63	3.72	71.81	232.66
22×22	0.50	3.08	58.24	281.87	3.10	58.29	281.41
26×26	0.70	2.81	45.79	309.57	2.88	46.10	310.12
30×30	0.95	2.62	35.48	319.34	2.58	35.44	320.48
34×34	1.21	2.79	27.42	310.98	2.62	27.58	310.30
38×38	1.51	2.83	21.30	307.54	2.68	21.42	307.66
42×42	1.84	2.95	16.70	294.52	2.02	16.49	294.42

that the runtime decreases and speed-up increases for increasing numbers of processors as predicted by our theoretical analysis. When the processor grids reach 30×30, close to $P_{\max}$, the minimal runtime and hence maximal speed-up can be achieved. After $P > P_{\max}$, the runtime increases and speed-up decreases for increasing number of processors. For parallel efficiency, for sufficiently large P, the communication time dominates the total runtime. When $P = P_{\max}$, it is not surprising to see that 67% is spent in communication! After $P > P_{\max}$, the percentage of communication time in total runtime increases for increasing number of processors. Also the theoretical model can predict very well on the performance of the parallel implementation.

For the IQMR method with overlapping consideration, the theoretical and measured runtime, efficiencies and speed-ups for one iteration step are given in Table 2. Here we consider processor grids between 6×6 and 42×42 as same as the approach without overlapping consideration to be the typical examples to investigate the parallel performance.

For $P_{\text{ovl}} = 102$, we see clearly for processor grid 6×6, there is no significant communication visible. The corresponding parallel efficiency equals almost to 100%. When the number of processors is increased from 6×6 to 10×10, the parallel efficiency remains almost constant around 100% which has been predicted by our previous analysis because we have $P < P_{\text{ovl}}$, so that the increase in the communication time is covered by the overlapping computation. During this stage, the communication time is not very important for small processor grids. For processor grids of P increases to 10×10 near P_{ovl} the communication time balance with the overlapping computation time. Accordingly, we can achieve near linear speed-up with near 100% parallel efficiency. But for larger processor grids we cannot overlap the communication which dominates the runtime. For the processor grids $P_{\text{ovl}} < P < \hat{P}_{\max}$, the runtime decreases and speed-up increases for increasing numbers of processors. Since the communication dominates the runtime, the parallel efficiency decreases slowly for increasing number of processors. When the processor grids reach 30×30, close to $\hat{P}_{\max}$, the minimal runtime and hence maximal speed-up can be achieved. After $P > \hat{P}_{\max}$, the runtime increases and speed-up decreases for increasing number of processors which are predicted very well by the theoretical analysis. With regards to parallel efficiency, when $P = \hat{P}_{\max}$, 63% is spent in communication! After $P > \hat{P}_{\max}$, the percentage of communication time in total runtime keeps increasing for

Table 2
Measured/estimated runtime, efficiencies and speed-ups without overlapping consideration

Processor grid	Theoretical results					Measured/estimated results		
	β	θ	T_p (ms)	E_p (%)	S_p	T_p (ms)	E_p (%)	S_p
6×6	0.21	0.04	24.17	100.00	36.00	24.42	98.22	34.91
10×10	1.98	0.11	8.70	100.00	100.00	8.18	99.48	99.08
14×14	1.00	0.21	4.95	89.62	175.65	5.06	89.17	175.66
18×18	1.00	0.35	3.55	75.54	244.74	3.68	75.36	244.84
22×22	1.00	0.53	2.96	60.67	293.65	3.04	60.92	293.38
26×26	1.00	0.74	2.72	47.28	319.64	2.88	47.49	319.46
30×30	1.00	0.98	2.65	36.37	327.31	2.56	36.62	327.26
34×34	1.00	1.26	2.69	27.95	323.05	2.62	27.58	323.37
38×38	1.00	1.58	2.78	21.61	312.08	2.88	21.44	312.28
42×42	1.00	1.93	2.92	16.89	297.91	2.89	16.81	297.86

increasing number of processors. The theoretical model can give a very precise prediction on the performance of the parallel implementation.

A direct comparison shows that the optimal performance for the IQMR method with overlapping consideration is better than the approach without overlapping. With overlapping we can run faster on fewer processors, which, of course, gives a large improvement in parallel efficiency. But the scalability is not improved by overlapping the communication.

From the theoretical and measured results, the quantitative parallel behavior is relatively well modeled for both implementations with and without overlapping consideration. There is an almost constant difference between the measured and theoretically estimated values.

7. Conclusions

In this paper, a theoretical model of computation and communication phases is presented to allow us to give a quantitative analysis of the parallel performance of the IQMR method on a massively distributed memory computer with two-dimensional grid topology. The efficiency, speedup, and runtime are expressed as functions of the number of processors scaled by the number of processors that gives the minimal runtime for the given problem size. This provides a natural way to analyze the performance characteristics for the range of the number of processors that can be used effectively. The model not only shows clearly the dramatic influence of global communication on the performance, but also evaluates effectively the improvements in the performance due to communication reductions by overlapping. The model also provides useful insight into the scalability of IQMR method. The numerical timing results agree well with the theoretically estimated results predicted by this performance model.

References

[1] Bücker HM. Isoefficiency analysis of parallel QMR-like iterative methods and its implications on parallel algorithm design. Technical Report KFA-ZAM-IB-9604. Central Institute for Applied Mathematics, Research Centre Julich, Germany, January 1996.

[2] Bücker HM, Sauren M. A parallel version of the quasi-minimal residual method based on coupled two-term recurrences. In: Proceedings of Workshop on Applied Parallel Computing in Industrial Problems and Optimization (Para96), LNCS184. Technical University of Denmark, Lyngby, Denmark, August 1996. Berlin: Springer, 1996.

[3] Bücker HM, Sauren M. A parallel version of the unsymmetric Lanczos algorithm and its application to QMR. Technical Report KFA-ZAM-IB-9605. Central Institute for Applied Mathematics, Research Centre Julich, Germany, March 1996.

[4] de Sturler E. A parallel variant of the GMRES(m), Proceedings of the 13th IMACS World Congress on Computational and Applied Mathematics. IMACS, Criterion Press, 1991.

[5] de Sturler E. Performance model for Krylov subspace methods on mesh-based parallel computers. Technical Report CSCS-TR-94-05, Swiss Scientific Computing Centre, La Galleria, CH-6928 Manno, Switzerland, May 1994.

[6] de Sturler E, van der Vorst HA. 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 Netherlands, 1994.

[7] Dongarra JJ, Duff IS, Sorenson DC, van der Vorst HA. Solving linear systems on vector and shared memory computers. Philadelphia, PA: SIAM, 1991.

[8] Freund RW, Golub GH, Nachtigal N. Iterative solution of linear systems. Acta Numer 1991:57–100.

[9] Freund RW, Gutknecht MH, Nachtigal NM. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. SIAM J Scient Stat Comput 1993;14:137–58.

[10] Freund RW, Nachtigal NM. QMR: a quasi-minimal residual method for non-Hermitian linear systems. Numerische Mathematik 1991;60:315–39.

[11] Freund RW, Nachtigal NM. An implementation of the QMR method based on coupled two-term recurrences. SIAM J Scient Stat Comput 1994;15(2):313–37.

[12] George A. Direct solution of sparse positive definite systems: some basic ideas and open problems. New York: Academic Press, 1981.

[13] Kim SK, Chronopoulos AT. An efficient non-symmetric Lanczos method on parallel vector computers. J Comput Appl Math 1992:357–74.

[14] Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J Res Natl Bur Stand 1950;45:255–82.

[15] Parlett BN, Taylor DR, Liu ZA. A look-ahead Lanczos algorithm for unsymmetric matrices. Math Comput 1985;44:105–24.

[16] Pommerell C. Solution of large unsymmetric systems of linear equations. PhD thesis. ETH, 1992.

[17] Taylor DR. Analysis of the look ahead Lanczos algorithm for unsymmetric matrices. PhD thesis. Department of Mathematics, University of California at Berkeley, November 1982.

[18] Yang T. Solving sparse least squares problems on massively parallel distributed memory computers. In: Proceedings of International Conference on Advances in Parallel and Distributed Computing (APDC-97), March 1997. Shanghai, P.R. China.

[19] Yang T, Lin HX. The improved quasi-minimal residual method on massively distributed memory computers. In: Proceedings of The International Conference on High Performance Computing and Networking (HPCN-97), April, Vienna, Austria, 1997.

[20] Yang T, Lin HX. The improved quasi-minimal residual method on massively parallel distributed memory computers. IEICE Trans Inform Syst 1997;E80-D(9):919–24 Special issue on Architectures, Algorithms and Networks for Massively Parallel Computing.