ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2021, Vol. 61, No. 11, pp. 1750–1775. © Pleiades Publishing, Ltd., 2021.
Russian Text © The Author(s), 2021, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2021, Vol. 61, No. 11, pp. 1786–1813.

GENERAL NUMERICAL METHODS

Iterative Preconditioned Methods in Krylov Spaces: Trends of the XXI Century

V. P. Il'in[a,b]

[a] *Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch, Russian Academy of Sciences, Novosibirsk, 630090 Russia*

[b] *Novosibirsk State Technical University, Novosibirsk, 630073 Russia*

e-mail: ilin@sscc.ru

Received February 11, 2020; revised March 16, 2021; accepted July 7, 2021

Abstract—A analytic review of major problems and new mathematical and technological discoveries in methods for solving SLAEs is given. This stage of mathematical modeling is a bottleneck because the amount of the required computational resources grows nonlinearly with the increasing number of degrees of freedom of the problem. It is important that the efficiency and performance of computational methods and technologies significantly depend on how well the specific features of the class of application problems—electromagnetism, fluid dynamics, elasticity and plasticity, multiphase filtering, heat and mass transfer, etc. are taken into account. The development of Krylov iterative processes is mainly intended for the construction of two-level algorithms with various orthogonal, projective, variational, and spectral properties, including not only polynomial but also rational and harmonic approximation techniques. Additional acceleration of such algorithms is achieved on the basis of deflation and augmenting approaches using various systems of basis vectors. The goal of intensive studies is to construct efficient preconditioning operators on the basis of various principles: new multigrid schemes and parallel domain decomposition methods, multipreconditioning, nested and alternate triangular factorizations, low-rank and other algorithms for approximating inverse matrices, etc. High-performance and scalable parallelization are based on hybrid programming using internode message passing, multithreaded computations, vectorization, and graphics processing units (GPUs). Modern trends in mathematical methods and software are aimed at the creation of an integrated environment designed for a long lifecycle and massive innovations in important applications.

Keywords: sparse SLAE, preconditioning, iterative methods, Krylov subspaces, symmetric and asymmetric matrices, decomposition algorithms, multigrid approaches, approximate factorization

DOI: 10.1134/S0965542521110099

1. INTRODUCTION

Preconditioned iterative methods in Krylov subspaces for solving systems of linear algebraic equations (SLAEs) is an important achievement of computational mathematics in the 20th century. Their significance becomes especially high with the development of supercomputer simulation and solution of interdisciplinary direct and inverse, nonlinear, and nonstationary multidimensional problems with complicated geometric and material properties. Modern requirements for the accuracy of solving problems with real life data lead to superlarge systems (10^{10}–10^{11} and greater) and huge conditioning numbers (10^{14} and higher) when the standard double precision computations are already on the verge of being available. In this case, the computational cost of solving SLAEs exceeds 80% of the total computational cost for large-scale computer simulations.

Of the most interest for researchers are algebraic systems with sparse matrices that arise as a result of approximation of boundary value problems using finite element and finite volume methods and discontinuous Galerkin algorithms of various orders on unstructured grid (see the review in [1]). In these cases, the portraits or graphs of matrices have an irregular structure, i.e., the distribution of nonzero elements is specified only by their enumeration, and their values are stored in compressed formats, which is an important feature in the implementation of algorithms on modern supercomputers with heterogeneous structure and hierarchical shared memory. The spectral and structure properties of SLAEs are significantly different in different applications—electromagnetism, fluid dynamics, elasticity and plasticity, heat and mass transfer equations, etc., which gives rise to a large variety of algorithms, the development of

which goes in the direction of both the study of new iterative processes with various orthogonal, variational and projection properties and the construction of effective preconditioned matrices. A combination of these methodologies determines success in this area. The results of studies are described in the books by Axelsson [2], Elman, Silvester, and Wathen [3], Marchuk and Kuznetsov [4], Saad, [5], Van der Vorst [6], Olshanskii and Tyrtyshnikov [7], Liesen and Strakos [8]), the author of his paper [9, 10], and in a huge number of journal publications and conference proceedings.

Speaking about the generalization of Krylov iterative methods, we note that there are various approaches to enriching the corresponding subspaces in which residual or direction vectors for finding new iterative approximations are determined on the basis of various orthogonal of projective principles, including deflation and augmenting algorithms in combination with spectral optimization or least squares principle. It is of interest that there were attempts at constructing "alternatives" to methods in Krylov subspaces, such as Anderson acceleration (which was initially proposed for solving nonlinear systems [11–13]) and Sonneveld subspaces [14]); however, in actual fact they turned out to be variations on the general theme.

The ever increasing stream of literature on methods for solving SLAEs contains a rich palette of ideas for constructing preconditioning matrices that should be easily invertible, increase the convergence rate of iterative algorithms, and ultimately ensure their overall performance. Here multigrid methods, which are asymptotically optimal with respect to order (the amount of computations is proportional to the number of degrees of freedom of the discrete problem), gained a second youth. The methodologies of matrix decomposition have also been significantly developed, including low-rank matrix approximations and nested and alternate triangular factorizations. Various approaches formed a uniform technology for constructing multipreconditioned processes in Krylov subspaces, which are generally multilevel, and additive domain decomposition methods (DDMs), which are the main tool for parallelizing multidimensional problems on multiprocessors. Methods of scalable parallelization and improving the performance of DDMs on supercomputers of heterogeneous architecture with distributed and hierarchical shared memory form the key problem in modern computational algebra [5, 16–19]. Note that the methods for solving SLAEs arising in implicit approximations of initial boundary value problems have some specific features (see the review in [20]). In particular, it was shown in [21] that the choice of initial iterative approximations at each time step using a generalization of the predictor-corrector algorithm can significantly improve the efficiency of computations.

The practical demand for algebraic calculations has led over a half-century history to a huge amount of software, both free and commercial; a fairly complete list of such software can be found in [22], and a classification of algorithms is given in [23]. Here we pay attention to the tools of general importance like SPARSE BLAS and widespread libraries PETSc, HYPRE, MKL INTEL, etc. The international community of experts in computational algebra developed standardized formats for representing matrices and corresponding converters, as well as large collections of matrices from real-life simulation problems that are indispensable for testing and comparative analysis of algorithms. An important trend in recent decades is the transition from concrete libraries and packages of application programs to integrated computational environments, examples of which are DUNE, INMOST, OPEN FOAM, and MATLAB, and the basic simulation system, the concept of which is described in [24] and includes principles of flexible extension of the composition of models and algorithms, as well as the adaption to the evolution of computer architectures, reuse of external software, and coordinated participation of various development groups, which should facilitate the creation of an effective new generation ecosystem with a long lifecycle that joins the communities of mathematicians, software developers, and users from various application domains. On the whole, the high-performance solution of SLAEs for a wide class of practical problems is a promising field of activity for the intellectualization of both research and technological areas.

This paper is organized as follows. In Section 2, we describe the general modern principles of constructing iterative methods in Krylov subspaces, including two-level methods. Section 3 contains a brief review of approaches to constructing preconditioned matrices. Section 4 describes parallelization and performance improving technologies for solving SLAEs, including issues of their software implementation and effective use for simulating real-live processes and phenomena. In the final section, we discuss the prospects of developing the issues under consideration.

2. ORTHOGONAL AND VARIATIONAL PRINCIPLES OF DESIGNING KRYLOV ITERATIVE METHODS

We consider real algebraic systems

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

with positive semi-definite matrices

$$(Av, v) \geq \delta \|v\|^2, \quad \delta \geq 0, \quad (v, w) = \sum_{i=1}^{N} v_i w_i, \quad \|v\|^2 = (v, v),$$

among which an important class consists of symmetric positive definite (s.p.d.) matrices with $\delta > 0$. The SLAEs under examination can be represented in block form

$$\begin{gathered} A_{q,q}u_q + \sum_{r \in \Omega q} A_{q,r}u_r = f_q, \quad q = 1, 2, \ldots, P, \\ A = \{A_{q,r}\}, \quad A_{q,r} \in \mathcal{R}^{N_q, N_r}, \quad f_q \in \mathcal{R}^{Nq}, \quad N_1 + \ldots + N_P = N, \end{gathered} \tag{2}$$

where Ω_q is the set of indices of matrix rows forming the qth block row of the matrix A and P is its block order.

One of important block structures gives saddle algebraic systems

$$Au \equiv \begin{bmatrix} D & C^{\mathrm{T}} \\ C & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ 0 \end{bmatrix} = f, \quad D \in \mathcal{R}^{N_1, N_1}, \quad C \in \mathcal{R}^{N_2, N_1}; \quad u_1, f_1 \in \mathcal{R}^{N_1}, \tag{3}$$

which arise from the approximation of mixed statements of boundary value problems and in optimization methods. The matrix A in (3) is invertible if the following conditions are fulfilled:

$$\operatorname{rank}(D) = N_1, \quad \operatorname{rank}(C) = N_2, \quad \ker(D) \cap \ker(C) = \{0\}, \quad N_2 \leq N_1.$$

Along with the original matrix, we will consider preconditioning matrices, which we will denote by the symbol B with super or subscripts. The choice of preconditioners is based on efficient invertibility and improvement of conditioning of preconditioned matrices of form $B^{-1}A$ or AB^{-1}, which together should improve the performance of the algorithm software implementation.

A general representation of the stationary iterative process for solving SLAE (1) is the so-called first canonical form

$$u^{n+1} = u^n + B^{-1}r^n, \quad r^n = f - Au^n,$$

where r^n is the residual vector. It is associated with the second canonical form

$$u^{n+1} = Tu^n + g, \quad T = (1 - B^{-1}A), \quad g = -B^{-1}f,$$

where T is called the transition matrix. A criterion for stopping the iterative process is the fulfillment of the following conditions (for details see [25, 26])

$$\left\|r^n\right\| \leq \varepsilon_1 \|f\| + \varepsilon_2 \|A\| \cdot \left\|u^n\right\|, \quad 0 < \varepsilon_1, \varepsilon_2 \ll 1;$$

however, it is often assumed that $\varepsilon_2 = 0$ for simplicity. The issue of estimating the iterative approximation error $\left\|u - u^n\right\|$ is nontrivial, especially if machine arithmetic precision is taken into account. This issue is considered in a large number of studies (see [8] and references therein).

2.1. Algorithms for Symmetric SLAEs

The inception and the first period of development of iterative methods in Krylov subspaces, which may be dated to 1950–1965, should be attributed to the authors of pioneering works by Lanczos (1950, 1952), Arnoldi (1951), Hestsenes and Stiefel (1951, 1952), Kreig (1954, 1955), Vorob'ev (1958), Faddeev and Faddeeva (1958, 1960), Fridman (1962), Golub (1965), and Varga (1962). The list of subsequent studies is in the hundreds.

We begin the consideration with algorithms for solving symmetric SLAEs with the purpose of giving a uniform view at this intensively developing area of research and discuss important issues of computational performance of methods.

Let the iterative approximation u^n and the corresponding residual vector $r^n = f - Au^n$ be calculated by the formulas

$$\begin{aligned} u^n &= u^{n-1} + \alpha_{n-1}p^{n-1} = u^0 + \alpha_0 p^0 + \ldots + \alpha_{n-1}p^{n-1}, \\ r^n &= r^{n-1} - \alpha_{n-1}Ap^{n-1} = r^0 - \alpha_0 Ap^0 - \ldots - \alpha^{n-1}Ap^{n-1}, \end{aligned} \tag{4}$$

where α_k and p^k are iteration parameters and direction vectors. In this paper, we describe the family of conjugate direction algorithms characterized by various variational and projective properties determined by the corresponding scalar products and norms of vectors

$$(u, v)_\gamma = (A^\gamma u, v), \quad \|u\|_\gamma^2 = (u, u)_\gamma, \tag{5}$$

where the exponent is $\gamma = 0, 1, 2$. Let the direction vectors be A^γ-orthogonal, i.e.,

$$(A^\gamma p^n, p^k) = \rho_k^{(\gamma)}\delta_{n,k}, \quad \rho_k^{(\gamma)} = (A^\gamma p^k, p^k) = \|p^k\|_\gamma^2, \tag{6}$$

where $\delta_{n,k}$ is the Kronecker symbol. Then, the residuals r^n satisfy the relations

$$\Phi_\gamma(r^n) = (A^{\gamma-2}r^n, r^n) = (r^0, r^0)_{\gamma-2} - \sum_{k=0}^{n-1}[2\alpha_k(r^0, A^{\gamma-1}p^k) - \alpha_k^2\rho_k]. \tag{7}$$

This implies that, at the values of the parameters

$$\alpha_k = \sigma_k/\rho_k, \quad \sigma_k = (r^0, p^k)_{\gamma-1} = (r^k, p^k)_{\gamma-1}, \tag{8}$$

the functionals Φ_γ take the minimum values

$$\Phi_\gamma(r^n) = \|r^0\|_{\gamma-2} - \sum_{k=0}^{n-1}(r^0, p^k)_{\gamma-1}^2/\rho_k. \tag{9}$$

Here and below, the index γ of the coefficients and vectors is dropped for simplicity, and the matrix A is assumed to be nonsingular for the time being.

There are two ways to satisfy conditions (6). The first one is to use the Lanczos A^γ-orthogonalization, which can be written as

$$\begin{aligned} p^0 &= 0, \quad \overline{\beta}_1 p^1 = f, \quad k = 1, 2, \ldots, n: \\ \overline{\beta}_{n+1}p^{n+1} &= Ap^n - \overline{\alpha}_n p^n - \overline{\beta}_n p^{n-1}, \quad \overline{\alpha}_n = (p^n, p^n)_{\gamma+1}, \end{aligned} \tag{10}$$

where $\overline{\beta}_n$ are chosen from the condition $\|p\|_\gamma = 1$. Another way (by Hestenes and Stiefel), which is most widespread due to its stability, uses the two-term recurrence

$$p^0 = r^0, \quad p^n = r^n + \beta_{n-1}p^{n-1}, \quad \beta_{n-1} = (A^\gamma r^n, p^{n-1})/\rho_{n-1}, \tag{11}$$

which, in combination with the formula for the residual in (4), yields other relations for the direction vectors:

$$p^n = (1 + \beta_{n-1})p^{n-1} - \alpha_{n-1}Ap^{n-1} - \beta_{n-2}p^{n-2}. \tag{12}$$

Using (4) and (11), we can obtain the property of $A^{\gamma-1}$-orthogonality of residual vectors and more convenient (for $\gamma = 1, 2$) formulas for the iteration coefficients

$$(A^{\gamma-1}r^n, r^k) = \sigma_k\delta_{n,k}, \quad \sigma_k = \|r^n\|_{\gamma-1}^2, \quad \beta_k = \sigma_{k+1}/\sigma_k. \tag{13}$$

However, α_k in the case $\gamma = 0$ must be calculated in a different way. Since the exact solution of the SLAE can be represented as the expansion in the basis

$$u = u^0 + \alpha_0 p^0 + \ldots + \alpha_{m-1}p^{m-1}, \quad m \le N,$$

The vector error of the iterative approximation and the corresponding residual are written as

$$\begin{aligned} v^n &= u - u^n = \alpha_n p^n + \ldots + \alpha_m p^m, \\ r^n &= Av^n = \alpha_n Ap^n + \ldots + \alpha_m Ap^m. \end{aligned} \tag{14}$$

Hence, using the A^γ-orthogonalization of the vectors p^k, we have

$$\alpha_n = (v^n, p^n)_\gamma / \left\|p^n\right\|_\gamma^2 = -\alpha_{n-1}(v^n, Ap^{n-1}) / \left\|p^n\right\|_\gamma^2 = -\alpha_{n-1}(r^n, p^{n-1})_\gamma / \left\|p^n\right\|_\gamma^2. \tag{15}$$

Here we used the orthogonality of p^n to the vectors p^{n-1} and p^{n-2} and its relation to the vector Ap^{n-1} in (12). The coefficient β_{n-1} must be calculated by formula (11).

The iterative processes of interest possess optimal properties thus ensuring the minimization of the corresponding functionals $\Phi_\gamma(r^n)$ of form (7) in Krylov subspaces

$$\mathcal{K}_n(r^0, A) = \mathrm{Span}(r^0, Ar^0, \ldots, A^{n-1}r^0). \tag{16}$$

In the case $\gamma = 0, 1, 2$, these algorithms are called minimum iteration or minimum error methods (see [8]) and conjugate gradients and conjugate residuals, respectively [5, 9, 27].

The approaches described above admit simple generalization for the SLAEs preconditioned using some s.p.d. matrices B. To preserve the symmetry of systems, this should be done by two-sided preconditioning with the help of the formally introduced matrix $B^{1/2}$. As a result, system (1) takes the form

$$\overline{A}\overline{u} = \overline{f}, \quad \overline{A} = B^{-1/2}AB^{-1/2}, \quad \overline{u} = B^{1/2}u, \quad \overline{f} = B^{-1/2}f. \tag{17}$$

As a result of applying the conjugate directions formulas to SLAE (17), we obtain after certain transformations the following iterative process for the case $\gamma = 1, 2$:

$$\begin{gathered} \hat{p}^0 = \hat{r}^0 = B^{-1}r^0 = B^{-1}(f - Au^0), \quad n = 1, 2, \ldots, \\ u^n = u^{n-1} + \alpha_{n-1}\hat{p}^{n-1}, \quad \hat{r}^n = \hat{r}^{n-1} - \alpha_{n-1}A\hat{p}^{n-1}, \\ \hat{p}^n = \hat{r}^n + \beta_{n-1}\hat{p}^{n-1}, \quad \alpha_{n-1} = \sigma_{n-1}/\rho_{n-1}, \quad \beta_{n-1} = \sigma_n/\sigma_{n-1}, \\ \sigma_{n-1} = (A^{\gamma-1}\hat{r}^{n-1}, \hat{r}^{n-1}), \quad \rho_{n-1} = (B^{-1}A\hat{p}^{n-1}, A^{\gamma-1}\hat{p}^{n-1}), \end{gathered} \tag{18}$$

where the new vectors are related to the old ones by the relations $\hat{p}^n = B^{-1}p^n$, $\hat{r}^n = B^{-1}r^n$. For the minimum iteration method with $\gamma = 0$, the parameters α_{n-1} and β_{n-1} should the calculated by formulas (15) and (11) in which r^k and p^k are replaced by $\overline{r}^k = B^{-1}r^k$ and $\overline{p}^k = B^{-1}p^k$, respectively.

Remark 1. In practice, it is often required to solve a series of SLAEs with the same matrix but successively computed different right-hand sides. In this case, a significant amount of computations can be avoided by memorizing the calculated vectors p^k and Ap^k after solving the first system and somehow using them for expanding the Krylov basis when solving the subsequent SLAEs. These issues are discussed in a large number of papers (see [28] and the list of 157 works therein). However, the significant improvement of "informativeness" of the basis of iterative processes requires further study.

Remark 2. In book [8], the following methodological circumstance was mentioned: Methods in Krylov subspaces are closely related to the problem of moments, which plays an important role in the theory of operators and many applications. The beginnings of this topic were laid by Chebyshev, Markov, and Stieltjes.

In many applications, algebraic systems with symmetric matrices that have a sign definite spectrum are of interest. Such SLAEs can be singular, in particular, consistent or inconsistent. Inconsistent systems have a generalized, rather than classical, solution in the sense of least squares. For these cases, the normal (or pseudoinverse) solution is defined that has the minimum norm and minimizes the residual:

$$\min \|u\|_2 : u = \operatorname{argmin} \|f - Au\|_2. \tag{19}$$

Such a solution always exists and is unique, and its formal representation is obtained using the left Gauss transformation of the original system ($A^{\mathrm{T}}Au = A^{\mathrm{T}}f$); this representation is

$$u^{+} = (A^{\mathrm{T}}A)^{+}f = (A^{2})^{+}Af, \tag{20}$$

where A^{+} is the psedouinverse or generalized inverse matrix of A.

For solving such SLAEs, Saunders with his colleagues (see review [29]) developed the methods SYMMLQ, MINRES, and MINRES–QLP based on the Lanczos A-orthogonalization, which, due to recurrences (10), can be written in matrix form

$$AP_k = P_{k+1}T_k, \quad T_k = \begin{bmatrix} \bar{\alpha}_1 & \bar{\beta}_1 & & \\ \bar{\beta}_2 & \bar{\alpha}_2 & & \\ & \ddots & \ddots & \bar{\beta}_k \\ & & \bar{\beta}_k & \bar{\alpha}_k \\ & & & \bar{\beta}_{k+1} \end{bmatrix}, \tag{21}$$

where $P_k = [p^0, p^1, \ldots, p^{k-1}]$. Then, the approximate solution is sought in the form $u^k = P_k q^k$ and is reduced to finding a vector $q^k \in \mathcal{R}^k$ that minimizes the residual

$$r^k = f - AP_k q^k = \beta_1 p^1 - P_{k+1}T_k y^k. \tag{22}$$

This problem is solved using the LQ- or QR-transformations of the tridiagonal matrix T_k with the orthogonal matrix Q on the basis of stable Hausholder reflection operations. The results of representative numerical experiments presented by the authors demonstrate good efficiency of the algorithms, and their software implementations are available on the Internet.

The convergence rate of iterations in Krylov-type methods for positive semidefinite SLAEs is characterized by the bound on the number of iterations $n(\varepsilon)$ required for suppressing the initial error by a factor of ε^{-1}:

$$n(\varepsilon) \le \frac{1}{2}\sqrt{\kappa}\log_e \frac{2}{\varepsilon}, \quad \kappa = \lambda_{\max}/\lambda_{\min};$$

here κ is generally the effective condition number, which for singular matrices is represented in terms of the minimum nonzero eigenvalue $\lambda_{\min}$.

For symmetric SLAEs, the problem of stable numerical solution was basically solved in the 20th century, including the case of sign indefinite, singular, and inconsistent systems. As for the classical conjugate gradient method, we want to mention Kaporin's ingenious theory of convergence [30] based on the condition K-number introduced by him; this number is expressed in terms of the trace and determinant of the matrix.

More specifically, consider the preconditioned conjugate gradient method in the following form:

$$r^0 = f - Au^0, \quad p^0 = B^{-1}r^n, \quad n = 0, 1, \ldots,$$

$$u^{n+1} = u^n + \alpha_n p^n, \quad r^{n+1} = r^n - \alpha_n Ap^n,$$

$$p^{n+1} = B^{-1}r^{n+1} + \beta_n p^n, \quad \alpha_n = \sigma_n/\rho_n,$$

$$\beta_n = \sigma_{n+1}/\sigma_n, \quad \sigma_n = (r^n, B^{-1}r^n), \quad \rho_n = (p^0, Ap^n).$$

In this case, it was proved in [31] that the condition

$$(r^n, B^{-1}r^n) \le \varepsilon^2 (r^1, B^{-1}r^n)$$

is fulfilled if the number of iterations is

$$n(\varepsilon) = \log_2 K + \log_2 \varepsilon^{-1},$$

where K is the condition K-number of the preconditioned matrix $B^{-1}A$ defined by

$$K = K(B^{-1}A) = \left[\frac{1}{N}\operatorname{trace}(B^{-1}A)\right]^N \Big/ \det(B^{-1}A).$$

In [32], iterative algorithms were optimized for a number of preconditioners so as to minimize the condition K-number.

2.2. Krylov Algorithms for Asymmetric SLAEs

We continue the description of specific approaches with a wide class of multipreconditioned semiconjugate direction (SCD) algorithms [31]. In the general block form, such iterative methods in Krylov subspaces are written as

$$\begin{gathered} r^0 = f - Au^0, \quad n = 0, \ldots: \quad u^{n+1} = u^n + P_n\overline{\alpha}_n, \\ r^{n+1} = r^n - AP_n\overline{\alpha}_n = r^q - AP_q\overline{\alpha}_q - \cdots - AP_n\overline{\alpha}_n, \quad 0 \le q \le n, \\ P_n = (p_1^n, \ldots, p_{M_n}^n) \in \mathcal{R}^{N,M_n}, \quad \overline{\alpha}_n = (\alpha_{n,1}, \ldots, \alpha_{n,M_n})^{\mathrm{T}} \in \mathcal{R}^{M_n}. \end{gathered} \tag{23}$$

Here $p_1^n, \ldots, p_{M_n}^n$ are the direction vectors constituting at the nth iteration the matrix P_n, and $\overline{\alpha}_n$ is the vector of iteration parameters. For the vectors p_k^n in (23), we assume for the time being that only the orthogonality conditions

$$\begin{gathered} (Ap_k^n, A^\gamma p_{k'}^{n'}) = \rho_{n,k}^{(\gamma)}\delta_{n,n'}^{k,k'}, \quad \rho_{n,k}^{(\gamma)} = (Ap_k^n, A^\gamma p_k^n), \\ \gamma = 0, 1, \quad n' = 0, 1, \ldots, n-1, \quad k, k' = 1, 2, \ldots, M_n \end{gathered} \tag{24}$$

are fulfilled. However, if we define the coefficients $\overline{\alpha}_n = \{\alpha_{n,l}\}$ by the formulas

$$\alpha_{n,l} = \sigma_{n,l}/\rho_{n,n}^{(\gamma)}, \quad \sigma_{n,l} = (r^0, A^\gamma \overline{p}_l^n), \tag{25}$$

then (23) implies the following expressions for the residual functionals:

$$\Phi_n^{(\gamma)}(r^{n+1}) \equiv (r^{n+1}, A^{\gamma-1}r^{n+1}) = (r^q, A^{\gamma-1}r^q) - \sum_{k=q}^{n}\sum_{l=1}^{M_n}(r^q, A^\gamma p_l^k)^2 \Big/ \rho_{k,l}^{(\gamma)}, \quad q = 0, 1, \ldots, n; \tag{26}$$

they attain their minimums in the block Krylov subspaces

$$K_M = \operatorname{Span}\{p_1^0, \ldots, p_{M_0}^0, Ap_1^1, \ldots, Ap_{M_1}^1, \ldots, Ap_1^n, \ldots, Ap_{M_n}^n\}, \quad M = M_0 + M_1 + \cdots + M_n, \tag{27}$$

at $\gamma = 1$, and in the case when the matrix A is skew symmetric also at $\gamma = 0$. Note that in the case of asymmetric matrices and $\gamma = 0$, the application of algorithms is actually restricted to SLAEs with the matrices that have a positive definite symmetric part $A_s = 0.5(A + A^{\mathrm{T}})$ because, e.g., for skew symmetric matrices, $(Au, u) = 0$ for $u = \{1\}$.

The orthogonality property of the direction vectors (24) can be ensured if they are defined using "multipreconditioned" recurrences in which each vector p_l^{n+1} is assigned a specific preconditioning matrix $B_{n+1,l}$:

$$\begin{gathered} p_l^0 = B_{0,l}^{-1}r^0, \quad p_l^{n+1} = B_{n+1,l}^{-1}r^{n+1} - \sum_{k=0}^{n}\sum_{l=1}^{M_k}\beta_{n,k,l}^{(\gamma)}p_l^k, \quad n = 0, 1, \ldots, \\ B_{n,l} \in \mathcal{R}^{N,N}, \quad i = 1, \ldots, M_n; \quad \gamma = 0, 1, 2, \\ \overline{\beta}_{n,k}^{(\gamma)} = \{\beta_{n,k,l}^{\gamma}\} = (\beta_{n,k,1}^{(\gamma)} \ldots \beta_{n,k,M_n}^{(\gamma)})^{\mathrm{T}} \in \mathcal{R}^{M_n}, \\ \beta_{n,k,l}^{(\gamma)} = -(A^\gamma p_l^k, AB_{n+1,l}^{-1}r^{n+1})\Big/\rho_{n,l}^{\gamma}, \quad n = 0, 1, \ldots, \quad k = 0, 1, \ldots, n, \quad l = 1, 2, \ldots, M_n. \end{gathered} \tag{28}$$

If the matrix A is symmetric, then the recurrence data turn from long ones into two-term data and we obtain the conjugate gradient and conjugate residual methods (for $\gamma = 0, 1$, respectively, in multipreconditioned and classical versions). For asymmetric SLAEs, these algorithms are called semi-conjugate gradient or semi-graduate residual (SCG or SCR) methods. In block (multipreconditioned) conjugate direc-

tion methods for solving symmetric SLAEs, formulas (23)–(25) remain the same, and the direction matrices P_n are calculated using the two-term recurrences

$$P_{n+1} = Q_{n+1} + P_n\overline{\beta}_n^{(\gamma)}, \quad Q_{n+1} = \{q_l^{n-1} = B_{n+1,l}^{-1}r^{n+1}\} \in \mathcal{R}^{N,M_{n+1}},$$
$$\overline{\beta}_n^{(\gamma)} = \{\beta_{n,l}^{(\gamma)} = \beta_{n,n,l}^{(\gamma)}\} \in \mathcal{R}^{M_n}.$$

These formulas, as well as (28), contain "built-in" preconditioning matrices. If we set $B_{n+1,l} = I$ and $M_n = 1$ for all n, then we obtain the Krylov process in its "pure form" without preconditioning. Note that formulas (28) implement the Gram–Schmidt orthogonalization algorithm, the stability of which can be improved by using the modified Gram–Schmidt (MGS) orthogonalization method [5, 33].

A feature of the algorithms considered here when they are applied to solving ill-conditioned asymmetric SLAEs is their high demand for resources in the sense of the amount of computations and memory when the number of iterations is large. A means for mitigating this drawback is to reduce the number of stored and used direction vectors, which can be achieved using two approaches. The first one is to reduce the recurrence by taking into account only its last m vectors. The second approach is to periodically restart the process after each m iterations, i.e., calculate the residual vector from the original equation as at the zero iteration rather than from the recurrence formula:

$$r^{n_t} = f - Au^{n_t}, \quad n_t = mt, \quad t = 0, 1, \ldots; \tag{29}$$

here t is the restart index (the further computations up to $n = n_{t+1}$ are performed using recurrence (23)). Both approaches significantly slow down the iterative process.

To eliminate this stagnant effect, it is proposed to add a second level of iterations using the least squares method (LSM) [34]. Suppose that we know the restart approximations $u^{n_0}, u^{n_1}, \ldots, u^{n_t}$, $n_0 = 0$. Then to correct the iteration vector u^{n_t}, which is the initial one for the next restart period of method (23)–(29), we will use the linear combination

$$\hat{u}^{n_t} = u^{n_t} + b_1v_1 + \ldots + b_tv_t = u^{n_t} + v^{n_t}, \quad v^{n_t} = V_t\overline{b}, \quad \overline{b} = (b_1, \ldots, b_t)^{\mathrm{T}},$$
$$V_t = \{v_k = u^{n_k} - u^{n_{k-1}}, k = 1, \ldots, t\} \in \mathcal{R}^{N,t}, \tag{30}$$

in which the vector of coefficients $\overline{b}$ is found by minimizing the norm $\|r^{u_t}\|$ of the residual by solving the overdetermined algebraic system

$$W_t\overline{b} = r^{n_t} \equiv f - Au^{n_t}, \quad W_t = AV_t. \tag{31}$$

This SLAE can be solved, e.g., using the QR- or SVD-decompositions of the matrix W_t. The normal solution with the minimum norm $\|\overline{b}\|$ is found by applying to (31) the left Gauss transformation

$$W_t^{\mathrm{T}}W_t\overline{b} = W_tr^{n_t}.$$

A simpler SLAE in the sense of its decreased condition number is obtained by multiplying system (31) by the matrix V_t^{T} on the left:

$$C_t\overline{b} \equiv V_t^{\mathrm{T}}AV_t\overline{b} = V_t^{\mathrm{T}}r^{n_t}. \tag{32}$$

If V_t is a full rank matrix, then the matrices A and C_t are simultaneously nonsingular. In this case, we have for the correction vector in (30) the formula $v^{n_t} = B_tr^{n_t} \equiv V_t(V_t^{\mathrm{T}}AV_t)^{-1}V_t^{\mathrm{T}}r^{n_t}$, where $B_t = V_t\hat{A}^{-1}V_t^{\mathrm{T}}$ ($\hat{A} = V_t^{\mathrm{T}}AV$) is a low-rank approximation of the matrix A^{-1}. In this approach, all restart vectors are stored in corrected form, and the corresponding residuals are found by the formula $r^{n_t} = f - Au^{n_t}$. If a matrix under consideration is not invertible, then a generalized inverse matrix is used. Numerous experiments with the use of LSM for accelerating Krylov processes with restarts show its high efficiency.

One more possibility of improving the performance of SCD methods with restarts is to store all direction vectors p^n and the vectors Ap^n obtained during the first restart period and use them in the computations in the subsequent restart periods rather than calculate new vectors p^n and Ap^k.

The class of semiconjugate direction method with dynamic multipreconditioning is equivalent in terms of convergence rate to other known algorithms of solving asymmetric SLAEs in Krylov subspaces, among which the most popular one is the generalized minimal residual method GMRES based on Arnoldi orthogonalization and its various versions. This algorithm was proposed by Saad and Schultz in 1986 [5], and it received wide and well-deserved popularity. Among numerous studies of this method, we distinguish the results by Knizhnerman [35] on estimating the error of Arnoldi orthogonalization.

Another principle of constructing Krylov iterative processes for solving asymmetric SLAEs is based on constructing sequences of biorthogonal vectors. In this case, the direction vectors are calculated by short (two-term) recurrences, but two computations of vector-matrix product is required at each iteration step. The biorthogonalization ideas underlie the algorithms BiCG, CGS, and BiCGStab and their various modifications; later, their analogs with A- biorthogonalization of direction vectors were invented (see review in [36]). The closely related family of induced dimension reduction (IDR) (see [37] and references therein) is based on Sonneveld spaces. Among other approaches, we recommend methods of quasi-minimal residuals and least squares method (LSQR and LSMR) [38]. In recent years, algorithms based on bidiagonal Kreig–Golub–Kahan transformations has also revived (they were originally proposed in 1955 and 1965); their informative study, generalization, and comparative analysis can be found in [25, 39]. Finally, we note flexible conjugate gradient methods (FCG) [40], which are a generalization of the classical CG methods (including preconditioned ones) to the asymmetric case.

3. METHODS OF SLAE PRECONDITIONING

In this section, in the description of algorithms we will discuss at the qualitative level issues of their potential parallelization. A more detailed discussion of technological aspects determining the performance of parallel implementation will be given later.

In addition to the use of a preconditioning matrix considered in (17) and (18), this also can be done by direct preconditioning (left, right, or two-sided) of the original SLAE:

$$\begin{gathered}\overline{A}u = \overline{f}, \quad \overline{A} = B^{-1}A, \quad \overline{f} = B^{-1}f,\\ \hat{A}\hat{u} = AB^{-1}Bu = f, \quad \hat{A} = AB^{-1}, \quad \hat{u} = Bu,\\ \breve{A}\breve{u} = B_1^{-1}AB_2^{-1}B_2u = B_1^{-1}f = \breve{f}, \quad \breve{A} = B_1^{-1}AB_2^{-1}, \quad \breve{u} = B_2u,\end{gathered} \tag{33}$$

where B, B_1, and B_2 are nonsingular matrices. Note that if A is symmetric, then to keep the matrix $\breve{A}$ symmetric, one should set $B_1 = B_2^{\mathrm{T}}$ in the last case. Since the matrices $\overline{A}$, $\hat{A}$, or $\breve{A}$ obtained from preconditioned algebraic systems are better conditioned, they should be solved using Krylov iteration formulas in pure form without additional preconditioning (although formally it is not prohibited, and multipreconditioning may be applied using a modification of both the iterative process (28) and the SLAE itself according to (33)).

Among the simple preconditioning methods, we mention scaling and binormalization of matrices. In the first method, a congruent diagonal transformation ($\tilde{A} = DAD$) is performed to obtain the unit principal diagonal; the second method is based on leveling the norms of rows and columns of the matrix (see review in [41, 42]). Such approaches may be used as preliminary ones in combination with other preconditioners.

For the block structure of SLAE (2), it seems the most natural to solve it using the Jacobi block method, which in a slightly generalized form is written as

$$B_{q,q}u_q^{n+1} \equiv (A_{q,q} + \theta D_q)u_q^{n+1} = f_q + \theta D_q u_q^n - \sum_{r\in\Omega_q} A_{q,r}u_r^n, \quad q = 1, 2, \ldots, P. \tag{34}$$

Here $\theta \in [0, 1]$ is an iteration (compensating) parameter, and D_q is the diagonal matrix determined by the equation

$$D_q e = \sum_{r\in\Omega_q} A_{q,r}e, \quad e = (1, \ldots, 1)^{\mathrm{T}} \in \mathcal{R}^{Nq}, \tag{35}$$

which is called the compensation or filtering condition (sometimes, the introduction of matrix D_q of form (35) is called lumping). In some cases, the choice of θ can be justified and described theoretically, and it helps significantly accelerate the iterative process. There are generalizations of the compensation

principle (35) based on a complication of the matrices D_q with banded structure and on the use of several compensating (filtering) vectors [2, 9, 10]:

$$D_q y_s = \sum_{r \in \Omega_q} A_{q,r} y_s, \quad s = 1, 2, \ldots, S; \tag{36}$$

however, this issue requires further study.

An alternative analog of the block Jacobi method, which belongs to the class of simultaneous displacement iterative or additive algorithms, is the Seidel method and its elaborations—successive over-relaxation and symmetric successive over-relaxation (SOR and SSOR) algorithms. The last method with some modification is also known under the names of alternate triangular, incomplete factorization, and explicit alternate direction scheme (see [2, 9]). These successive displacement or multiplicative methods (the new approximation u_q^{n+1} can be made only after the preceding subvectors $u_1^{n+1}, \ldots, u_{q-1}^{n+1}$ have been found) have improved convergence rate, and the accelerating compensation principle (35), (36) can be applied to them. However, such algorithms are based on inverting triangular matrices, which is difficult to parallelized. For this reason, in the age of multiprocessor supercomputers, they turned out to be less popular, and we do not consider them in this paper. On the other hand, it was shown in [43] that the application of adaptive successive displacement downstream methods can significantly reduce the number of iterations.

Numerous preconditioning methods have theoretical justification and estimates of the convergence rate (see review in [44–46]). A special case is the dynamic or flexible preconditioning with variable matrices [17, 47–51]. In iterative processes applied to solving SLAEs arising from approximation of boundary value problems on a grid with the characteristic step h that have the condition number $O(h^{-2})$, its magnitude is usually reduced to $O(h^{-1})$ due to preconditioning. As a result, the number of iterations in Krylov-type methods is estimated by $n(\varepsilon) = O(h^{-1/2})$. However, it should be taken into account that these asymptotic expressions can include large constants in "bad" problems with a large spread of values of the matrix A elements.

3.1. Domain Decomposition Algorithms

When grid SLAEs (especially of node type) are solved, each component of the vector to be found is associated with one grid node, and the structure and portrait of the matrix (the configuration of its nonzero elements) are isomorphic to the grid graph (which is directed for asymmetric matrixes and undirected for symmetric ones). In this case, the matrix block structure of form (2) is geometrically represented by the decomposition of the grid computation domain Ω into grid subdomains Ω_q in each of which a subproblem algebraically corresponding to the diagonal block $A_{q,q}$ is formed. Historically, the priority in decomposition methods belongs to the theory of multidimensional boundary value problems since the 19th century, when the solution of differential equations in complex computational domains using the Schwarz alternating method was reduced to examining a sequence of auxiliary problems in overlapping subdomains using the Dirichlet conditions on the introduced internal boundaries. Later, such approaches were generalized on the continuous level in the works by Smelov, Lions, Matsokin, Widlund, Nataf with his colleagues, Vasilevski, Olshanskii, and many other researchers (see reviews of literature in [5, 7, 17–19, 52–57]), and they also were extended to discrete or grid level. In the age of multiprocessor supercomputers, decomposition methods became the main tool for parallelizing computations in the solution of multidimensional initial boundary problems with complex real-life data that require high resolution and speed of computations.

The DDMs may be classified according to three main characteristics. The first one is the dimension of decomposition depending on one, two, or three families of coordinate planes are used for decomposition. The second characteristic is the size of subdomain overlappings, which in a simple case are specified without overlapping, and parameterized overlappings are constructed on the grid level by the layer-by-layer expansion of each subdomain by elementary grid cells. The third direction of DDM development is based on generalizing the types of interface boundary conditions when iterating over subdomains.

On the whole, DDMs are implemented as a two-level iterative process in Krylov subspaces—the upper level is the additive Jacobi–Schwarz block method (34), which for simplicity in its stationary version can be written as

$$u^{n+1} = u^n + B_1^{-1}(f - Au^n), \quad u^n = \{u_q^n\}, \quad B_1 = \text{block} - \text{diag}\{B_{q,q}\}, \tag{37}$$

where B_1 is the corresponding preconditioner that will actually be used in the Krylov process. At each nth iteration (37), auxiliary problems are simultaneously or concurrently solved in the subdomains Ω_q; formally, this solution is reduced to inverting the preconditioner blocks $B_{q,q}$. This can be done using direct and iterative methods among which it is natural to choose, in the case of large algebraic systems, preconditioned algorithms in the corresponding Krylov spaces. Note that in the last case we have the problem of solving multiple SLAEs with the same matrix but different right-hand sides, which can be efficiently done by reusing bases of Krylov sunspaces, as was mentioned in remark 1.

From the viewpoint of parallelizing computations on multiprocessors, it is preferable to decompose the computation domain into a greater number of subdomains Ω_q ($q = 1, 2, \ldots, P$) to be able to solve the corresponding subsystems synchronously on P processors. However, the number of external iterations will obviously grow in this case; this number can be estimated as $O(P^{1/d})$, where d is the geometric dimension of the domain (assuming that the decomposition forms a d-dimensional "macronet" consisting of subdomains). The number of external iterations can be reduced by increasing the size of adjacent subdomains overlapping; however, the implementation of each step will be costlier (in practice, the size of overlapping net subdomains has the size of several steps h).

To accelerate the Krylov-type iterations in DDMs, there are various approaches, which can be interpreted as the application of additional preconditioners, including smoothing preconditioners; for this reason, such methods are sometimes called hybrid (see [58, 59]). By way of example, consider the coarse-grid correction method based on the deflation principle of choosing an initial approximation u^0.

Let we have an information basis consisting of vectors $v_1, \ldots, v_m$, which compose the matrix $V = (v_1 \ldots v_m) \in \mathcal{R}^{N,m}$, $m < N$. By choosing an arbitrary vector u^{-1}, we will seek an initial approximation and the corresponding residual vector in the form

$$u^0 = u^{-1} + Vc, \quad r^0 = r^{-1} - AVc, \quad r^{-1} = f - Au^{-1},$$

where the vector of unknown coefficients $c = (c_1, \ldots, c_m)^{\mathrm{T}}$ is determined by minimizing the residual r^0 from the overdetermined equation

$$AVc = r^{-1}. \tag{38}$$

By multiplying both sides of (38) on the left by $V^{\mathrm{T}}A^{\gamma}$ with the parameter $\gamma = 0$ or $\gamma = 1$, we obtain the family of generalized solutions

$$c = (V^{\mathrm{T}}A^{\gamma+1}V)^{+}V^{\mathrm{T}}A^{\gamma}r^{-1},$$

where the case $\gamma = 1$ corresponds to the normal solution, which minimizes the Euclidean norms of the residual $r^0 = r^{-1} - AVc$ and the solution itself c. Hence, we obtain for the corrected residual

$$V^{\mathrm{T}}A^{\gamma}r^0 = 0, \quad r^0 = Tr^{-1}, \quad T = I - AV(V^{\mathrm{T}}A^{\gamma+1}V)^{-1}V^{\mathrm{T}}A^{\gamma}, \tag{39}$$

which corresponds to the minimum of the functional

$$\Phi_{\gamma}(r^0) = (A^{\gamma-1}r^0, r^0).$$

It is easy to see that here the vectors v_s play the role of direction vectors in Krylov subspaces, i.e., they provide the residual vector r^0 with orthogonal and variational properties similar to the properties in the conjugate gradient and conjugate residual methods at $\gamma = 0$ and $\gamma = 1$, respectively.

Following this reasoning, we find using r^0 the initial direction vector for the conjugate direction method from the orthogonality conditions

$$V^{\mathrm{T}}A^{\gamma+1}p^0 = 0, \quad p^0 = B_2^{-1}r^0, \quad B_2^{-1} = I - A^{\gamma+1}V(V^{\mathrm{T}}A^{2\gamma+2}V)^{-1}V^{\mathrm{T}}A^{\gamma+1}. \tag{40}$$

The matrix B_2 here plays the role of the second preconditioner in addition to B_1 in (37), and they both can be used in the subsequent iterations by the formulas of multipreconditioned conjugate directions (see [17]). Note that the exponents of the matrix A in (40) are chosen in such a away that they ensure the symmetry of B_2.

The choice of the basis vectors $v_1, \ldots, v_m$ that effectively extend the Krylov subspace is important in many accelerating iterative procedures. For example, the coarse-grid correction algorithm that is used in DDMs is based on the simple case of "geometric" decomposition of the grid domain Ω_q, and its components are 1 or 0, depending on whether or not the corresponding node lies in the qth subdomain. The further elaboration of this approach is in the transition from piecewise constant basis functions v_q to piecewise polynomial functions of higher orders (see [53]).

3.2. Multigrid Methods

In the pioneering works by Fedorenko and Bakhvalov, the multigrid approaches were based on spectral-approximation principles with a separate suppression of the error components corresponding to low and high frequencies. The further development of these approaches took the geometric, algebraic (the most popular name is AMG—Algebraic Multi-Grid) and combinatorial directions (see [5, 56, 60–67]).

These approaches occupy an exceptional place in computational algebra, since they are asymptotically optimal in order, i.e. the required computer resources are proportional to the SLAE size. The most general algebraic methodology is based on the universal principles of matrix preconditioning. The implementation of technologies with various number of nested grids is interpreted as a recursive application of a two-grid algorithm. For simplicity, consider the sequence of nested grids $\Omega_m \subset \ldots \subset \Omega_2 \subset \Omega_1$; i.e., the nodes of the coarser grids are nodes of the "nearest" finer grid Ω_k. We assume that Ω_k is obtained from Ω_{k+1} by "refining by a factor of two," so that the numbers of grid nodes are related as $N_k \approx 2^d N_{k+1}$, where d is the domain dimension. To solve the SLAE $Au = f$ on Ω, which may be renamed as $A_1 u_1 = f_1$, a preconditioner B_1 is implicitly constructed by approximately solving the system with a specially constructed matrix A_2 for the grid Ω_2, and so on; i.e., the process goes in a "loop."

In the general modern interpretation, the AMG method can be represented by the following stages of iterative solution of the algebraic system $A_k u_k = f_k$ on Ω_k.

1. Given an approximation of the vector u_k^n, the residual on the "fine" grid Ω_k is found:

$$r_k^n = f_k - A_k u_k^n, \quad A_1 = A. \tag{41}$$

2. For the vector r_k^n, preprocessing (presmoothing) is performed usually by making several iterations using a simple algorithm. More precisely, this procedure is performed in two steps. At the first step, the direction vector

$$\hat{S}_k p_k^n = r_k^n \tag{42}$$

is calculated, where $\hat{S}_k$ is the operator (matrix) of this presmoothing step. Actually, $\hat{S}_k$ is an approximation of the matrix A_k (a formal representation of the results of several iteration steps of the smoothing method). At the second step, the next (smoothed) approximate solution and the corresponding residual are found:

$$\tilde{u}_k^n = u_k^n + \tilde{p}_k^n, \quad \tilde{r}_k^n = f - A_k \tilde{u}_k^n = r_k^n - A_1 \tilde{p}_k^n. \tag{43}$$

3. Using the vector $\tilde{r}_k^n$ determined on the coarse grid Ω_k, the residual vector $\tilde{r}_{k+1}^n$ on the fine grid Ω_{k+1} is formed:

$$\tilde{r}_{k+1}^n = R_k \tilde{r}_k^n, \quad R_k \in \mathcal{R}^{N_k, N_{k+1}}, \quad \tilde{r}_k^n \in \mathcal{R}^{N_k}, \tag{44}$$

where R_k is a restriction operator (the restriction stage). In the simplest case, this operation is the projection of the vector from the grid Ω_k onto Ω_{k+1}.

4. On the fine grid, the direction vector $\hat{p}_{k+1}^n$ is found from the solution of the SLAE

$$A_{k+1} \hat{p}_{k+1}^n = \tilde{r}_{k+1}^n, \quad A_{k+1} \in \mathcal{R}^{N_{k+1}, N_{k+1}}, \quad \hat{p}_{k+1}^n, \hat{r}_{k+1}^n \in \mathcal{R}^{N_{k+1}}, \tag{45}$$

where A_{k+1} is matrix of the SLAE for the grid Ω_{k+1}, which in a certain sense inherits the approximation and algebraic properties of the operator A_k on Ω_k. There are various methods for constructing the matrix A_k, which is sometimes called the coarse grid correction matrix.

5. The vector $\hat{p}_{k+1}^n$ found from the solution of system (45) is continued from the coarse grid Ω to the fine grid Ω_k (the prolongation stage):

$$\check{p}_k^n = P_k \hat{p}_{k+1}^n, \quad P_k \in \mathcal{R}^{N_k, N_{k+1}}, \quad \check{p}_k^n \in \mathcal{R}^{N_k}. \tag{46}$$

In a certain sense, a consistent definition of the operators above is such that satisfies the relation $A_k^{-1} \approx P_k A_{k+1}^{-1} R_k$. If, for example, the matrix A_k is symmetric, then in order to inherit this property on the fine grid Ω_{k+1}, it is reasonable to use $P_k = R_k^{\mathrm{T}}$. In particular, this gives the so-called Galerkin approximation $A_{k+1} = P_k^{\mathrm{T}} A_k P_k$.

6. For the vector $\check{p}_k^n$, the corresponding vectors of the approximate solution and residual on the fine grid (residual update) are found by

$$\check{u}_k^n = \tilde{u}_k^n + \check{p}_k^n, \quad \check{r}_k^n = f_k - A_k \check{u}_k^n = r_k^n - A_k \check{p}_k^n, \quad \check{r}_k^n \in \mathcal{R}^{N_k}. \tag{47}$$

7. Postprocessing, i.e., repeated smoothing, for the new direction vector and the iterative approximation of the solution on the fine grid Ω_k is performed with the simultaneous calculation of the new direction vector $\hat{p}_k^n$ on the basis of the solution of the auxiliary SLAE with the matrix $\check{S}_k$ (the repeated smoothing operator, the postsmoothing stage)

$$\check{S}_k \hat{p}_k^n = \check{r}_k^n \tag{48}$$

similarly to (42).

8. The ultimate direction vector is determined as a result of the first AMG iteration:

$$p_k^n = \hat{p}_k^n + \check{p}_k^n + \tilde{p}_k^n = B_k^{-1} r_k^n,$$

where B_k is the preconditioning matrix of this two-grid stage of the method which can be explicitly represented in terms of the smoothing, restriction, coarse-grid correction, and prolongation operators; however, its specific representation is of no importance because the general computations and its implementation are described by formulas (41)–(48). There are a lot of specific versions of the two-grid method, and all of them (in the framework of the stationary iterative process) have the form

$$u_k^{n+1} = u_k^n + p_k^n = u_k^n + B_k^{-1} r_k^n, \tag{49}$$

which is a two-level iterative process since the operation of multiplying by B_k^{-1} includes the solution of a SLAE with the matrix A_{k+1}, which is unreasonable to do by the direct algorithm in the case of large N_{k+1}. It is clear that a preconditioned method in Krylov subspaces can be constructed on the basis of algorithm (49).

However, the idea of the multigrid approach is to use the recursive principle: to approximately solve the equation on a coarse grid Ω_{k+1}, the methods described above are applied on the coarser grid Ω_{k+2}, and so on. The computations on each grid Ω_k by formulas (41)–(48) may be repeated a prescribed number of times γ, which can be written as a generalization of relation (49):

$$u_k^{n+1} = u_k^n + B_k^{-\gamma} r_k^n. \tag{50}$$

In practice, the number of such repetitions is $\gamma = 1$ or $\gamma = 2$. In the first case, the computation scheme is called V-loop, and in the second case it is W-loop. In [64, 65], an elaboration of this approach was proposed: K-loop, in which at $\gamma = 2$ a Krylov-type (rather than stationary) iterative process is used (more specifically, the "flexible" conjugate gradient FCG method described in [40]). In the last formula, it should be taken into account that the preconditioner B_k is expressed recursively for $k = 1, 2, \ldots, m-1$, and on the last mth grid SLAE (45) with the matrix A_{m+1} is solved directly (by a direct or iterative method).

Specific implementations of multigrid approaches, which have already became classical, differ in the way of choosing the matrix operators determining the successive stages of the computation scheme described above.

3.3. Incomplete Factorization Algorithms

The most classical methods of SLAE preconditioning are numerous explicit and implicit matrix factorization methods; a thorough analysis of such methods without parallelization aspects can be found in [10]. Modern approaches to these technologies, including promising approximations of matrices that are inverse of triangular factors in incomplete matrix decompositions are described in [68–70] and references therein.

The typical form of easily invertible preconditioned matrix is obtained by the approximate factorization

$$B = (G + L)G^{-1}(G + U) = G + L + U + LG^{-1}U. \tag{51}$$

Here L and U are the lower and upper triangular parts of the original matrix $A = D + L + U$, and D is a block diagonal or diagonal (in a special case) matrix. On the basis of the general requirement $B \approx A$, the block diagonal matrix G is constructed using the relation

$$G = D - \overline{LG^{-1}U}, \tag{52}$$

where the overscore above a matrix denotes its banded approximation.

Based on formulas (51) and (52), various successive symmetric upper relaxation methods are constructed (in this case, $G = 2\omega^{-1}D$), where ω is an iteration parameter; another group of methods is explicit and implicit incomplete factorization, which are analyzed in books [2, 5, 10, 68] and in special reviews [51, 69, 70]. A large family of algorithms is based on the incomplete LU-factorization of the original matrix, which in the symmetric case is reduced to the approximate Cholesky decomposition. The quality of the matrix approximation and the convergence rate are improved due to "compressing" the matrix factors; however, each iteration step becomes costlier.

A similar approach, but focused on parallelization, is to construct a sparse approximation of the inverse matrix, thus eliminating the need to solve triangular systems at each iteration. These methods were studied in both pointwise and block versions. Novel results in this direction were obtained by Eremin, Kaporin, Kolotilina, Kon'shin, and other researchers in [31, 32, 71, 72]. Another promising direction of research is based on the factorization (including approximate one) of hierarchical matrices in HSS formats using low-rank block structures in the LU decomposition (see [73, 74]).

Finally, pay attention to one more promising family of algorithms for constructing preconditioners that are based on network programming or graph transformation methods (see [75–77]). An important point is that as traditionally the incomplete triangular decomposition of matrices is realized for nonsingular matrices, now there are its modifications for positive semidefinite cases.

A general approach to speeding up the iterations is based on the compensation principle or matching row sums, which is to find a matrix G in (51), (52) such that

$$By^{(l)} = Ay^{(l)}, \quad l = 1, 2, \ldots, m, \tag{53}$$

on a given set of m trial vectors (see [49, 78] and references therein). In some works, this method is also called filtering.

To satisfy condition (53), the matrix G is transformed to the form

$$G = D - \overline{LG^{-1}U} - \theta S, \tag{54}$$

where $\theta \in [0, \theta \in [0,1]]$ is a compensating parameter and S is a block diagonal matrix formed on the basis of the condition

$$Sy^{(l)} = (LG^{-1}U - \overline{LG^{-1}U})y^{(l)}, \quad l = 1, 2, \ldots, m. \tag{55}$$

Among the incomplete factorization methods for solving network SLAEs, implicit methods are the fastest; they are based on triadiagonal matrix algorithms for solving auxiliary tridiagonal systems. Such algorithms can be effectively parallelized on multiprocessors using a multithreading implementation of recursive triadiagonal matrix procedures based on the reduction of original SLAEs or on bordering methods. Note that in [57] a two-thread block version of alternate-triangular matrix factorization method was proposed, in which each factor is not a lower or upper triangular matrix but rather consists of block rows of different orientations—some of them are lower triangular and the others are upper triangular; this decomposition is called twisted decomposition (see review in [27]).

If A is a symmetric matrix, i.e., $D = D^{\mathrm{T}}$ and $L = U^{\mathrm{T}}$, then it is natural to choose symmetric matrices G and B and use two-sided preconditioning for the original SLAE with preserving the symmetry of the final system:

$$\begin{gathered}\overline{A}\overline{u} = \overline{f}, \quad \overline{A} = L_B^{-1}AU_B^{-1}, \quad \overline{u} = U_B\overline{u}, \quad \overline{f} = L_B^{-1}f, \\ B = L_BU_B, \quad L_B = (G+L)U_G^{-1}, \quad G = L_GU_G, \quad U_G = L_G^{\mathrm{T}}.\end{gathered} \tag{56}$$

The preconditioned matrix $\overline{A}$ can be reduced by simple transformations to the form

$$\begin{gathered}\overline{A} = (I+\overline{L})^{-1} + (I+\overline{U}\,)^{-1} + (I+\overline{L})^{-1}(\overline{D}-2I)(I+\overline{U})^{-1}, \\ \overline{D} = L_G^{-1}DU_G^{-1}, \quad \overline{L} = L_G^{-1}LU_G^{-1}, \quad \overline{U} = L_G^{-1}UU_G^{-1},\end{gathered} \tag{57}$$

in which the vector-matrix multiplication can be implemented in the form

$$\overline{A}v = (I+\overline{L})^{-1}[v + (\overline{D}-2I)w] + w, \quad w = (I+\overline{U})^{-1}v; \tag{58}$$

in many cases, this saves a significant amount of computations. Note that the matrix representations (56)–(58) also hold without the requirements $L_B = U_B^{\mathrm{T}}$ and $L_G = U_G^{\mathrm{T}}$; i.e., these formulas also apply to asymmetric SLAEs.

The direct inversion of triangular matrices used in the algorithms described above is poorly parallelized. There are special tricks for mitigating this drawback (e.g., see [79] and the review in [80]). We consider another approach that uses alternate-triangular matrices $\hat{L}$ and $\hat{U}$ instead of L and U, which have nonzero elements only on the left of the principal diagonal in some rows and nonzero elements only on the right in the other rows. We illustrate the construction of such preconditioners by way of a block tridiagonal matrix

$$Au = \begin{bmatrix} D_1 & U_1 & & & 0 \\ L_2 & D_2 & U_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & & U_{N_x-1} \\ 0 & & & L_{N_x} & D_{N_x} \end{bmatrix}. \tag{59}$$

In this case, we define the alternate-triangular matrices by

$$\hat{L} = \begin{bmatrix} 0 & 0 & & & & & \\ L_2 & 0 & 0 & & & 0 & \\ & L_3 & 0 & 0 & & & \\ & & L_4 & 0 & U_4 & & \\ & & & 0 & 0 & U_5 & \\ & 0 & & & 0 & 0 & U_6 \\ & & & & & 0 & 0 \end{bmatrix}, \quad \hat{U} = \begin{bmatrix} 0 & U_1 & & & & & \\ 0 & 0 & U_2 & & & 0 & \\ & 0 & 0 & U_3 & & & \\ & & 0 & 0 & 0 & & \\ & & & L_5 & 0 & 0 & \\ & 0 & & & L_6 & 0 & 0 \\ & & & & & L_7 & 0 \end{bmatrix}. \tag{60}$$

In this case, $A = D + \hat{L} + \hat{U}$; therefore, the incomplete factorization formulas (51)–(58) for determining the preconditioner B remain valid. Only L and U should be replaced by $\hat{L}$ and $\hat{U}$, respectively. In this case, the tridiagonal blocks of the matrix $\overline{G} = \{\overline{G}_i\}$ are found by formulas similar to (28) and (29) but using opposite recursions (block tridiagonal method [25]), which we write for an arbitrary odd $N_x = 2m+1$:

$$\begin{gathered}G_1 = D, \quad G_i = D_i - L_iG_{i-1}^{-1}U_{i-1} - \theta S_i, \quad i = 2, \ldots, m, \\ G_{N_x} = D_{N_x}, \quad G_i = D_i - U_iG_{i+1}^{-1}rU_{i+1} - \theta S_i, \quad i = N_x-1, \ldots, m+2, \\ S_iy^{(l)} = L_i(G_{i-1}^{-1} - G_{i-1}^{-1})U_{i-1}y^{(l)}, \quad l = 1, \ldots, l_i, \\ G_{m+1} = D_{m+1} - L_{m+1}G_m^{-1}U_m - U_{m+1}G_{m+2}^{-1}L_{m+2} - \theta S_{m+1}, \\ S_{m+1}y^{(l)} = [L_{m+1}(G_m^{-1} - G_m^{-1})U_m - U_{m+1}(G_{m+2}^{-1} - G_{m+2}^{-1})]y^{(l)},\end{gathered} \tag{61}$$

where S_i are diagonal matrices for $l = 1$ and tridiagonal for $l = 2$.

When the matrix $\overline{B}$ is used as a preconditioner for an iterative process, the following auxiliary system must be solved at each step:

$$Bp \equiv (G + \hat{L})G^{-1}(G + \hat{U})p = r.$$

Its solution can be found from the successive relations

$$(G + \hat{L})v = r, \quad Gw = \hat{U}p, \quad p = v - w,$$

which are implemented by twisted decomposition by the formulas

$$\begin{gathered} v_1 = G_1^{-1}r_1, \quad i = 2, \ldots, m : v_i = G_1^{-1}(r_i - \hat{L}_i v_{i-1}), \\ v_{N_x} = G_{N_x}^{-1}r_{N_x}, \quad i = N_x - 1, \ldots, m+2 : v_i = G_i^{-1}(r_i - \hat{U}_i v_{i+1}), \\ v_{m+1} = G_{m+1}^{-1}(r_{m+1} - \hat{L}_{m+1}v_m - \hat{U}_{m+1}v_{m+2}) = p_{m+1}, \\ i = m, \ldots, 1 : w_i = G_i^{-1}\hat{U}_i p_{i+1}, \quad p_i = v_i - w_i, \\ i = m+2, \ldots, N_x : w_i = G_i^{-1}\hat{L}_i p_{i-1}, \quad p_i = v_i - w_i. \end{gathered} \tag{62}$$

In recurrences (61) and (62), the computations with increasing index i and with its decreasing may be called forward and backward recursions, respectively. It is clear that they can be executed simultaneously in two threads. This limited parallelism is based on the alternate-triangulation (60) for the matrices L and U. This approach can be generalized for the case of P-alternate triangulation; i.e., we can define matrices $\hat{L}$ and $\hat{U}$ satisfying the condition $\hat{L} + \hat{U} = L + U$ as consisting of P block rows each of which successively changes the triangulation type. Formulas for the block tridiagonal matrix algorithm (61), (62) are more complicated, but they can be parallelized to P processors.

For example, matrix (59) corresponds to the five-point Laplace equations on a rectangular grid with $N = N_x \cdot N_y$ nodes. In this case, D_i are tridiagonal blocks and L_i and U_i are diagonal blocks. In the case of three-dimensional problems on a box grid with $N = N_x \cdot N_y \cdot N_z$ nodes, the matrix A can be represented in block tridiagonal form (59) in which each diagonal block D_k has the size $N = N_x \cdot N_y$ and is a matrix of the same structure as (59).

The elaboration of the above approaches for solving SLAEs on three-dimensional grids received the name of nested factorization methods; they were proposed in 1983 by Appleyard and Cheshire and later studied by many researchers (see [57, 82, 83] and references therein). Let the matrix A have the form

$$A = D + L_1 + U_1 + L_2 + U_2 + L_3 + U_3, \tag{63}$$

where D is a diagonal and L_k and $U_k, k = 1, 2, 3$, are lower and upper triangular matrices. Let us find the preconditioning matrix B using the recursive factorization (51):

$$\begin{gathered} B = (P + L_3)P^{-1}(P + U_3) = P + L_3 + U_3 + L_3P^{-1}U_3, \\ P = (T + L_2)T^{-1}(T + U_2) = T + L_2 + U_2 + L_2T^{-1}U_2, \\ T = (M + L_1)M^{-1}(M + U_1) = M + L_1 + U_1 + L_1M^{-1}U_1; \end{gathered} \tag{64}$$

as a result, we obtain

$$B = M + A - D + L_1M^{-1}U_1 + L_2T^{-1}U_2 + L_3P^{-1}U_3. \tag{65}$$

Depending on the definition of M, different preconditioners B can be formed. We consider a simple case when matrix (63) corresponds to the seven-point approximation of the Dirichlet problem for the Poisson equation in a box on a box grid. Then, in the case of natural ordering of the nodes, the matrices M, T and P can be made diagonal, tridiagonal, and five-diagonal, respectively, and the preconditioner can be found by the formulas

$$\begin{gathered} M = D - L_1M^{-1}U_1 - \theta_2S_2 - \theta_3S_3, \\ B = A + L_2T^{-1}U_2 + L_3P^{-1}U_3 - \theta_2S_2 - \theta_3S_3. \end{gathered} \tag{66}$$

Here θ_2 and θ_3 are iteration (relaxing) parameters, and, S_2 and S_3 are diagonal matrices determined from the conditions of matching row sums similarly to how this was done above:

$$S_2 e = L_2 T^{-1} U_2 e, \quad S_3 e = L_3 P^{-1} U_3 e. \tag{67}$$

Note that instead of (67) we may use matching column sums rather than row sums (see [83] and references therein):

$$e^{\mathrm{T}} S_2 = e^{\mathrm{T}} L_2 T^{-1} U_2, \quad e^{\mathrm{T}} S_3 = e^{\mathrm{T}} L_3 P^{-1} U_3. \tag{68}$$

The three-level nested factorization considered above can be structurally simplified by reducing it to the two-level one. To this end, we rewrite the original matrix A (63) in the form

$$A = D_3 + L_3 + U_3, \quad D_3 = D_2 + L_2 + U_2, \quad D_2 = D + L_1 + U_1. \tag{69}$$

In this case, D_3 is a block diagonal matrix with five-diagonal blocks $D_{3,i}$ of size $N_y N_z$ each of which corresponds to a plane problem in the section $x = \mathrm{const}$ and has the same structure as the matrix A in (59). Then, the matrix P in (64) is defined by the formula

$$P = \{G_i = D_{3,i} - \theta S_i\}, \quad S_i e = L_{3,i} G_{i-1}^{-1} U_{i-1} e,$$

which corresponds to the definition $L_3 P^{-1} U_3 = 0$ in (64). Note that L_3 and U_3 can be defined as alternate-triangular matrices, and then the algorithm implementation can be parallelized using twisted decomposition.

3.4. Solving Saddle SLAEs

Consider a SLAE with a saddle matrix

$$A \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \quad A = \begin{bmatrix} D & C^{\mathrm{T}} \\ C & 0 \end{bmatrix}, \tag{70}$$

where $A \in \mathcal{R}^{N,N}$, $D \in \mathcal{R}^{N_1,N_1}$, $C \in \mathcal{R}^{N_2,N_1}$, $N = N_1 + N_2$, $f \in \mathcal{R}^{N_1}$, and $p, g \in \mathcal{R}^{N_2}$. The matrix D is assumed to be symmetric and positive semidefinite. A necessary and sufficient condition for the nonsingularity of A is $\ker(D) \cap \ker(C) = \{0\}$, and a sufficient condition is the positive definiteness of D on $\ker(C)$ and $\ker(C^{\mathrm{T}}) = \{0\}$, which we assume to be fulfilled.

Problems with a saddle point often arise in mathematical formulations in modeling, including initial boundary mixed formulations for differential classical or generalized equations, optimization problems, and computational algebra (see [84–87] and references therein). We focus on methods for solving saddle SLAEs with large sparse matrices that arise in grid approximations of multidimensional boundary value problems in many applications—electromagnetism, fluid dynamics, elasticity, plasticity, multiphase filtering in porous media, in optimization problems, etc.

Due to special features of the block structure of saddle algebraic systems, methods for their solution are studied in numerous works, e.g., see the reviews by Golub and his colleagues [84], Brezzi [85], Vassilevski [72], and Notay [86], book by Bychenkov and Chizhonkov [87], series of papers by C. Greif and his colleagues (including the case of asymmetric saddle SLAEs [68, 85]), and Arioli with coauthors [25, 75] (also see the reviews in [90–92]).

Note that without loss of generality the saddle SLAE can be considered in the form

$$\begin{bmatrix} D & C^{\mathrm{T}} \\ C & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}. \tag{71}$$

Indeed, if we take a particular solution of system $C\hat{u} = g$, then the vector $u = \breve{u} + \hat{u}$, which is a solution of SLAE (70), satisfies the system

$$\begin{bmatrix} D & C^{\mathrm{T}} \\ C & 0 \end{bmatrix} \begin{bmatrix} \breve{u} \\ p \end{bmatrix} = \begin{bmatrix} f - D\hat{u} \\ 0 \end{bmatrix}.$$

Note that any solution of SLAE (71) simultaneously satisfies the system

$$\tilde{A}v = \tilde{A}\begin{bmatrix} u \\ p \end{bmatrix} \equiv \begin{bmatrix} \tilde{D} & C^{\mathrm{T}} \\ C & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \equiv \tilde{f}, \quad \tilde{D} = D + R, \quad R = C^{\mathrm{T}}K_0^{-1}C, \tag{72}$$

where $v, \tilde{f} \in \mathcal{R}^N$, and $K_0 \in \mathcal{R}^{N_1,N_1}$ is an arbitrary nonsingular matrix. Since the last system is formally a regularization (or generalization) of SLAE (71), we below discuss algorithms for solving Eq. (72).

Using the Schur complement

$$S = -C\tilde{D}^{-1}C^{\mathrm{T}},$$

the matrix of system (72) is factorized as

$$\tilde{A} = \begin{bmatrix} \tilde{D} & 0 \\ C & S \end{bmatrix}\begin{bmatrix} I & \tilde{D}^{-1}C^{\mathrm{T}} \\ 0 & I \end{bmatrix}.$$

If the matrices $\bar{D}$ and S are replaced by their approximations (preconditioners) B_d and B_s, then we obtain a preconditioner for the matrix B in the form of incomplete block factorization

$$B_1 = \begin{bmatrix} B_d & 0 \\ C & -B_s \end{bmatrix}\begin{bmatrix} I & B_d^{-1}C^{\mathrm{T}} \\ 0 & I \end{bmatrix}. \tag{73}$$

A somewhat coarser approximation when only one (left or right) factor is used in (73) gives an incomplete block triangular preconditioning with the matrix

$$B_2 = \begin{bmatrix} B_d & C^{\mathrm{T}} \\ 0 & -B_s \end{bmatrix} \tag{74}$$

or the incomplete Uzawa preconditioner

$$B_3 = \begin{bmatrix} B_d & 0 \\ C & -B_s \end{bmatrix}. \tag{75}$$

The implementation of each step of the corresponding iterative processes can be divided into several stages at which only one block component of the desired solution is recalculated (for this reason, these methods are sometimes called segregative). In a slightly generalized form, we represent such a stationary algorithm (without the Krylov acceleration) by the following three stages (see [83, 85, 86]):

$$\begin{aligned} \hat{u}_d^{n+1} &= u_d^n + Q_d^{(1)}(f_d - \bar{D}u_d^n - C^{\mathrm{T}}u_c^n), \\ u_s^{n+1} &= u_s^n - B_s^{-1}(f_s - C\hat{u}_d^{-1}), \\ u_d^{n+1} &= \hat{u}_d^{n+1} + Q_d^{(2)}(f_d - \bar{D}A\hat{u}_d^{n+1} - C^{\mathrm{T}}u_s^{n+1}). \end{aligned} \tag{76}$$

Here $Q_d^{(1)}$ and $Q_d^{(2)}$ are approximations of inverse or generalized inverse matrix of the preconditioner B_d. In particular, if $Q_d^{(1)} = B_d^{-1}$, $Q_d^{(2)} = 0$ or $Q_d^{(1)} = 0$, $Q_d^{(2)} = B_d^{-1}$, then we obtain from (76) the Uzawa algorithm with the preconditioner B_3 defined in (75) (in this case, the third stage is omitted, i.e., $u_d^{n+1} = \hat{u}_d^{n+1}$) or the incomplete block triangular preconditioning with the matrix B_2 defined in (74) (in this case, the first stage in (76) is omitted and $u_d^{n+1} = u_d^n$).

If the matrix $Q_d = Q_d^{(1)} + Q_d^{(2)} - Q_d^{(2)}\bar{D}Q_d^{(1)}$ is singular, then the iterative process (76) is associated with the preconditioner

$$B_4 = \begin{pmatrix} I & 0 \\ CQ_d^{(1)} & I \end{pmatrix}\begin{pmatrix} Q_d^{-1} & 0 \\ 0 & -B_s \end{pmatrix}\begin{pmatrix} I & Q_d^{(2)}C^{\mathrm{T}} \\ 0 & I \end{pmatrix}. \tag{77}$$

In this case, for $Q_d^{(1)} = Q_d^{(2)} = B_d^{-1}$ (77) implies the so-called symmetrized Uzawa method with the preconditioned matrix

$$B_5 = \begin{pmatrix} I & 0 \\ CB_d^{-1} & I \end{pmatrix} \begin{pmatrix} B_d(2B_d - \bar{D})^{-1}B_d & 0 \\ 0 & B_s \end{pmatrix} \begin{pmatrix} I & B_d^{-1}C^{\mathrm{T}} \\ 0 & I \end{pmatrix}.$$

A promising block diagonal preconditioner for solving SLAE (72) is

$$B_6 = \begin{bmatrix} \tilde{D} + C^{\mathrm{T}}K_1^{-1}C & 0 \\ 0 & K_2 \end{bmatrix}, \tag{78}$$

where K_1 and K_2 are symmetric nonsingular matrices.

The eigenvalues and eigenvectors of the preconditioned matrix $\bar{A} = B_6^{-1}\tilde{A}$ from the "disturbed" SLAE(72) are determined by the eigenvalue problem

$$\lambda B_6 z = \tilde{A}z, \quad z = (z_1^{\mathrm{T}}, z_2^{\mathrm{T}})^{\mathrm{T}}, \quad z_k \in \mathcal{R}^{N_k}, \quad k = 1, 2,$$

which in componentwise form is written as

$$\begin{aligned} (D + C^{\mathrm{T}}K_0C)z_1 + C^{\mathrm{T}}z_2 &= \lambda(\tilde{D} + C^{\mathrm{T}}K_1^{-1}C)z_1, \\ Cz_1 &= \lambda K_2 z_2. \end{aligned} \tag{79}$$

1 The analysis of this eigenvalue problem yields interesting results for various sdecial cases. In particular, it was shown in [81, 82] that unique spectral properties of the matrix $\bar{A}$ are obtained if the block $\tilde{D}$ is strongly singular, which is important for algorithms designed for solving systems of Maxwell equations. For example, we have the following results.

Proposition 1. *Let B_6 be an spd matrix and $\{z_i,\ i = 1, 2, \ldots, N_1 - N_2\}$ be a basis of the kernel of* the matrix C. *Then, the preconditioned matrix $B_6^{-1}\tilde{A}$ has $N_1 - N_2$ linearly independent vectors $(z_i^{\mathrm{T}}0) \in \mathcal{R}^N$ associated with $N_1 - N_2$ multiple eigenvalue $\lambda = 1$.*

Proposition 2. *Let $K_1 = K_2 = K$, and let $\tilde{D}$ be a symmetric positive semidefinite matrix with the kernel dimension equal to r. Then, $\lambda = 1$ is an eigenvalue of the matrix $B_6^{-1}\tilde{A}$ with the multiplicity N_1, $\lambda = -1$ is an eigenvalue of multiplicity r, and the other $N_2 - r$ eigenvalues lie in the interval $\lambda \in (-1, 0)$ and satisfy the relation*

$$\lambda = -\mu/(\mu + 1),$$

where μ are $N_2 - r$ positive generalized eigenvalues

$$\mu\tilde{D}z = C^{\mathrm{T}}KCz.$$

Let $\{z_i,\ i = 1, 2, \ldots, N_1 - N_2\}$ be a basis of the kernel of C, $\{x_i,\ i = 1, 2, \ldots, N_2 - 2\}$ be a basis of the kernel of $\tilde{D}$, and $\{y_i,\ i = 1, 2, \ldots, N_2 - r\}$ be the set of linearly independent vectors complementing $\ker(\tilde{D}) \cup \ker(C)$ in the basis $\mathcal{R}^{N_1}$. Then, $N_1 - N_2$ vectors $(z_i, 0)$, r vectors $(x_i, K^{-1}Cx_i)$, and $N_2 - r$ vectors $(y_i, K^{-1}y_i)$ are linearly independent eigenvectors corresponding to the eigenvalues $\lambda = 1$, and r vectors $(x_i, K^{-1}x_i)$ are the eigenvectors corresponding to the eigenvalues $\lambda = -1$.

On the whole, the presence of different matrices K_0, K_1, and K_2 in the preconditioner B_6 provides wide opportunities for constructing efficient algorithms in particular cases.

We consider a family of iterative methods for solving saddle symmetric SLAEs with the matrix $\tilde{A}$ defined by (72); this family is based on the efficient Golub–Kahan $G - K$-bidiagonalization, which was originally proposed for the singular decomposition of rectangular matrices but later was successfully used for solving algebraic systems, including the case of block saddle structure, in the works by Saunders, Arioli, Greif, and other researchers.

Without loss of generality, we write the SLAE under examination in the form

$$\tilde{A}\begin{bmatrix} u \\ p \end{bmatrix} \equiv \begin{bmatrix} \tilde{D} & C^{\mathrm{T}} \\ C & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix}, \quad \tilde{D} = D + C^{\mathrm{T}}K^{-1}C. \tag{80}$$

It is easy to verify that, if the function u in (72) is replaced by $u + \tilde{D}^{-1}f$, then this system takes form (80) with the right-hand side $g = -C\tilde{D}^{-1}f$. It is assumed that $\tilde{D}$ and K in (80) are spd matrices and $N_1 \geq N_2$.

The $G - K$-bidiagonalization method is based on the construction of $\tilde{D}$-orthogonal vectors v_k and K-orthogonal vectors q_k that satisfy the conditions

$$\begin{aligned} C^{\mathrm{T}}Q &= \tilde{D}V[B^{\mathrm{T}}0]^{\mathrm{T}}, \quad V^{\mathrm{T}}\tilde{D}V = I_{N_1}, \\ CV &= KQ[B^{\mathrm{T}}0], \quad Q^{\mathrm{T}}KQ = I_{N_2}, \end{aligned} \tag{81}$$

where $V = [v_1, \ldots, v_{N_1}] \in \mathcal{R}^{N_1,N_2}$, $Q = [q_1, \ldots, q_{N_1}]$, and $B \in \mathcal{R}^{N_2,N_2}$ is a two-diagonal matrix

$$B = \begin{bmatrix} \alpha & \beta_1 & 0 & \ldots & 0 \\ 0 & \alpha_2 & \beta_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & \alpha_{N_2-1} & \beta_{N_2-1} \\ 0 & \ddots & 0 & 0 & \alpha_{N_2} \end{bmatrix}.$$

By introducing new unknown functions

$$u = Vz, \quad p = Qg \tag{82}$$

and multiplying system (80) by the block diagonal matrix block-diag$(V^{\mathrm{T}}, Q^{\mathrm{T}})$, we obtain

$$\begin{bmatrix} I_{N_2} & 0 & B \\ 0 & I_{N_1-N_2} & 0 \\ B^{\top} & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ Q^{\top}g \end{bmatrix}, \quad z_1 \in \mathcal{R}^{N_2}, \quad z_2 \in \mathcal{R}^{N_1-N_2}. \tag{83}$$

It follows from (83) that the vector u depends on N_2 columns of the matrix V because $z_2 = 0$. Thus, SLAE (83) is reduced to the form

$$\begin{bmatrix} I_{N_2} & B \\ B^{\top} & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ Q^{\mathrm{T}}g \end{bmatrix}. \tag{84}$$

Hence, we put

$$q_1 = K^{-1}g/\|g\|_{K^{-1}}, \quad \alpha_1\tilde{D}^{-1}v_1 = C^{\mathrm{T}}q_1, \quad \|g\|_{K^{-1}} = (g, K^{-1}g)^{1/2}$$

to obtain the initial vector v_1:

$$v_1 = w/\alpha_1, \quad \alpha_1\sqrt{w^{\mathrm{T}}C^{\mathrm{T}}q_1}, \quad w = \tilde{D}C^{\mathrm{T}}q_1. \tag{85}$$

The further vectors v_n, q_i, and the elements α_i, β_i of the matrix B are calculated from the recurrences ($n = 1, 2, \ldots$)

$$\begin{aligned} s &= K^{-1}(Cv_n - \alpha_i Kq_n), \quad \beta_{n-1} = \sqrt{s^{\mathrm{T}}Ks}, \\ q_{n+1} &= s/\beta_{n+1}, \quad w = \tilde{D}^{-1}(C^{\mathrm{T}}q_{n+1} - \beta_{n+1}\tilde{D}v_n), \\ \alpha_{n+1} &= (w^{\mathrm{T}}\tilde{D}w)^{1/2}, \quad v_{n+1} = w/\alpha_{n+1}. \end{aligned} \tag{86}$$

The successive approximations u^n, according to (82), are found from the first n columns of the matrix V, i.e.,

$$u^n = V_n z^n = \sum_{j=1}^{n} z_1^j v_j, \quad z^n = (z_1^1, \ldots, z_1^n), \tag{87}$$

where z_1^j are the components of the vector z_1 in (84) calculated by the formulas

$$z_1^n = \|g\|_{N^{-1}}/\alpha_1, \quad z_{j+1}^n = -\beta_{j+1}z_j^n/\alpha_{j+1}, \quad j = 1, 2, \ldots, N_2. \tag{88}$$

We omit the details of derivation of these formulas (see [87]) and only present the final recurrences for the iterative solution

$$\begin{aligned} u^{n+1} &= u^n + z_1^{n+1} v_{n+1}, \quad n = 1, 2, \ldots, N_2, \\ p^{n+1} &= p^n - z_1^{n+1} d_{n+1}, \quad d_{n+1} = (q_{n+1} - \beta_{n+1}\alpha_n)/\alpha_{n+1}, \end{aligned} \tag{89}$$

where d_n is the nth column of $D = QB^{-1}$. Note that the algorithm just described has the following optimization properties: at each nth step, the iterative approximation error has the minimum

$$\begin{aligned} &\min_{\substack{u_n \in \mathcal{U}_n \\ Cu^n - g \perp \mathcal{Q}_n}} \left\| u - u^n \right\|_{\tilde{D}}, \\ &\mathcal{U}_n = \operatorname{Span}\{v_1, \ldots, v_n\}, \quad \mathcal{Q}_n = \operatorname{Span}\{q_1, \ldots, q_n\}. \end{aligned} \tag{90}$$

As is mentioned in [71], this method has high convergence rate in the case of SLAEs arising in finite element approximations of continuous multidimensional saddle problems, i.e., in mixed statements. An important factor is that at each iteration algebraic subsystems with the matrices $\tilde{D}$ and K must be solved; in a certain sense, these matrices play the role of preconditioners. Their approximate inversion actually leads to two-level iterative processes in certain subspaces.

4. PARALLELIZATION AND PERFORMANCE ISSUES OF ITERATIVE METHODS

The modern understanding of algorithm quality includes two main characteristics—mathematical efficiency and performance of implementation on a particular supercomputer architecture. The first characteristic includes the construction and optimization of iterative methods with a high convergence rate and theoretical estimates of the required computational resources (the number of arithmetic operations and amount of memory). The second characteristic is purely practical—it is determined by the actual execution time of the algorithm for a certain class of problems, which depends on the scalability of the algorithm parallelization and on the programming technology on a specific computer platform.

The SLAEs of greatest interest for us have large size (10^8–10^{11}) and sparse matrices with large condition numbers and irregular structure. This not only leads to a large number of iterations, but also forces one to work with systems of distributed and hierarchical shared memory and also significantly speeds down data access.

The main quantitative characteristic of parallelization is the acceleration of computations

$$S_p = T_1/T_p, \quad T_p = T_p^a + T_p^c,$$

where T_p is the time of solving the problem on p processors, which is the sum of data exchange time and arithmetic operations execution time; in turn, these times are found by the approximate formulas

$$T^a = \tau_a N_a, \quad T^c = N_t(\tau_0 + \tau_c N_c).$$

Here τ_a and N_a are the average time of an arithmetic operation execution and their total number, respectively, N_t is the number of data exchanges, τ_0 and τ_c are the waiting time and the time of sending one number, respectively, and N_c is the average size of one communication.

Machine constants satisfy the inequalities $\tau_0 \gg \tau_c \gg \tau_0$; therefore, for the algorithms to be constructed, we have obvious recommendations: try to minimize the size of communications, and make data exchanges in large portions, i.e., pre-accumulate data buffers. These conclusions are all the more true because interprocessor data transfers not only slow down the computational process but also are the most energy-consuming operations, and this becomes a significant factor in the costs of operating a supercomputer [93].

The strategy of parallelizing large grid SLAEs arising as a result of approximation of multidimensional boundary value problems is based on hybrid programming and additive domain decomposition methods with two-level iterations in Krylov subspaces. The upper level iterations (over subdomains) are performed using the message passing interface MPI between processes each of which solves (simultaneously) an algebraic subsystem in the corresponding subdomain. At each iteration, the values of approximate solutions on the interface between subdomains are exchanged. Naturally, all matrix and vector data for the subsystems are preliminary distributed between processes. The solution of the SLAE in each subdomain is parallelized using multithreading computations (like OpenMP) on multicore processors with shared

memory. Additional acceleration can be achieved by vectorizing operations (AVX-like instruction sets based on SIMD (Single Instruction Multiple Data) technologies (see reviews in [18]). Unfortunately, presently there are no ready-to-use software development systems with automatic parallelization of algorithms; hence the success in computations speedup and scalability significantly depends on the skills of the programmer.

Note that the traditional block Jacobi–Schwarz methods underlying parallel domain decomposition algorithms are based on the spectral optimization of iterative processes. A possible alternative to them are projection block Cimmino methods [8, 94], which still are not sufficiently well studied and rarely used in practice.

From the parallelization viewpoint, implicit alternating direction or Peaceman–Rachford methods (see [90] and the review in [1]), which were intensively developed in the 1960–1970s seem promising. They have a record convergence rate of iterations for model grid boundary value problems. In recent decades, this direction has found a second wind in connection with the development of rational approximations in Krylov subspaces (see [95]), including the solution of Lyapunov and Sylvester matrix equations. Note that in [93] a method for increasing the degree of parallelization on the basis of expanding rational functions in partial fractions was proposed.

Computations can be significantly accelerated by using variable precision arithmetic. Traditionally, the solution of large SLAEs uses double precision arithmetic in which floating point numbers are represented by 64 bits; however, for extremely poorly conditioned matrices, quad precision (128 bits) is required. By contrast, at certain stages of the algorithm, single and even half precision (32 and 16 bits, respectively), which is significantly faster may be used. This approach is quite natural at first iterations when the error of the approximate solution is relatively high. Another option of using low precision is in DDMs when solving auxiliary problems in subdomains. It should be taken into account that such tricks require a thorough analysis of numerical errors of the method to ensure the stability of computations as a whole.

A further reserve for improving performance is code optimization, which can be achieved by using high-quality computational tools (e.g., SPARSE BLAS), by using various compiler options, and special properties of a particular supercomputer platform. In recent decades, a large number of high-quality software libraries for computational algebra have been developed (PETSc, HYPRE, MKL INTEL, and others, see the review in [22, 23]). Also pay attention to the works performed in the All-Russian Research Institute of Experimental Physics, Russian Federal Research Center (Sarov) [96] and to projects on integrated computational environments DUNE [97], INMOST [98], and BSM [1, 24].

The creation, maintenance, and development of mathematical methods and software for the efficient solution of SLAEs is a continuous science-intensive technological process, which includes regular experimental research that requires intelligent means of support, such as systems for automation of verifying and testing algorithms and their implementations, collections of problems (such as popular Sparse Matrix Collections offered by various developers) and numerous demo and training versions of software (in addition to traditional monographs, textbooks, and user guides with recommendations on the application of various algorithms for solving various practical problems).

5. CONCLUSIONS: PROBLEMS AND PROSPECTS OF DEVELOPING ITERATIVE PROCESSES

From the systems point of view, the high-performance solution of SLAEs is a widely demanded area of intellectual activity that is far from being limited by writing and justifying formulas of the algorithm; it also includes statements of new practical problems or theoretical ideas, technological aspects of software implementations, and issues of their effective application in supercomputer experiments.

New practical challenges of mathematical modeling will drastically increase the demand for solving interdisciplinary and inverse high resolution problems on real-life data, which makes the high-performance solution of SLAEs with tens or hundreds of billions of degrees of freedom and extremely poor conditioning. In recent decades, the computational algebra methods have been rapidly developing both theoretically and experimentally, and promising directions have developed here: low-rank approximations of matrices, including rational approximations, tensor methodologies graph-theoretic and combinatorial approaches, aggregation and segregation methods; moreover, novel ideas in traditional matrix factorizations, algebraic decompositions, multigrid technologies, etc. are appearing.

Another important factor is the ongoing accumulation of a vast number of various problems (interdisciplinary, methodological, practical, typical, and unique) and methods and technologies for their solution on computers of various classes. At the same time, there is a transition from the amount of processed data to quality, which requires the concept of developing a new generation of mathematical techniques and

software. An important task is to create an active base of mathematical knowledge to ensure the automation or optimization of algorithm construction and their mapping to supercomputer architecture. A prototype of such a base is the project ALGOWIKI developed in the framework of Wikipedia technologies under the guidance of J. Dongarra and Vl.V. Voevodin. This task is within the realm of artificial intelligence and machine learning which, along with big data technologies, forms the foundation of modern modeling. The achieved scales of the triad *problems–methods–computers* lead to the fact that the integrated computing environment for solving SLAEs should become the form and content of a hierarchical infrastructure that constitutes the industrial-type instrumental environment and implements further stages of the development of high-tech computing and information technologies, the outlines of which are outlined in [1, 24, 93]. To prevent deep specialization from leading to the separation of computer theorists, programmers, end users, their joint efforts should be aimed at creating an intelligent ecosystem that supports interprofessional and human-machine interfaces both to intensify fundamental research and to quickly implement their innovations.

FUNDING

This work was supported by the Russian Foundation for Basic Research, project no. 19-11-50073 within the competition of review articles "Expansion" and by the Ministry for Science and Higher Education, project no. 2020-0012.

ACKNOWLEDGMENTS

I am grateful to two anonymous reviewers whose remarks helped improve the presentation.

REFERENCES

1. V. P. Il'in, *Mathematical Modeling. Part I: Continuous and Discrete Models* (SBRAS, Novosibirsk, 2017).
2. O. Axelsson, Iterative Solution Methods (Cambridge Univ. Press, Cambridge, 1994).
3. H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics. Numerical Mathematics and Scientific Computation,* 2nd ed. (Oxford Univ. Press, Oxford, 2014.
2 4. G. I. Marchuk and Yu. A. Kuznetsov, *Iterative Methods and Quadratic Functionals* (CC SBRAS, Novosibirsk, 1972) [in Russian].
5. Y. Saad, *Iterative Methods for Sparse Linear Systems,* 2nd ed. (SIAM, 2003).
6. H. A. Van der Vorst, *Iterative Krylov Methods for Large Linear Systems* (Cambridge Univ. Press, Cambridge, 2003_.
7. M. A. Olshanskii and E. E. Tyrtyshnikov, *Iterative Methods for Linear Systems Theory and Applications* (SIAM, Philadelphia, 2014).
8. J. Liesen and Z. Strakos, *Krylov Subspace Methods, Principles and Analysis* (Oxford Univ. Press, Oxford, 2013).
9. V. P. Il'in, *Finite Element Methods and Technologies* (ICM&MG SBRAS, Novosibirsk, 2007) [in Russian].
10. V. P. Il'in, Iterative Incomplete Factorization Methods. (World Science, Singapore, 1992).
11. H. Fang and Y. Saad, "Two classes of multisecant methods for nonlinear acceleration," Numer. Linear Algebra Appl. **16** (3), 197–221 (2009).
12. P. P. Pratara and J. E. Suryanarayana, "Anderson acceleration of the Jacobi iterative method. An efficient alternative to Krylov methods for large, sparse linear systems," J. Comput. Phys. **306**, 43–54 (2016).
13. H. F. Walkert and P. Ni, "Anderson acceleration for fixed-point iterations," SIAM J. Assoc. Comput. Math. **49**, 1715–1735 (2011).
14. V. P. Il'in, "Iterative processes in the Krylov–Sonneveld subspaces," J. Math. Sci. **24**, 890–899 (2017).
15. D. S. Butyugin, Y. I. Gurieva, V. P. Il'in, and D. V. Perevozkin, "Some geometric and algebraic aspects of domain decomposition methods," Lect. Notes Comput. Sci. Eng. **104**, 117–126 (2016).
16. Domain Decomposition Methods. http://ddm.org.
17. V. P. Il'in, "Multi-preconditioned domain decomposition methods in the Krylov subspaces," Lect. Notes Comput. Sci. **10187**, 95–106 (2017).
18. V. P. Il'in, "Problems of parallel solution of large systems of linear algebraic equations," J. Math. Sci. **216**, 795–804 (2016).
19. A. Toselli and O. Widlund, *Domain Decomposition Methods: Algorithms and Theory* (Springer, Berlin, 2005).
20. P. N. Vabishchevich, "Incomplete iterative implicit schemes." Comput. Meth. Appl. Math. 2019.

21. V. P. Il'in, "High-performance computation of initial boundary value problems,": PCT 2018, CCIS 910, ed. by L. Sokolinsky and M. Zymbler (Springer, 2018), pp. 186–199.
22. J. Dongarra, List of freely available software for linear algebra on the web (2006). http://netlib.org/utk/people/JackDongarra/la-sw.html.
23. R. Barret, M. Bery, T. F. Chan. J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. A. van der Vorst, *Templates for the Solution of Linear Systems. Building Blocks for Iterative Methods* (SIAM, Philadelphia, 1994).
24. V. P. Il'in, "On an integrated computational environment for numerical algebra," PCT 2019, CCIS 1063, ed. by L. Sokolinsky and M. Zymbler (Springer, 2019), pp. 91–106.
25. M. Arioli, "Generalized Golub–Kahan diagonalization and stopping criteria," SIAM J. Matrix Anal. Appl. **34** (2), 57–92 (2013).
26. N. J. Higham. *Accuracy and Stability of Numerical Algorithms* (SIAM, Philadelphia, 2002).
27. V. P. Il'in, *Finite Difference and Finite Volume Methods for Elliptic Equations* (Inst. Vychisl. Mat. Mat. Geofiz., Ross. Akad. Nauk, Novosibirsk, 2001) [in Russian].
28. K. M. Soodhalter, E. Sturler, and M. Kilmer, "A survey of subspace recycling iterative methods." https://arxiv.org/abs/2001.10347.
29. S.-C. T. Choi, C. C. Paige, and M. A. Saunders, "INRES-QLP: A Krylov subspace method for indefinite or singular symmetric systems," SIAM J. Sci. Comput. (2015). arXiv:1003.4042 [math.NA] **2**.
30. I. E. Kaporin, "New convergence results and preconditioning strategies for the conjugate gradient method," J. Numer. Linear Algebra Appl. **1**, 179–210 (1994).
31. I. E. Kaporin, "High quality preconditioning of a general symmetric positive definite matrix based on its UTU+UTR+RTU-?Decomposition," J. Numer. Linear Algebra Appl. **5**, 483–509 (1998).
32. I. E. Kaporin and I. N. Konshin, "A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems," J. Numer. Linear Algebra Appl. **9**, 141–162 (2002).
33. V. P. Il'in, "Methods of semiconjugate directions," Russ. J. Numer. Anal. Math. Model. **23**, 369–387 (2008).
34. V. P. Il'in, "Two-level least squares methods in Krylov subspaces," J. Math. Sci. **232**, 892–901 (2019).
35. L. A. Knizhnerman, "Error bounds for the Arnoldi method: A set of extreme eigenpairs," J. Numer. Linear Algebra Appl. **296**, 191–211 (1999).
36. V. P. Il'in, "Biconjugate direction methods in Krylov subspaces," J. Appl. Indust. Math. **4** (1), 65–78 (2010).
37. M. P. Neuenhofen and C. Greif, "MSTAB: stabilized inducted dimension reduction for KRYLOV subspace recycling," SIAM J. Sci. Comput. **40**, 554–571 (2018).
38. Saunders M. A., Simon H. D., Yip. E.L. "Two conjugate-gradient-type methods for unsymmetric linear equations," SIAM J. Numer. Anal. **25**, 927–940 (1988).
39. R. Estrin and C. Greif, :SPMR: a family of saddle-point minimum residual solvers," SIAM J. Sci. Comput. **40**, 1884–1914 (2018).
40. Y. Notay, "Flexible conjugate gradients," SIAM J. Sci. Comput. **22**, 1444–1460 (2000).
41. A. M. Bradley, Algorithms of the equilibration of matrices and their application to limited-memory quasi-Newton methods, PhD thesis (ICME, Stanford Univ., 2010).
42. O. E. Livne and G. H. Golub, "Scaling by binormalization," J. Numer. Algorithms **35**, 97–120 (2004).
43. V. P. Il'in and R. Kellog, "Analysis of flow directed iterations," J. Comp. Math. **10** (1), 1–18 (1992).
44. D. N. Arnold, R. S. Faik, and R. Winther, "Preconditioning in H(div) and applications," J. Math. Comput. **66**, 937–984 (1997).
45. M. Benzi, "Preconditioning techniques for large linear systems: A survey." J. Comput. Phys. **82**, 418–477 (2002).
46. N. J. Higham and T. Mary, "A new preconditioner that exploits low-rank approximations for factorization error," SIAM J. Sci. Comput. **41**, 59–82 (2019).
47. R. Hiptmair, "Operator preconditioning," J. Comput. Math. Appl. **52**, 699–706 (2006).
48. K. Mardal and R. Winther, "Preconditioning discretizations of systems of partial differential equations," J. Numer. Linear Algebra Appl. **18**, 1–40 (2011).
49. P. S. Vassilevski, *Multi-Level Block Factorization Preconditioners* (Springer, New York, 2008).
50. A. Wathen, "Preconditioning," Acta Numer., No. 24, 329–376 (2015).
51. R. Bridson and C. Greif, "A multipreconditioned conjugate gradient algorithm," SIAM J. Matrix Anal. Appl. **27**, 1056–1068 (2006).
52. V. Dolean, P. Jolivet, and F. Nataf, *An Introduction to Domain Decomposition Methods: Algorithms, Theory and Parallel Implementation* (SIAM, Philadelphia, 2015).
53. Y. L. Gurieva and V. P. Il'in, "On coarse grid correction methods in the Krylov subspaces," J. Math. Sci. **232**, 74–82 (2018).

54. Y. M. Laevsky and A. M. Matsokin, "Decomposition methods for solving elliptic and parabolic boundary value problems," Sib. Zh. Ind. Mat. **2**, 361–372 (1999).
55. Y. Vassilevski, "A hybrid domain decomposition method based on aggregation," Numer. Linear Algebra Appl. **11**, 327–341 (2004).
56. Y. Vassilevski and M. A. Olshanskii, Short Course on Multi-Grid and Domain Decomposition Methods (MAKS, Moscow, 2007).
2 57. L. Yu. H. Xiang and F. Nataf, "Two-level algebraic domain decomposition preconditioners using Jacobi–Schwarz smoother and adaptive coarse grid corrections," J. Comput. Appl. Math. **261**, 1–13 (2014).
58. A. Gaul, M. H. Gutknecht, J. Hesenta, and R. Nabben, "A framework for deflated and augmented Krylov subspace methods," SIAM J. Matrix Anal. Appl. **34**, 495–518 (2013).
59. R. Nabben and C. Vuik, "A comparison of deflation and coarse grid correction applied to porous media flow," SIAM J. Numer. Anal. **42**, 631–647 (2004).
60. R. Bank, R. Falgout, T. Jones, T. Manteuffel, S. McCormick, J. Ruge, "Algebraic multigrid domain and range decomposition (AMG-DD/AMG-RD)," SIAM J. Sci. Comput. **37**, 113–136 (2015).
61. A. Brandt, "Algebraic multigrid theory: The symmetric case," J. Appl. Math. Comput., No. 19, 23–56 (1986).
62. V. V. Shaidurov, "Some estimates of the rate of convergence for the cascadic conjugate-gradient method," J. Comput. Math. Appl. **31** (4–5), 161–171 (1996).
63. Y. L. Gurieva, V. P. Il'in, and A. V. Petukhov, "On multigrid methods for solving two-dimensional boundary-value problems," J. Math. Sci. **249** (2), 118–127 (2020).
64. Y. Notay, "Algebraic multigrid for Stokes equations," SIAM J. Sci. Comput. **39**, 88–111 (2017).
65. Y. Notay and A. A. Napov, "An efficient multigrid method for graph Laplacian systems II: robust aggregation," SIAM J. Sci. Comput. **39**, 379–403 (2017).
66. M. A. Olshanskii, *Lecture Notes on Multigrid Methods* (Inst. Vychisl. Mat., Ross. Akad. Nauk, Moscow 2012).
67. J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga, "Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods," J. Sci. Comput., No. 39, 340–370 (2009).
68. R. S. Varga, *Iterative Analysis* (Springer, 1962).
2 69. L. Yu. Kolotilina, "The convergence of certain incomplete block factorization splittings," J. Math. Sci. **86**, 2828–2834 (1997).
2 2 70. A. Yu. Yeremin, L. Yu. Kolotilina, and A. A. Nikishin, "Factorized sparse approximate inverse preconditioning III. Iterative construction of preconditioners," J. Math. Sci. **101**, 3237–3254 (2000).
71. P. Kumar, L. Grigori, F. Nataf, and Q. Nui, "On relaxed nested factorization and combination preconditioning," Int. J. Comput. Math. **93** (1), 178–199 (2016).
72. P. S. Vassilevski, "Preconditioning mixed finite element saddle-point elliptic problems," J. Numer. Linear Algebra Appl. **3** (1), 1–20 (1996).
73. W. Hackbusch, *Hierarchische Matrizen: Algorithmen und Analysis* (Springer, Berlin, 2009).
74. S. A. Solovyev, "Multifrontal hierarchically solver for 3D discretized elliptic equations" in *FDM 2014,* Dimov, I., Faragi, I., Vulkov, L. (eds.) LNCS, (Springer, Cham, 2015), vol. 9045, pp. 371–378.
75. M. Arioli and M. Manzini, "A network programming approach in solving Darcy's equations by mixed finite elements methods," Electron. Trans. Numer. Anal. **22**, 41–70 (2006).
76. M. Bern, J. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo "Support-graph preconditioners," SIAM J. Matrix Anal. Appl. **27**, 930–951 (2006).
77. C. Ponce and P.S. Vassilevski, "Solving graph Laplacian systems through recursive partitioning and two-grid preconditioning," SIAM J. Matrix Anal. Appl. **38**, 621–648 (2017).
2 78. V. P. Il'in and K. Yu. Laevsky, "Generalized compensation principle in incomplete factorization methods," Russ. J. Numer. Anal. Math. Model. **12**, 399–412 (1997).
79. M. Heath and C. Romine, "Parallel solution of triangular systems on distributed-memory multiprocessors," SIAM J. Sci. Stat. Comp. **9**, 558–588 (1988).
80. E. Hutter and E. Solomonik, "Communication-avoiding Cholesky QR-2 for rectangular matrices," arXiv: 1710.08471v6.
81. A. Ruhe, "Rational Krylov sequence methods for eigenvalue computation," J. Linear Algebra Appl. **58**, 39–45 (1984).
82. N. N. Kuznetsova, O. V. Diyankov, S. V. Kotegov, I. V. Krasnogorov, V. Y. Pravilnikov, and S. Y. Maliassov, "The family of nested factorizations," Russ. J. Numer. Anal. Math. Model. **22**, 393–412 (2007).
83. R. Wang, Q. Nu, and L. Lu, "A twisted block tangential filtering decomposition preconditioner," Math. Prob. Eng. Article ID 282307 (2009).
84. M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," Acta Numerica **14**, 1–137 (2005).
85. F. Brezzi, "Stability of saddle points in finite dimensions, frontiers in numerical analysis," (Springer, Berlin, 2013), pp. 17–61.

86. Y. Notay and A. Napov, "Further comparison of additive and multiplicative coarse grid correction," J. Appl. Numer. Math. **65**, 53–62 (2013).
87. Y. V. Bychenkov and E. V. Chizhonkov, *Iterative Methods for Solving Saddle Point Problems* (Binom, Moscow, 2010) [in Russian].
88. C. Greif and D. Schotzau, "Preconditioners for saddle point linear systems with highly singular (1.1) Blocks," in *Special Volume on Saddle Point Problems,* Electron. Trans. J. Nшner. Anal. **22**, 114–121 (2006).
89. C. Greif and D. Schotzau, "Preconditioners for the discretized harmonic Maxwell equations in mixed form," Numer. Linear Algebra Appl. **14**, 281–287 (2007).
90. Y. Notay, "Convergence of some iterative methods for symmetric saddle point linear systems," SIAM J. Matrix Anal. Appl. **40**, 122–146 (2019).
91. V. P. Il'in and G. Y. Kazantsev, "Iterative solution of saddle-point systems of linear equations," J. Math. Sci. **249**, 199–208 (2020).
92. Y. Notay, "Algebraic two-level convergence theory for singular systems," SIAM J. Matrix Anal. Appl. **37**, 1419–1439 (2016).
93. J. Dongarra, L. Grigori, and N. J. Higham, "Numerical algorithms for high performance computational science," Phil. Trans. R. Soc. A **378**, (2020).
94. V. P. Il'in, "Projection methods in Krylov subspaces," J. Math. Sci. **240**, 772–782 (2019).
95. N. I. Gorbenko and V. P. Il'in, "The additive Peaceman–Rachford method," J. Math. Sci. **216**, 753–760 (2016).
96. A. Y. Aleinikov, R. A. Barabanov, Y. G. Bartenev, et al., "An application of parallel solvers for SLAEs in the applied packages for engineering," *Proc. Int. Conf. "Supercomputing and Mathematical Modeling,"* (Unicef, 2015), pp. 102–110.
97. P. Bastian and M. Blatt, Iterative solver template library (DUNE). https://www.dune-project.org/.
98. Y. V. Vassilevskii, I. N. Kon'shin, G. V. Kopytov, and K. M. Terekhov, INMOST—A Software Platform and Graphical Medium for the Development of Parallel Numerical Models on General-Type Grids (Moscow State Univ., Moscow, 2018) [in Russian].

Translated by A. Klimontovich

SPELL: 1. sdecial, 2. Yu