Parallel approaches and technologies of domain decomposition methods *

Y.L.Gurieva[1)], V.P.Il'in[1,2)]

[1)] *Institute of Computational Mathematics and Mathematical Geophysics SB RAS, e-mail:yana@lapasrv.sscc.ru*

[2)] *Novosibirsk State University, e-mail:ilin@sscc.ru*

Abstract

The efficiency of two-level iterative processes in the Krylov subspaces is investigated as well as their parallelization in solving large sparse non-symmetric systems of linear algebraic equations arising from grid approximations of two-dimensional boundary value problems for diffusion-convection equations with different coefficient values. A special attention is being given to optimization of the subdomain intersection size, to the types of boundary conditions on adjacent boundaries in the domain decomposition method, and to the aggregation (or coarse grid correction) algorithms. An outer iterative process is based on the additive Swartz algorithm, while a parallel solution of subdomain algebraic systems is affected by the direct or the preconditioned Krylov method. A crucial point in a programming realization of these approaches is a 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 data. Some issues related to the scalability of parallelization are discussed.

Keywords: domain decomposition, parallel two-level methods, Krylov subspaces, preconditioning matrices, aggregation algorithms, subdomain intersections, interface conditions

1 Introduction

Creating parallel iterative algorithms to solve grid systems of linear algebraic equations (SLAEs) arizing from finite element or finite volume approximations of boundary value problems is based on decomposition of a computational domain and represents a manifold mathematical and technological problem. On the one hand, a high convergence rate of an applied iterative process should be provided, so that a large number of the algorithmic approaches exists. On the other hand, the final performance of a computational tool is largely determined by a formed data structure and program implementation of the algorithms on a multiprocessor computer system [1].

*The work is supported by Russian Science Foundation grant N 14-11-00485. The experimental segment of the paper is supported by the RFBR grant N 14-07-0128.

The objective of this paper is to experimentally investigate an impact on the parallelization scalability of three algorithmic factors: the size of intersections of the adjacent subdomains, the type of iterated boundary conditions on their inner boundaries, and application of aggregation (or a coarse grid correction) methods [2]. A rather simple but representative SLAE family was choosen as a methodical test suit, namely, five-point approximations of the linear diffusion-convection equations on a uniform grid in a rectangular domain [3], [4]. It is somewhat an idealized situation that allows one to clearly describe computating information issues arising as well as the ways to resolve them.

When solving in parallel very big sparse SLAEs with the orders of about 10^9, a two-level iterative process in the Krylov subspaces is one of the main computational tools. It uses a block Jacobi method, which is an additive Swartz method, as its preconditioner.

Section 2 contains the formulation of the initial problems and a description of the computational methods to solve them. Section 3 deals with presentation of some algorithms and technologies for "extended" subdomains, which are the basis for the domain decomposition parallelization. The results of numerical experiments for different initial data and algorithmic parameters are discussed in the last section.

2 Problem statement and description of parallel algorithms

Let the Dirichlet problem for the diffusion-convection equation

$$\begin{array}{l} -\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial^2 u}{\partial y^2} + p\dfrac{\partial u}{\partial x} + q\dfrac{\partial u}{\partial y} = f(x,y), \;\; (x,y) \in \Omega, \\ u|_\Gamma = g(x,y), \end{array} \tag{1}$$

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

The given boundary value problem is approximated on a uniform grid

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

by a five-point scheme of the form

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

where l is a "global", or continuous, number of an inner grid node:

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

A particular view of the coefficients can be different, and specific formulae versions can be found in [3], [4]. Equations (3) are written for the inner grid nodes, moreover, for the nodes near the boundary, whose numbers are from a set of the indices $i = 1, N_x$ or $j = 1, N_y$, the values known from the boundary conditions of the solution are substituted into the corresponding equations and moved to their right-hand sides, so that the corresponding coefficients $a_{l,l'}$ in (3) be equal to zero. SLAE (3) is written down in a vector-matrix 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, we will denote by Ω not only the computational domain but a set of grid nodes $(x_i, y_j) \in \Omega$ as well (we will use the term "grid computational domain"), as well as sets of the indices $l = 1, ..., N$, of vectors u, f of dimension N.

We now decompose the domain Ω, i.e. let it be at first represented as a union of identical (for simplicity) non-intersecting rectangle subdomains:

$$\Omega = \bigcup_{s=1}^{P} \Omega_s, \;\; P = P_x P_y,$$

each containing an equal number of grid nodes

$$M = m_x m_y, \;\; N_x = P_x m_x, \;\; N_y = P_y m_y, \;\; N = PM.$$

One can consider that the subdomains form a two-dimensional macrogrid, where each macrovertex can be numbered by a pair of indices p, q (similarly to the grid node indices i, j), and a "continuous" number of a subdomain is defined as

$$\begin{array}{l} s = s(p,q) \equiv p + (q-1)P_x = 1, ..., P, \\ p = 1, ..., P_x; \;\; q = 1, ..., P_y. \end{array} \tag{6}$$

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

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

where I_p and J_q are the begin numbers of the grid nodes in x and y directions in the (p, q)-th subdomain, and the global grid numbers $l(i, j)$ are calculated with the help of (4). Every subdomain Ω_s has its own four faces which form in the aggregate a boundary that does not pass through the grid nodes.

We now turn from continuous numbering of nodes to their subdomain by the subdomain ordering: at first, we number all the nodes in Ω_1, then in Ω_2, etc. The vector components u, f are ordered correspondingly, so that the original SLAE (3) takes the following block-matrix form:

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

Here $\bar{u}_s \in \mathcal{R}^{N_s}$ means a subvector of the vector u, whose components correspond to the nodes from the subdomain Ω_s, and Q_s means a set of numbers of the subdomains adjacent to the subdomain Ω_s. Hereinafter we assume that a local node ordering in every subdomain is a natural one: local pairs of indices $i' = 1, ..., m_x; \;\; j' = 1, ..., m_y$ are introduced and a 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 a reordering of the nodes from the local to the global ordering is done with the help of the values introduced in (6) according to the relations $i = i' + I_{p-1}, \;\; j = j' + J_{q-1}$.

Let us note that the formalism given above pertains to a grid domain decomposition without subdomains overlapping and without usage of separator nodes common to adjacent subdomains. However, to rise the generality and an efficiency of the algorithms discussed below it is necessary to move to constructing the "extended" subdomains with intersections.

Let ω_l denote a gird stencil or a set of nodes adjacent to the l-th node, i.e. a set of numbers of the sought for solution components involved in the corresponding l-th equation of form (3). For a grid subdomain Ω_s, let us denote by $\Gamma_s = \Gamma_s^0$ its boundary, i.e. a set of the nodes external to Ω_s such that at least one of their neighboring nodes lies in Ω_s ($\bar{\Omega}_s = \bar{\Omega}_s^0 = \Omega_s \cup \Gamma_s^0$ is a closure of the grid subdomain Ω_s). Further let Γ_s^1 denote the first extended boundary or the first external front $\bar{\Omega}_s$, i.e. a set of the nodes which do not lie in $\bar{\Omega}_s$ but have at least one neighbour node from $\bar{\Omega}_s$ ($\bar{\Omega}_s^1$ is the first extension of $\bar{\Omega}_s^0$). Let us similarly define the next subsequent stages of a grid subdomain extension and call the number of such stages Δ a parameter of the extended subdomain $\bar{\Omega}_s^\Delta = \Omega_s^\Delta \bigcup \Gamma_s^\Delta$, where the nodes from Γ_s^Δ do not belong to Ω_s^Δ whose number of nodes is denoted by $\bar{N}_s$. An illustration of an extended subdomain with the parameter $\Delta = 3$ is presented in Figure 1.

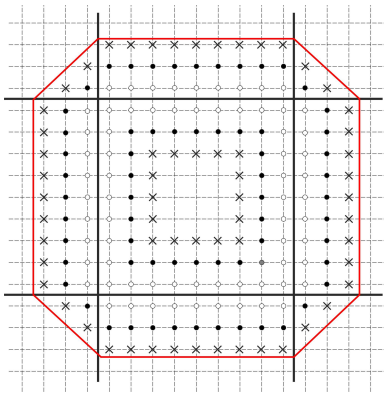

Figure 1: An example of subdomain extension

When building an iterative Swartz process in grid sundomains, it is possible to take differently into account the interface links between adjacent subdomains. Let the l-th node be a near-boundary one in the subdomain Ω_s^Δ, i.e. $l \in \Gamma_s^{\Delta-1}$. Let us write down the corresponding equation of the algebraic system in the following form:

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

Here n is iteration number and the terms which have the same coefficients are both added to the right-hand and left-hand sides of relation (9) and which contain a factor θ_l being an iterative process parameter (Figure 2). Let us note that the case $\theta_l = 0$ can be interpreted as using the Dirichlet boundary condition when solving an auxiliary subproblem in Ω_s. Similarly, the case with $\theta_l = 1$ relates to the Neumann condition, and the value $\theta_l \in (0, 1)$ — to the boundary condition of the third type (or the Robin condition).

In a matrix form, the given algorithm can be presented as a block Jacobi method

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

Here the subvectors $\tilde{u}_s^n$ and $\tilde{f}_s^n$ are from the extended subdomains and have the dimensions $\bar{N}_s$, and $\bar{B}_s \in \mathcal{R}^{\bar{N}_s, \bar{N}_s}$ are the preconditioning matrices whose diagonal entries depend on the values of the parameters θ_l.

The iterative process, presented in the form of (10), is underdetermined as unknowns of $\tilde{u}_s^{n+1}$ have non-unique values in the intersections of the subdomains. We will use a

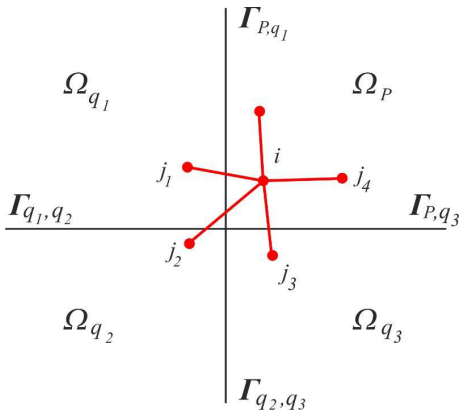

Figure 2: A grid stencil of a near-boundary node

restricted additive Swartz (RAS) method when the next iterative solution is uniquely defined as $u^{n+1} = \bigcup\limits_s u_s^{n+1}$, where $u_s^{n+1} \in \Omega_s$ are a set of values of subvector $\tilde{u}_s^{n+1}$, which is defined in the extended subdomain $\bar{\Omega}_s$ but whose nodes belong to Ω_s (for the s-th subdomain, a restriction operator R_s can be defined as $R_s : \bar{\Omega}_s \to \Omega_s$). The RAS method can be written in the following form

$$\begin{aligned} &u^{n+1} = u^n + B_{ras}^{-1} r^n, \\ &B_{ras}^{-1} = R\hat{A}^{-1}W^T, \;\; \hat{A} = W^T A W = \text{ block-diag } \{A_s \in \mathcal{R}^{\bar{N}_s,\bar{N}_s}\}, \end{aligned} \tag{11}$$

$W = [w_1 ... w_P] \in \mathcal{R}^{N,P}$ being a rectengular matrix, whose each column w_s has ones in the nodes from $\bar{\Omega}_s$ and has zeros otherwise. Let us note that generally even if the original SLAE is symmetric, a preconditioning matrix B_{ras} from (11) is not a symmetric one. In addition, the inversion of the blocks A_s of the matrix $\hat{A}$ is actually reduced to the solution of independent subsystems in the corresponding subdomains which is the basis for parallelization of the additive Swartz or the block Jacobi method.

The rate of convergence of the given iterative process depends on the number of the subdomains, or more precisely, on the diameter of a graph representing a macrogrid formed by decomposition. This can be clearly explained by the fact that by a single iteration the solution perturbation in one subdomain is transmitted only to neighbouring, or adjacent, subdomains. To speed up the iterative process, it is natural to use at every step not only the nearest but also the remote subdomain couplings. For this purpose, different approaches are used in decomposition algorithms: methods of deflation, coarse grid correction, aggregation, etc., which to some extent are close to the multigrid principle as well as the low-rank approximations of matrices, see numerous publications cited at a special site [10].

We will consider the following approach based on an interpolation principle. Let Ω_c be a coarse grid with the number of nodes $N_c \ll N$ in the computational domain Ω, moreover, the nodes of the original grid and the coarse grid may not match.

Let us denote by $\varphi_1, ..., \varphi_{N_c}$ a set of basis interpolating polynomials of order M on the grid Ω_c which are supposed to be finite and without loss of generality forming an expansion of the unit, i.e.

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

Then a sought for solution vector of SLAE (5) can be represented in the form of an expansion in terms of the given basis:

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

where $\hat{u} = \{c_k\} \in \mathcal{R}^{N_c}$ is a vector of the coefficients of the expansion in terms of the basis functions, ψ is an approximation error, and $\Phi = [\varphi_1 ... \varphi_{N_c}] \in \mathcal{R}^{N,N_c}$ is a rectangular matrix with every k-th column consisting of the values of the basis function $\varphi_k(x_i, y_j)$ at the nodes of the original grid Ω (most of the entris of Φ equal zero in virtue of the finiteness of the basis). The columns, or the functions φ_k, can be treated to be orthonormal but not necessarily. If at some k-th node P_k of the coarse grid Ω_c only one basis function is a nonzero one ($\varphi_k(P_{k'}) = \delta_{k,k'}$), then $\hat{u}_k = c_k$ is the exact value of the sought for solution at the node P_k. With substitution of (12) into the original SLAE, one can obtain the system

$$A\Phi\hat{u} = f - A\psi, \tag{13}$$

and if to multiply it by Φ^T one can obtain

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

Assuming further that the error ψ in (12) is sufficiently small and omitting it, one can obtain a system for an approximate coarse grid solution $\check{u}$:

$$\hat{A}\check{u} = \Phi^T f \equiv \check{f}. \tag{15}$$

If the matrix A is a non-singular matrix and Φ is the full-rank matrix(the rank is much less than N), we assume these facts to hold further, then from (14) we have

$$u \approx \tilde{u} = \Phi\check{u} = \Phi\hat{A}^{-1}\hat{f} = B_c^{-1} f, B_c^{-1} = \Phi(\Phi^T A\Phi)^{-1}\Phi^T,$$

moreover for the error of the approximate solution we have

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

The error of the approximate solution can also be presented via the error of the approximation ψ. Subtracting equations (14) and (15) term by term we have

$$\hat{A}(\hat{u} - \check{u}) = -\Phi^T A\psi$$

that yields the required equation:

$$u - \tilde{u} = \Phi\hat{u} + \psi - \Phi\check{u} = \psi - B_c^{-1} A\psi.$$

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} we can choose an initial guess as

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

In doing so, the corresponding initial residual $r^0 = f - Au^0$ will be orthogonal to a coarse grid subspace

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

in the sense of fulfilling the condition

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

The relations given in [7] are the basis for the conjugate gradient method with deflation, wherein an initial direction vector is chosen by the formula

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

which ensures that the following orthogonality condition holds:

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

Further iterations are implemented using the relations:

$$\begin{array}{l} u^{n+1} = u^n + \alpha_n p^n, \;\; r^{n+1} = r^n - \alpha_n A p^n, \\ p^{n+1} = r^{n+1} + \beta_n p^n - B_c^{-1} A r^{n+1}, \\ \alpha_n = (r^n, r^n)/(p^n, Ap^n), \;\; \beta_n = (r^{n+1}, r^{n+1})/(r^n, r^n). \end{array} \tag{22}$$

In this method, which we will refer to as DCG, at every step the following relations hold:

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

If now we turn back to the additive Swartz method (11), we can try to accelerate it by the coarse grid preconditioner B_c^{-1} (in addition to the preconditioner B_{ras}^{-1}). We will consider this point in a more general formulation assuming that matrix A is a non-symmetric one and that there are several but not only two preconditioning matrices. Moreover, the preconditioners can change from iteration to iteration what corresponds to the so-called dynamic or flexible preconditioning.

To solve a SLAE with a non-symmetric matrix A, let us build a family of multi-preconditioned semi-conjugate residuals, which are based on a union of two ideas presented in [5], [6].

Let $r^0 = f - Au^0$ be an initial residual of the algebraic syatem, and $B_0^{(1)}, ..., B_0^{(m)}$ – be a set of some non-singular easily inversible preconditioning matrices. Using them, let us define a rectangular matrix composed of the initial direction vectors $p_k^0, \;\; k = 1, ..., m$:

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

which are assumed to be linearly independent.

Let us note that it is easy to generalize the considered algorithms to the block iterative methods in the Krylov subspaces when not a single but m different initial guesses u_l^0 are taken. Then the initial direction vectors in (24) can be defined as

$$p_l^0 = (B_0^{(l)})^{-1} r_l^0, \;\; r_l^0 = f - Au_l^0, \;\; l = 1, ..., m,$$

however we will not dwell further on this issue.

Successive approximations u^n and the corresponding residuals $r^n = f - Au^n$ will be sought for with the help of the recursions

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

Here $\bar{\alpha}_n = (\alpha_n^1, ..., \alpha_n^m)^T$ are m-dimensional vectors. The direction vectors p_l^n forming the columns of the rectangular matrices $P_n = [P_1^n \cdots P_m^n] \in \mathcal{R}^{N,m}$ will be defined as orthogonal ones in the sence of satisfying 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 if the matrices P_k have the full rank as is supposed.

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

$$\begin{array}{l} (r^{n+1}, r^{n+1}) = (r^0, r^0) - \\ - \sum\limits_{k=0}^{n} [2(r^0, AP_k\bar{\alpha}_k) - (AP_k\bar{\alpha}_k, AP_k\bar{\alpha}_k)]. \end{array} \tag{27}$$

From here it follows that under the condition of a minimum of the functional

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

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

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

For such values of $\bar{\alpha}_n$ it is easy to check that the vectors p_k^n, r_k^n satisfy the semi-conjugation condition, i.e.

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

In this case, the following relations are valid for the functionals of the residuals:

$$\begin{array}{l} (r^{n+1}, r^{n+1}) = (r^n, r^n) - (C_n r^0, r^0) = \\ (r^0, r^0) - (C_0 r^0, r^0) - ... - (C_n r^0, r^0), \;\; C_n = P_n A D_{n,n}^{-1} A^T P_n^T. \end{array} \tag{30}$$

We will look for the matrices composed of the direction vectors from the recurrent relations

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

where the auxiliary matrices

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

are introduced, $B_{n+1}^{(l)}$ are the preconditioning matrices and $\bar{\beta}_{k,n}$ are the coefficient vectors which are defined after substitution of (31) into orthogonality conditions (26) by the formula

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

Let us consider the relations

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

with the help of (31) and (25). It follows from these relations that if $k = n$, the equality

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

holds thus allowing us to obtain a new formula instead of (28):

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

And if $k < n$, the property of semiconjugacy of the residuals follows from (34):

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

which gave the name of the method under considerstion.

3 Some features of parallel implementation technologies

The objectives of this paper are: verification, testing, and a comparative analysis of the efficiency of different algorithms of solving big sparse SLAEs aimed at their optimization and including into the KRYLOV library [11] of parallel algebraic solvers. The main requirements to develop a proper software are high and scalable performance and no formal restrictions on the orders of the SLAEs to be solved and on the number of the processors or computational cores used. Let us note that according to [8] a strong and a weak scalability can be distinguished. The first notion describes a decrease in the execution time of one big problem with an increase of the number of computing devices, while the second notion stands for approximate preservation of the solution time, while increasing the dimension (the number of degrees of freedom) of the problem and the number of devices.

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

Computations are carried out in the following natural way: if a computational domain is divided into P subdomains than the solution is performed on $P + 1$ CPUs (one is the root processor and any of the rest processors corresponds to its own subdomain), and the same number of MPI processes are formed as well. The solutions to auxiliary algebraic subsystems in the subdomains are simultaneously obtained on the multicore CPUs with the usage of multithread OpenMP calculations.

As algorithms from KRYLOV library are designed to solve big sparse SLAEs arising from an approximation of multidimensional boundary value problems on non-structured grids, then the well-known compressed sparse row (CSR) format of the matrix storage is used to keep non-zero matrix entries. In doing so, the global matrix A is formed on the root processor at the preliminary stage, and then the distributed storage of the block rows $\bar{A}_s$ from (10) is performed for the s-th extended subdomains in the respective processors.

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

A Scalable parallelization of the algorithms is provided by synchronization of the calculations in subdomains by means of MPI and by minimization of the time losses during interprocessors communication. A solution to the isolated SLAEs in Ω_s is produced by the direct or iterative method requiring $(N/P)^{\gamma_1}, \gamma_1 > 0$ operations at every step of the two-level process. As it is necessary to exchange the data corresponding to peripheral nodes of the adjacent subdomains only, the volume of such an information is much less and proportional to $(N/P)^{\gamma_1/2}$ thus allowing one to carry out arithmetic and communication operations simultaneously.

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

4 Numerical results

We present the results of methodical experiments on solving five-point SLAEs for the Dirichlet problem in a square on the square grids with the number of nodes 128^2 and 256^2. Calculations were carried out on $P = 2^2, 4^2, 8^2$ processors each of which corresponding to one of subdomains forming the square macrigrid. Iterations over the subdomains were reslised with the help of BiCGStab algorithm [13] with the stopping criterion

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

Solving of the auxiliary subdomain subsystems was carried out by the direct algorithm embodied in multithread program PARDISO from Intel MKL. Moreover, the most time-consuming part of LU matrix decomposition was done only once before the iterations.

In the Table 1, each cell contains the numbers of iterations over the subdomains and the times of SLAEs solving on the grids 128^2 and 256^2 (two upper and bottom pairs of figures respectively), moreover the first and the third lines correspond to the zero convection while the second and the fourth lines – to the convection coefficients $p = q = 4$) according to the different values of overlapping parameter Δ.

The results demonstrate that with Δ increasing up to 5 the number of the iterations reduces 3 - 4 fold, but when the overlapping value is big the time of subdomain solving begins to increase. So for almost all grids and the numbers of processors, the optimal Δ value is approximately 3 – 4. If the convection coefficients p, q have nonzero values, the number of the iterations increases by approximately 30–50%.

In the Tables below, for the sake of brevity, the results for the Poisson equation only are presented, i.e. when there are no convection coefficients in equation (1). As the experiments show, with the moderate values of p, q ($|p| + |q| < 50$) the behavior of the iterative process varies slightly.

The Table 2 presents the numbers of the iterations for different values of θ from the equation (9), which define the interface boundary conditions for the adjacent subdomains (in each cell, the left and the right fugures are for the grids 128^2 and 256^2 respectively).

As one can see from these data, the given data for different grids and the numbers of the subdomains contain the optimal value of θ close to one, but the gain is only within 10-40%. These calculations were carried out without subdomain overlapping and for $\Delta \geq 1$ the best θ value is zero what corresponds to the Dirichlet conditions on the adjacent boundaries.

$P \setminus \Delta$	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 the solution times (in seconds) on the grids 128^2 and 256^2

$P \setminus \theta$	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 number of the iterations on the grids 128^2 and 256^2 for different θ values

The data above show that the behavior of iterations varies slightly when the initial error varies. The experiments given above were hold for the initial guess $u^0 = 0$ and the exact SLAE solution $u = 1$.

Table 3 shows the effect of applying two deflation methods when the conjugate gradient method without any additional preconditioning is used, and without additive Schwarz method for three square grids with different numbers of nodes N including. The basis functions $\phi_k(x, y)$ were the bilinear finite functions. Three right columns have the number of the iterations for the single orthogonalization of the form (17), (21) as the upper figure in each cell while the iteration number for the orthogonalization (22), (23) on every iteration is the bottom figure. If to compare these data with the algorithm when the deflation is not used at all (the column with $N_c = 0$) one can see the acceleration up to three times when P increases. However, it should be taken into account that an implementation of multiple orthogonalization makes each iteration more expensive, so an additional investigation is required to optimize the algirithms practically.

The results from Table 4 present the same data but when using the additive Swartz method with the domain decomposition into P subdomains. The coarse grid Ω_c nodes are taken in the vicinity of the subdomain corners, i.e. when $P = 2^2, 4^2, 8^2$ the numbers of the coarse grid nodes or the values of N_c are $3^2, 5^2$ and 9^2 respectively. THe basis functions $\phi_k(x, y)$ as in the previous series of experiments from Table 3 were the bilinear finite ones. Every cell of Table 4 contains the numbers of the iterations carried out without deflation as the upper figure and the numbers of the iterations for the single orthogonalization of

$N \setminus N_c$	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 deflation influence in the conjugate gradient mathod

the initial guess as the bottom figure.

$N \setminus P$	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. Aggregation influence in additive Swartz method (the decomposition without subdomain intersection)

The presented results for the considered grids and macrogrids have approximately the same character as in Table 3 when the increasing of the deflation space yields to the decreasing of the iteration number together with the increasing of the amount of computations at each step. In these experiments, the outer iterations were carried out by the conjugate gradient method, wherein it was possible to apply it because the decomposition was done without the subdomain intersections.

Let us note that the experiments whose data are presented in Tables 3, 4 were hold 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 behaviour of the solution sought for. For example, if is of the form $u(x_i, y_j) = x - y$, then the usage of the bilinear basis functions $\varphi_k(x, y)$ for $N_c \geq 4$ yields to the convergence in one iteration.

References

[1] Il'in V.P. Parallel methods and technologies of domain decomposition. – Vestnik YuUrGU. Series "Computational mathematics and informatics", N 46(305), 2012, 31-44.

[2] Toselli A., Widlund O. Domain Decomposition Methods – Algorithms and Theory. Springer Series in Comput. Math., v. 34, 2005.

[3] Andreeva M.Yu., Il'in V.P., Itskovich E.A. Two solvers for nonsymmetric SLAE.–Bull. NCC, series: Num. Anal., iss. 12, 2003, 1-16.

[4] Il'in V.P. Finite difference and finite volume methods for elliptic equations.– Novosibirsk, ICMMG Publisher, 2001, 318 p.

[5] Bridson R., Greif C. A multipreconditioned conjugate gradient algorithm.–SIAM J. Matrix Anal. Appl., v. 27, N 4, 2006, 1056-1068.

[6] Il'in V.P., Itskovich E.A. On semi-conjugate directions mathod with a dynamic presonditioning. – Novosibirsk, SibJIM, v. 10, N 4, 2007, 41-54.

[7] Chapman A., Saad Y. Deflated and augmented Krylov subspace technique. Numer. Linear Algebra Applic., v. 4, N 1, 1997, 43-66.

[8] Dubois O., Gander M.J., St-Cyr A., Loisel S., Szyld D. The Optimized Schwarz Method with a Coarse Grid Correction.–SIAM Journal on Scientific Computing, v. 34, N 1, 2012, 421-458.

[9] URL:https://software.intel.com/en-us/intel-mkl

[10] URL:http://www.ddm.org

[11] Butyugin D.S., Gurieva Y.L., Il'in V.P., Perevozkin D.V., Petukhov A.V. Functionality and algebraic solvers technologies in Krylov library.— Vestnik YuUrGU. Series "Computational mathematics and informatics", v. 2, N 3, 2013, 92-105.

[12] URL:http://www2.sscc.ru

[13] Saad Y. Iterative Methods for Sparse Linear Systems.–N.Y.:PWS Publ., 2002.