DOI 10.1007/s10958-015-2395-4
Journal of Mathematical Sciences, Vol. 207, No. 5, June, 2015

PARALLEL APPROACHES AND TECHNOLOGIES OF DOMAIN DECOMPOSITION METHODS

Y. L. Gurieva* **and V. P. Il'in***,† UDC 519.612

The efficiency of two-level iterative processes in the Krylov subspaces is investigated, as well as their parallelization in solving large sparse nonsymmetric systems of linear algebraic equations arising from grid approximations of two-dimensional boundary-value problems for convection-diffusion equations with various coefficient values. Special attention is paid to optimization of the sizes of subdomain intersections, to the types of boundary conditions on adjacent boundaries in the domain decomposition method, and to the aggregation (or coarse grid correction) algorithms. The outer iterative process is based on the additive Schwarz algorithm, whereas parallel solution of the subdomain algebraic systems is effected by using a direct or a preconditioned Krylov method. The key point in the programming realization of these approaches is the technology of forming the so-called extended algebraic subsystems in the compressed sparse row format. A comparative analysis of the influence of various parameters is carried out based on numerical experiments. Some issues related to the scalability of parallelization are discussed. Bibliography: 13 *titles.*

1. Introduction

Creating parallel iterative algorithms for solving grid systems of linear algebraic equations (SLAEs) arising from finite element or finite volume approximations of boundary-value problems is based on decomposition of a computational domain and is a multifaceted mathematical and technological problem. On the one hand, a high convergence rate of the applied iterative process should be provided. On the other hand, the overall performance of a computational tool is highly dependent on the data structure used and on the program implementation of algorithms on a multiprocessor computer system [1].

The aim of this paper is an experimental investigation of the impact on the parallelization scalability of the following three algorithmic factors: the sizes of intersections of adjacent subdomains, the type of iterated boundary conditions on their inner boundaries, and application of aggregation (or coarse grid correction) methods [2]. As the test suit, we used a rather simple but representative SLAE family, namely, five-point approximations of linear convection-diffusion equations on a uniform grid in a rectangular domain [3, 4]. It is this somewhat idealized situation that allows one to describe the related computational and information issues, as well as methods for coping with them.

One of the main computational tools used in solving very large (of orders of about 10^9) sparse SLAEs in parallel is the two-level iterative process in the Krylov subspaces with the block Jacobi preconditioning, which is the additive Schwarz method.

Section 2 states the original problems and describes the computational methods for solving them. Section 3 presents some algorithms and implementation technologies for the so-called "extended" subdomains, based on which domain decomposition is parallelized. Results of numerical experiments for different initial data and parameters of algorithms are discussed in the last section.

*Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia, e-mail: yana@lapasrv.sscc.ru.

†Novosibirsk State University, Novosibirsk, Russia, e-mail: ilin@sscc.ru.

Translated from *Zapiski Nauchnykh Seminarov POMI*, Vol. 428, 2014, pp. 89–106. Original article submitted November 10, 2014.

2. Problem statement and description of parallel algorithms

Consider the Dirichlet problem for the convection-diffusion equation

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + p\frac{\partial u}{\partial x} + q\frac{\partial u}{\partial y} = f(x,y), \quad (x,y) \in \Omega, \quad u|_\Gamma = g(x,y), \tag{1}$$

in a computational domain $\Omega = (a_x, b_x) \times (a_y, b_y)$, where Γ is the boundary of Ω, and, for simplicity, the convection coefficients p, q are given constants.

The above boundary-value problem is discretized on a uniform grid

$$\begin{aligned} &x_i = a_x + ih_x, \;\; y_j = a_y + jh_y, \quad i = 0, 1, \ldots, N_x + 1; \;\; j = 0, 1, \ldots, N_y + 1; \\ &h_x = (b_x - a_x)/(N_x + 1), \quad h_y = (b_y - a_y)/(N_y + 1), \end{aligned} \tag{2}$$

using the five-point stencil, which yields linear algebraic equations of the form

$$(Au)_l = a_{l,l}u_l + a_{l,l-1}u_{l-1} + a_{l,l+1}u_{l+1} + a_{l,l-N_x}u_{l-N_x} + a_{l,l+N_x}u_{l+N_x} = f_l, \tag{3}$$

where

$$l = l(i,j) \equiv i + (j-1)N_x = 1, \ldots, N = N_xN_y \tag{4}$$

is the "global" (or continuous) number of a node of the inner grid. The coefficients can have different forms, and specific versions of the formulas can be found in [3, 4]. Equations (3) are written for the inner grid nodes; for the nodes near the boundary whose numbers belong to the index set $i = 1, N_x$ or $j = 1, N_y$ the values of the solution known from the boundary conditions are substituted into the corresponding equations and moved to their right-hand sides, so that the corresponding coefficients $a_{l,l'}$ in (3) vanish. SLAE (3) is written in matrix-vector form as

$$Au = f, \;\; A = \{a_{l,l'}\} \in \mathcal{R}^{N,N}, \;\; u = \{u_l\}, \;\; f = \{f_l\} \in \mathcal{R}^N. \tag{5}$$

Hereinafter, by Ω we denote not only the computational domain but also the set of grid nodes $(x_i, y_j) \in \Omega$ (we will use the term "grid computational domain"), as well as the set of indices $l = 1, \ldots, N$ of the vectors u and f of dimension N.

Now we decompose the domain Ω, i.e., represent it, first, as a union of identical (for simplicity) nonintersecting rectangular subdomains

$$\Omega = \bigcup_{s=1}^{P} \Omega_s, \quad P = P_xP_y,$$

each of which contains the same number of grid nodes

$$M = m_xm_y, \quad N_x = P_xm_x, \quad N_y = P_ym_y, \quad N = PM.$$

In a natural way, the subdomains form a two-dimensional macrogrid, whose macrovertices are numbered by pairs of indices p, q (similar to the grid node indices i, j), whereas the "continuous" number of a subdomain is defined as follows:

$$s = s(p,q) \equiv p + (q-1)P_x = 1, \ldots, P; \quad p = 1, \ldots, P_x, \quad q = 1, \ldots, P_y. \tag{6}$$

Thus, a subdomain with number $s(p,q)$ contains the grid nodes with indices

$$\begin{aligned} &i = I_{p-1} + 1 \equiv (p-1)m_x + 1, \ldots, pm_x \equiv I_p, \\ &j = J_{q-1} + 1 \equiv (q-1)m_y + 1, \ldots, qm_y \equiv J_q, \end{aligned} \tag{7}$$

where I_p and J_q are the initial numbers of the grid nodes in the (p,q)th subdomain in x and y directions, and the global grid numbers $l(i,j)$ are computed in accordance with (4). Every subdomain Ω_s has its own four faces, which jointly form a boundary not passing through the grid nodes.

Now from the continuous ordering of nodes we turn to their subdomain-by-subdomain ordering: all the nodes in Ω_1 are numbered first, then the nodes in Ω_2 are numbered, etc. The components of the vectors u and f are ordered consistently, and the original SLAE (3) takes the following block form:

$$A_{s,s}\overline{u}_s + \sum_{s' \in Q_s} A_{s,s'}\overline{u}_{s'} = f_s, \quad s = 1, \ldots, P. \tag{8}$$

Here, $\overline{u}_s \in \mathcal{R}^{N_s}$ is the subvector of the vector u whose components correspond to the nodes from the subdomain Ω_s, and Q_s is the set of numbers of the subdomains adjacent to the subdomain Ω_s. Hereinafter, it is stipulated that the local node ordering in every subdomain is the natural one; the local pairs of indices $i' = 1, \ldots, m_x$; $j' = 1, \ldots, m_y$ are introduced, and the continuous number is determined by the formula $l' = i' + (j' - 1)m_x$, similar to (4). If the subdomain number equals $s(p, q)$, then the reordering of the nodes from the local order to the global one is effected with the use of the quantities introduced in (7) in accordance with the relations $i = i' + I_{p-1}$, $j = j' + J_{q-1}$.

Note that the above formalism pertains to a decomposition of the grid domain without subdomains overlapping and without usage of separator nodes shared by adjacent subdomains. However, in order to increase the generality and efficiency of the algorithms discussed below, it is necessary to turn to constructing "extended" subdomains with nonempty intersections.

Let ω_l denote a grid stencil or a set of nodes adjacent to the lth node, i.e., the set of numbers of those components of the desired grid solution which are involved in the corresponding lth equation of the form (3). For a grid subdomain Ω_s, by $\Gamma_s = \Gamma_s^0$ we denote its boundary, i.e., the set of nodes external with respect to Ω_s and having at least one of their neighboring nodes in Ω_s ($\overline{\Omega}_s = \overline{\Omega_s^0} = \Omega_s \cup \Gamma_s^0$ is the closure of the grid subdomain Ω_s). Further, let Γ_s^1 denote the first extended boundary, or the first external front $\overline{\Omega}_s$, i.e., the set of nodes that do not lie in $\overline{\Omega}_s$ but have at least one neighboring node in $\overline{\Omega}_s$ ($\overline{\Omega}_s^1$ is the first extension of $\overline{\Omega}_s^0$). The subsequent stages of extending a grid subdomain are defined similarly; the number of such stages Δ is called the parameter of the extended subdomain $\overline{\Omega}_s^\Delta = \Omega_s^\Delta \bigcup \Gamma_s^\Delta$, where the nodes from Γ_s^Δ no longer belong to Ω_s^Δ; the number of nodes in Ω_s^Δ is denoted by $\bar{N}_s$. An example of an extended subdomain with $\Delta = 3$ is presented in Fig. 1.

In building an iterative Schwarz process in grid subdomains, the interface links between adjacent subdomains can be taken into account in different ways. Let the lth node be a near-boundary one in the subdomain Ω_s^Δ, i.e., $l \in \Gamma_s^{\Delta-1}$. Write the corresponding equation of the algebraic system in the following form:

$$\Big(a_{l,l} + \theta_l \sum_{l' \notin \Omega_s^\Delta} a_{l,l'}\Big)u_l^n + \sum_{l \in \Omega_s^\Delta} a_{l,l'}u_{l'}^n = f_l + \sum_{l' \notin \Omega_s^\Delta} a_{l,l'}\Big(\theta_l u_l^{n-1} - u_{l'}^{n-1}\Big). \tag{9}$$

Here, n is the iteration number, and to the right-hand and left-hand sides of relation (9) terms with the same coefficients and involving the factor θ_l, which is a parameter of the iterative process, are added (see Fig. 2). Observe that the case $\theta_l = 0$ can be interpreted as using the Dirichlet boundary condition when solving the auxiliary subproblem in Ω_s. Similarly, the case $\theta_l = 1$ corresponds to the Neumann condition, and the value $\theta_l \in (0, 1)$ corresponds to the boundary condition of the third type (or the Robin condition).

In matrix form, the algorithm considered can be represented as the block Jacobi method

$$\bar{B}_s(\widetilde{u}_s^{n+1} - \widetilde{u}_s^n) = \widetilde{f}_s^n - (\bar{A}\widetilde{u}^n)_s \equiv \widetilde{r}_s^n. \tag{10}$$

Here, the subvectors $\widetilde{u}_s^n$ and $\widetilde{f}_s^n$ correspond to the extended subdomains and have dimension $\bar{N}_s$; $\bar{B}_s \in \mathcal{R}^{\bar{N}_s, \bar{N}_s}$ are preconditioning matrices, whose diagonal entries depend on the parameters θ_l.

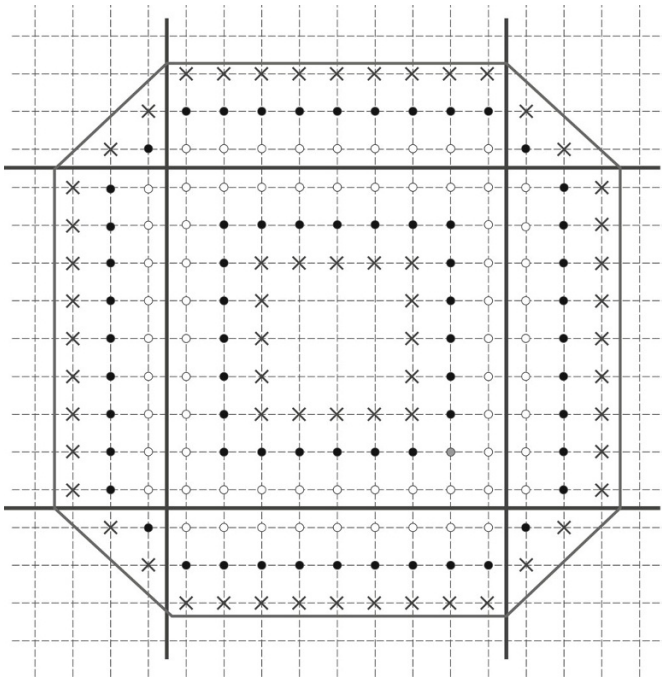

Fig. 1. An example of subdomain extension.

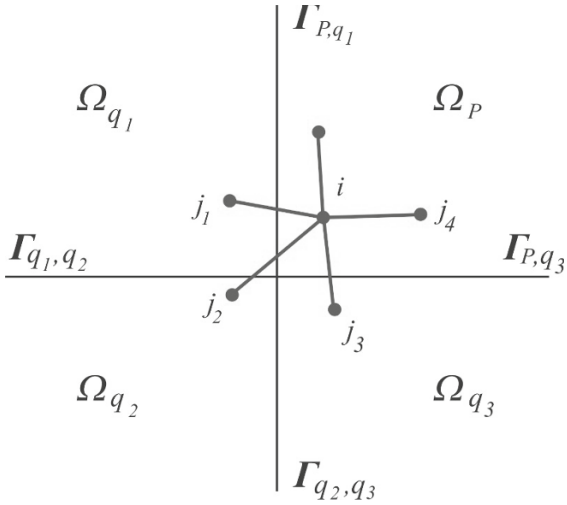

Fig. 2. A grid stencil of a near-boundary node.

The iterative process of the form (10) is underdetermined because the unknowns of $\widetilde{u}_s^{n+1}$ are not uniquely determined in the intersections of the subdomains. We will use the restricted additive Schwarz (RAS) method, in which the subsequent iterative solution is uniquely determined by $u^{n+1} = \bigcup\limits_s u_s^{n+1}$, where $u_s^{n+1} \in \Omega_s$ form a set of values of the subvector $\widetilde{u}_s^{n+1}$, which is defined in the extended subdomain $\bar{\Omega}_s$, though its nodes belong to Ω_s (for the sth subdomain, one can define a restriction operator $R_s : \bar{\Omega}_s \to \Omega_s$). The RAS method can be written in the following form:

$$u^{n+1} = u^n + B_{\text{ras}}^{-1} r^n, \quad B_{\text{ras}}^{-1} = R\widehat{A}^{-1}W^T, \quad \widehat{A} = W^T A W = \text{ block-diag } \{A_s \in \mathcal{R}^{\bar{N}_s,\bar{N}_s}\}, \quad (11)$$

where $W = [w_1 \dots w_P] \in \mathcal{R}^{N,P}$ is a rectangular matrix, and each of its columns w_s has unit components in the nodes from $\bar{\Omega}_s$ and zeros elsewhere. Note that even if the original SLAE is symmetric, the preconditioning matrix B_{ras} from (11) is in general unsymmetric. In addition, the inversion of the blocks A_s of the matrix $\widehat{A}$ actually reduces to solving decoupled subsystems in the corresponding subdomains, which is the basis for parallelization of the additive Schwarz, or the block Jacobi method.

The rate of convergence of the above iterative process depends on the number of subdomains or, more exactly, on the diameter of the graph representing the macrogrid resulting from the domain decomposition. This is due to the fact that a single iteration transmits the solution perturbation from a subdomain to the neighboring (adjacent) subdomains only. In order to speed up the iterative process, it is natural to use, at every step, not only the nearest but also some remote subdomain couplings. To this end, in decomposition algorithms, different approaches (deflation, coarse grid correction, aggregation, etc.), which are similar to some extent to the multigrid principle, as well as to low-rank approximations of matrices, are used (see the references provided at the special site [10]).

We will consider the following approach based on the interpolation principle. Let Ω_c be a coarse grid with N_c nodes, $N_c \ll N$, in the computational domain Ω; the nodes of the original and coarse grids are not necessarily the same.

By $\varphi_1, \dots, \varphi_{N_c}$ denote a set of basis interpolating polynomials of order M on the grid Ω_c. Without loss of generality, we assume that they are finite and form an expansion of unity, i.e.,

$$\sum_{k=1}^{N_c} \varphi_k(x, y) = 1.$$

In this basis, the desired solution of SLAE (5) can be represented as

$$u = \left\{ u_{i,j} \approx u^c_{i,j} = \sum_{k=1}^{N_c} c_k \varphi_k(x_i, y_j) \right\} = \Phi \widehat{u} + \psi, \tag{12}$$

where $\widehat{u} = \{c_k\} \in \mathcal{R}^{N_c}$ is the vector of coefficients of the expansion in terms of the basis functions; ψ is the approximation error, and $\Phi = [\varphi_1 \dots \varphi_{N_c}] \in \mathcal{R}^{N,N_c}$ is the rectangular matrix whose kth column, $k = 1, \dots, N$, consists of the values of the basis function $\varphi_k(x_i, y_j)$ at the nodes of the original grid Ω (most of the entries of Φ vanish because the basis is finite). The columns, or the functions φ_k can be treated as orthonormal but not necessarily. If at a certain kth node P_k of the coarse grid Ω_c only one basis function is not vanishing ($\varphi_k(P_{k'}) = \delta_{k,k'}$), then $\widehat{u}_k = c_k$ is the exact value of the desired solution at the node P_k. On substituting (12) into the original SLAE, we obtain the system

$$A\Phi\widehat{u} = f - A\psi; \tag{13}$$

premultiplying the latter by Φ^T, we have

$$\widehat{A}\widehat{u} \equiv \Phi^T A \Phi \widehat{u} = \Phi^T f - \Phi^T A \psi \equiv \widehat{f} \in \mathcal{R}^{N_c}. \tag{14}$$

In what follows, we assume that the error ψ in (12) is sufficiently small and omit it. In this way, for an approximate coarse grid solution $\breve{u}$ we obtain the system

$$\widehat{A}\breve{u} = \Phi^T f \equiv \breve{f}. \tag{15}$$

If the matrix A is nonsingular and Φ is a full-rank matrix (the rank of Φ is much less than N), which is stipulated below, then from (14) it follows that

$$u \approx \widetilde{u} = \Phi\breve{u} = \Phi\widehat{A}^{-1}\widehat{f} = B_c^{-1} f, \quad B_c^{-1} = \Phi(\Phi^T A \Phi)^{-1}\Phi^T.$$

Moreover, for the error of the approximate solution we have

$$u - \widetilde{u} = (A^{-1} - B_c^{-1})f. \tag{16}$$

The error of the approximate solution can also be represented in terms of the approximation error ψ. By subtracting Eqs. (14) and (15) term by term, we obtain

$$\widehat{A}(\widehat{u} - \breve{u}) = -\Phi^T A\psi,$$

which yields the desired equation

$$u - \widetilde{u} = \Phi\widehat{u} + \psi - \Phi\breve{u} = \psi - B_c^{-1}A\psi.$$

The matrix B_c^{-1} introduced above can be regarded as a low-rank approximation to the matrix A^{-1} and used as a preconditioner to build an iterative process. In particular, for an arbitrary vector u^{-1}, the initial guess can be chosen in accordance with the formula

$$u^0 = u^{-1} + B_c^{-1}r^{-1}, \quad r^{-1} = f - Au^{-1}. \tag{17}$$

In this case, the corresponding initial residual $r^0 = f - Au^0$ is orthogonal to the coarse-grid subspace

$$\Phi = \text{ span } \{\varphi_1, \ldots, \varphi_{N_c}\} \tag{18}$$

in the sense that

$$\Phi^T r^0 = \Phi^T(r^{-1} - A\Phi\widehat{A}^{-1}\Phi^T r^{-1}) = 0. \tag{19}$$

The relations provided in [7] form the basis for the deflated conjugate gradient method, wherein the initial direction vector is given by the formula

$$p^0 = (I - B_c^{-1}A)r^0, \tag{20}$$

implying that the following orthogonality condition is fulfilled:

$$\Phi^T A p^0 = 0. \tag{21}$$

Subsequent iterations are performed based on the relations

$$\begin{gathered} u^{n+1} = u^n + \alpha_n p^n, \quad r^{n+1} = r^n - \alpha_n Ap^n, \quad p^{n+1} = r^{n+1} + \beta_n p^n - B_c^{-1}Ar^{n+1}, \\ \alpha_n = (r^n, r^n)/(p^n, Ap^n), \quad \beta_n = (r^{n+1}, r^{n+1})/(r^n, r^n). \end{gathered} \tag{22}$$

At every step of this method, which will be referred to as the DCG, the following relations hold:

$$\Phi^T r^{n+1} = 0, \quad \Phi^T A p^{n+1} = 0. \tag{23}$$

If now we turn back to the additive Schwarz method (11), then we can attempt to accelerate it by the coarse-grid preconditioner B_c^{-1} (in addition to the preconditioner B_{ras}^{-1}). We will consider this issue in a more general context, assuming that the matrix A is unsymmetric and a few (rather than two) preconditioning matrices are available. Moreover, the preconditioners can change from iteration to iteration, which corresponds to the so-called dynamic (or flexible) preconditioning.

In order to solve a SLAE with an unsymmetric matrix A, we build a family of multi-preconditioned semiconjugate residuals methods, which are based on combining two ideas presented in [5, 6].

Let $r^0 = f - Au^0$ be the initial residual of the algebraic system, and let $B_0^{(1)}, \ldots, B_0^{(m)}$ be some nonsingular easily invertible preconditioning matrices. Using them, compose the rectangular matrix of the initial direction vectors p_k^0, $k = 1, \ldots, m$,

$$P_0 = [p_1^0 \cdots p_m^0] \in \mathcal{R}^{N,m}, \quad p_l^0 = \left(B_0^{(l)}\right)^{-1} r^0, \tag{24}$$

which are assumed to be linearly independent.

(Note that the algorithms considered can readily be extended to block iterative methods in the Krylov subspaces, in which case m different initial guesses u_l^0 rather than an only one are used. The initial direction vectors in (24) are then defined by the formulas

$$p_l^0 = \left(B_0^{(l)}\right)^{-1} r_l^0, \quad r_l^0 = f - Au_l^0, \quad l = 1, \ldots, m,$$

but we do not dwell on this any further.)

Successive approximations u^n and the corresponding residuals $r^n = f - Au^n$ are sought for with the use of the recursive relations

$$\begin{aligned} u^{n+1} &= u^n + P_n\bar{\alpha}_n = u^0 + P_0\bar{\alpha}_0 + \cdots + P_n\bar{\alpha}_n, \\ r^{n+1} &= r^n - AP_n\bar{\alpha}_n = r^0 - AP_0\bar{\alpha}_0 - \cdots - AP_n\bar{\alpha}_n. \end{aligned} \tag{25}$$

Here, $\bar{\alpha}_n = (\alpha_n^1, \ldots, \alpha_n^m)^T$ are m-dimensional vectors. The direction vectors p_l^n, which are the columns of the rectangular matrices $P_n = [P_1^n \cdots P_m^n] \in \mathcal{R}^{N,m}$, are orthogonal in the sense that they satisfy the relations

$$P_n^T A^T A P_k = D_{n,k} = 0 \quad \text{for} \quad k \neq n, \tag{26}$$

where $D_{n,n}$ is a symmetric positive-definite matrix, provided that the matrices P_k are of full rank, which is stipulated.

It is obvious that under conditions (26) the residuals satisfy the relations

$$\left(r^{n+1}, r^{n+1}\right) = \left(r^0, r^0\right) - \sum_{k=0}^{n} [2(r^0, AP_k\bar{\alpha}_k) - (AP_k\bar{\alpha}_k, AP_k\bar{\alpha}_k)]. \tag{27}$$

Then the functional minimization conditions

$$\partial\left(r^{n+1}, r^{n+1}\right)/\partial\alpha_k^{(l)} = 0, \quad k = 0, 1, \ldots, n; \quad l = 1, \ldots, m,$$

yield the following formula for the "vector coefficients" $\bar{\alpha}_n$:

$$\bar{\alpha}_n = \left(D_{n,n}^{-1}\right)^{-1} P_n^T A^T r^0. \tag{28}$$

For such values of $\bar{\alpha}_n$, as is readily verified, the vectors p_k^n and r_k^n satisfy the semiconjugation conditions

$$P_k^T A^T r^{n+1} = 0, \quad k = 0, 1, \ldots, n. \tag{29}$$

In this case, for the functionals of the residuals we have the relations

$$\begin{gathered} \left(r^{n+1}, r^{n+1}\right) = \left(r^n, r^n\right) - \left(C_n r^0, r^0\right) = \left(r^0, r^0\right) - \left(C_0 r^0, r^0\right) - \cdots - \left(C_n r^0, r^0\right), \\ C_n = P_n A D_{n,n}^{-1} A^T P_n^T. \end{gathered} \tag{30}$$

The matrices composed of the direction vectors will be computed from the recursive relations

$$P_{n+1} = Q_{n+1} + \sum_{k=0}^{n} P_k\bar{\beta}_{k,n}, \tag{31}$$

where

$$Q_{n+1} = \left[q_1^{n+1} \ldots q_m^{n+1}\right], \quad q_l^{n+1} = \left(B_{n+1}^{(l)}\right)^{-1} r^{n+1}, \tag{32}$$

are auxiliary matrices; $B_{n+1}^{(l)}$ are preconditioning matrices, and $\bar{\beta}_{k,n}$ are coefficient vectors, which are determined from (31) with account for the orthogonality conditions (26) by the formula

$$\bar{\beta}_{k,n} = -D_{k,k}^{-1} P_k^T A^T A Q_{n+1}. \tag{33}$$

By using (31) and (25), we obtain the relations

$$Q_k^T A^T r^n = \Big(P_k^T A^T - \sum_{j=0}^{k-1} \bar{\beta}_{i,k}^T P_i^T A^T\Big)\Big(r^0 - \sum_{i=0}^{k-1} A P_i \bar{\alpha}_i\Big), \tag{34}$$

implying that for $k = n$,

$$Q_n^T A^T r^n = P_n^T A^T r^0.$$

This allows us to obtain, instead of (28), the new formula

$$\bar{\alpha}_n = D_{n,n}^{-1} Q_n^T A^T r^n.$$

For $k < n$, from (34) we infer the semiconjugacy property of the residuals,

$$Q_k^T A^T r^n = 0, \quad k < n, \tag{35}$$

which justifies the name of the method under consideration.

3. Some features of parallel implementation technologies

The purpose of this paper is to verify, test, and compare the efficiency of various algorithms for solving large sparse SLAEs in order to optimize them and then incorporate into the KRYLOV library [11] of parallel algebraic solvers. The main requirements to an adequate software are its high and scalable performance, and no formal restrictions on the orders of the SLAEs to be solved and on the number of processors or computational cores used. Note that according to [8], the strong and weak scalabilities are distinguished. The first notion characterizes a decrease in the execution time for a large problem as the number of computing devices increases, whereas the second notion means that the solution time remains approximately the same as the problem dimension (the number of degrees of freedom) and the number of devices increase.

The algorithms were coded with account for the architecture of the SSCC SD RAS cluster [12] (where the KRYLOV library is available) but without using the GPGPU, as their utilization in the domain decomposition methods considered has its own technological computational complexity and needs a special study.

Computations were carried out in the following natural way: if a computational domain was divided into P subdomains, then the solution was performed on $P + 1$ CPUs (one being the root processor, and every other processor corresponding to its own subdomain), and the same number of MPI processes was formed. The auxiliary algebraic subsystems in the subdomains were solved simultaneously on the multicore CPUs with the usage of multithread OpenMP calculations.

The algorithms from the KRYLOV library are designed for solving large sparse SLAEs arising from approximation of multidimensional boundary-value problems on nonstructured grids. For this reason, the well-known compressed sparse row (CSR) format of the matrix storage is used to store the nonzero matrix entries. At the preliminary stage, the global matrix A is stored on the root processor, and then the distributed storage of the block rows $\bar{A}_s$ from (10) for the sth extended subdomains on the respective processors is used.

Note that for the two-dimensional grid boundary-value problems under consideration, a two-dimensional balanced domain decomposition into subdomains is considered, where, for approximately equal numbers of nodes $N_S \approx N/P$ in every subdomain, the macrogrid diameter d (for the macrogrid composed of the subdomains) is equal, approximately, to $\sqrt{P}$. As the number of iterations of the additive Schwarz method, even with the usage of the Krylov methods, is proportional to $d^\gamma, \gamma > 0$, this yields a significant advantage over the one-dimensional decomposition, for which $d \approx P$.

A scalable parallelization of the algorithms is provided by synchronizing computations in the subdomains by means of MPI and by minimizing the time losses during interprocessors communications. The isolated SLAEs in Ω_s are solved by a direct or an iterative method, requiring $(N/P)^{\gamma_1}, \gamma_1 > 0$, operations at every step of the two-level process. Since only the data corresponding to the peripheral nodes of adjacent subdomains must be exchanged, the volume of this information is much less and proportional to $(N/P)^{\gamma_1/2}$, allowing one to carry out arithmetic and communication operations simultaneously.

A high performance of the code based on the approach presented is ensured by an active usage of the standard functions and matrix-vector operations from BLAS and SPARSE BLAS included into MKL INTEL [9].

4. Numerical results

We present results of numerical solution of five-point SLAEs corresponding to the Dirichlet problem in a square on square grids with 128^2 and 256^2 nodes. Computations were carried out on $P = 2^2, 4^2, 8^2$ processors, each one corresponding to one of the subdomains forming a square macrogrid. Iterations over the subdomains were realized with the use of the BiCGStab algorithm [13] and the stopping criterion

$$||r^n||_2 \le 10^{-8}||f||_2.$$

The auxiliary subdomain subsystems were solved by the direct algorithm from the multi-thread program PARDISO from Intel MKL [9]. It should be emphasized that the most time-consuming part of the LU matrix decomposition was performed only once prior to iterations.

In Table 1, every cell contains the numbers of iterations over subdomains and the times of solving SLAEs on the grids of sizes 128^2 and 256^2 (two upper and lower pairs of figures, respectively); the first and third lines correspond to the zero convection, whereas the second and fourth lines correspond to the convection coefficients $p = q = 4$.

$P \setminus \Delta$	0		1		2		3		4		5	
4	18	1.75	11	1.45	9	1.37	7	1.26	7	1.26	6	1.20
	31	2.45	17	1.77	13	1.66	12	1.53	11	1.50	10	1.35
	27	6.85	16	4.37	12	3.51	10	3.02	9	2.82	8	2.49
	46	11.37	25	6.53	19	5.16	17	4.74	15	4.28	13	3.76
16	32	1.42	18	1.18	14	1.19	12	1.09	11	0.89	9	0.79
	41	2.23	25	2.6	19	2.44	16	1.90	14	1.28	14	1.78
	41	3.85	24	2.83	20	2.20	17	1.80	14	1.38	14	1.66
	58	5.96	35	3.55	28	3.03	22	2.58	19	1.99	18	1.99
64	43	1.56	26	1.66	19	1.39	16	1.50	14	1.56	12	0.86
	57	2.02	34	1.91	26	1.78	21	1.98	20	1.69	18	1.35
	60	4.75	36	4.16	27	3.35	22	3.11	20	3.00	18	4.66
	87	7.04	47	5.61	38	4.89	31	4.13	28	4.02	25	4.48

Table 1. The numbers of iterations and solution times (in seconds) on the grids of sizes 128^2 and 256^2.

The results obtained demonstrate that as the overlapping parameter Δ increases up to 5, the number of iterations reduces 3–4 times; however, for large values of Δ, the solution time increases. Thus, for almost all grids and numbers of processors, the optimal value of Δ is about 3–4. If the convection coefficients p, q have nonzero values, then the number of iterations increases approximately by 30–50%.

In the tables below, for the sake of brevity, only the results for the Poisson equation (with zero convection coefficients in Eq. (1)) are presented. As the experiments show, for moderate values of p and q ($|p| + |q| < 50$) the behavior of the iterative process varies slightly.

Table 2 presents the numbers of iterations for different values of θ from Eq. (9), which specify the interface boundary conditions for the adjacent subdomains (in every cell, the left and right figures correspond to the grids with 128^2 and 256^2 nodes, respectively).

$P \setminus \theta$	0		0.5		0.6		0.7		0.9975	
4	18	27	16	26	16	24	14	23	10	12
16	32	41	28	40	27	39	27	40	31	75
64	43	60	42	56	40	55	41	55	93	86

Table 2. The numbers of iterations on the grids with 128^2 and 256^2 nodes for different values of θ.

As can be seen from Table 2, for both grids and different numbers of subdomains, there is an optimal value of θ, close to one, but the gain is only within 10-40%. These computations were carried out without subdomain overlapping, whereas for $\Delta \geq 1$ the best value of θ is zero, which corresponds to the Dirichlet conditions on the adjacent boundaries.

The above data show that the behavior of iterations varies slightly as the initial error varies. The experiments presented above were conducted for the zero initial guess, $u^0 = 0$, and the exact SLAE solution $u = 1$.

Table 3 shows the effect of applying two deflation methods when using the unpreconditioned conjugate gradient method (without the additive Schwarz method) for three square grids with different numbers of nodes N. The basis functions $\phi_k(x, y)$ are the bilinear finite functions. In the three rightmost columns, the upper and lower numbers in every cell correspond to the single orthogonalization of the form (17), (21) and to the orthogonalization (22), (23) at every iteration, respectively. If these data are compared with the algorithm using no deflation (the column with $N_c = 0$), one can see acceleration up to three times as P increases. However, it should be taken into account that the implementation of multiple orthogonalization makes each iteration more expensive. For this reason, practical optimization of the algorithms needs additional investigation.

$N \setminus N_c$	0	2^2	4^2	8^2
64^2	176	167	166	103
		118	87	56
128^2	338	309	255	181
		220	159	104
256^2	609	544	442	276
		376	294	190

Table 3. The influence of deflation on the conjugate gradient method.

The results in Table 4 present the same data as in Table 3 but with the additive Schwarz method and domain decomposition into P subdomains. The nodes of the coarse grid Ω_c are

chosen near the subdomain corners, i.e., for $P = 2^2, 4^2, 8^2$ the numbers of the coarse grid nodes (the values of N_c) are 3^2, 5^2, and 9^2, respectively. As in the previous series of experiments from Table 3, the basis functions $\phi_k(x, y)$ are bilinear and finite. In every cell of Table 4, the number of iterations carried out without deflation and those corresponding to the single orthogonalization of the initial guess are the upper and lower ones, respectively.

$N \setminus P$	2^2	4^2	8^2
64^2	19	26	37
	23	21	28
128^2	29	35	51
	24	26	36
256^2	38	53	71
	31	35	40

Table 4. The influence of aggregation in the additive Schwarz method (decomposition without subdomain intersection).

The results presented for the grids and macrogrids considered have approximately the same character as in Table 3, namely, as the dimension of the deflation space increase, the iteration number decreases, but the amount of computations at every step grows. In these experiments, the outer iterations were the conjugate gradient iterations, which was feasible because the domain was decomposed into nonoverlapping subdomains.

Note that the experiments described in Tables 3 and 4 were carried out for the initial guess $u^0 = 0$ and the exact SLAE solution $u(x_i, y_j) = x_i^2 - y_j^2$. Naturally, the efficiency of the considered "interpolation" deflation depends on the behavior of the solution sought for. For example, if it is of the form $u(x_i, y_j) = x - y$, then the bilinear basis functions $\varphi_k(x, y)$ for $N_c \geq 4$ yield convergence in one iteration.

This work was supported by the Russian Science Foundation (project No. 14-11-00485).

Translated by Y. L. Gurieva and V. P. Il'in.

REFERENCES

1. V. P. Il'in, "Parallel methods and technologies of domain decomposition," *Vestnik YuUrGU. Ser. Vychisl. Matem. Inform.*, **46** (305), 31–44 (2012).
2. A. Toselli and O. B. Widlund, *Domain Decomposition Methods – Algorithms and Theory* (Springer Ser. Comput. Math., **34**), Springer–Verlag (2005).
3. M. Yu. Andreeva, V. P. Il'in, and E. A. Itskovich, "Two solvers for nonsymmetric SLAE," *Bull. NCC, Ser. Numer. Anal.*, **12**, 1–16 (2003).
4. V. P. Il'in, *Finite Difference and Finite Volume Methods for Elliptic Equations* [in Russian], IVMMG Publ., Novosibirsk (2001).
5. R. Bridson and C. A. Greif, "A multipreconditioned conjugate gradient algorithm," *SIAM J. Matrix Anal. Appl.*, **27**, No. 4, 1056–1068 (2006).
6. V. P. Il'in and E. A. Itskovich, "On the semiconjugate directions methods with dynamic preconditioning," *Sib. Zh. Industr. Mat.*, **10**, No. 4, 41–54 (2007)

7. A. Chapman and Y. Saad, "Deflated and augmented Krylov subspace techniques," *Numer. Linear Algebra Appl.*, **4,** No. 1, 43–66 (1997).
8. O. Dubois, M. J. Gander, S. Loisel, A. St-Cyr, and D. Szyld, "The optimized Schwarz method with a coarse grid correction," *SIAM J. Sci. Comput.*, **34,** No. 1, 421–458 (2012).
9. URL: https://software.intel.com/en-us/intel-mkl
10. URL: http://www.ddm.org
11. D. S. Butyugin, Y. L. Gurieva, V. P. Il'in, D. V. Perevozkin, A. V. Petukhov, and I. N. Skopin, "Functionality and algebraic solvers technologies in the Krylov library," *Vestnik YuUrGU. Ser. Vychisl. Matem. Inform.*, **2,** No. 3, 92–105 (2013).
12. URL: http://www2.sscc.ru
13. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publ., New York (1996).