Mathematical Modelling and Analysis. Ed. A.Buikis
Volume 12 Number 1, 2007, pages 1–16
 ISSN 1392-6292 print, ISSN 1648-3510 online

PARALLEL VARIATIONAL ITERATIVE LINEAR SOLVERS

RAIM. ČIEGIS[1], REM. ČIEGIS[2,3], A. JAKUŠEV[1] and G. ŠALTENIENĖ[1]

[1] *Vilnius Gediminas Technical University*

Saulėtekio al. 11, LT-10223, Vilnius, Lithuania

E-mail: rc@fm.vtu.lt, Aleksandr.Jakushev@fm.vtu.lt

[2] *Vilnius University, Kaunas Faculty of Humanities*

Muitinės st. 8, LT-44280, Kaunas, Lithuania

E-mail: remigijus.ciegis@vukhf.lt

[3] *Vytautas Magnus University*

K. Donelaičio st. 58, LT-44248, Kaunas, Lithuania

Received September 15, 2006; revised November 25, 2006; published online February 10, 2007

Abstract. In this work we consider parallel variational algorithms for solution of linear systems. Theoretical analysis explains the superlinear convergence rate for two step gradient descent method. A new modification of the algorithm is proposed. Results of computational experiments are given for a linear system of equations approximating 3D elliptic boundary value problem. All algorithms are implemented using parallel array object tool *ParSol*, then a parallel algorithm follows semi-automatically from the serial one. Results of the scalability analysis are presented and the efficiency of the presented parallel algorithm is investigated experimentally.

Key words: variational iterative methods, parallel algorithms, linear algebra problems, software tools

1. Introduction

Application of finite difference or finite volume methods for elliptic or parabolic problems of PDE lead to linear algebraic systems of equations of the form

$$AX = F, \tag{1.1}$$

where A is a positive definite $A > 0$, symmetric matrix of dimension $N \times N$, F is the right-hand side vector, and X is the solution vector. We note that in systems obtained after approximation of PDE matrices A are sparse and in many cases they are also banded.

To solve such systems efficiently is an important part of mathematical modelling technology. Iterative methods take advantage of the fact that matrices are sparse and therefore they are at the center of most modern numerical codes and software tools. We also note that parallelization of iterative algorithms can be done more easily than to parallelize direct algorithms.

Among iterative algorithms various modifications of Krylov subspace methods are most popular. Very robust methods can be built by using this general approach. It is important to note that modifications of many iterative Krylov type algorithms are known for solution of linear systems with non-symmetrical matrix [7].

For completeness of the presentation we will describe briefly a general scheme for one-step variational iterative algorithms. It is well known that system of linear equations (1.1) is equivalent to minimization of quadratic functional $Q(Y) = (A(Y-X),\, Y-X)$, where (X, Y) defines an inner product of two vectors [7]. We consider the variational iterative method defined by

$$X^{n+1} = X^n - \tau_n P^n, \quad n \geq 0, \tag{1.2}$$

where some initial approximation of the solution X^0 is given. Here P^n are known vectors and they define the direction vector for minimization of the functional $Q(X^{n+1})$. By solving the obtained quadratic equation with respect to τ_n we get the optimal value of the parameter

$$\tau_n = \frac{(AX^n - F, P^n)}{(AP^n, P^n)}.$$

The methods of *Steepest Descent* (SD) and *Minimal Residuals* (MR) are most popular examples of such algorithms [7].

Let A be symmetric and positive definite matrix having eigenvalues

$$0 < \lambda_1 \leq \lambda_1 \leq \cdots \leq \lambda_N.$$

Then $\kappa(A) = \lambda_N/\lambda_1$ is the condition number of A. In applications for solution of second order elliptic PDEs this number $\kappa(A) = \mathcal{O}(h^{-2}) \gg 1$, where h is a space step of the grid [13]. Let us denote the vector norm

$$\|Y\|_A = (AY,\, Y)^{1/2}.$$

The convergence rate of the SD method is linear and the following error estimate is valid [7]:

$$\|X^{n+1} - X\|_A \leq \rho \|X^n - X\|_A,$$

$$\rho = \frac{1-\eta}{1+\eta}, \quad \eta = \frac{1}{\kappa(A)}.$$

Thus the number of iterations K required to reduce the error of initial approximation $1/\varepsilon$ times, i.e. to get the estimate

$$\|X^K - X\|_A \leq \varepsilon \|X^0 - X\|_A,$$

is given by $K = \frac{1}{2}\kappa(A)\ln(1/\varepsilon)$.

Very similar estimates are valid for the MR method, only the error is estimated in the following norm

$$\|A(X^{n+1} - X)\| = \|R^n\|.$$

Much faster convergence is obtained for the *Conjugate Gradient* (CG) method. The obtained iterations X^n are the best approximation of the exact solution X of (1.1) in the so-called Krylov space

$$K_n(A; F) = \text{ span } \{F, AF, \ldots, A^{n-1}F\},$$

consisting of linear combinations of the $\{A^i F,\ i = 0, \ldots, n-1\}$. The error is estimated as [7]:

$$\|X^n - X\|_A \leq 2\left(\frac{1-\sqrt{\eta}}{1+\sqrt{\eta}}\right)^n \|X^0 - X\|_A.$$

The number of iterations K_{CG} required to reduce the initial error $1/\varepsilon$ times is given by $K = \frac{1}{2}\sqrt{\kappa(A)}\ln(2/\varepsilon)$.

Let us write an iterative algorithm in the canonical form

$$X^n = C_n X^{n-1} + \tau_n F.$$

where $C_n = I - \tau_n A$. Two types of methods are used to construct iterative algorithms and to investigate its convergence rate. The first method is based on the *minimax* estimates of the spectrum of operator

$$\min_{\tau_1,\ldots,\tau_k} \max_{a_m \leq \lambda \leq a_M} |(1-\tau_1\lambda)\cdots(1-\tau_k\lambda)|,$$

where a_m, a_M are the estimates of the spectrum of matrix A. The explicit sequence of parameters $\tau_1, \ldots, \tau_k$ is calculated by using roots of the Chebyshev polynomials [13]. The main advantage of this method is that parameters are computed *a priori* if some estimates of the spectrum of matrix A are known. We note that minimization is done with respect to the worst case of the spectrum distribution, which may be not the case for a given matrix A.

Iterative methods of the second group are based on variational formulation of the systems of linear equations. These algorithms use the paradigm of greedy methods, i.e. they minimize the error in a given trial direction. It is well known that the convergence rate of variational algorithms reduces after each iteration and the vector of global error asymptotically converges to a linear combination of two eigenvectors of matrix A, corresponding to the largest and smallest eigenvalues. The convergence rate of variational iterative algorithms is estimated from above by using this asymptotical property of the generated sequence of iterations.

Recently a few new iterative algorithms are proposed, which are based on combination and modification of well known SD and MR methods. Some preliminary computational experiments show a super-linear convergence rate.

Our goal is to make asymptotical analysis of these iterative algorithms and to present some explanations why their convergence is such fast. This analysis is based only on qualitative analysis and gives only heuristical proofs. The importance of the given analysis follows from the ability to construct new iterative algorithms with better convergence rate than the original methods.

The second goal of the paper is to develop parallel versions of new iterative algorithms and to investigate their efficiency and scalability. It is well known that variational iterative methods are sensitive to the accuracy of calculation of iterative parameters τ_n. These parameters depend on global residual vectors, since dot products of some vectors are computed. In parallel algorithms the new source of perturbations is introduced when these dot products are computed on distributed processors and local sums are accumulated.

The rest of the paper is organized as follows. In Section 2, we formulate and investigate the two step gradient descent algorithm. It is explained why this method has a superlinear convergence rate. A new modification of the steepest descent algorithm is proposed by using results of the theoretical analysis. The third modification of the SD method is obtained by using the sub-relaxation of the iterative parameter. Convergence rate of these three variational methods is tested experimentally by solving a discrete 3D elliptic problem. In Section 3, we construct and investigate parallel variational algorithms. They are developed by using data parallel decomposition method and implemented with the help of the new parallel array object tool *ParSol*. A parallel version of the code follows semi-automatically from the serial one. By applying the scalability analysis we investigate the efficiency of the obtained parallel algorithms. At the end of this section results of computational experiments are presented and discussed. A discrete approximation of 3D elliptic problem is taken as a test problem. The spectrum of the obtained matrix is quite complicated and it simulates very well the distribution of eigenvalues of matrix obtained in solving many real-world applications.

2. Modifications of the SD Method

In this section we will investigate two modifications of the SD and MR iterative methods. The two step gradient descent (TSGD) method, was proposed in [6]. The conclusions on the convergence rate of this iterative algorithm were based on computational experiments, including 2D elliptic problems. Some theoretical explanation of its superlinear convergence rate was given in [2]. Another interesting modification of the SD method is proposed in [8]. It is based on the fixed sub-relaxation of the iterative parameter. No results on the convergence of this new algorithm are presented in [8].

2.1. Formulation of the TSGD method

TSGD method is a combination of two variational iterative algorithms, described above. The method of *Steepest Descent* (SD) is obtained when vectors

$P^n = \nabla Q(X^n)$ coincides with a gradient of the functional Q. Let us denote the residual $R^n = AX^n - F$, then the SD method is defined as:

$$X^{n+1} = X^n - \tau_n R^n, \quad \tau_n = \frac{(R^n,\, R^n)}{(AR^n,\, R^n)}. \tag{2.1}$$

In the method of *Minimal Residuals* (MR) the quadratic functional of residuals $Q_2(Y) = (AY - F, AY - F)$ is minimized and the new iteration is defined as

$$X^{n+1} = X^n - \tau_n R^n, \quad \tau_n = \frac{(AR^n,\, R^n)}{(AR^n,\, AR^n)}. \tag{2.2}$$

At odd iterations of the TSGD method the SD method is used and at even iterations the MR method is applied.

TSGD method

TSGD (Matr A, Vec x, Vec f, ε)
begin
(1) Set initial values $i = 0, \quad r = Ax - f$;
(2) **while** ($\|r\| > \varepsilon$) **do**
(3) $v = Ar$;
(4) **if** ($i == 0$) **then**
(5) $\tau = (r, r)/(v, r)$;
(6) $i = 1$;
else
(7) $\tau = (v, r)/(v, v)$;
(8) $i = 0$;
end if
(9) $x := x - \tau\, r; \quad r := r - \tau\, v$;
end do
end TSGD

Convergence analysis

We divide the theoretical analysis of the convergence rate of the TSGD method into two steps. *First* we investigate the case when the residual vector $R^0 = AX^0 - F$ for an initial vector X^0 has a form

$$R^0 = c_1 V_1 + c_N V_N,$$

where V_j are the eigenvectors of the matrix A corresponding to the eigenvalues λ_j. Simple computations give that after two iterations of the TSGD method the residual of vector X^2 is given by

$$R^2 = \frac{c_1^2 c_N^2 (\lambda_N - \lambda_1)^2 \lambda_N}{(c_1^2 \lambda_1 + c_N^2 \lambda_N)(c_1^2 \lambda_1^2 + c_N^2\, \lambda_N^2)} \Big(c_1 V_1 + c_N \frac{\lambda_1}{\lambda_N} V_N \Big).$$

We see that $R^2 \neq cR^0$ and the convergence is essentially nonlinear. After few iterations the residual vectors $R^j \approx d_j V_1$, i.e. a linear combination of two

vectors reduces to a vector parallel to the first eigenvector. It is well known that in this case the SD and MR methods converge after one iteration [7].

Not reducing the generality of the analysis let us take the following initial residual vector

$$c_1 = 1, \quad c_N = 1, \quad \eta = \frac{\lambda_1}{\lambda_N} \ll 1.$$

Then we get the following convergence rates $\|R^j\| \leq \rho_j \|R^0\|$:

$$\rho_2 = \mathcal{O}(1), \quad \rho_4 = \mathcal{O}(\eta^{-1}), \quad \rho_6 = \mathcal{O}(1), \quad \rho_{6+2k} = \mathcal{O}(\eta^{2k}).$$

The presented analysis explains a superlinear convergence rate of the TSGD method in the case of very special initial vector.

Remark 1. We note that for a linear combination of two eigenvectors the CG algorithm converges to the exact solution in two iterations.

In the *second* part of the convergence analysis we take into account a special asymptotical property of the SD and MR methods. Starting from a general initial vector

$$R^0 = c_1 V_1 + c_2 V_2 + \ldots + c_N V_N$$

iterations monotonically converge to the approximation consisting of only two eigenvectors:

$$R^j \approx d_1 V_1 + + d_N V_N.$$

Then a super-linear speed-up of the convergence is obtained in few iterations, after that we get an approximation, where all eigenvectors are important in the spectral representation of the residual vector and a full cycle is repeated.

In conclusion we state, that a combination of these two properties, i.e. the superlinear convergence of the TSGD method for initial vectors belonging to a special subspace of vectors and asymptotical convergence of all initial vectors to the neighbourhood of this subspace, explains the fast convergence of the TSGD method.

The presented analysis is not oriented to obtain strict estimates of the convergence rate (we note that only estimates from above are usually obtained in convergence analysis) but we investigated a qualitative behaviour of the iterative sequences. It explains why the super-linear convergence rates can be expected and, even more, it helps to construct new efficient algorithms. One example will be given in the next section.

2.2. Modified TSDG method

By using the results of the theoretical analysis presented above we propose a modification of the TSGD method, when each consecutive $(m+n)$ iterations are implemented in the following way: a) first m iterations of the SD method are computed, b) then n iterations of the TSGD method are computed. We denote this iterative algorithm by MSD(m, n).

MSD(m,n) algorithm

MSD (Matr A, Vec x, Vec f, ε, m, n)
begin
(1) Set initial values $i = 1; \quad j = 0;$
(2) $r = Ax - f;$
(3) **while** ($\|r\| > \varepsilon$) **do**
(4) $j := j + 1; \quad v = Ar;$
(5) **if** ($j \le m \;||\; i == 0$) **then**
(6) $\tau = (r, r)/(v, r);$
(7) $i = 1;$
else
(8) $\tau = (v, r)/(v, v);$
(9) $i = 0;$
end if
(10) $x := x - \tau r; \quad r := r - \tau v;$
(11) **if** ($j == (n + m)$) $j = 0;$
end do
end MSD

2.3. Sub-relaxation of the SD method

A new interesting modification of the SD method is proposed in [8]. The following computational algorithm is used

$$X^{n+1} = X^n - 0.9\,\tau_n R^n, \quad \tau_n = \frac{(R^n,\, R^n)}{(AR^n,\, R^n)}. \tag{2.3}$$

The main idea is to restrict the length of the step, predicted by the classical SD method. The factor 0.9 was fitted experimentally. We denote this algorithm by SRSD (Sub-Relaxation Steepest Descent).

Remark 2. We note that this algorithm is an interesting example demonstrating that the greedy variational SD algorithm is not optimal. In fact in the Seidel method exactly opposite technique, i.e. over-relaxation, is used to speed-up the convergence rate.

SRSD algorithm

SRSD (Matr A, Vec x, Vec f, ε)
begin
(1) Set initial values $i = 1; \quad r = Ax - f;$
(2) **while** ($\|r\| > \varepsilon$) **do**
(3) $v = Ar;$
(4) $\tau = (r, r)/(v, r);$
(5) $x := x - 0.9\tau r;$
(6) $r := r - 0.9\tau v;$
end do
end SRSD

Convergence analysis

Computational experiments for different tests, including 2D elliptic problems, are reported in [8]. The convergence rate of the SRSD method appears to be similar to the convergence rate of the CG method. Till now no theoretical results are given to prove this hypothesis.

Again, we apply two step convergence analysis methodology, which is proposed in previous sections. Let us consider the case when an initial residual $R^0 = AX^0 - F$ has a form

$$R^0 = c_1 V_1 + c_N V_N.$$

Results of computational experiments show that after nondeterministic number of iterations we periodically get approximations where only the first eigenvector is dominant in a linear combination of two eigenvectors and a superlinear speed-up is achieved at such iterations. In Table 1 we present a basic information on convergence of iterations. We take the following eigenvalues of the matrix:

$$\lambda_1 = 1, \quad \lambda_N = 1000.$$

Two cases of initial vectors are considered

$$1)\, c_1 = 1, \;\; c_N = 1, \quad 2)\, c_1 = 1, \;\; c_N = 0.1.$$

The values of coefficients d_1^k, d_N^k in the representation of residual vector after k-th iteration

$$R^k = d_1^k V_1 + d_N^k V_N$$

and the convergence rate ρ^k are given in Table 1.

Table 1. Convergence analysis of SRSD algorithm.

k	case	d_1^k	d_N^k	ρ^k
49	1	1.00	3.4776	0.9991
50	1	1.00	0.0893	0.8990
106	1	1.00	-2.8350	0.9990
107	1	1.00	0.0336	0.5770
108	1	1.00	24.580	0.9991
13	2	1.00	3.1184	0.9991
14	2	1.00	0.0236	0.4215
27	2	1.00	3.2015	0.9991
28	2	1.00	0.0394	0.6465

A problem of finding the optimal value of sub-relaxation factor d is very complicated. First we need to define strictly the objective function. As example this objective function can be specified as

$$\min_{0<d\leq 1} \max_{c_1,\ldots,c_N} \max_{\lambda_1\leq\ldots\leq\lambda_N} K(c_1,\ldots,c_N;\lambda_1,\ldots,\lambda_N;d),$$

where $K(c_1,\ldots,c_N;\lambda_1,\ldots,\lambda_N;d)$ is the number of iterations. In Table 2 we present the numbers of iterations required to solve 2×2 dimension system of linear equations, when

$$\lambda_1 = 1, \quad \lambda_N = 1000$$

and the accuracy of computations $\varepsilon = 10^{-4}$.

Table 2. The dependence of K on the sub-relaxation factor d.

c_1	c_N	0.88	0.89	0.90	0.91	0.92	0.93	0.94	1.0
1	1	359	138	401	248	230	147	216	4606
1	10	204	435	387	317	171	120	283	190
1	0.5	242	212	198	251	171	113	125	2951
1	0.1	116	152	159	153	125	132	237	189

It follows that $d = 0.93$ is optimal for this small test suit.

2.4. Computational experiments

Let us consider a three–dimensional Poisson problem

$$\begin{cases} -\sum\limits_{\alpha=1}^{3} \dfrac{\partial^2 u}{\partial x_\alpha^2} = f, \quad x \in Q := (0,1)\times(0,1)\times(0,1)\,, \\ u(x) = g, \quad x \in \partial Q, \end{cases} \tag{2.4}$$

We discretize it by using the finite–volume method and get a system of linear equations

$$-a_{i+1,j,k}U_{i+1,j,k} - a_{i-1,j,k}U_{i-1,j,k} - a_{i,j+1,k}U_{i,j+1,k} - a_{i,j-1,k}U_{i,j-1,k} \tag{2.5}$$

$$-a_{i,j,k+1}U_{i,j,k+1} - a_{i,j,k-1}U_{i,j,k-1} + a_{ijk}U_{ijk} = F_{ijk}, \quad 1 \leq i,j,k \leq n.$$

The 7-point stencil is used for the approximation. The resulting system of equations may be written as $AU = F$, where A is a symmetric positive definite matrix. More details about the properties of such discrete approximations are given by Samarskii [13], Golub and Van Loan [7] .

We note that the convergence of variational iterative algorithms strongly depends not only on the stiffness parameter $\kappa(A)$, but also on the distribution and clustering of all eigenvalues of matrix A (see, e.g. [5]). This test problem is a typical example of 3D elliptic problems solved in many real-world applications by using variational iterative algorithms.

In Table 3 we present the numbers of iterations, required to reduce the initial error by factor $1/\varepsilon$, $\varepsilon = 10^{-4}$ for different variational iterative methods.

The size of the discrete grid is equal to $n \times n \times n$ and it is scaled from $n = 20$ till $n = 160$. An initial vector is such, that all eigenvectors present in its spectral decomposition. It consist of smooth and non-smooth parts. Artificially we add a few components corresponding to largest eigenvalues of the matrix (all eigenvectors of the matrix are known analytically, see [13]). The same initial vector is used in all numerical experiments.

Table 3. The numbers of iterations for solution of the test problem.

Method	$n = 20$	$n = 40$	$n = 80$	$n = 160$
SD	337	1273	3920	7702
MR	341	1255	3283	6130
TSGD	78	140	318	552
MSD(30, 10)	74	118	196	240
SRSD	101	153	252	262
CG	38	69	118	203

The presented computational results confirm the prediction that all modifications of the SD and MR methods have superlinear convergence rate, which is close to the convergence rate of the CG method. It is important to note that this conclusion is obtained for a 3D elliptic problem, thus simple, robust variational methods can be recommended for usage in general software tools.

Parameters M_1, M_2 of the MSD(M_1, M_2) method are not optimized in these experiments. Experimental results show that the convergence rate of MSD is not sensitive to the selection of these parameters (see Table 4).

Table 4. Convergence rate of MSD method for different parameters M_1, M_2.

n	$(20, 10)$	$(30, 10)$	$(40, 10)$	$(30, 15)$	$(30, 20)$
40	116	118	128	124	134
80	202	198	202	168	182

3. Parallel TSGD Algorithm

The parallel versions of variational algorithms from the previous section are developed by using data parallel decomposition method (see [9]). For example we consider the TSGD algorithm:

procedure The serial TSGD algorithm
begin
(1) $X^0, \quad n = 0, \quad R^0 = AX^0 - F$
(2) $RN = (R^n, R^n)$
(3) **while** $\left(RN > \varepsilon\,(R^0, R^0)\right)$ **do**
(4) $G^n = AR^n$,
(5) **if** (n is even) **then**
(6) $\tau_{n+1} = RN/(G^n, R^n)$,
else
(7) $\tau_{n+1} = (G^n, R^n)/(G^n, G^n)$,
end if
(8) $X^{n+1} = X^n - \tau_{n+1} R^n$,
(9) $R^{n+1} = R^n - \tau_{n+1} G^n$,
(10) $RN = (R^{n+1}, R^{n+1}), \quad n := n + 1$.
end while
end

As a preconditioner we use a diagonal part of the matrix **A**. Diagonal preconditioners can be parallelized very efficiently. It is well known (see [7]) that a faster convergence rate can be obtained with more complicated preconditioners, such as ILU or multigrid preconditioners, but they are not very efficient in parallel computing.

A parallel version of ILU is obtained by doing the factorization of a local part of matrix at each processor. It is well known that such strategy reduces the convergence rate of the preconditioned iterative algorithms, but a parallelism of the obtained preconditioner is the same as for the diagonal one. Various ordering techniques are used to overcome the trade–off between parallelism and convergence in incomplete factorization. Development and analysis of parallel preconditioners for CG algorithm are investigated in [3, 11, 12].

3.1. Formulation of the algorithm

Let us assume that we have p processors, which are connected by three dimensional mesh, i.e. $p = p_1 \times p_2 \times p_3$. The grid ω_h (a data set) is decomposed into p 3D subgrids by using a block distribution scheme. Then each subgrid ω_{np} has

$$\frac{n}{p_1} \times \frac{n}{p_2} \times \frac{n}{p_3} = \frac{n^3}{p}$$

computational points and it is assigned to one processor. All processors simultaneously perform the same code but with different data sets. Each processor is responsible for all computations of the local part of vector X.

The update of vector G^n at grid points which lie beside cutting planes (i.e. boundary nodes of the local part of the vector U) needs a special attention, since information from the neighbouring processors is required to compute new values. A star-stencil of seven points is used in (2.5), therefore in each dimension two *ghost* points are added to local subgrids. Each processor exchanges data with its neighbours in the specified topology of processors.

Similarly, parallel computation of the inner product of two vectors requires global communication among all processors.

3.2. Complexity of the parallel algorithm

In this section we present scalability analysis of the parallel TSGD algorithm. According to the definition of the isoefficiency function, we should find the rate at which the problem size W needs to grow with p for a fixed efficiency of the algorithm.

Let denote $S_p = W/T_p$, $E_p = S_p/p$, here T_p is the complexity of the parallel algorithm when p processors are used, S_p is the speed-up of the parallel algorithm and E_p is the efficiency coefficient. Let $H(p,W) = pT_p - W$ be the total overhead of a parallel algorithm. Then the isoefficiency function $W = g(p, E_p)$ is defined by the implicit equation (see [9]):

$$W = \frac{E_p}{1 - E_p} H(p, W)\,. \tag{3.1}$$

For simplicity of notation we take $E_p = 0.5$. First we will estimate the complexity of the parallel TSGD algorithm.

1. At steps (8), (9) of the parallel TSGD algorithm *saxpy* type operations are computed. Only local data is used by each processor and no communication among processors is required. The complexity of this part of computations is given by
$$T_{1p} = g_1 \frac{n^3}{p}.$$
2. During matrix-vector multiplication at step (4) processors exchange ghost elements of vector R^n, corresponding to its local boundary grid points. The complexity of this step is given by
$$T_{2p} = g_2 \frac{n^3}{p} + 6\Big(\alpha + \beta \frac{n^2}{p^{2/3}}\Big),$$
where α is the message startup time and β is the time required to send one element of data.
3. Parallel computation of the inner product of two vectors at steps (2), (6), (7) and (10) requires global communication among all processors during summation of local parts of the product. The complexity of this step is given by
$$T_{3p} = g_3 \frac{n^3}{p} + 2R(p)(\alpha_b + \beta_b),$$
where $R(p)$ depends on the algorithm used to implement the *ReduceAll* operation and the architecture of the computer. For the simplest algorithm we have $R(p) = p$.

Summing up all obtained estimates we compute the complexity of the parallel TSGD algorithm

$$T_p(J) = g\frac{n^3}{p} + 5R(p)(\alpha_b + \beta_b) + 6\Big(\alpha + \beta\frac{n^2}{p^{2/3}}\Big).$$

The problem size of the serial algorithm is equal to $W = gn^3$.

The total overhead of the parallel TSGD algorithm is given by

$$H(p, W) = 6\alpha p + 6\beta p^{1/3}n^2 + 5pR(p)(\alpha_b + \beta_b).$$

We analyze the influence of each individual term. The component that requires the problem size to grow at the fastest rate determines the overall asymptotic isoefficiency function. After simple computations we get the following three isoefficiency functions

$$W = \mathcal{O}(p), \quad W = \mathcal{O}(p), \quad W = \mathcal{O}\big(pR(p)\big)\,.$$

Thus the the overall asymptotic isoefficiency function is defined by the overheads of the global reduction operation.

Let us consider the most simple algorithm, when all processors send their local results to the master processor, which computes the global inner product and broadcasts it to all processors. Then $R(p) = p$ and $W = W = \mathcal{O}(p^2)$. In three–dimensional mesh network the global *reduce* and *broadcast* operations can be implemented with $R(p) = p^{1/3}$. Thus the problem size W has to grow as $\mathcal{O}(p^{4/3})$ to maintain a certain efficiency. For a hypercube mesh we have smaller costs of the global reduction operation $R(p) = \log p$, then isoefficiency function is close to linear $W = \mathcal{O}(p \log p)$. We note, that in the case of a moderate number of processors the costs of global reduction operation can be ignored and the isoefficiency function $W = \mathcal{O}(p)$ linearly depends on p.

3.3. Results of computational experiments

In this section we present some results of computational experiments. Computations were performed on IBM SP5 computer at CINECA, Bologna.

Parallel numerical objects.

Special tools are developed to simplify the parallelization of numerical algorithms, e.g. *Diffpack* tool [10] and *PETSc* toolkit [1]. We have developed a tool *ParSol* of parallel numerical arrays, which can be used for semi–automatic parallelization of data parallel algorithms, that are implemented in `C++`. Such algorithms are usually constructed for solving PDEs and systems of PDEs on logically regular rectangular grids. *ParSol* is a library of parallel array objects, a functionality of which is similar to *Distributed Arrays* in *PETSc*. Its main features and some applications are described in [4, 14].

As it follows from the theoretical analysis given in previous section the superlinear convergence of the TSGD method follows due to high nonlinear effect of the obtained iterative operator. In fact we deal with the bifurcation of the function describing the error or residual of the iteration sequence. A smooth trend of these functions are changed with abrupt jumps at some iterations. This feature is sensitive to small perturbations of iterative parameters.

Thus we can expect that the numbers of iterations of the parallel versions of these algorithms will depend on the number of processors. This prediction is confirmed by computational experiments for a test problem with difference grid $80 \times 80 \times 80$ (see Table 5, where the numbers of iterations for parallel variational algorithms are presented). The classical iterative algorithms as well as the MSD algorithm are stable with respect to the number of processors, while the convergence rate of the SRSD and TSGD methods depends on p.

Table 5. The numbers of iterations for parallel variational algorithms.

Method	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$
CG	118	118	118	118	118
SD	3920	3920	3920	3920	3920
MSD(30, 10)	198	198	198	198	198
SRSD	252	268	279	234	218
TSGD	318	342	320	412	382

Next we consider the efficiency of parallel CG and MSD(30,10) algorithms. They are typical examples of two–step recurrence and one–step iterative algorithms. The results are given in Tables 6 and 7. Here we present the values of experimental speedup $S_p(n) = \dfrac{T_1(n)}{T_p(n)}$ and efficiency $E_p(n) = S_p(n)/p$ coefficients for scaled sizes of the discrete problem.

Table 6. The speedup and efficiency coefficients for the MSD(30,10) algorithm.

p	$S_{p,80}$	$E_{p,80}$	$S_{p,120}$	$E_{p,120}$	$S_{p,160}$	$E_{p,160}$
2	1.920	0.960	1.962	0.981	1.971	0.986
4	3.688	0.922	3.776	0.941	3.808	0.952
8	7.104	0.878	7.232	0.904	7.922	0.911
16	13.15	0.822	13.62	0.851	14.58	0.865
32	24.99	0.781	25.70	0.803	26.01	0.813

It follows from results, presented in Tables 6 and 7 that both parallel algorithms scale well, and experimental results confirm the theoretical prediction that isoefficiency function of the parallel MSD algorithm is close to linear one.

Table 7. The speedup and efficiency coefficients for parallel CG algorithm.

p	$S_{p,80}$	$E_{p,80}$	$S_{p,120}$	$E_{p,120}$	$S_{p,160}$	$E_{p,160}$
2	1.891	0.945	1.926	0.963	1.962	0.981
4	3.612	0.903	3.688	0.922	3.760	0.940
8	6.768	0.846	6.968	0.871	7.160	0.895
16	12.82	0.801	13.17	0.823	13.49	0.843
32	24.03	0.751	24.96	0.780	25.38	0.793

Acknowledgment

The work has been partially performed under the Project HPC–EUROPA (RII3–CT-2003-506079), with the support of the European Community – Research Infrastructure Action under the FP6 "Structuring the European Research Area" Programme. Raim. Čiegis gratefully acknowledge the hospitality and excellent working conditions in CINECA, Bologna.

This work was also supported by the Lithuanian State Science and Studies Foundation within the framework of the Eureka Project EUREKA E!3691 OPTCABLES.

The authors would like to thank the referee for his constructive criticism which helped to improve the clarity of this note.

References

[1] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L. Curfman McInnes, B.F. Smith and H. Zhang. *PETSc Users Manual. ANL-95/11 - Revision 2.3.0, Argonne National Laboratory.* 2005.

[2] R. Čiegis. On the convergence rate of some iterative methods of variational type. *Mathematical modelling*, **9**(4), 115–125, 1997.

[3] R. Čiegis. Analysis of parallel preconditioned conjugate gradient algorithms. *Informatica*, **16**(3), 317–332, 2005.

[4] R. Čiegis, A. Jakušev, A. Krylovas and O. Suboč. Parallel algorithms for solution of nonlinear diffusion problems in image smoothing. *Math. Modelling and Analysis*, **10**(2), 155–172, 2005.

[5] R. Čiegis and V. Pakenienė. On the solution of one minimization problem. *Liet. matem. rink.*, **42**(Special Issue), 604–608, 2002.

[6] V.V. Ermakov and N.N. Kalitkin. Two stage gradient descent method. *J. Comput. Math. and Math. Physics*, **20**(4), 1040–1045, 1980.

[7] G.H. Golub and Ch. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 1996.

[8] O. Kachar, A. Boltnev and N. Kalitkin. Speeding up the convergence of simple gradient methods. In: *Mathematical Modelling and Analysis 2005. Proceedings 10th Int. Conf. MMA2005 & CMAM2, Trakai, Lithuania, 2005.* Vilnius, Technika, 349–354, 2005.

[9] V. Kumar, A. Grama, A. Gupta and G. Karypis. *Introduction to parallel computing: design and analysis of algorithms.* Benjamin/Cummings, Redwood City, 1994.

[10] H.P. Langtangen and A. Tveito. *Advanced Topics in Computational Partial Differential Equations. Numerical Methods and Diffpack Programming.* Springer, Berlin, 2003.

[11] S. Ma. Comparisons of the parallel preconditioners for large nonsymmetric sparse linear systems on a parallel computer. *International Journal of High Speed Computing*, **12**(1), 55–68, 2004.

[12] M. Monga-Made and H. A. van der Vorst. A generalized domain decomposition paradigm for parallel incomplete LU factorization preconditionings. *Future Generation Computer Systems*, **17**, 925–932, 2001.

[13] A.A. Samarskii. *The theory of difference schemes.* Marcel Dekker, Inc., New York–Basel, 2001.

[14] V. Starikovičius, R. Čiegis and A. Jakušev. Analysis of upwind and high resolution schemes for solving convection dominated problems in porous media. *Math. Modelling and Analysis*, **11**(4), 447–472, 2006.