Incomplete Factorization Approach in Algebraic Domain Decomposition Methods

Yana Gurieva[1], Valery Il'in[1], and Ruslan Kardash[2(✉)]

[1] Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia
yana@lapasrv.sscc.ru, ilin@sscc.ru
[2] Novosibirsk State University, Novosibirsk, Russia
r.kardash@g.nsu.ru

Abstract. Iterative methods for domain decomposition methods in Krylov subspaces to solve large systems of linear algebraic equations arising from grid approximations of multidimensional boundary value problems are considered. The algorithms under study are based on purely algebraic approaches with special variants of approximate factorization of matrices arizing from grid division by a single-layer or two-layer separating macrogrid subsets. Traditional interface boundary conditions between contacting subdomains are replaced by matrix approximations with the compensation principle exploiting. The implementation of preconditioning matrices is carried out by naturally parallelizable forward and back sweep algorithms on the macrogrid. The issues of assessing the efficiency and performance of the proposed methods and technologies for two-dimensional and three-dimensional problems are discussed, including the cases of parallelizing calculations. The results of numerical experiments for a set of methodical problems are presented.

Keywords: Iterative method · Domain decomposition · Matrix factorization · Krylov subspaces · Performance · Forward and backward sweep

1 Introduction

Domain decomposition methods (DDM) have been studied in many monographs [1–7], review articles [8–13], as well as materials of regular international conferences from DDM.ORG website.

The main motivations of DDM are the parallelization of algorithms on multiprocessor systems (MPS) and the distribution of supercomputer resources when solving large multidimensional problems. In algebraic terms, such approaches consist in a construction of preconditioned iterative algorithms in the Krylov subspaces, including conjugate direction methods for symmetric systems of linear algebraic equations (SLAEs) and generalized methods of minimal residuals

V. Voevodin et al. (Eds.): RuSCDays 2024, LNCS 15406, pp. 362–376, 2025.
https://doi.org/10.1007/978-3-031-78459-0_26

(various versions of GMRES) or semi-conjugate direction for asymmetric ones, see [14–17].

In this paper, we develop the purely algebraic approach proposed in [18] for solving SLAEs with large sparse matrices arising from grid approximations of the nodal type for multidimensional boundary value problems, when each grid node corresponds to one component of the original vector. An iterative process is constructed based on the use of incomplete factorization as a preconditioner of the original system with a special block decomposition of the problem using a set of separator nodes as a macrogrid formally considered as one of the resulting grid subdomains. Economical algorithms for solving the generated auxiliary subsystems are implemented using sweep methods for tridiagonal SLAEs [19].

This work is structured as follows. Section 2 discusses principles of a decomposition of 2D and 3D grid domains, as well as the corresponding algebraic data structures. The next section is devoted to a brief presentation of the preconditioned conjugate gradient and conjugate residual methods (CG and CR) for solving symmetric SLAEs. Section 4 contains a description of algorithmic aspects, including issues of parallelization efficiency and performance of the methods and technologies under study. Section 5 presents the results of numerical experiments on a series of methodological problems demonstrating the efficiency of the proposed approaches. In conclusion, a possible generalization of the methods to a wider class of problems and plans for further research are discussed.

2 Matrix Structures for Grid Decomposition Methods

We will consider methods for solving SLAEs with the real data

$$Au = f, \quad A = \{a_{l,l'}\} \in \mathcal{R}^{N,N}, \quad u = \{u_l\}, f = \{f_l\} \in \mathcal{R}^N, [1p] \tag{1}$$

with symmetric positive definite (s.p.d.) matrices arising in grid approximations of multidimensional boundary value problems [20]. For brevity, we will limit ourselves to only three-dimensional regular grids of the form

$$\Omega^h : x = x_i,\ i = 1, \ldots, N_x,\ y = y_j,\ j = 1, \ldots, N_y, \quad z = z_k,\ k = 1, \ldots, N_z, \tag{2}$$

forming a computational domain with the shape of a parallelepiped Ω. We assume that the system (1) consists of grid seven-point equations written using indices i, j, k in the following form:

$$\begin{aligned}(Au)_{i,j,k} &= a^{(0)}_{i,j,k}u_{i,j,k} - a^{(1)}_{i,j,k}u_{i-1,j,k} - a^{(2)}_{i,j,k}u_{i,j-1,k} - a^{(3)}_{i,j,k}u_{i+1,j,k} \\ &\quad - a^{(4)}_{i,j,k}u_{i,j+1,k} - a^{(5)}_{i,j,k}u_{i,j,k-1} - a^{(6)}_{i,j,k}u_{i,j,k+1} = f_{i,j,k}, \\ a^{(1)}_{1,j,k} &= a^{(2)}_{i,1,k} = a^{(3)}_{N_x,j,k} = a^{(4)}_{i,N_y,k} = a^{(5)}_{i,j,1} = a^{(6)}_{i,j,N_z} = 0.\end{aligned} \tag{3}$$

We will assume that this SLAE is of positive type, i.e., it is indecomposable, has the property of the diagonal dominance, and all coefficients in (3) are

non-negative. Let us divide the grid domain Ω^h into its subdomains $\Omega_m^D,\ m = 1,\dots,M_D$ by using a nested macrogrid

$$\Omega^H:\ x_{i_1},\dots,x_{i_{M_x}};\ y_{j_1},\dots,y_{j_{M_y}};\ z_{k_1},\dots,z_{k_{M_z}};\quad \Omega^H=\Omega_V^H\bigcup\Omega_E^H\bigcup\Omega_F^H,\quad (4)$$
$$1<i_1<i_{M_x}<N_x;\ 1<j_1<j_{M_y}<N_y;\ 1<k_1<k_{M_z}<N_z,$$

where Ω_V^H is a set of its macronodes, or macrovertices (intersection points of the coordinate planes forming it), Ω_E^H is a set of macroedges (grid segments connecting macronodes), and Ω_F^H is a set of macrofaces (a set of nodes on a rectangular segment of a plane bounded by macroedges). Thus, we have

$$\Omega^h=\Omega^H\bigcup\Omega^D,\ \Omega^D=\bigcup_{m=1}^{M_D}\Omega_m^D,\ \Omega_V^H=\bigcup_{k=1}^{M_V}\Omega_k^V,\ \Omega_E^H=\bigcup_{l=1}^{M_E}\Omega_l^E,\ \Omega_F^H=\bigcup_{n=1}^{M_F}\Omega_n^F, \quad (5)$$

where M_V, M_E, M_F, M_D is the number of macrovertices, macroedges, macrofaces and subdomains (the latter consisting of internal grid nodes).

In the future, when describing algorithms, we will focus on a model SLAE corresponding to the approximation of the Dirichlet problem for the Poisson equation in a cubic computational domain on a cubic grid with a step h. We will also consider two-dimensional problems obtained from three-dimensional ones as a special case: macrofaces play the role of subdomains, and the grid equations are five-point ones, with the values of non-zero coefficients for the model problem $a_{i,j}^{(0)}=4,\quad a_{i,j}^{(1)}=\dots=a_{i,j}^{(4)}=1$. In addition, to simplify the notation, we will limit ourselves to grids with the same number of nodes N_e on each macroedge and with the following characteristics:

$$M_x=M_y=M_z=M_c,\quad N_x=N_y=N_z=N_c=N_e(M_C+1)+M_C,$$
$$N=N_c^3=N_H+N_D,\ N_H=N_V+N_F+N_T,\ N_V=M_c^3,\ N_T=N_eM_E,\quad (6)$$
$$N_D=N_e^3M_D,\ M_e=3M_c(M_c+1),\ M_F=3(M_c+1)^2,\ M_D=(M_c+1)^3.$$

Here N_V, N_T, N_F, N_D are the number of macronodes, the total number of nodes on macroedges, macrofaces and subdomains, respectively.

Note that for the two-dimensional and three-dimensional cases there is some difference in the structure of the Ω^H marcogrid, which is defined as a set of subdomain separator nodes. In the two-dimensional case it consists of macronodes and macroedges, and in the three-dimensional case it also includes macrofaces.

We begin a construction of the algebraic domain decomposition method (ADDM) with the two-dimensional problems. Numbering first the nodes and corresponding vector components from the macrogrid, and then from the subdomains, we write the SLAE (1) in the following second-order block form:

$$Au=\begin{bmatrix}A_{1,1}\ A_{1,2}\\ A_{2,1}\ A_{2,2}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}=\begin{bmatrix}f_1\\ f_2\end{bmatrix}\ u_k,\ f_k\in\mathcal{R}^{N_k},\quad k=1,2\ ,\ N_1+N_2=N.\quad (7)$$

Let us carry out a more detailed partition of the subvectors, numbering the components u, f first sequentially for each macroedge, and then for all macronodes.

In addition, dividing $u_2 = \{u_2^m\}$, $f_2 = \{f_2^m\}$ by subdomains, we transform the blocks $A_{1,1}$, $A_{2,2}$ to the bordered, and the submatrix – block-diagonal form:

$$A = \left[\begin{array}{c|ccc} A_{1,1} & A_{1,2}^{(1)} & \dots & A_{1,2}^{(M_D)} \\ \hline A_{2,1}^{(1)} & A_{2,2}^{(1)} & O & \\ \vdots & & \ddots & \\ A_{2,1}^{(M_D)} & O & & A_{2,2}^{(M_D)} \end{array}\right], \quad A_{1,1} = \left[\begin{array}{ccc|c} T_1 & & & C_1 \\ & \ddots & & \\ & & T_{M_e} & C_{M_e} \\ \hline C_1^T & \dots & C_{M_e}^T & C \end{array}\right] = \begin{bmatrix} T & C_{1,2} \\ C_{2,1} & C \end{bmatrix}. \quad (8)$$

Here $T = block\text{-}diag\{T_l\}$ and each of T_l is a tridiagonal matrix of order N_e corresponding to a horizontal or vertical edge, $A_{2,2}^{(m)}$ are five-diagonal matrices of order N_e^2. Besides, $C \in R^{N_V, N_V}$ is a diagonal "macronode" matrix, and off-diagonal blocks establish connections between different types of variables and are very sparse. For the three-dimensional case, it is natural to represent the system (1) in the third-order block form

$$Au = \begin{bmatrix} A_{1,1} & A_{1,2} & 0 \\ A_{2,1} & A_{2,2} & A_{2,3} \\ 0 & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}. \quad (9)$$

where the subvectors u_1, u_2, u_3 correspond to sets of nodes from macronodes and macroedge, from macrofaces and subdomains. The block $A_{1,1}$ has a bordered structure. The matrices $A_{2,2}, A_{3,3}$ are five- and seven-diagonal ones.

3 Incomplete Factorization Methods on Macrogrid

To solve the SLAEs, we construct iterative methods based on determining the preconditioning matrix B by an approximate factorization of the matrix A, which for the two-dimensional case has the form (see motivation in [17]):

$$B = \begin{bmatrix} A_{1,1} & 0 \\ A_{2,1} & G \end{bmatrix} \begin{bmatrix} A_{1,1}^{-1} & 0 \\ 0 & G^{-1} \end{bmatrix} \begin{bmatrix} A_{1,1} & A_{1,2} \\ 0 & G \end{bmatrix} \approx A \quad (10)$$

$$G = A_{2,2} - [H]_b - \theta R, \quad Re = (H - [H]_b)e, \quad H = A_{2,1}A_{1,1}^{-1}A_{1,2}.$$

where R is a diagonal (compensating) matrix, $e = \{1\}$ is a vector with the unit entries, $[H]_b$ denotes a band approximation (with the bandwidth b)of the matrix H with its specification given below, and θ is an iterative (compensating) parameter. The diagonal compensation principle used here ensures the relation $Be = Ae$ for $\theta = 1$, see more details in [9]. The inversion of the preconditioner B is carried out by the following block approach:

$$B \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} A_{1,1} & 0 \\ A_{2,1} & G \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \begin{bmatrix} I & A_{1,1}^{-1}A_{1,2} \\ 0 & I \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \quad (11)$$

which is implemented using its component relations

$$A_{1,1}v_1 = g_1, \quad Gq_2 = g_2 - A_{2,1}v_1 = \bar{g}_2, \quad A_{1,1}w_1 = A_{1,2}q_2, \quad q_1 = v_1 - w_1. \quad (12)$$

For a block third-order SLAE (9), for the three-dimensional problem, a construction of a factorized preconditioner can be represented as follows, see [9]:

$$B = \begin{bmatrix} G_1 & 0 & 0 \\ A_{2,1} & G_2 & 0 \\ 0 & A_{3,2} & G_3 \end{bmatrix} \begin{bmatrix} G_1^{-1} & 0 & 0 \\ 0 & G_2^{-1} & 0 \\ 0 & 0 & G_3^{-1} \end{bmatrix} \begin{bmatrix} G_1 & A_{1,2} & 0 \\ 0 & G_2 & A_{2,3} \\ 0 & 0 & G_3 \end{bmatrix}, \quad G_1 = A_{1,1},$$
$$G_2 = A_{2,2} - [H_2]_b - \theta_2 R_2,\ G_3 = A_{3,3} - [H_3]_b - \theta_3 R_3,\ R_2 e_2 = (H_2 - [H_2]_b)e,$$
$$R_3 e_3 = (H_3 - [H_3]_b)e,\ H_2 = A_{2,1}A_{1,1}^{-1}A_{1,2},\ H_3 = A_{3,2}G_2^{-1}A_{2,3}. \tag{13}$$

Here, R_2 and R_3 are the diagonal matrices, e_2 and e_3 are the vectors with the unit entries, and $\theta \in [0,1]$ is compensating parameters. For $\theta = 1$ we obtain the complete diagonal compensation $Be = Ae$ which is matching the row sums of the preconditioned and original matrices. Let us remark, that the alternative to the considered approach is block SSOR preconditioning, see [9,14].

A quick analysis of the block-component relations (12),(13) shows, firstly, that their implementation includes the solution of subsystems with the matrix $A_{1,1}$ of the bordered type and corresponding to macrogrid nodes from Ω^H. Moreover, we will need to calculate the columns of the inverse matrix $A_{1,1}^{-1}$, which can be done economically using parallel sweep algorithms. Much more difficult problems arise when forming the matrices G, G_2, G_3 and solving the corresponding algebraic subsystems. Note that with the exact factorization $[H]_b = H$ these matrices of the block order M_D have all their blocks as non-zero ones, although their original matrices $A_{2,2}, A_{3,3}$ are block diagonal ones.

As it follows from the vector-matrix structures indicated above, the block configuration of the "frame matrix" $A_{1,1}$ presented in (8) is externally the same for two-dimensional and three-dimensional problems, although the topology of the macroedges and macronodes differ qualitatively and quantitatively. Therefore, to solve the first of the equations in (12), the matrix factorization can be written uniformly using the following formulas:

$$\begin{bmatrix} T & C_{1,2} \\ C_{2,1} & C \end{bmatrix} \begin{bmatrix} v_T \\ v_c \end{bmatrix} = \begin{bmatrix} T & 0 \\ C_{2,1} & S_c \end{bmatrix} \begin{bmatrix} w_T \\ w_c \end{bmatrix} = \begin{bmatrix} g_T \\ g_c \end{bmatrix}, \begin{bmatrix} I & T^{-1}C_{1,2} \\ 0 & I \end{bmatrix} \begin{bmatrix} v_T \\ v_c \end{bmatrix} = \begin{bmatrix} w_T \\ w_c \end{bmatrix},$$
$$S_c = C - C_{2,1}T^{-1}C_{1,2}, \tag{14}$$

which in component form are implemented by the relations:

$$Tw_T = g_T, \quad S_c v_c = g_c - C_{2,1}w_T = f_c,\ v_T = w_T - T^{-1}C_{1,2}v_c. \tag{15}$$

Here, the subvectors with the indexes "T" and "c" correspond to macroedge and macronode variables, and the matrix S_c (Schur's complement) is a symmetric five- and seven-diagonal matrix in the two-dimensional and three-dimensional cases, respectively. It's easy to show that if A is a matrix of the positive type, and the matrix T has the strict diagonal dominance ($Te \geq \delta e$, where the value $\delta > 0$ does not depend on h) , then the matrix S_c has the same property. Consequently, its condition number is finite, that is, it does not depend on h as well, and the corresponding SLAE for finding w_c in (15) can be economically

solved by the iterative Chebyshev acceleration method, which in this case will be optimal in terms of the order (the total number of operations is proportional to the order of the system, see [21], [22]). Note also that the solution of SLAEs with tridiagonal matrices, as well as their inversions, can be effectively implemented in (15) using naturally parallelizable algorithms, see [23]. Let a system of N_e three-point equations be defined on some $l-$th macroedge

$$\begin{aligned} &-a_t u_{t-1} + b_t u_t - c_t u_{t+1} = f_t, \quad t = 1, 2, \ldots, N_e, \\ &u_0 = u_b, u_{N+1} = u_e, \end{aligned} \tag{16}$$

where the quantities u_b, u_e play the role of the Dirichlet boundary conditions for the corresponding grid boundary value problem and represent the value of the solution of the macronodal SLAE in the adjacent macronodes. The solution of the subsystem (16) is constructed using one of the recursions of the form

$$u_t = \begin{cases} \hat{\beta}_t u_{t+1} + \hat{z}_t, & t = N_e - 1, \ldots, 1; \quad u_{N_e} = \hat{z}_{N_e}, \\ \check{\beta}_t u_{t-1} + \check{z}_t, & t = 2, \ldots, N_e; \quad u_1 = \check{z}_1. \end{cases} \tag{17}$$

Here the coefficients of the recursions are determined by the following formulas:

$$\begin{aligned} &\hat{\beta}_1 = c_1 \hat{d}_1, \quad \hat{d}_1 = b_1^{-1}, \quad \hat{\beta}_t = c_t \hat{d}_t, \quad \hat{d}_t = (b_t - a_t \hat{\beta}_{t-1})^{-1}, \ t = 2, \ldots, N_e, \\ &\check{\beta}_{N_e} = a_{N_e} \check{d}_{N_e}, \ \check{d}_{N_e} = b_{N_e}^{-1}, \ \check{\beta}_t = a_t \check{d}_t, \ \check{d}_t = (b_t - c_t \check{\beta}_{t+1})^{-1}, \ t = N_e, \ldots, 1, \\ &\hat{z}_1 = (f_1 + a_1 u_0)\hat{d}_1, \ \hat{z}_t = (f_t + a_t \hat{z}_{t-1})\hat{d}_t, \ t = 2, \ldots, N_e - 1, \\ &\hat{z}_{N_e} = (f_{N_e} + c_{N_e} u_{N_e+1} + a_{N_e} \hat{z}_{N_e-1})\hat{d}_{N_e}, \ \check{z}_{N_e} = (f_{N_e} + c_{N_e} u_{N_e+1})\check{d}_{N_e}, \\ &\check{z}_t = (f_t + c_t \check{z}_{t+1})\check{d}_t, \ t = N_e - 1, \ldots, 2, \quad \check{z}_1 = (f_1 + a_1 u_1 + c_1 \check{z}_2)\check{d}_1. \end{aligned} \tag{18}$$

The formation of a macronodal subsystem with the matrix S_c and the right-hand side f_c from (15) actually uses the principle of superposition, that is, the general solution of the grid boundary value problem is represented as a sum:

$$u = u_0 \check{u} + u_{N_e+1} \hat{u} + \bar{u} = \{u_t = u_0 \hat{u}_t + u_{N_e+1} \check{u}_t + \bar{u}_t\} \tag{19}$$

where $\check{u}, \hat{u}$ are the solutions to a homogeneous SLAE (16) ($f_t = 0$ for all t) with the boundary conditions $\check{u}_0 = 1$, $\check{u}_{N_e+1} = 0$, $\hat{u}_0 = 0$, $\hat{u}_{N_e+1} = 1$, and $\bar{u}$ is the solution to (16) with the given values of f_t and homogeneous boundary conditions, which are most simply calculated using the counter sweep formulas

$$\begin{aligned} &\hat{u}_t = \hat{\beta}_t \hat{u}_{t+1}, \ t = N_e - 1, \ldots, 1, \ \hat{u}_{N_e} = c_{N_e} \hat{d}_{N_e}, \\ &\check{u}_t = \check{\beta}_t \check{u}_{t-1}, \ t = 2, \ldots, N_e, \ \check{u}_1 = a_1 \hat{d}_1. \end{aligned} \tag{20}$$

Note that if it is necessary to solve the SLAE (16) multiple times, as well as when finding the columns of the inverse matrix $T^{-1} = \{\widetilde{u}^{(l)} = \widetilde{u}_t^{(l)}, t = 1, \ldots, N_e\}$, corresponding to the right-hand sides $f_t^{(l)} = \delta_{t,e}$, where $\delta_{t,e}$ is the Kronecker symbol, it is advisable to use economical formulas

$$\begin{aligned} &\widetilde{u}_{(l)}^{(l)} = (b_l - a_l \hat{\beta}_{l-1} - c_l \check{\beta}_{l+1})^{-1}, \ l = 2, \ldots, N_e - 1, \\ &\widetilde{u}_{(1)}^{(l)} = (b_1 - c_1 \check{\beta}_2)^{-1}, \ \widetilde{u}_{(N_e)}^{(N_e)} = (b_{N_e} - a_{N_e} \hat{\beta}_{N_e-1})^{-1}. \end{aligned} \tag{21}$$

After substituting the expressions (19) into (2) with $t = 1$ and $t = N_e$ for all macroedges, simple relations are obtained for the elements of the matrix S_c and the right-hand side vector $\bar{f}_c$ of the macronodal SLAE. We present the derivation of the corresponding formulas for the two-dimensional case. For a macronode with the coordinates (x_{i_k}, y_{j_l}) we denote the macroedges adjacent to the left, down, right and top (up) as $E^l_{k,l}$, $E^d_{k,l}$, $E^r_{k,l}$, $E^u_{k,l}$ respectively. Obviously, the opposite ends of these four macroedges are the macronodes with the indices (i_{k-1}, j_l), (i_k, j_{l-1}), (i_{k+1}, j_l) , (i_k, j_{l+1}). Let us write the initial equation for the (i_k, j_l)-th macronode in a form similar to (2):

$$\begin{gathered} a^{(0)}_{i_k,j_l} u_{i_k,j_l} - a^{(1)}_{i_k,j_l} u_{i_k-1,j_l} - a^{(2)}_{i_k,j_l} u_{i_k,j_l-1} - a^{(3)}_{i_k,j_l} u_{i_k+1,j_l} - a^{(4)}_{i_k,j_l} u_{i_k,j_l+1} = f_{i_k,j_l}, \\ i_k, j_l \in \Omega^H_V, \quad a^{(1)}_{i_1,j_l} = a^{(2)}_{i_k,j_1} = a^{(3)}_{i_{M_c},j_l} = a^{(4)}_{i_k,i_{M_c}} = 0, \quad k,l = 1,\ldots,M_c. \end{gathered} \tag{22}$$

The off-diagonal terms included in this equation and corresponding to the internal nodes of various macroedges can be expressed in terms of partial solutions of the corresponding tridiagonal algebraic systems. As a result, the five-point equations for macronodes are obtained in the following form:

$$(S_c u_c)_{i_k,j_l} = \bar{f}^c_{i_k,j_l}, \tag{23}$$

$$\begin{gathered} (S_c u_c)_{i_k,j_l} = s^{(0)}_{i_k,j_l} u_{i_k,j_l} - s^{(1)}_{i_k,j_l} u_{i_k-1,j_l} - s^{(2)}_{i_k,j_l} u_{i_k,j_l-1} - s^{(3)}_{i_k,j_l} u_{i_k+1,j_l} - s^{(4)}_{i_k,j_l} u_{i_k,j_l+1} \\ s^{(0)}_{i_k,j_l} = a^{(0)}_{i_k,j_l} - a^{(1)}_{i_k,j_l} \hat{u}^l_{i_k-1,j_l} - a^{(2)}_{i_k,j_l} \hat{u}^d_{i_k,j_l-1} - a^{(3)}_{i_k,j_l} \check{u}^r_{i_k+1,j_l} - a^{(4)}_{i_k,j_l} \check{u}^u_{i_k,j_l+1}, \\ s^{(1)}_{i_k,j_l} = a^{(1)}_{i_k,j_l} \check{u}^l_{i_k-1,j_l}, \; s^{(2)}_{i_k,j_l} = a^{(2)}_{i_k,j_l} \check{u}^d_{i_k,j_l-1}, \\ s^{(3)}_{i_k,j_l} = a^{(3)}_{i_k,j_l} \hat{u}^r_{i_k+1,j_l}, \; s^{(4)}_{i_k,j_l} = a^{(4)}_{i_k,j_l} \hat{u}^d_{i_k,j_l+1}, \\ (\bar{f}_c)_{i_k,j_l} = (f_c)_{i_k,j_l} + a^{(1)}_{i_k,j_l} \bar{u}^l_{i_k-1,j_l} + a^{(2)}_{i_k,j_l} \bar{u}^d_{i_k,j_l-1} + a^{(3)}_{i_k,j_l} \bar{u}^r_{i_k+1,j_l} + a^{(4)}_{i_k,j_l} \bar{u}^u_{i_k,j_l+1}. \end{gathered} \tag{24}$$

Here the superscripts l, d, r, u denote the partial solutions of the tridiagonal systems associated with the macroedges adjacent to the (i_k, j_l)-th macronode on the left, down, right and top(up), respectively, so they are left, bottom, right and top neighbors, respectively. After solving the resulting subsystem, that is, calculating the vector $w_c = u_c$ in (15) of the form (22) with the unit right-hand sides using a direct or iterative method, determining the macroedge components vector V_T from (15) is most easily done using the representation (19).

Let us now consider the details of the formation of the matrix G for the two-dimensional case from (10) and the solution of the corresponding SLAE in (12) when constructing the preconditioning. The key point here is to find the inverse matrix $A^{-1}_{1,1}$, or more precisely, its columns $h^{(l')}$ defined by the equations:

$$A_{1,1} h^{(l')} = f^{(l')} = \{\delta_{l',l}\} = \mathbf{e}^{(l')}, \quad l' = 1,\ldots,N_E, \tag{25}$$

where $\mathbf{e}^{(l')}$ is the l'th column of the identity matrix. The solution of each SLAE (25), as before during the implementation of (14),(15), is carried out by reducing it to a macronodal system with the matrix S_c, which is the same in all the cases. We assume that the columns of its inverse $S^{-1}_c = h^{(r)}_c, r = 1,\ldots,N_V$, are

calculated in advance by a parallel solution of N_V subsystems. By dividing the vector $h^{(l')}$ into macronode and macroedge subvectors $h_c^{(l')}, h_T^{(l')}$, we can state that the first of them is a linear combination of two vectors $h_c^{(r_1)}, h_c^{(r_2)}$:

$$h_c^{(l')} = \widetilde{u}_1^{(l')} a_{1,r_1}^{(l')} h_c^{(r_1)} + \widetilde{u}_{N_e}^{(l')} a_{N_e,r_2}^{(l')} h_c^{(r_2)}, \tag{26}$$

where r_1, r_2 are the numbers of macronodes adjacent to the nodes of some macroedge, containing the l'-th node with its local numbers $1, N_e$, respectively, $a_{1,r_1}^{(l')}$ and $a_{N_e,r_2}^{(l')}$ are the coefficients of the original matrix A connecting the corresponding components of the desired vector, and $\widetilde{u}_1^{(l')}, \widetilde{u}_{N_e}^{(l')}$ – are the first and the last components of the partial solution from (19) for the macroedge, containing the l'th node. Further, taking in (26) the values $h_{c,r_1}^{(l')} = u_0, \quad h_{c,r_2}^{(l')} = u_{N_e+1}$ for the boundary conditions for the macroedge subsystem, we find from the superposition principle all the components of the subvector $h_T^{(l')} = \{h_{T,t}^{(l')},\ t = 1, \ldots, N_e\}$ using the formulas (20)–(22), taking $l = l'$ in them.

Denoting the numbers of macronodes adjacent to macroedges by r_1 and r_2, to calculate the corresponding elements of the inverse matrix $A_{1,1}^{-1}$, one can also use the formulas (19) – (22) taking into account the fact that in this case we will divide $\bar{u}_t = 0$ into subsets of the near-boundary nodes Ω_m^1 adjacent to macroedges and of the internal ones Ω_m^2, numbering ner-boundary nodes first and then the internal nodes. Denoting the corresponding vectors by v_m^1, v_m^2, we can write the second equation from (12) as follows:

$$Gv_2 = \begin{bmatrix} G_{1,1}\ G_{1,2} \\ G_{2,1}\ G_{2,2} \end{bmatrix} \begin{bmatrix} v_2^{(1)} \\ v_2^{(2)} \end{bmatrix} = \begin{bmatrix} g_2^{(1)} \\ g_2^{(2)} \end{bmatrix}. \tag{27}$$

Here the matrix $G_{1,1}$ is a dense one for $[H]_b = H$ and it has its off-diagonal entries decreasing as they are distanced from the main diagonal. For the two-dimensional grid, the matrix block $G_{1,2} = G_{2,1}^T$ contains at most one non-zero element in each row, and $G_{2,2} = block\text{-}\ diag\{G_{2,2}^{(m)}\}$ for each subdomain has a five-diagonal matrix of order $(N_e - 2)^2$, which is a submatrix of $A_{2,2}^{(m)}$. A natural way to build the preconditioning matrix B in (10) is to construct a five-diagonal approximation of $[H]_b$ such that the structures of the matrices G and $A_{2,2}$ (with the same ordering of the variables) are the same.

Let k', k'' be the numbers of neighboring (i.e. $a_{k',k''} \neq 0$) near-boundary nodes from the m-th subdomain that are not corner nodes and that are adjacent to the l'-th and l''-th macroedge nodes, respectively. Then for $b = 5$ in (10) for the entries of the matrix $G = \{g_{k',k''}\}$ we have:

$$g_{k',k''} = a_{k',k''} - [H]_5(k', k'') - \theta((H - H_5)e)_{k'}, \ : [H]_5(k', k'') = a_{k',l'} h_{l''}^{(l')} a_{l'',k''}, \tag{28}$$

where $a_{l',k'}$ and $h_{l''}^{(l')}$ denote the element of the original matrix A and the l''-th component of the l'-th column of the matrix $A_{1,1}^{-1}$. If $k' = k''$ and a given node is a corner node in a subdomain, then it has two neighboring macroedge

nodes, whose numbers we denote by l' and l''. In this case, to determine the corresponding diagonal element $G_{1,1}$, it is necessary to calculate two columns of the matrix $A_{1,1}^{-1}$, namely, the $h^{(l')}$-th and the $h^{(l'')}$-th, and instead of (28) we get

$$g_{k',k''} = a_{k',k''} - [H]_5(k',k'') - \theta((H-H_5)e)_{k'},$$

$$[H]_5(k',k'') = a_{k',l'}(h_{l'}^{(l')}a_{l',k'} + h_{l''}^{(l')}a_{l',k'}) - a_{k',l''}(h_{l'}^{(l')}a_{l',k'} + h_{l''}^{(l')}a_{l'',k''}). \quad (29)$$

If node k' is a corner node, but k'' is not, then:

$$g_{k',k''} = a_{k',k''} - [H]_5(k',k'') - \theta((H-H_5)e)_{k'}, \quad (30)$$

$$[H]_5(k',k'') = a_{k',\hat{l}'}h_{l''}^{(\hat{l}')}a_{l',k''} + a_{k',l''}h_{l''}^{(\check{l}')}a_{l'',k''}.$$

Here $\hat{l}', \check{l}'$ denote the numbers of two macroedge nodes adjacent to node k', and l'' is the number of the macroedge neighbor to node k''. When only node k'' is in the corner, and $\hat{l}''$ and $\check{l}''$ are the numbers of its two macroedge neighbors, then in formula (30) only the last relation should be changed (in this case l' is the number of the neighbor to node k' on the macroedge):

$$[H]_5(k',k'') = a_{k',l'}(h_{\hat{l}''}^{(l')}a_{\hat{l}'',k''} + h_{\check{l}''}^{(l')}a_{\check{l}'',k''}). \quad (31)$$

The matrix G from (27) transfroms into the block-diagonal $\bar{G} = block\text{-}diag\{G^{(m)}\}$, and the solution the corresponding subsystem in (12) is carried out independently across subdomains using direct or some iterative algorithms.

As for the approximation of the preconditioner B from (13) for the three-dimensional case, first of all, we note that the matrix G_2 has a block structure similar to G from (27), but it is more complex, since there are many macrofaces now and they can be of three different coordinate orientations. By choosing the five-diagonal approximation $[H]_5$ we get $\bar{G} = block\text{-}diag\{\bar{G}_2^{(m)}\}$, where the diagonal blocks $\bar{G}_2^{(m)}$ generate auxiliary SLAEs that are relatively easy to solve independently in subdomains. However, in the three-dimensional case, these blocks also need to be inverted when finding the matrix G_3, and here it is necessary to use an additional approximation, using either a new incomplete factorization.

4 Iterative Methods in the Krylov Subspaces for ADDM

This section presents a description of classical preconditioned methods of the Krylov type and outlines a general scheme for parallelizing the proposed algorithms on multiprocessor computing systems (MCS).

Let $B = L_B U_B$, $L_B = U_B^T$ be some easily factorizable symmetric matrix, with the help of which we form a bilaterally preconditioned symmetric SLAE

$$\bar{A}\bar{u} = \bar{f}, \quad \bar{A} = L_B^{-1}AU_B^{-1}, \quad \bar{u} = U_B u, \quad \bar{f} = L_B^{-1}f. \quad (32)$$

To solve it, we apply one of the conjugate direction methods

$$\bar{u}^n = \bar{u}^{n-1} + \alpha_{n-1}\bar{p}^{n-1} = \bar{u}^0 + \alpha_0\bar{p}^0 + \ldots + \alpha_{n-1}\bar{p}^{n-1},$$
$$\bar{r}^n = \bar{r}^{n-1} - \alpha_{n-1}\bar{A}\bar{p}^{n-1} = \bar{r}^0 - \alpha_0\bar{A}\bar{p}^0 - \ldots - \alpha_{n-1}\bar{A}\bar{p}^{n-1}, \quad (33)$$

where $\bar{u}^0$ and $\bar{r}^n = \bar{f} - \bar{A}\bar{u}^n$ are an arbitrary initial guess and a residual vector, respectively. If for any $\bar{p}^0$ the direction vectors $\bar{p}^k$ in (13) satisfy the conditions of $\bar{A}^\gamma$-orthogonality $(\bar{p}^n, \bar{p}^k)_\gamma \equiv (\bar{A}^\gamma \bar{p}^n, \bar{p}^k) = \rho_k^{(\gamma)} \delta_{n,k}, \quad \rho_k^{(\gamma)} = \|\bar{p}^k\|_\gamma^2$ where $\delta_{n,k}$ is the Kronecker symbol and $\gamma = 1, 2$ for the conjugate gradient and conjugate residual methods, respectively, then for the parameter values

$$\alpha_k = \sigma_k / \rho_k, \quad \sigma_k = (\bar{r}^0, \bar{p}^k)_{\gamma-1} = (\bar{r}^k, \bar{p}^k)_{\gamma-1} \tag{34}$$

the functionals $\Phi_\gamma(\bar{r}^n) = (\bar{A}^{\gamma-2}\bar{r}^n, \bar{r}^n)$ take their minimal values in the Krylov subspaces $\mathcal{K}_n(\bar{r}^0, \bar{A}) = Span(\bar{r}^0, \bar{A}\bar{r}^0, \ldots, \bar{A}^{n-1}\bar{r}^0)$.

To calculate the direction vectors that satisfy the orthogonality conditions (34), we use the two-term Hestenes-Stiefel recursion

$$\bar{p}^0 = \bar{r}^0, \quad \bar{p}^n = \bar{r}^n + \beta_{n-1}\bar{p}^{n-1}, \quad \beta_{n-1} = (\bar{A}^\gamma \bar{r}^n, \bar{p}^{n-1})/\rho_{n-1}. \tag{35}$$

Note that with the help of the relations (32)–(35) the $\bar{A}^{\gamma-1}$-orthogonality property of residual vectors is easily verified, from which more economical formulas for calculating iterative coefficients follow:

$$(\bar{A}^{\gamma-1}\bar{r}^n, \bar{r}^k) = \sigma_n \delta_{n,k}, \quad \sigma_n = ||\bar{r}^n||^2_{\gamma-1}, \quad \beta_n = \sigma_n/\sigma_{n-1}. \tag{36}$$

The condition to stop the iterative processes (13)–(19) is the inequality $\|r^n\| \le \varepsilon \|f\|_0$, $\varepsilon \ll 1$ (for more details see [9]). The corresponding number of iterations is estimated by the quantity $n(\varepsilon) \le \frac{1}{2}\sqrt{cond(\bar{A})}\ log(2/\varepsilon)$, where $cond(\bar{A}) = \lambda_{max}/\lambda_{min}$ is the spectral condition number of the preconditioned matrix $\bar{A}$ in (31), and λ_{max} and λ_{min} are its maximum and minimum non-zero eigenvalues.

Note that if it is technologically more convenient to implement the considered Krylov methods with one-side preconditioning, that is, without using a factorized representation of the matrix B, then instead of (31)–(35) one can use, for example, the conjugate gradient method with an arbitrary initial vector u^0:

$$\begin{gathered}\tilde{p}^0 = \tilde{r}^0 = B^{-1}r^0, \quad r^0 = f - Au^0, \quad n = 1, 2, \ldots \\ u^n = u^{n-1} + \alpha_{n-1}\tilde{p}_{n-1}, \quad \alpha_n = \rho_n/\rho_n, \quad \tilde{r}^n = \tilde{r}^{n-1} - \alpha_{n-1}A\tilde{p}^{n-1}, \\ \sigma_n = (B^{-1}r^n, r^n), \quad \rho_n = (A\tilde{p}^n, \tilde{p}^n), \quad \tilde{p}^n = B^{-1}\tilde{r}^n + \beta_n\tilde{p}^{n-1}, \quad \beta_n = \sigma_n/\sigma_{n-1}.\end{gathered} \tag{37}$$

The solution of the auxiliary SLAEs with the matrices G and $A_{1,1}$ can be done using either direct or iterative algorithms. In the latter case, we will have two-level processes in the Krylov subspaces, which requires a careful control of the accuracy of successive approximations. A similar approach with the two-side preconditioning is also implemented for the three-dimensional case.

Now, we will provide a brief analysis of the performance of the proposed iterative methods when parallelizing their various stages on an a multi-processor system with distributed and shared memory. It is assumed that the calculations for each subdomain (including the macrogrid) are implemented simultaneously on the corresponding multi-core nodes by forming computational processes using MPI system which provide data exchanges at each iteration. The second level of parallelization is performed via multi-threaded computing tools (OpenMP) on the shared hierarchical memory of each node.

In general, the problem of optimizing algorithms and their mapping onto a multi=processor architecture is too complex, and we can only talk about individual aspects that contribute to improving a computing performance. If we talk about the main parameters of the considered s.p.d. SLAE, then there are few of them: N is the number of unknowns, nz is the number of non-zero entries of the matrix and $cond(A)$ is its spectral condition number. For the most relevant three-dimensional problems with large sparse matrices, we have $N \approx 10^8 - 10^{11}$, nz – from 7 to 100 coefficients for each equation, $cond(A) = 10^{17} - 10^{15}$ (in practice, this value is unknown and only rough estimates or hypotheses are possible regarding its value). Note that with the traditionally used double precision for 64-bit machines, an increasing the condition number to 10^{16} no longer guarantees an acceptable accuracy of the results. With the simplest model of a supercomputer, there are very few characteristics of its configuration: N_n is the number of nodes, N_k is the number of cores on one node, t_a is the average execution time of one arithmetic operation, t_c is the time of sending one number and t_d - the delay time, or setup, of one transaction. Taking into account the relations $t_a << t_c << t_d$, it is necessary to minimize the volume and the number of communications, and if possible, to combine data transfer with calculations.

Let us now consider the features of the algorithms we are developing to solve SLAEs on a given multi-processorsystem. Here it is necessary to clearly distinguish between the mathematical properties of the method (the resource intensity of one iteration and their total number) and the pecularities of its software implementation, which is largely related to the generated data structure.

First of all, the total amount of calculations is divided into two parts: those performed once before the start of the iterative process and those that are performed then at each iteration. The first group in our case includes those operations that are associated with the implementation of preconditioning matrices involved in the formulas (12), (13) for two-dimensional and three-dimensional cases. Not counting the initial data for the SLAE being solved 1, this includes the quantities $\hat{\beta}_t, \check{\beta}_t, \hat{u}_t, \check{u}_t$ from the sweep formulas (18), as well as the elements of the inverse matrices S_c^{-1} and $A_{1,1}^{-1}$, with the help of which the Schur complement is determined for equations on the macronetwork and in subdomains. The second part of solving large sparse SLAEs is responsible for the implementation of the iterative process. In ADDM approache, one of the most important issue consists in the balancing the total volumes of the arithmetic calculation and data communication. First of all, it depends on grid domain dimension (on the degree of freedoms (d.o.f.)) and on the number of subdomains.

Roughly speaking, the problem of optimally tuning ADDM parameters for an MPS architecture can be defined as minimizing the time to solve a given SLAE on a specific multi-node computing cluster architecture.

Let an algebraic system of order N be divided into M subsystems (subdomains), each of which is solved simultaneously at its "own" node during n_ϵ iterations. It is assumed that each node has P multi-core processors with shared memory, which allows parallelizing at the "internal" level the solutions of the

algebraic subsystem formed for the subdomains. The total implementation time of the ADDM under consideration can be represented by the formula

$$T = T_0 + n_\epsilon[(Q_1^a + Q_2^a)\tau_a + \tau_0 + (Q_1^c + Q_2^c)\tau_c] \tag{38}$$

where $\tau_a \ll \tau_c \ll \tau_0$ denotes the execution time of one arithmetic operation (on average), the transfer of one number between computing nodes and the delay (settings) of one transaction (transfer of one data array). The quantities T_0, n_ϵ, Q_1^a and Q_2^a, Q_1^c and Q_2^c are the duration of preliminary calculations before iterations, the number of operations performed, the number of arithmetic operations per each step when implementing the "macrogrid" part of the algorithm in subdomains, as well as the amount of information transferred from the macrogrid to the subdomains and vice versa.

The value T in (38) has a complex functional dependence on numerous parameters: the number of iterations n_ϵ increases with increasing values of N, M, the number of arithmetic operations Q_1^a and Q_2^a are proportional to the volumes of the subdomains, and Q_1^c and Q_2^c are proportional to the size of their "surfaces" (in terms of the number of nodes), etc. An approximate optimization of the performance of the software implementation of the algorithm can be carried out for a specific configuration of a multi-processor system when solving a given SLAE of higher dimension by appropriately selecting the number of subdomains. Obviously, if they increase excessively, both communication losses and the number of iterations will increase, so an optimum must exist.

5 Numerical Experiments

We will consider the results of numerical experiments on solving methodological s.p.o. SLAEs that approximate two-dimensional Dirichlet boundary value problems for the Poisson equation on square grids in a square computational domain. The main goal of the research in this case is to determine the rate of convergence of ADDM iterations on Krylov subspaces with the applied preconversion devices depending on d.o.f. tasks and the number of specified subdomains. All calculations are carried out with standard double precision; the parameter $\epsilon = 10^{-7}$ is selected as the stopping criterion for external conjugate gradient iterations on subdomains.

The Table 1 shows different values of N_e, $N_x = N_y = N_c$ and $M_x = M_y - M_c$ and the corresponding numbers n of iterations in subdomains. For the cases of applying the LU and CG methods, the results are almost the same.

Calculations were carried out for a model problem with an exact solution $u = 1$ with various initial approximations u^0, which weakly affect the number of iterations over subdomains (in the experiments the preconditioned conjugate gradient method was used). The compensating parameter in the incomplete factorization was taken as $\theta = 1$ (note that in this case, for $u^0 = 0$, the process converges in one iteration due to the equality $Be = Ae$). Auxiliary SLAEs in subdomains were solved either by the direct LU algorithm from the library EIGEN.

In all the runs, the error $\delta = sup|1 - u_{i,j}|$ does not exceed the value of 1.8e-06.

In each cell of the Table 1, the upper number is N_c, the mid number is N_e, the lower number is the number of iterations n. The empty cells here are explained by the hardware conditions of the experiments.

Table 1. The numbers of iterations in Krylov subspaces for different grid subdomains

$M_c = 2$	101	200	401	800	1601
	33	66	133	266	533
	20	29	41	57	81
$M_c = 4$	104	204	404	804	1604
	20	40	80	160	320
	28	39	55	75	104
$M_c = 8$	107	206	404	800	
	11	22	44	88	
	37	52	72	101	
$M_c = 16$	101	203	407		
	5	11	23		
	45	67	95		

Table 2. Computation times for different number of the grid points N, subdomains M_d, and the number processor (threads) $P = 1, 4, 16, 32$ (in each cell row by row from the left to the right).

N_e	32	64	128	256
M_d \ N	131^2	259^2	515^2	1027^2
16	1.19 0.67	9.07 4.94	71.9 37.6	525 287
	0.24 0.16	1.55 1.15	18.2 8.6	98 63.9
64	1.34 0.74	8.52 4.55	58.4 32.1	467 296
	0.35 0.16	1.66 0.86	9.8 6.82	87.8 56.6
256	1.44 0.91	9.74 5.52	60.2 33.4	422 285
	0.87 0.23	4.26 1.14	12.9 6.65	88.6 46.8
1024	1.95 1.77	10.8 7.23	71.3 44.2	448 252
	0.91 0.64	5.3 3.19	35.4 14.3	149 66.3

In the Table 2 we present the running times (in seconds, four values in each sell) of parallel computations for the different numbers of processors $P = 1, 4, 16, 32$.

The results demonstrate the good enough speedup $S_p = T_1/T_p$ of parallel computing, where T_p means the calculation time on P processors. The experiments were made on the server with 72 cores Intel Xeon Platinum 8354H.

An analysis of data from preliminary experimental studies suggests a fairly high efficiency of the proposed ADDM approach using a macrogrid in the sense that the number of iterations $n(\varepsilon)$ grows slightly as the value of the grid step h decreases and the number $N_D = (M_c + 1)^2$ of subdomains increases.

6 Conclusion

A new family of fully algebraic domain decomposition methods is proposed. It is based on using the set of macrogrid nodes provides a special separation of the subdomains. The original s.p.d. SLAE is presented in block-tridiagonal form which is solved by incomplete factorization with diagonal compensation conjugate gradient method. The subsystem for macroedges and macronodes is solved by the parallel direct algorithm. The auxiliary subdomain SLAEs are solved in parallel by direct methods. The efficiency of the iterative process and the performance of its parallel implementation are demonstrated by the results of numerical experiments for set of model grid boundary value problems.

Further increase in the performance of the software implementation of algorithms is planned on the basis of a hybrid parallelization of the computing process on supercomputers with distributed and hierarchical shared memory, as well as by using an asymptotically order-optimal algebraic multigrid (AMG) method for solving SLAEs in subdomains. Another possibility to increase the efficiency of the iterative process consists in an improvement of the preconditioning matrix B. And, of course, a generalization of the proposed ADDM approaches to a wider class of applied problems is a topic of a great interest.

Acknowledgements. Methodological results of the work were carried out under state contract with ICMMG SB RAS FWNM-2022-0001. Applied part the study was financially supported by RSF (Project No. 24-21-00402).

References

1. Dolean, V., Jolivet, P., Nataf, F.: An Introduction to Domain Decomposition Methods: Algorithms, Theory and Parallel Implementation. SIAM, Philadelphia (2015). https://doi.org/10.1137/1.9781611974065
2. Vassilevski, Y.V., Olshanskii, M.A.: Short Course on Multi-grid and Domain Decomposition Methods. MAKS Press Publ, Moscow (2007)
3. Quarteroni, A., Valli, A.: Domain Decomposition Methods for Partial Differential Equations. Clarendon Pecss, Oxford (1999)
4. Agoshkov, V.I., Lebedev, V.I.: Variational algorithms of the domain decomposition method. Sov. J. Num. Anal. Math. Model. **5**(1), 27–46 (1990)
5. Smith, B.F., Bjorstad, P.F., Group, W.D.: Domain Decomposition Parallel Multilevel Methods for Elliptic Differential Equations. Cambridge University. Press, Cambridge (1996). https://doi.org/10.1016/s0898-1221(97)90035-3
6. Widlind, O., Toselli, A.: Domain Decomposition Methods, Algorithms and Theory. In: Springer Series in Comput, Math. vol. 74, Berlin (2005). https://doi.org/10.1007/b137868
7. Korneev, V., Langer, U.: Domain Decomposition Methods and Preconditioning. In: Encyclopedia of Computational Mechanics, vol. 1, pp. 617–647. John Wiley and Sons, Ltd., Hoboken (2004). https://doi.org/10.1002/0470091355.ecm019
8. Laevsky, Y.M., Matsokin, A.M.: Decomposition methods for solving elliptic and parabolic boundary value problems. Sib. Zh. Vychisl. Mat. **2**, 361–372 (1999)

9. Il'in, V.P.: Iterative preconditioned methods in krylov spaces: trends of the 21st century. Comput. Math. Math. Phys. **61**(11), 1750–1775 (2021). https://doi.org/10.1134/S0965542521110099
10. Gander, G., Margoles, F., Nataf, F.: Optimized Schwarz methods without overlap for the Helmholtz equation. SIAM J. Sci. Comput. **21**, 38–60 (2002). https://doi.org/10.1137/S1064827501387012
11. Bern, M., Gilbert, J., Hendrickson, B., Nguyen, N., Toledo, S.: Support graph preconditioners. SIAM J. Matrix Anal. Appl. **27**, 930–951 (2006). https://doi.org/10.1137/S0895479801384019
12. Kuznetsov, Y.A.: Algebraic multigrid domain decomposition methods. Preprint N 232, Academy of Sciences of the USSR. Department of Computational Mathematics. Moscow (1989)
13. Nepomnyaschikh, S.V.: Decomposition and fictitious domains methods for elliptic boundary value problems. In: Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations (Norfolk, VA, 1991), SIAM, pp. 62–72, January 1992
14. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd ed. SIAM, New Delhi (2003)
15. D'yakonov, E.G.: On some direct and iterative methods based on matrix bordering. Numerical methods in mathematical physics. In: Proceedings of the seminar "Methods of computational and applied mathematics" under the leadership of G.I.Marchuk. Issue 4. Novosibirsk, pp. 45–68 (1979)
16. Il'in, V.P.: Problems of parallel solution of large systems of linear algebraic equations. J. Math. Sci. **216**, 795–804 (2016)
17. Il'in, V.P.: Multi-Preconditioned Domain Decomposition Methods in the Krylov Subspaces, LNCS, vol. 10187, pp. 95–106, Springer, 2017
18. Gurieva, Y.L., Ilyin, V.P., Kozlov D.I.: Parallel domain decomposition methods with graph preconditioning In: Parallel computing technologies (PAVT'2023). Short articles and descriptions of posters. Materials of the XVII All-Russian Scientific Conference with international participation. YuUrGU, Chelyabinsk, pp. 215–228 (2023)
19. Il'in, V.P., Kuznetsov, Y.I.: Tridiagonal Matrices and Their Applications, p. 205. Nauka, Moscow (1985). (in Russian)
20. Il'in, V.P.: Mathematical Modelling. Part 1. Continuous and discrete models (in Russian). Novosibirsk, izd. SO RAN, p. 428 (2017)
21. Il'in, V.P.: Methods of finite differences and finite volumes for elliptic equations. – Novosibirsk, ed. ICM&MG SB RAS, p. 318 (2001)
22. Batalov, M., Gurieva, Y., Ilyin, V., Petukhov, A.: On parallel multigrid methods for solving systems of linear algebraic equations. In: Sokolinsky, L., Zymbler, M. (eds.) Parallel Computational Technologies. PCT 2023. Communications in Computer and Information Science, vol. 1868, pp. 93–109 (2023)