Parallel Domain Decomposition Methods with Graph Preconditioning

Y.L. Gurieva[1], V.P. Il'in[1], and D.I. Kozlov[1,2]

[1)] Institute of Computational Mathematics and Mathematical Geophysics SB RAS,
[2)] Novosibirsk State University
yana@lapasrv.sscc.ru, ilin@sscc.ru, di_kozlov@sscc.ru

Abstract. To solve symmetric positive definite SLAEs with large sparse matrices that arise under approximation of two-dimensional and three-dimensional problems for differential equations of the second order on regular structured grid, additive methods of domain decomposition without intersection of subdomains are considered. The algorithm is constructed by defining a subset of the separator nodes between the subdomains, which forms the macrogrid and formally is taken as a special subdomain. A general iterative process is described as a block incomplete factorization method in the Krylov subspaces using the compensation principle. In the two-dimensional case, algebraic system for a macrogrid is solved by an economical direct method using parallelizable sweeps. In other subdomains a scalable parallelization is achieved by solving auxiliary SLAEs by direct or iterative algorithms that are implemented by means of hybrid programming on cluster architectures with distributed and shared memory. For 3D problems, this "two-dimensional" algorithm serves as a part of a three-level computational scheme for solving algebraic systems on separating the grid faces. Estimates of the efficiency of parallelization of the proposed algorithms, features of the generated data structures and a possibility of a minimization the volume of resource-intensive and energy-consuming communications perations are given.

Keywords: parallel domain decomposition · iterative processes · Krylov subspaces · performance · graph preconditioners

1 Introduction

Domain decomposition algorithms have a rich history, beginning with the alternating Schwarz method applied to solving and investigating multidimensional boundary value problems for partial differential equations. These equations were formulated for computational domains with a complex geometry and different types of boundary conditions. This research direction became a direction of current interest and was significantly updated and intensified at the end of the last century after the development of multiprocessor computing systems (MCSs) with wide opportunities for parallelizing algorithms.

The additive domain decomposition method (DDM), being historically a development of the Schwarz method, proved to be an effective tool for synchronization of arithmetic operations while solving the auxiliary problems in subdomains for computations on modern MCSs of a heterogeneous architecture with distributed and hierarchical shared memory. Numerous monographs and articles on the ADD topic are available. They are focuses on ADD with a parameterized intersection of subdomains, with different iteration conditions on internal boundaries, with various approaches to the construction of iteration processes and with numerous applications. Regular international conferences and an extensive bibliography (see ddm.org), as well as works [1] – [22] and the literature referenced there, are devoted to domain decomposition. Here we highlight the studies [23] – [26] based on P. Vaidya's idea of constructing an easily reversible preconditioning matrix by building a spanning tree, based on graph interpretation of grid systems of linear algebraic equations (SLAEs), although, strictly speaking, this approach differs in methodology from the principle of domain decomposition that we use.

In this paper, we consider an additive decomposition method without intersections of subdomains, in which separating nodes form a "macrogrid" and formally constitute their own subdomain whose SLAE is economically solved using parallel sweep algorithms see [27] – [29]. The algebraic systems to be solved are assumed to be symmetric positively determined (s.p.d.) and they are derived from some approximations of two- or three-dimensional boundary value problems for the second order partial derivative equations on structured or unstructured grids. SLAEs are solved by a conjugate directions method using an additive domain decomposition method (without intersection of subdomains) as a preconditioner which algebraically corresponds to the large-block algorithm of incomplete factorization using the compensation principle.

This paper is structured as follows. Section 2 is devoted to the description of the algebraic structures for decomposition method of grid domains with separating nodes (for the two- and three-dimensional cases). Section 3 describes the proposed variant of a parallel domain decomposition method in the Krylov subspaces based on utilizing large-block incomplete factorization and the compensation principle to form the interface conditions between subdomains. Section 4 is devoted to the presentation of a parallel direct algorithm to solve SLAEs on macrographs. Section 5 presents the results of preliminary investigations on performance and efficiency estimations of algorithm parallelizing. Finally, the issues of generalization of algorithms to solve a wider class of problems are discussed.

2 Algebraic structures in domain decomposition with separating nodes

We consider a SLAE with s.p.d. matrices of the form

$$Au = f, \quad A = \{a_{l,m}\} \in \mathbb{R}^{N,N}, \quad u = \{u_l\}, \quad f = \{f_l\} \in \mathbb{R}^N, \tag{1}$$

obtained from the discretization of multidimensional boundary value problems using compact approximation schemes of a node type. This means, first, that each vector component u_l and f_l in (1) corresponds to its own grid node, and second, that each grid equation contains only terms corresponding to the "nearest" neighboring nodes. The last condition means that in the one-dimensional case only three-point schemes are allowed, i.e., terms with the components $u_{l\pm\Delta}, |\Delta| \geqslant 2$ cannot be present in the l-th equation.

2.1 Classification of variables for 2D case

Let us illustrate the classification of grid nodes and the corresponding block structure of SLAEs with a simple example. Let $\Omega = [x_0, x_{N_x+1}] \times [y_0, y_{N_y+1}]$ be a computational rectangular domain with the rectangular grid

$$\Omega^h : x = x_i, \quad i = 1, \ldots, N_x, \quad y = y_j, \quad j = 1, \ldots, N_y.$$

On this grid, let us consider a system of $N = N_x N_y$ five-point equations

$$(Au)_{i,j} = a^{(0)}_{i,j} u_{i,j} - a^{(1)}_{i,j} u_{i-1,j} - a^{(2)}_{i,j} u_{i,j-1} - a^{(3)}_{i,j} u_{i+1,j} - a^{(4)}_{i,j} u_{i,j+1} = f_{i,j},$$
$$i, j \in \Omega^h, \quad a^{(1)}_{1,j} = a^{(2)}_{i,1} = a^{(3)}_{N_x,j} = a^{(4)}_{i,N_y} = 0, \quad (2)$$

approximating some boundary value problem in $\bar{\Omega} \in \Omega \cup \Gamma$ with Dirichlet boundary conditions, for example, for a second order elliptic differential equation. Let us introduce separating coordinate lines in Ω^h

$$x_{i_1}, \ldots, x_{i_{M_x}}; \quad y_{j_1}, \ldots, y_{j_{M_y}}; \quad 1 < i_1 < i_{M_x} \leqslant N_x; \quad 1 < j_1 < j_{M_y} \leqslant N_y.$$

They form a macrogrid, or a macrograph, including macronodes, or macrovertexes, and macroedge nodes. The remaining nodes of the grid represent the inner nodes of the formed subdomains $\Omega^h_1, \ldots, \Omega^h_M, M = (M_x + 1)(M_y + 1)$. It is easy to calculate that on such a macrogrid the number of macronodes is $M_v = M_x M_y$, and the number of macroedges is $K = M_x + M_y + 2M_x M_y$.

An example of a decomposition of the grid computational domain for a square grid is shown in Fig. 1, where the symbols $\bullet$, $\times$, $\circ$ denote macronodes, macroedge nodes and internal nodes in subdomains, respectively. Hence, it is obvious that the compactness condition for the solved grid equations can be formulated as follows: the equation for an internal node from the subdomain $\Omega_s, s = 1, \ldots, M$, cannot contain terms with unknown variables corresponding to other subdomains. Note that each equation for a macronode contains four terms corresponding to the outermost nodes of the adacent macroedges, and each "macroedge" equation includes two terms corresponding to the internal nodes from different adjacent subdomains.

Here we denote by $\hat{u} = \{\hat{u}_k, k = 1, \ldots, K\}$ and $\check{u} = \{\check{u}_m, m = 1, \ldots, M_v\}$ the subvectors with the components corresponding to grid variables defined on macroedges (here $\hat{u}_k$ is a subvector corresponding to one macroedge) and macronodes, and introduce one big (defined on the whole macrogrid) subvector

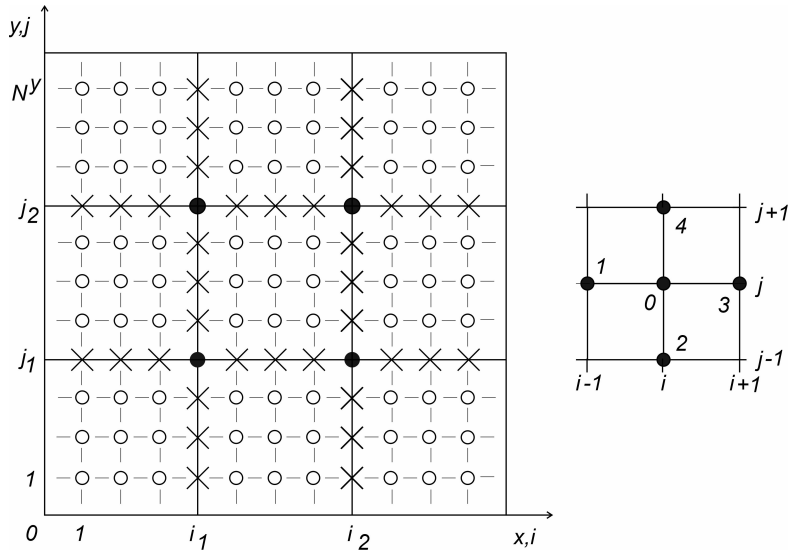

Fig. 1. An example of decomposition of a grid area with separators in a 2D problem

$\bar{u}_1 = \left(\hat{u}^\top, \check{u}^\top\right)^\top$ and subvector $\bar{u}_2 = \{\bar{u}_{2,s}, s = 1, \ldots, M\}$ with the components corresponding to all Ω_s subdomains. Then the original algebraic system (1) can be written in the following form:

$$\bar{A}\bar{u} = \begin{bmatrix} \bar{A}_{1,1} & \bar{A}_{1,2} \\ \bar{A}_{2,1} & \bar{A}_{2,2} \end{bmatrix} \begin{bmatrix} \bar{u}_1 \\ \bar{u}_2 \end{bmatrix} = \begin{bmatrix} \bar{f}_1 \\ \bar{f}_2 \end{bmatrix}. \tag{3}$$

Here the matrix $\bar{A}$ differs from A only by the permutation of rows and columns, and $\bar{A}_{2,2}$ = block-diag $\{D_s, s = 1, \ldots, M\}$, where D_s are five-diagonal matrix blocks, each of which characterizes an autonomous SLAE in its "own" subdomain. Note that if each macroedge contains (for simplicity) N_e nodes, then the total numbers of macrogrid nodes and the internal nodes in subdomains equal

$$\bar{N}_1 = M_x M_y + K N_e, \quad \bar{N}_2 = (M_x + 1)(M_y + 1)N_e^2 = M N_e^2. \tag{4}$$

The block structure $\bar{A}_{1,1}$ in (3) has the following characteristic arrowhead representation:

$$\bar{A}_{1,1} = \left(\begin{array}{cccccccccc|c}
T_1^x & . & . & . & 0 & 0 & . & . & . & 0 & C_1^x \\
. & . & & & . & . & & & & . & . \\
. & & . & & . & . & & & & . & . \\
. & & & . & . & . & & & & . & . \\
0 & . & . & . & T_{M_x}^x & 0 & . & . & . & 0 & C_{M_x}^x \\
\hline
0 & . & . & . & 0 & T_1^y & . & . & . & 0 & C_1^y \\
. & & & & . & . & . & & & . & . \\
. & & & & . & . & & . & & . & . \\
. & & & & . & . & & & . & . & . \\
0 & . & . & . & 0 & 0 & . & . & . & T_{M_y}^y & C_{M_y}^y \\
\hline
(C_1^x)^\top & . & . & . & (C_{M_x}^x)^\top & (C_1^y)^\top & . & . & . & (C_{M_y}^y) & S
\end{array}\right) \quad (5)$$

It is assumed here that the nodes of the horizontal macroedges are numbered first, then the vertical ones, and then macronodes. All diagonal blocks $T_{k'}^x$ and $T_{k''}^y$ $(1 \leqslant k' \leqslant M_x, 1 \leqslant k'' \leqslant M_y)$ are tridiagonal matrices, and S is a diagonal block. The off-diagonal blocks $C_{k'}^x, C_{k''}^y$ have a rectangular shape and one nonzero element in their first and last rows (more precisely, some of them, which are near-boundary, are zero). All last M_v rows of matrix $\bar{A}$ contain four off-diagonal elements.

2.2 Matrix structures for decomposition of 3D problems

In this subsection, as in the previous one, we restrict ourselves to the consideration of a methodological problem, assuming for simplicity that the computational domain is a parallelepiped $\Omega = [x_0, x_{N_x+1}] \times [y_0, y_{N_y+1}] \times [z_0, z_{N_z+1}]$, which is discretized using a regular grid $\Omega^h : x = x_i, i = 1, \ldots, N_x; y = y_j, j = 1, \ldots, N_y; z = z_k, k = 1, \ldots, N_z$.

We will consider grid seven-point equations, which are a natural generalization of five-point SLAEs (2), defined on a stencil shown in Fig. 2. We assume that the Dirichlet boundary conditions are set on the boundary Γ of the computational domain $\bar{\Omega} = \Omega \cup \Gamma$.

In the grid area Ω^h, we introduce the dividing planes with coordinates $x = x_{i_1}, \ldots, x_{i_{M_x}}; y = y_{j_1}, \ldots, y_{j_{M_y}}; z = z_{k_1}, \ldots, z_{k_{M_z}}$ forming a macrogrid, a fragment of which is shown in Fig. 3. Here the symbols $\bullet$, $\times$, $\circ$ denote macronodes, macroedge and macroface nodes, respectively. If the subvectors of unknown and known vectors defined on the macrogrid and in the internal nodes of the subdomains formed with its help are denoted by $\bar{u}_1, \bar{u}_2, \bar{f}_1, \bar{f}_2$, then the original SLAE (1) is written similarly to (3) in the block form.

In this case, the block $\bar{A}_{2,2}$ is, as in (3), a block-diagonal matrix, but now each of its $M = (M_x + 1)(M_y + 1)(M_z + 1)$ blocks is a seven-diagonal matrix

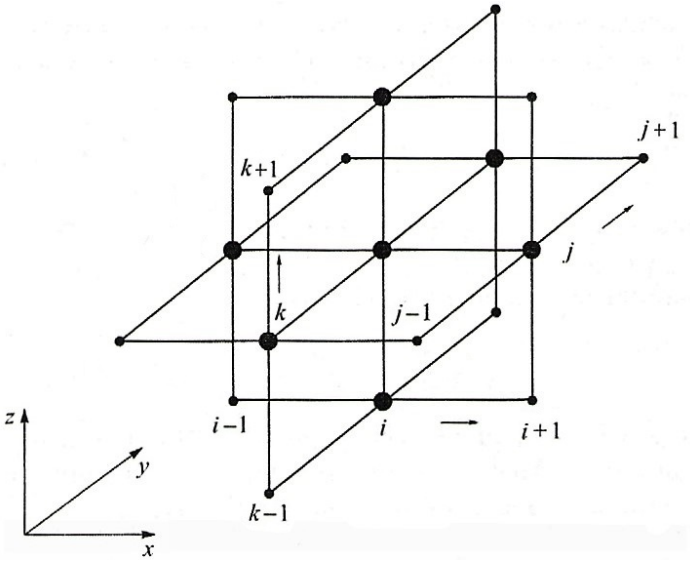

Fig. 2. Grid 3D seven-point stencil

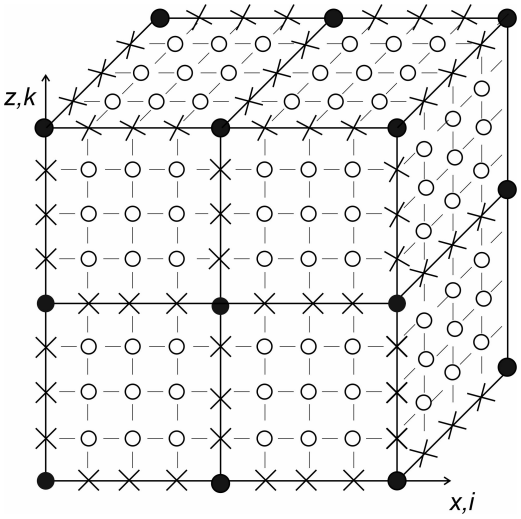

Fig. 3. A decomposition fragment of a 3D grid domain

obtained from the approximation of the three-dimensional boundary value problem in the corresponding subdomain. At the same time, the matrix block $\bar{A}_{1,1}$ has a more complex structure than in the two-dimensional version, since the three-dimensional separating macrogrid contains, in addition to macrovertices and macroedges, the macroface nodes. Thus, the entire block (5) corresponds to the matrix structure. Generally speaking, a "3D Macrogrid" block $\bar{A}_{1,1}$ can also be represented as an arrow-like matrix (5), but now the diagonal T-blocks correspond to macrofaces and hence have the block structure (5) exactly. And macroedge nodes and macrovertices now form a macrograph, which, unlike the 2D case, is not flat but three-dimensional. The total number of macrovertices here is equal to $M_v = M_x M_y M_z$, and each of them is incident to six macroedges.

3 General scheme of the iterative domain decomposition method in the Krylov subspaces

A universal form of the additive decomposition algorithm for a symmetric SLAE is the preconditioned method of conjugate directions. To construct a preconditioning symmetric matrix, we use an approximation of the matrix $\bar{A}$ from (3) obtained with the help of incomplete factorization [16] and the compensation principle:

$$\begin{gathered} B = \begin{bmatrix} G_1 & 0 \\ \bar{A}_{2,1} & G_2 \end{bmatrix} G^{-1} \begin{bmatrix} G_1 & \bar{A}_{1,2} \\ 0 & G_2 \end{bmatrix}, \\ G_1 = \bar{A}_{1,1} - \theta_1 S_1, \quad G_2 = \bar{A}_{2,2} - \overline{\bar{A}_{2,1} G_1^{-1} \bar{A}_{1,2}} - \theta_2 S_2, \\ S_1 e = \bar{A}_{1,2} e, \quad S_2 e = \left(\bar{A}_{2,1} G_1^{-1} \bar{A}_{1,2} - \overline{\bar{A}_{2,1} G_1^{-1} \bar{A}_{1,2}} \right) e, \end{gathered} \tag{6}$$

where a bar over a matrix means its approximation, $\theta_1, \theta_2 \in [0,1]$ are iterative (compensating) parameters, S_1, S_2 are diagonal matrices, and e is a trial vector with unit entries. Specific matrix approximations will be made using band representations, depending on the dimension of the problem. Having an arbitrary initial approximation $u^0 = \left((\bar{u}_1^0)^\top, (\bar{u}_2^0)^\top \right)^\top$ to the solution of the SLAE (3), we utilize the preconditioned conjugate gradient method described by the following formulas:

$$\begin{gathered} r^0 = f - \bar{A} u^0, \quad p^0 = B^{-1} r^n, \quad n = 0, 1, \cdots : \\ u^{n+1} = u^n + \alpha_n p^n, \quad r^{n+1} = r^n - \alpha_n \bar{A} p^n, \\ p^{n+1} = B^{-1} r^{n+1} + \beta_n p^n, \quad \alpha_n = \sigma_n / \rho_n, \\ \beta_n = \sigma_{n+1} / \sigma_n, \quad \sigma_n = (r^n, B^{-1} r^n), \quad \rho_n = (p^0, \bar{A} p^n), \end{gathered} \tag{7}$$

in which each vector, like u^0, consists of two subvectors, while the first one corresponds to macrogrid (separating) variables. At each iteration, to calculate the vector $q^{n+1} = B^{-1} r^{n+1}$, one need to solve an auxiliary SLAE with a preconditioning matrix B, which when using subvectors

$$r_1^n = \bar{f}_1 - \bar{A}_{1,1} u_1^n - \bar{A}_{1,2} \bar{u}_2^n, \quad r_2^n = \bar{f}_2 - \bar{A}_{2,1} u_1^n - \bar{A}_{2,2} \bar{u}_2^n$$

is implemented in block form as follows:

$$G_1 v_1^n = r_1^{n+1}, \quad G_2 v_2^n = r_2^{n+1} - \bar{A}_{2,1} v_1^n,$$
$$q_2^{n+1} = v_2^n, \quad G_1 q_1^{n+1} = v_1^n - \bar{A}_{1,2} v_2^{n+1}. \tag{8}$$

Note that, unlike the classical block method of Jacobi or Schwartz, the presence in the proposed DDM of a macrogrid allocated in a special block that provides connections between all subdomains at each iteration is designed to significantly speed up the iterative process (7). At the same time, the implementation of formulas (8) differs significantly in terms of resource consumption for two-dimensional and three-dimensional cases. More specifically, the solution of an auxiliary SLAE with matrix G_1 on a planar graph is very economically implemented using an easily parallelizable algorithm. This procedure is an integral part of the algebraic problem for a three-dimensional macrogrid. These algorithms are described in more detail in the next section.

A feature of the introduction of a separating macrogrid is that during the execution of each iteration, according to the formulas (7), all subdomains are connected through interfaces only with the nodes of the macrograph. It is important to note that the amount of transmitted information for a particular subdomain is directly proportional to the number of its surface nodes, and the number of arithmetic operations performed is proportional to the number of internal (volume) nodes. Because of this, it becomes possible to combine direct calculations and exchanges in time, which benefits in computation speedup as a whole. Here, of course, it is important that the elements of the initial and preconditioning matrices have been determined in advance and distributed over the corresponding memory areas of the MPI processes. Reducing the volume of communications is important not only because they slow down the entire computing process, but also because of their high energy consumption, which seriously affects the operating costs of the MCS. This raises the non-standard problem of finding such algorithms that operate with a smaller amount of data, albeit at the expense of complicating analytics.

4 A parallel direct method for solving SLAE on a macrograph

We consider in a uniform form an algorithm to solve an algebraic system

$$A_{1,1} u = \begin{bmatrix} T & C \\ C^\top & S \end{bmatrix} \begin{bmatrix} u_e \\ u_v \end{bmatrix} = \begin{bmatrix} f_e \\ f_v \end{bmatrix}, \quad u_e, f_e \in \mathbb{R}^{N_e}, \quad u_v, f_v \in \mathbb{R}^{N_v}, \tag{9}$$

where $T = \text{block-diag}\{T_l, l = 1, \ldots, N_e/n\}$ is a block-diagonal matrix where each block T_l is a tridiagonal matrix of order N_e/n, n is the dimension of the area. For ease of notation, all macroedges are considered to contain the same number of nodes, so that the dimensions of vectors u_e, f_e are equal to the total number of macroedge nodes N_e. Each of the N_v equations for macrovertices (naturally, $N_v \ll N_e$) contains four links to macroedge variables in the 2D case (and six

links in the 3D case). These numbers are the numbers of nonzero elements in a row of triangular matrix $C^\top \in \mathbb{R}^{N_v,N_e}$. The matrix S is diagonal one, and in matrix C only some rows have a single nonzero element (corresponding to the edge nodes of each macroedge). Note that a multiple solution of this system with the same matrix and different sequentially calculated right-hand sides is often required.

The idea of this algorithm (evidently first published in the work of I.V. Fryazinov in 1970 [28], and in 1978 applied by N.N. Yanenko and A.N. Konovalov with colleagues [29] to parallelizing the sweep) consists in the preliminary exclusion of the subvector u_e from the (9) system by an economical sweep method.

Omitting the superfluous indices, we write the system of three-point equations on one macroedge in the form

$$(Tu)_t = -a_t u_{t-1} + b_t u_t - c_t u_{t+1} = f_t, \quad t = 1, 2, \ldots, N_e, \tag{10}$$

where u_0, u_{N_e+1} are the unknowns in the adjacent macrovertices, through which we express the required u_t from (10). Obviously, by the principle of superposition, we can write

$$u = \{u_t = u_0\hat{u}_t + u_e\check{u}_t + \bar{u}_t\}, \quad u = \hat{u} + \check{u} + \bar{u},$$

where $\hat{u}_t$ corresponds to the SLAE solution (10) at $u_0 = 1, u_{N_e+1} = 0, f_t = 0, \check{u}_t$ – the solution at $u_0 = 0, u_{N_e+1} = 1, f_t = 0$, and $\bar{u}_t$ is the solution for $u_0 = u_{N_e+1} = 0$.

The general solution of the tridiagonal system can be represented via one of two recursions

$$u_t = \begin{cases} \hat{\beta}_t u_{t+1} + \hat{z}_t, & t = N_{e-1}, \ldots, 1; \quad u_{N_e} = \hat{z}_{N_e}, \\ \check{\beta}_t u_{t-1} + \check{z}_t, & t = 2, \ldots, N_e; \quad u_1 = \check{z}_1, \end{cases} \tag{11}$$

where the auxiliary quantities are found as follows:

$$\hat{\beta}_1 = c_1\hat{d}_1, \quad \hat{d}_1 = b_1^{-1}, \quad \hat{\beta}_t = c_t\hat{d}_t, \quad \hat{d}_t = (b_t - a_t\hat{\beta}_{t-1})^{-1}, \quad t = 2, \ldots, N_e,$$
$$\check{\beta}_{N_e} = a_{N_e}\hat{d}_{N_e}, \quad \hat{d}_{N_e} = b_{N_e}^{-1}, \quad \check{\beta}_t = a_t\check{d}_t, \quad \check{d}_t = (b_t - c_t\check{\beta}_{t+1})^{-1}, \quad t = N_e, \ldots, 1,$$

$$\hat{z}_1 = f_1\hat{d}_1, \quad \hat{z}_t = (f_1 - a_t\hat{z}_{t-1})\hat{d}_t, \quad t = 2, \ldots, N_e,$$
$$\check{z}_{N_e} = f_{N_e}\check{d}_{N_e}, \quad \check{z}_t = (f_t - c_t\check{z}_{t+1})\check{d}_t, \quad t = N_e - 1, \ldots, 1. \tag{12}$$

Note that in these relations, the "ordinary" recursions (related to the values $\hat{\beta}_t, \hat{z}_t$) and back ones can be calculated on two processors for each macroedge in parallel, moreover, when solving the SLAE multiple times, it suffices to calculate the coefficients $\hat{\beta}_t, \hat{d}_t, \check{\beta}_t, \check{d}_t$ only once.

An additional doubling acceleration here can be obtained by reducing the length of recursions using the counter-sweep algorithm. Let $t_0 = N_e/2$. Let us calculate the values $\hat{\beta}_t, \hat{d}_t, \hat{z}_t$, for $t = 1, \ldots, i_0 - 1$, and the values $\check{\beta}_t, \check{d}_t, \check{z}_t$ for

$t = N_e, \ldots, i_0+1$. Then we substitute the solution in the form (11) into equation (10), resulting in the formula

$$u_{t_0} = (f_{t_0} + a_{t_0-1}\hat{z}_{t_0-1} + c_{t_0+1}\check{z}_{t_0+1})/(t_0 b_{t_0} - a_{t_0}\hat{\beta}_{t_0-1} - c_{t_0}\check{\beta}_{t_0+1}). \qquad (13)$$

After u_{t_0} is found, the remaining components of the solution are calculated for $t < t_0$ and $t > t_0$ synchronously via recursions (11).

A further increase in the scalability of parallelization can be achieved by economically calculating the columns of the matrix $G = \{g_{t,t'}\}$, that is inversion to the tridiagonal one. Obviously, such a t'-th column can be determined through a particular solution $\bar{u}_t$, with the right-hand side of the form $f_{t'} = \delta_{t',t''}$, where $\delta_{t',t''}$ is the Kronecker symbol. In this case, formula (13) is significantly simplified, and we get the diagonal element of the inverse matrix

$$g_{t,t'} = \gamma_{t'}^{-1}, \quad g_{t,t'} = \begin{cases} \hat{\beta}_{t''}\gamma_{t'}, & t'' = t'-1, \ldots, 1, \\ \check{\beta}_{t''}\gamma_{t'}, & t'' = t'+1, \ldots, N_e. \end{cases}$$

Hence, for the solution $\bar{u}$ we have a rather resource-intensive formula

$$\bar{u} = Gf = \left\{ \bar{u}_t = \sum_{t'=1}^{N_e} g_{t,t'} f_t \right\},$$

which, however, in the presence of N_e^2 processes, is calculated on the shared memory taking time $\tau_a \ln N_e$, where τ_a is the average time to fulfill one arithmetic operation. It is important to note that after eliminating the subvector u_e from (9) for $\hat{u}$, we obtain a "regular" banded SLAE with a five-diagonal or seven-diagonal matrix depending on the dimension of the problem.

Note that so far we have considered the parallelization for a single macroedge. Obviously, with a sufficient number of processors, all these operations can be performed simultaneously for all N_e macroedges, and we will get the corresponding speedup factor.

To find the subvector $\hat{u}$ from (9), it is a standard problem of solving a low-order N_v SLAE with a band matrix which can be implemented using a standard direct solver.

The formulas (6) – (8) allow us to make two contradictory general remarks. On the one hand, the convergence speedup for the iterative process requires a reduction in the condition number of the matrix $B^{-1}A$. However, this requires improving the approximation properties of the preconditioner B with respect to A that entails a complication of the matrix G_2 structure in (6), (8) and increase of the resource consumption for each iteration. We will demonstrate achieving a possible compromise via an example of a two-dimensional problem.

In this case, the inversion of the G_1 matrix in (8) actually represents the solution of the SLAE on the macrogrid, whose economical algorithm is described in the next section. A formation of the matrix G_2 will be carried in a manner that preserves the five-diagonal structure of the matrix $\bar{A}_{2,2}$. Due to the specifics of our problem, it suffices to consider this procedure for an example of one-diagonal

block D_s. Moreover, the matrix entries will be modified only for those matrix rows that correspond to the near-boundary nodes of the subdomain (we assume that its boundary consists of adjacent macroedges). More specifically, here only diagonal entries have to be recalculated as well as those off-diagonal ones that are responsible for connections with the nearest (neighboring) node. This is illustrated by the five-point pattern shown in Fig. 1 near-boundary nodes of the given subdomain. We can call this algorithm a tridiagonal modification of the D_s matrix block.

5 Performance and scalability issues of DDM parallelization

In this section, we focus on the issues of mathematical efficiency of iterative processes and technologies for achieving high-performance computing using the example of three-dimensional problems, as the most resource-intensive and practically relevant.

A natural approach to DDM parallelization is the use of hybrid programming with the allocation of each sub-area of a computing node containing several CPU devices with hierarchical shared memory. At the same time, at each iteration, information exchanges between subdomains are carried out by means of the MPI library for transmitting inter-node messages. With the help of the formed MPI processes, an external iterative algorithm is organized with synchronization of the solutions of auxiliary SLAEs in subdomains. The second level of parallelization of calculations is organized with the help of multi-threaded technologies (Open MP system).

The presence of a macrogrid as a special subdomain in our proposed approach determines the variety of algorithmic fragments and generated or data structures. Optimization of computational processes of this nature, their parallelization and mapping to the MCS architecture is a rather complex problem. Here we put a goal to outline the ways of experimental research and its solution.

One of the priority issues in DDM is optimization of the number of subdomains comprising the computational domain. It would seem that with an increase in their number, in the presence of a sufficient set of processes, it is possible to significantly increase the degree of parallelism and, consequently, the acceleration of calculations. However, when using, e.g., the Jacobi or Schwartz block method, the number of external iterations over subdomains increases significantly. Intuitively, we can assume that if the number of subdomains for $M_x = M_y (= M_z) = M_H$ is equal to M_H^2 in the two-dimensional case and M_H^3 — in the three-dimensional case, then the number of iterations in Krylov subspaces will be approximately proportional to M_H for both variants. The presence of a macrogrid subdomain, which implements information links between all subdomains at each iteration, is designed to significantly increase the rate of convergence of the iterative process.

An analysis of the efficiency of DDM parallelization can be carried out through the estimates of scalability in the strong and weak senses. The first means speeding up calculations by increasing the number of processors when

solving a large fixed problem, and the second – saving (ideally) the calculation time while increasing the resource intensity of the task and the number of processors.

The strong scaling strategy in the proposed decomposition method when solving a fixed large SLAE (e.g., $N > 10^9$) means an increase in the number of M subdomains and corresponding MPI processes. In this case, the size of each subdomain, the number of its nodes, and the dimension of the corresponding SLAE decrease. The exception is the separating subdomain, for which exchanges with all other subdomains are carried out at each external iteration.

The total time for solving the problem can be represented by a simple formula $t = t_a + t_c$, where the first term is the time of arithmetic operations, and the second is the time of communications (we can assume that with $M = 1$ we have $t_c = 0$). Since t_a decreases as M grows, while t_c grows, it is obvious that there is an optimal number of M_0 subdomains for which the time t is minimal. The functional dependencies of these characteristics are qualitatively represented by the formulas

$$t_a = \tau_a N_a, \quad t_c = \tau_0 N_0 + \tau_c N_c, \quad \tau_a \ll \tau_c \ll \tau_0,$$

where τ_a, τ_c, τ_0, are the average times for performing an arithmetic operation, transmitting one number, and the duration of setting (delay) for one communication; N_a, N_c and N_0 — total number of arithmetic operations, passed numbers and number of transactions. A simple analysis of this record shows the expediency of pre-buffering data and carrying out the exchanges themselves in large portions.

The scaling of the DDM in the weak sense assumes that as M increases, the number of grid nodes in each subdomain is preserved: in this case, the problem dimension and the number of processors involved change proportionally. If we assume that exchanges between subdomains are carried out only by boundary (surface) data, then their total volume at one iteration is proportional to $M^{2/3}$. The latter means that as M increases, the relative communication losses will decrease, since the time t_a per one MPI process will increase only due to an increase in the number of iterations.

6 Conclusions

The paper proposes variants of the domain decomposition method with a formal definition of a separating subdomain that forms a macrogrid and consists of interface (surface) nodes for all subdomains. The inversion of the corresponding algebraic subsystem is carried out using economical parallel algorithms, and the general iterative process is formed on the basis of block incomplete factorization methods in the Krylov subspaces. Consideration of the algorithms is done with the examples of two-dimensional and three-dimensional regular grids with the formation of SLAE with a fairly simple structure. A qualitative analysis of the scalability of parallelization of the described DDMs in the strong and weak

senses is carried out. The proposed methods can be generalized to more complicated boundary value problems, unstructured grids, and various approximation approaches. The main technological difficulties that arise in these case, are related to the automatic construction of domain decomposition procedure as well as forming the corresponding data structures.

References

1. Dolean, V., Jolivet, P., Nataf, F.: An introduction to domain decomposition methods: algorithms, theory and parallel implementation. SIAM, Philadelphia (2015). doi:10.1137/1.9781611974065
2. Gurieva, Y.L., Il'in, V.P.: On coarse grid correction methods in the Krylov subspaces, J. Math. Sci. **232**(6), 774–182 (2018) doi:10.1007/s10958-018-3907-9
3. Laevsky, Y.M., Matsokin, A.M.: Decomposition methods for solving elliptic and parabolic boundary value problems. In: Sib. Zh. Vychisl. Mat. **2**, 361–372 (1999).
4. Vassilevski, Y.: A hybrid domain decomposition method based on aggregation. Numer. Linear. Alg. Appl. **11**, 327–341 (2004). doi:10.1002/nla.351
5. Vassilevski, Y.V., Olshanskii, M.A.: Short course on multi-grid and domain decomposition methods. Moscow, MAKS Press Publ (2007).
6. Xiang, H., Nataf, F.: Two-level algebraic domain decomposition preconditioners using Jacobi-Schwarz smoother and adaptive coarse grid corrections. J. Comput. Appl. Math. **261**, 1–13 (2014). doi:10.1016/j.cam.2013.10.027
7. Quarteroni, A., Valli, A.: Domain Decomposition Methods for Partial Differential Equations. Oxford: Clarendon Pecss (1999).
8. Savchenko, A.O., Il'in, V.P., Butyugin, D.S.: A method for solving an exterior three-dimensional boundary value problem for the Laplace equation. Sib. Zh. Vychisl. Mat. **19**(2), 88–99 (2016). doi:10.17377/SIBJIM.2016.19.208
9. Sveshnikov, V.M.: Construction of direct and iterative decomposition methods. J. Appl. Industr. Math. **4**, 431–440 (2010). doi:10.1134/S1990478910030166
10. Agoshkov, V.I., Lebedev, V.I.: Variational algorithms of the domain decomposition method. Sov. J. Numer. Anal. Math. Modelling. **5**(1), 27–46 (1990).
11. Smelov, V.V.: The principle of iteration over subdomains in problems involving the transport equation. In: U.S.S.R. Comput. Math. Math. Phys., vol. 21, pp. 131–143 (1981). doi:10.1016/0041-5553(81)90158-0
12. Lions, P.L.: On the Schwarz Alternating Method III: A Variant for Nonoverlapping Subdomains. SIAM Philadelphia, PA, 202–233 (1900).
13. Schoberl, J., Melenk, J.M., Pechstein, C., Zaglmayr, S.: Additive Schwarz preconditioning for p-version triangular and tetrahedral finite elements. IMA J. Numer. Anal., 1–24 (2008). doi:10.1093/imanum/drl046
14. Ilin, V.P.: Multi-Preconditioned Domain Decomposition Methods in the Krylov Subspaces. In: LNCS, 10187, Springer, pp. 95–106 (2017). doi:10.1007/978-3-319-57099-0_9
15. Gurieva, Y.L., Il'in, V.P., Perevozkin, D.V.: Deflated Krylov Iterations in Domain Decomposition Methods. In: Lee CO. et al. (eds) Domain Decomposition Methods in Science and Engineering XXIII. Lecture Notes in Computational Science and Engineering, vol 116. Springer, Cham, pp. 1–13 (2017).
16. Il'in, V.P.: Iterative Preconditioned Methods in Krylov Spaces: Trends of the XXI Century. Comput. Math. Math. Phys., vol. 61, N. 11, pp. 1750–1775. Pleiades Publishing, Ltd (2021). doi:10.1134/S0965542521110099

17. Smith B. F. Bjorstad P. F., Group W. D.: Domain Decomposition Parallel Multilevel Methods for Elliptic Differential Equations. Cambridge Univer. Press. (1996). doi:10.1016/s0898-1221(97)90035-3
18. Widlind, O., Toselli, A.: Domain Decomposition Methods, Algorithms and Theory. In: Springer Series in Comput, Math. vol. 74, Berlin (2005). doi:10.1007/b137868
19. Nepomnyashikh, S.V.: Domain Decomposition Methods. In: Radon Series Comput. Appl. Math. **1**, 89–159 (2007).
20. Korneev, V., Langer, U:. Domain Decomposition Methods and Preconditioning. In: Encyclopedia of Computational Mechanics, vol. 1, John Wiley and Sons, Ltd., pp. 617–647 (2004) doi:10.1002/0470091355.ecm019
21. Qin, L., Xu, X.: On a Parallel Robin-Type Nonoverlapping Domain Decomposition Method. SIAM J. Numer. Anal. **44**(6), 2539–2558 (2006). doi:10.1137/05063790X
22. Gander, G., Margoles, F, Nataf, F.: Optimized Schwarz Methods without Overlap for the Helmholtz Equation. SIAM J. Sci. Comput. **21**, 38–60 (2002). doi: 10.1137/S1064827501387012
23. Bern, M., Gilbert, J., Hendrickson, B., Nguyen, N., Toledo, S.: Support Graph Preconditioners. SIAM J. Matrix Anal. Appl. **27**, 930–951 (2006). doi: 10.1137/S0895479801384019
24. Chen, D., Toledo, S.: Vaidya's Preconditioners: Implementation and Experimental Study. ETNA **16**, 30–49 (2003).
25. Vaidya, P.: Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. An unpublished manuscript. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computations, Minneapolis (1991).
26. Perevozkin, D.V., Omarova, G.A.: Preconditioning Methods Based on Spanning Tree Algorithms. In: MSR 2020, J. Phys.: Conf. Ser., vol. 1715 (2021). doi: 10.1088/1742-6596/1715/1/012005
27. Ili'n, V. P., Kuznetsov, Yu.I: Tridiagonal Matrices and its Applications. Moscow, Nauka Publ. (1985). [in Russian].
28. Fryazinov, I.V.: An algorithm for the solution of difference problems by graphs. U.S.S.R. Comput. Math. Math. Phys. **10**(2), 268–273 (1970). doi:10.1016/0041-5553(70)90164-3
29. Yanenko, N.N., Konovalov, A.N., Bugrov, A.N., Shustov, G.V.: About organizing parallel computing and "paralleling" sweep. Numerical methods for continuum mechanics **9**(7), 139–146 (1978). [in Russian].