Parallel Methods for Solving Saddle Type Systems

V. P. Il'in[1] and D. I. Kozlov[1,2(✉)]

[1] Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia
{ilin,di_kozlov}@sscc.ru
[2] Novosibirsk State University, Novosibirsk, Russia

Abstract. Parallel methods for solving saddle-type algebraic systems that are relevant for modeling processes and phenomena in the problems of electromagnetism, hydro-gas dynamics, elastoplasticity, filtration and other applications are considered. Preconditioned iterative processes in the Krylov subspaces, including the efficient generalization of the Golub-Kahan-Arioli bidiagonalization method, are investigated as applied to large SLAEs with sparse matrices that arise when approximating multi-dimensional boundary value problems with a complex geometric configuration of computational domains and the contrasting material properties of various media on unstructured grids. It is supposed to store the matrices in compressed formats that require special technologies for working with big data. The parallelization of the proposed class of block algorithms is carried out by means of hybrid programming on supercomputers of a heterogeneous architecture with distributed and hierarchical shared memory, using the means of inter-node message transmission, multi-threaded computing, operation vectorization. A comparative analysis of various algorithmic approaches is carried out on the basis of the estimates of the performance and resource intensity of the corresponding software implementations.

Keywords: large sparse SLAEs · saddle matrices · iterative processes · algorithm parallelization · Krylov subspaces · computing performance

1 Introduction

Systems of linear algebraic equations (SLAEs) of the saddle type in their classical version are associated with second order block matrices having a zero lower-right block and written in the following form (we limit ourselves to a real case for simplicity):

$$Au \equiv \begin{bmatrix} D & C \\ C^\top & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \equiv f, \tag{1}$$

$$u, f \in \mathcal{R}^N; \quad u_1, f_1 \in \mathcal{R}^{N_1}; \quad u_2, f_2 \in \mathcal{R}^{N_2}; \quad N = N_1 + N_2,$$

$$D \in \mathcal{R}^{N_1,N_1}, \quad C \in \mathcal{R}^{N_1,N_2}, \quad A \in \mathcal{R}^{N,N}.$$

L. Sokolinsky and M. Zymbler (Eds.): PCT 2022, CCIS 1618, pp. 85–98, 2022.
https://doi.org/10.1007/978-3-031-11623-0_7

For a number of characteristic applications, the matrix A from (1) has the following property, see [1]–[2].

Property A. The matrix D is symmetric and positive definite (s.p.d.), the null spaces of the matrices D and $C^\top$ do not intersect, i.e., $ker(D) \cap ker(C^\top) = \{0\}$, which ensures the non-singularity of the matrix A.

The saddle-type matrices A from (1) are sometimes considered in a more general form

$$A = \begin{bmatrix} D & C_1 \\ C_2^\top & -\varepsilon H \end{bmatrix}, \quad \varepsilon > 0, \tag{2}$$

where $H \in \mathcal{R}^{N_2,N_2}$ is the positive-semi-definite matrix, in the sense of fulfilling the inequality $(Hv, v) \geq 0$ for $v \in \mathcal{R}^{N_2}$, introduced either by the conditions of the problem statement, or for the reasons of the regularization of original SLAEs (1). In addition, the matrices D, H, A can be asymmetric, i.e., $C_1 \neq C_2, D \neq D^\top, H \neq H^\top, A \neq A^\top$.

Note that without loss of generality, instead of (1), we can consider saddle SLAEs of the form

$$\begin{bmatrix} D & C \\ C^\top & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ 0 \end{bmatrix}. \tag{3}$$

Indeed, if we take any particular solution of the subsystem $C\hat{u}_1 = f_2$, the vector $u = \check{u} + \hat{u}$ is the solution of systems of linear algebraic equations (1), satisfying the system

$$\begin{bmatrix} D & C \\ C^\top & 0 \end{bmatrix} \begin{bmatrix} \check{u}_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 - D\hat{u}_1 \\ 0 \end{bmatrix}.$$

Note also that any solution to SLAEs (1) simultaneously satisfies the system

$$\begin{aligned} \tilde{A}v = \tilde{A} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \equiv \begin{bmatrix} \tilde{D} & C \\ C^\top & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ 0 \end{bmatrix} \equiv \tilde{f}, \\ \tilde{D} = D + \gamma R, \quad R = CK^{-1}C^\top, \quad \gamma \geq 0, \end{aligned} \tag{4}$$

where $v\tilde{f} \in \mathcal{R}^N$ and $K \in \mathcal{R}^{N_1,N_1}$ is an arbitrary non-degenerate matrix. Since the latter system is formally a regularization, or generalization, of SLAEs (1), we further focus on the algorithm for solving equation (4). The parameter γ is introduced for the convenience of varying the algorithm, in particular, $\gamma = 0$ means no regularization.

Note that the non-degenerate matrix of the form of (1) can have an alternating spectrum, which creates its own difficulties in the iterative solution of the corresponding algebraic system.

Without loss of generality, the studied SLAEs can be written down in the following form:

$$\tilde{A} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \equiv \begin{bmatrix} \tilde{D} & C \\ C^\top & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix}, \quad \tilde{D} = D + \gamma CK^{-1}C^\top. \tag{5}$$

It is easy to check that if in (4), the vector u_1 is replaced by $u_1 + \tilde{D}^{-1} f_1$, then this system will take the form of (5) with the right hand side $g = -C^\top \tilde{D}^{-1} f_1$. It is assumed that in (5), $\tilde{D}$ and K are the s.p.d. matrices, and the inequality $N_1 \geq N_2$ also holds.

Algebraic systems of the form of (1), (2) are in demand in many topical problems of electromagnetism, hydro-gas dynamics, heat and mass transfer, elasticity, multiphase filtration in porous-fractured media and in other applications. In particular, they arise in mixed classical or generalized formulations for initial boundary value problems. A large number of papers are devoted to the study of the saddle SLAEs under consideration and methods for solving them, see [3–12] and an extensive list of literature given therein. In recent years, due to the increasing role of the predictive modeling of real processes and phenomena with big data, there has been a significant increase in the interest in high-performance methods and technologies for solving large SLAEs with sparse matrices arising from the approximations of multi-dimensional boundary value problems with complex configurations of computational domains and the contrasting material properties of different media using finite difference methods, finite volumes, finite elements and discontinuous Galerkin algorithms of various orders of accuracy on unstructured grids [13]. The resulting algebraic systems have $10^8 - 10^{10}$ sizes and are poorly conditioned (the conditioning numbers of matrices reach 10^{13} and greater), thus, their numerical solution in practice takes up to 80% of the total machine resources. Therefore, the main reserve for speeding up calculations is the scalable parallelization of algorithms by means of hybrid programming with the inter-node message transmission, multi-threaded messages and vectorization of operations (MPI, OpenMP, AVX systems, respectively) of heterogeneous architecture supercomputers with distributed and hierarchical shared memory, see [14–18]. It should be noted that due to the large-block structure of saddle matrices, two-level iterative algorithms with specific features of data organization and memory access methods are characteristic of solving the corresponding SLAEs, and optimization is critical for improving the performance of software implementations.

This paper is structured as follows. Section 2 discusses the main approaches to constructing preconditioned block iterative methods in the Krylov subspaces for the effective solution of the considered algebraic systems. Section 3 deals with the analysis of the performance of the scalable parallelization of the studied computational processes in the weak and strong senses. In conclusion, the problems of improving the performance of algorithms for solving saddle SLAEs are discussed in the light of the current trends in the development of supercomputer architectures.

2 Iterative Algorithms in the Krylov Subspaces for Solving Saddle SLAEs

In this section, we first characterize the general property of Krylov-type iterative processes and then focus on their features when solving SLAEs with saddle matrices.

2.1 General Scheme of the Krylov Approaches for Symmetric and Non-symmetric Algebraic Systems

For solving symmetric or non-symmetric SLAEs

$$Au = f, \quad A \in \mathcal{R}^{N,N}; \quad f \in \mathcal{R}^N \tag{6}$$

iterative methods in the Krylov subspaces can be written down as follows:

$$u^{n+1} = u^n + \alpha_n p^n = u^0 + \alpha_0 p^0 + ... + \alpha_n p^n,$$
$$r^{n+1} = r^n - \alpha_n A p^n = r^0 - \alpha_0 A p^0 - ... - \alpha^n A p^n, \tag{7}$$

Here u^0 is an arbitrary initial vector, $r^0 = f - Au^0$ is the corresponding residual, α_n and p^n are the iterative parameters and guiding vectors. In the absence of a precondition for system (6), $p^0 = r^0$ is conventionally assumed (to be used below), although formally the initial guiding vector can be arbitrary. Assume that the direction vectors are A^γ-orthogonal, i.e.,

$$(p^n, p^k)_\gamma = (A^\gamma p^n, p^k) = \rho_k^{(\gamma)} \delta_{n,k}, \quad \rho_k^{(\gamma)} = (A^\gamma p^k, p^k) = ||p^k||_\gamma^2, \tag{8}$$

where $\delta_{n,k}$ is the Kronecker symbol, and the exponents are equal to $\gamma = 0, 1, 2$. Then the residuals are r^n for the value of the parameters

$$\alpha_n = \sigma_n / \rho_n, \quad \sigma_n = (r^0, p^n)_{\gamma-1} = (r^n, p^n)_{\gamma-1} = ||r^n||_{\gamma-1}^2, \tag{9}$$

providing a minimum of the functional $\Phi_\gamma(r^n) = (A^{\gamma-2} r^n, r^n)$ in the Krylov subspaces

$$\mathcal{K}_n(r^0, A) = \mathrm{Span}(r^0, Ar^0, ..., A^{n-1} r^0). \tag{10}$$

If the matrix A is symmetric, then orthogonality conditions (8) are provided when determining the direction vectors p^n by the two-term recursive formulas

$$p^{n+1} = r^{n+1} + \beta_n p^n, \quad \beta_n = \sigma_{n+1} / \sigma_n. \tag{11}$$

For the case $\gamma = 0$, however, the calculation of α_n must be done in a different way. Since the exact solution of SLAEs can be represented as a basis decomposition

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

then the iterative vectors and the corresponding residual vectors are presented in the following form:

$$v^n = u - u^n = \alpha_n p^n + ... + \alpha_m p^m,$$
$$r^n = A v^n = \alpha_n A p^n + ... + \alpha_m A p^m. \tag{12}$$

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

$$\alpha_n = (v^n, p^n)_\gamma / ||p^n||_\gamma^2 = -\alpha_{n-1} (v^n, A p^{n-1}) / ||p^n||_\gamma^2$$
$$= -\alpha_{n-1} (r^n, p^{n-1})_\gamma / ||p^n||_\gamma^2. \tag{13}$$

Here the orthogonality of the vectors p^{n-2}, p^{n-1} and p^n is used. The calculation of the coefficient β_n in this case should be carried out according to formula (11).

These algorithms for $\gamma = 0, 1, 2$ have the names of the methods of minimal iterations, or errors [17], as well as conjugate gradients and conjugate residuals, respectively.

The described approaches allow for a simple generalization to SLAEs, preconditioned with the help of some s.p.d. matrices B. To preserve the symmetry of the systems, it is advisable to do this by two-way preconditioning using the formally introduced matrix $B^{1/2}$. As a result, system (1) takes the form

$$\bar{A}\bar{u} = \bar{f}, \quad \bar{A} = B^{-1/2}AB^{-1/2}, \bar{u} = B^{1/2}u, \quad \bar{f} = B^{-1/2}f. \tag{14}$$

A result of applying the conjugate direction formulas to SLAE (14), after certain transformations for $\gamma = 1, 2$, we obtain the following iterative process:

$$\begin{aligned} \hat{p}^0 = \hat{r}^0 = B^{-1}r^0 = B^{-1}(f - Au^0), \quad n = 0, 1, 2, ...; \\ u^{n+1} = u^n + \alpha_n\hat{p}^n, \quad \hat{r}^n = \hat{r}^n - \alpha_n A\hat{p}^n; \\ \hat{p}^{n+1} = \hat{r}^{n+1} + \beta_n\hat{p}^n, \quad \alpha_n = \sigma_n/\rho_n, \quad \beta_n = \sigma_{n+1}/\sigma_n; \\ \sigma_n = (A^{\gamma-1}\hat{r}^n, \hat{r}^n), \quad \rho_n = (B^{-1}A\hat{p}^n, A^{\gamma-1}\hat{p}^n), \end{aligned} \tag{15}$$

where the new vectors are related to the previous relations $\hat{p}^n = B^{-1}p^n$, $\hat{r}^n = B^{-1}r^n$. In the method of minimum iterations with $\gamma = 0$, the calculation of the parameters α_n, β_n must be carried out according to formulas (9), (10), with the replacement values of $\bar{r}^n = B^{-1}r^n$, $\bar{p}^n = B^{-1}p^n$, respectively.

For non-symmetric algebraic systems, the methods of their solution become significantly more complicated. We briefly present a description of specific approaches for a fairly wide class of multi-preconditioned algorithms of semi-conjugate direction [18]. In general, these iterative processes in the block Krylov subspaces can be presented as follows:

$$r^0 = f - Au^0, \quad n = 0, \dots : \quad u^{n+1} = u^n + P_n\bar{\alpha}_n,$$

$$P_n = (p_1^n, \dots, p_{M_n}^n), \quad r^{n+1} = r^n - AP_n\bar{\alpha}_n, \quad \bar{\alpha}_n = (\alpha_{n,1}, \dots, \alpha_{n,M_n})^\top.$$

Here $p_1^n, ..., p_{M_n}^n$ are the guiding vectors that make up the matrix P_n of the n-th iteration, and $\bar{\alpha}_n$ is the vector of the iterative parameters. With respect to the vectors p_k^n in the above relations, only orthogonality conditions are assumed to be fulfilled

$$(Ap_k^n, A^\gamma p_{k'}^{n'}) = \rho_{n,k}^{(\gamma)}\delta_{n,n'}^{k,k'}, \quad \rho_{n,k}^{(\gamma)} = (Ap_k^n,\ A^\gamma p_k^n),$$

$$\gamma = 0, 1, \quad n' = 0, 1, \dots, n-1, \quad k, k' = 1, 2, \dots, M_n.$$

If, at the same time, the coefficients $\bar{\alpha}_n = \{\alpha_{n,l}\}$ are defined by the formulas

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

then the functional of the residual $\Phi_n^{(\gamma)}(r^{n+1}) \equiv (r^{n+1}, A^{\gamma-1}r^{n+1})$ reaches its minima in the Krylov block subspaces

$$\mathcal{K}_M = \mathrm{Span}\{p_1^0, ..., p_{M_0}^0, Ap_1^1, \ldots, Ap_{M_1}^1, \ldots, Ap_1^n, \ldots, Ap_{M_n}^n\},$$

$$M = M_0 + M_1 + \cdots + M_n,$$

for $\gamma = 1$, and in the case of symmetry of the matrix A and for $\gamma = 0$.

The orthogonality properties of the guiding vectors can be provided if they are determined using "multi-conditional" recurrence relations in which each vector p_l^{n+1} corresponds to "its" preconditioning matrix $B_{n+1,l}$:

$$p_l^0 = B_{0,l}^{-1} r^0, \;\; p_l^{n+1} = B_{n+1,l}^{-1} r^{n+1} - \sum_{k=0}^{n} \sum_{l=1}^{M_k} \beta_{n,k,l}^{(\gamma)} p_l^k, \quad n = 0, 1, \ldots;$$

$$B_{n,l} \in \mathcal{R}^{N,N}, \quad i = 1, \ldots, M_n; \;\; \gamma = 0, 1,$$

$$\bar{\beta}_{n,k}^{(\gamma)} = \{\beta_{n,k,l}^{\gamma}\} = \left(\beta_{n,k,1}^{(\gamma)} \ldots \beta_{n,k,M_n}^{(\gamma)}\right)^{\top} \in \mathcal{R}^{M_n},$$

$$\beta_{n,k,l}^{(\gamma)} = -\left(A^{\gamma} p_l^k, AB_{n+1,l}^{-1} r^{n+1}\right) / \rho_{n,l}^{\gamma}, \quad n = 0, 1, \ldots;$$

$$k = 0, \ldots, n; \quad l = 1, \ldots, M_n.$$

The peculiarity of the algorithms under consideration when solving poorly conditioned asymmetric SLAEs is of a high resource intensity, in terms of both the amount of calculations and the required memory, when conducting a large number of iterations. The remedy for this disadvantage can be carried out in two ways by reducing the number of used and saved direction vectors. The first of them is to reduce the recursion taking into account only its last m vectors. The second way consists in periodic restarts when using a given number of m iterations, the residual vector being calculated from the recurrence formula, and the original equation being to zero as the iteration:

$$r^{n_t} = f - Au^{n_t}, \quad n_t = mt, \quad t = 0, 1, ...,$$

where t is the number for the restart. Further calculations up to $n = n_{t+1}$ are carried out according to usual recursions. Both of these approaches lead to a significant slowdown in the iterative process.

To eliminate such a stagnating effect, it is proposed to add the second level of iterations using the least squares method (LSM) [17]. Let the "restart" approximations $u^{n_0}, u^{n_1}, ..., u^{n_t}, \; n_0 = 0$ be known. Then to correct the iterative vector u^{n_t}, which is the initial one for the next restart period, we use the following linear combination:

$$\hat{u}^{n_t} = u^{n_t} + b_1 v_1 + \ldots + b_t v_t = u^{n_t} + v^{n_t}, \;\; v^{n_t} = V_t \bar{b}, \quad \bar{b} = (b_1, \ldots, b_t)^{\top},$$

$$V_t = \{v_k = u^{n_k} - u^{n_{k-1}}, \;\; k = 1, \ldots, t\} \in \mathcal{R}^{N,t},$$

the vector of the coefficients $\bar{b}$ of which is determined by the condition of the minimum norm of residuals $||r^{n_t}||$ from the generalized solution of the overdefined algebraic system

$$W_t^\top W_t \bar{b} = r^{n_t} \equiv a^{n_t}, \quad W_t = AV_t.$$

The solution to this problem can be obtained, for example, using the QR - or SVD - decomposition of the matrix W_t. The normal solution with the minimum norm $||\bar{b}||$ is determined after applying the left Gauss transformation:

$$W_t^\top W_t \bar{b} = W_t r^{n_t}.$$

A more lightweight SLAEs format, in the sense of reducing its condition number, follows after multiplying the system on the left by the matrix V_t:

$$C_t \bar{b} \equiv V_t^\top A V_t \bar{b} = V_t^\top r^{n_t}.$$

If the matrix V_t has a full rank, then the matrices A and C_t will be non-degenerate at the same time. In this case, for a correction vector we have $v^{n_t} = B_t r^{n_t} \equiv V_t (V_t^T A V_t)^{-1} V_t^T r^{n_t}$, where the matrix $B_t = V_t \hat{A}^{-1} V_t^T$, $\hat{A} = V_t^T A V$ is a low-rank approximation of the matrix A^{-1}. In the approach considered, all restart vectors are stored in the corrected form, and the corresponding residuals are calculated using the formula $r^{n_t} = f - A\hat{u}^{n_t}$.

If there is no inverse for some matrix under consideration, then a generalized inverse matrix is used. Numerous experiments using the LSM to speed up the Krylov processes with restarts show its high efficiency.

We also note the following possibility of improving the performance of the SCD (Semi-Conjugate Direction) methods with restarts: when iterating the first restart period, remember all the p^n direction vectors, as well as the Ap^n vectors, and when calculating subsequent restart periods, we do not consider new vectors p^n and Ap^k, but use the previous ones.

The described class of SCD-methods with dynamic multi-conditionality in terms of the rate of convergence of iterations is equivalent to other well-known algorithms for solving asymmetric SLAEs in the Krylov subspaces, among which the generalized minimum residuals method (GMRES) based on the Arnoldi orthogonalization and existing in various versions is the most popular. The research into iterative methods for solving algebraic systems with saddle-type matrices involves the use of a wide variety of block preconditioners. The starting point for their construction is the following formula for the factorization of the matrix $A = D + L + U$, where D is block-diagonal, and L, U are the strictly lower and upper triangular matrices:

$$A = (G + L)G^{-1}(G + U), \quad G = D - LG^{-1}U,$$

$$G_1 = D, \quad G_2 = -\varepsilon H - C_2^\top D^{-1} C_1.$$

In particular, if the matrix A has a block structure of the form of (2), then $G = block - diagonal\{G_1, G_2\}$ has non-zero diagonal blocks only

$$G_1 = D, \quad G_2 = -\varepsilon H - C_2^\top D^{-1} C_1.$$

Note that if the matrix A is symmetric, i.e., $C_1 = C_2$ and $L = U^\top$, D and R in (2) are s.p.d. matrices, the given factorization is a congruence transformation of the block-diagonal matrix G. Since it obviously has an alternating sign spectrum, the matrix A has the same property, which causes certain difficulties in constructing methods for solving the corresponding SLAEs. If the definition of the matrix G is replaced by some approximation that allows simple calculations, then we get a family of preconditioners $B \approx A$. This implies, in particular, an iterative method of the Uzava type (see [14–20]). Another promising way is to construct block-diagonal preconditioners of the form

$$B = \begin{bmatrix} \tilde{D} + CK_1^{-1}C^\top & & 0 \\ & \ddots & \\ 0 & & K_2 \end{bmatrix},$$

where K_1 and K_2 are some s.p.d. matrices, see [17,18].

2.2 Generalized G-K-A – Bidiagonalization Method

Next, we consider a family of iterative methods for solving saddle symmetric SLAEs with the matrix $\tilde{A}$ from (4), based on the efficient approach of the G-K – Golub-Kahan bidiagonalization, which was originally proposed for a singular decomposition of rectangular matrices, but then in the publications by M. Saunders, M. Arioli, C. Greif and some other authors was successfully used to solve algebraic systems, including those with allowance for a block saddle structure.

More specifically, we present a generalization of the Golub-Kahan-Arioli algorithm, which is published in [9] under the title <<generalized G-K–bidiagonalization method>>, based on the construction of $\tilde{D}$-orthogonal vectors v_k and P-orthogonal vectors q_k, which satisfy the conditions

$$\begin{aligned} CQ_n &= \tilde{D}V_nB_n, \quad V_n^\top \tilde{D}V_n = I_{N_1}, \\ C^\top V_n &= PQ_nB_n^\top, \quad Q_n^\top PQ_n = I_{N_2}, \end{aligned} \tag{16}$$

where $V_n = [v_1, ..., v_n] \in \mathcal{R}^{N_1,n}$, $Q_n = [q_1, ..., q_n] \in \mathcal{R}^{N_2,n}, P \in \mathcal{R}^{N_2,N_2}$ – s.p.d. matrices and $B_n \in \mathcal{R}^{n,n}$ is the bidiagonal matrix

$$B_n = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ 0 & \alpha_2 & \beta_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \alpha_{n-1} & \beta_n \\ 0 & \ddots & 0 & 0 & \alpha_n \end{bmatrix}.$$

Let us note that in [9] only the case of $K = P$ is considered, which is not is mandatory, see formula (5). Introducing new unknown vectors $\mu^n =$

$(\mu_1, ..., \mu_n)$, $\nu^n = (\nu_1, ..., \nu_n) \in \mathcal{R}^n$, substituting the expressions

$$u_1 = V_n \mu^n, \quad u_2 = Q_n \nu^n \tag{17}$$

in (5) and multiplying system (5) on the left by the block-diagonal matrix *block-diag*$(V^\top, Q^\top)$, we get

$$V_n^\top \tilde{D} V_n (\mu^n + B_n \nu^n) = 0, \quad Q_n^\top P Q_n B_n^\top \mu^n = Q_n^\top g. \tag{18}$$

Thus, SLAEs (18) are reduced to the form

$$\begin{bmatrix} I_n & B_n \\ B_n^\top & 0 \end{bmatrix} \begin{bmatrix} \mu^n \\ \nu^n \end{bmatrix} = \begin{bmatrix} 0 \\ Q^\top g \end{bmatrix}. \tag{19}$$

Assuming further $Q_n^\top g = e_1 ||g||_{P^{-1}}, \quad e_1 = (1, 0, ...)^\top$, we define the vector

$$q_1 = P^{-1} g / ||g||_{P^{-1}}, \quad ||g||_{P^{-1}} = (g, P^{-1} g)^{1/2}.$$

Let us find the initial vector v_1:

$$\alpha_1 \tilde{D}^{-1} v_1 = C q_1, \quad v_1 = w/\alpha_1, \quad \alpha_1 = \sqrt{w^\top C q_1}, \quad w = \tilde{D}^{-1} q_1. \tag{20}$$

Note that the vector $\mu = B_n^{-\top} Q_n^\top g$ is determined up to a constant by the first column of the matrix $B_n^{-1} = (B_n^\top)^{-1}$.

Further the vectors v_n, q_n and the matrix B entries α_n, β_n are calculated from the following recurrent relations, $n = 1, 2, ...$:

$$\begin{aligned} s = P^{-1}(C v_n - \alpha_n P q_n), \quad \beta_{n+1} = \sqrt{s^\top P s}, \\ q_{n+1} = s/\beta_{n+1}, \quad w = \tilde{D}^{-1}(C^\top q_{n+1} - \beta_{n+1} \tilde{D} v_n), \\ \alpha_{n+1} = (w^\top \tilde{D} w)^{1/2}, \quad v_{n+1} = w/\alpha_{n+1}. \end{aligned} \tag{21}$$

Successive approximations u^n, beginning with (17), (18), are determined by the first n columns of the matrix V, according to

$$u_1^{n+1} = \sum_{j=1}^{n} \mu_j v_j = u_1^n + \mu_n v_n, \tag{22}$$

where μ_j are the components of the vector μ^n from (19), calculated by the formulas

$$\mu_1 = ||g||_{P^{-1}}/\alpha_1, \quad \mu_{j+1} = -\beta_{j+1} \mu_j / \alpha_{j+1} \quad j = 1, 2..... \tag{23}$$

Omitting the details of the derivation of the formula (see [10]), we present the resulting recurrence relation for the iterative solution:

$$u_2^{n+1} = u_2^n - \nu_n d_{n+1}, \quad d_1 = q_1/\alpha_1, \quad d_{n+1} = (q_{n+1} - \beta_{n+1}\alpha_n)/\alpha_{n+1}, \tag{24}$$

where d_n is the n-th column $D = QB^{-1}$. This approach is called the generalized G-K-A – bidiagonalization algorithm. At each step of such an iterative process, the error norm $||u - u^n||$ is minimized. As noted in [14], the two-level iterative method demonstrates high performance and convergence rate when solving saddle-type SLAEs obtained in grid approximations of the mixed formulations of multi-dimensional boundary value problems.

3 Scalable Parallelization of Iterative Methods

In general, the computational quality of the algorithm can be characterized by means of two different features. The first one is mathematical efficiency, which can be estimated by the total number of arithmetical operations. The second characteristic is more practical and is measured by the run time of the method implementation on a specific computer configuration. In other words, in this case we speak about the quality of mapping algorithms onto an architecture that usually has a heterogeneous structure with distributed and hierarchical shared memory. A significant complexity of the problem of studying the performance of an executable program code lies in the actual absence of a mathematical model of supercomputer calculations, and optimization attempts require some experience and skill gained in order to avoid repeated trials and errors. Scalable parallelization is conventionally understood in either a strong or weak sense. The first means a reduction in the calculation time of a fixed task with an increase in the number of computing devices, for example, cores. In the second case, a simultaneous proportional increase in the resource intensity of the problem (the number of degrees of freedom) and the number of arithmetic devices are considered (ideally, the estimated time remains approximately constant). The SLAEs of most interest to us have high orders and sparse matrices with large conditionality numbers and an irregular structure. This does not only lead to an increase in the number of iterations, but also forces one to work with distributed and/or hierarchical shared memory systems, and also significantly slows down the access to data. It should be said that the large-block structure of saddle matrices strongly affects the computational scheme of iterative algorithms and the ways of parallelizing them when changing the type of a preconditioner. In this section, we briefly focus on the general current problems of parallelization and in more detail on the proposed generalization of the G-K-A – bidiagonalization algorithm, as applied to the solution of saddle SLAEs obtained from a finite element approximation of the three-dimensional initial boundary value problem for the two-phase filtration proposed in [24]. In this case, the order of SLAEs (1)–(5) is equal to $N \cong 4h^{-3}$, where h is the characteristic element of the grid, and the dimensions of the diagonal blocks are equal to $N_1 \cong 3h^{-3}$ and $N_2 \cong h^{-3}$. In physical terms, the subvector $u_1 = \{u^x, u^y, u^z\}$ consists of components of the velocity vector along different axes of the Cartesian coordinate system referred to the midpoints of the faces of the cubic grid, and u_2 is a set of values of the scalar pressure function at the centers of the grid cells. The matrix D is block-diagonal, and its non-zero blocks are easily invertible s.p.d. tridiagonal matrices with a strict diagonal dominance. The off-diagonal matrix is represented as three block rows $C = (C_1^\top, C_2^\top, C_3^\top)$, and each $C_k,\ k = 1, 2, 3$, is a two-diagonal matrix. The parallelization of the G-K-A – bidiagonalization method, presented in Sect. 2.2., consists of the following main stages. The auxiliary orthogonal vectors v_n, q_n and the entries α_n, β_n of the matrix B by formulas (20), (21) are calculated. The tedious procedure of this stage essentially depends on the structure of the matrices $\tilde{D}$ and K introduced in (5), and the matrices of the form P of (16). We get a simple case at $\gamma = 0$ in (5), i.e., the matrix K

is missing, and P is a diagonal matrix. At the same time, the appeal $\tilde{D} = D$ is implemented by efficient iteration-free runs, which can be performed in parallel without any additional expenditure of machine resources. If $\gamma \neq 0$, then even with the diagonal character of K, the inversion of the matrix $\tilde{D}$ (more precisely, the solution of auxiliary SLAEs with such a matrix) requires the introduction of a two-level iterative process. The issue of optimizing these algorithm parameters, which can potentially significantly reduce the number of iterations, seems to be non-trivial and requires a special study.

We denote the possibilities of parallelization:

1. The parallelization of vector-matrix operations of the Krylov processes.
2. The calculation of two-term recursions for the vectors u_1^n, u_2^n is performed by formulas (24). These vector operations are naturally parallelized with a linear speedup.
3. The formation of data structures and buffers is based on the separation of computational vectors, which geometrically corresponds to the decomposition of computational and grid domains to subdomains. For big data tasks, reducing communications at each iteration is critical to scalable parallelization.

The performance of parallel computing is defined mainly by the speedup, which is determined by the formulas

$$S_p = T_1/T_p, \quad T_p = T^a + T^c. \tag{25}$$

Here, T_p means the run time of solving the task on p processors. This value consists of two parts: the times of data exchanges and the implementation of the arithmetic operations. The latter can be described approximately by the following relations:

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

In these formulas, τ_a means the average run time of an arithmetic operation, N_a is their total number, N_t is the number of communications, τ_0 and τ_c are the memory system waiting time and the transfer duration of one value, and N_c is the average volume of one data exchange. Since the machine constants are satisfied to the conditions $\tau_0 \gg \tau_c \gg \tau_a$, we can propose the following recommendations for the algorithms being constructed: we should try to minimize the volume of communications, and the exchanges should be carried out not in small, but in large portions, i.e., if possible, to carry out the preliminary accumulation of data buffers. These conclusions are even true because interprocessor information transfers not only slow down the computing process, but are also the most energy-consuming operations, and this becomes a significant factor in the cost of operating a supercomputer.

One of the important practical problems of scalable parallelization is due to solving large SLAEs with sparse matrices that arise from the grid approximations of multi-dimensional boundary or initial-boundary value problems. Here, the main approaches are additive domain decomposition methods with two-level iterative processes and the use of hybrid programming tools. The top level

iterations implemented over the subdomains are carried out by means of MPI (Massage Passing Interface) for communications between the contacting subdomains. The low level of the algorithm includes the simultaneous solution of algebraic subsystems in the corresponding subdomains. This stage is parallelized by multi-threaded computing (OpenMP). At each such iteration, the values of approximate solutions are exchanged on the interface boundary surfaces of the contacting subdomains. Naturally, all matrix and vector data for subsystems are performed in the process-distributed form. The solution to SLAEs in each of the subdomains is parallelized using multi-threaded computing (OpenMP-type systems) on multi-core processors with shared memory. Additional speedup here can be achieved by vectorizing operations (AVX-type command systems based on SIMD – Single Instruction Multi Data technologies), see reviews in [20–25]. Unfortunately, here we can state the absence of regular programming systems with the automatic parallelization of algorithms, so that the success in the scalability of speeding up calculations largely depends on the art and skill of a mathematician-programmer. One of the common modern supercomputer configurations is a network of multi-core servers, with a number of cores in several tens or hundreds, which have several memory levels with different information exchange rates. For quite understandable reasons, larger memory devices have a lower data transfer rate (the fastest are the registers of arithmetic units - AU). In this situation, communication channels between different levels of memory are represented as a bottleneck, the access to which dramatically degrades the performance of the computing process. The most successful algorithmic solutions are those that use a small-block structure of data with the maximum use of AU registers, as well as the features of implementing their communications with lower-level memory.

Strictly speaking, the main objective of code optimization is not parallelism, but high performance. A significant speedup of calculations can be attained using variable precision machine arithmetic. The conventional way to solve large SLAEs is to use standard double precision with a 64-bit floating-point representation length. Many years of numerical experience show that this accuracy is sufficient in practice. However, for some ill-conditioned algebraic problems, it is necessary to use quadruple precision (128 bits). On the other side, at many stages of the algorithms, it is enough to apply single (32 bits) and even half precision (16 bits) which can be performed much faster. Nevertheless, with this approach, it is necessary to check the stability of computations. Hopefully, such an intelligent problem will be solved in the near future. Another way to obtain high performance and code optimization can be achieved through the efficient use of reliable implemented numerical tools (from SPARSE BLAS, for example) that are adapted to various computer platforms.

4 Conclusion

Parallel iterative methods in the Krylov subspaces for solving large saddle-type SLAEs with various matrices, relevant in the problems of electromagnetism,

strength, hydro-gas dynamics and other applications, are considered. The issues of constructing preconditioners for symmetric or non-symmetric systems using universal least squares algorithms to speed up the Krylov iterations are discussed. Approaches to the generalization of Golub-Kahan-Arioli bidiagonalization methods are described. Scalable distributed technologies in the strong and weak senses, focused on minimizing communication losses, based on the use of hybrid programming tools, namely, the transmission of inter-node messages, multi-threaded computing and the vectorization of operations (MPI, OpenMP, and AVX systems), on supercomputers of a heterogeneous architecture with distributed and hierarchical shared memory, are considered.

References

1. Benzi, M., Golub, G.H., Liesen, J.: Numerical Solution of Saddle Point Problems. Acta Numerica **14**, 1137 (2005). https://doi.org/10.1017/S0962492904000212
2. Brezzi, F.: Stability of saddle points in finite dimensions. In: Blowey, J.F., Craig, A.W., Shardlow, T. (Eds.) Frontiers in Numerical Analysis. Universitext, pp. 17–61. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-55692-0_2
3. Vassilevski, P.S.: Preconditioning mixed finite element saddle - point elliptic problems. J. Numer. Linear Algebra App. **3**(1), 1–20 (1996). https://doi.org/10.1002/(SICI)1099-1506(199601/02)
4. Bychenkov, Y.V., Chizhonkov, E.V.: Iterative Methods for Solving Saddle Point Problems. Binom Publication, Moscow (2010). (in Russian)
5. Popov, P.E., Kalinkin, A.A.: The method of separation of variables in a problem with a saddle - point. Russian J. Numer. Anal. Math. Model. **23**(1), 97–106 (2008). https://doi.org/10.1515/rnam.2008.007
6. Mardal, K., Winther, R.: Preconditioning discretizations of systems of partial differential equations. Numer. Linear. Algebra Appl. **18**, 1–40 (2011). https://doi.org/10.1002/nla.716
7. Arioli, M., Manzini, M.: A network programming approach in solving Darcy's equations by mixed finite elements methods. Electron. Trans. Numer. Anal. **22**, 41–70 (2006)
8. Golub, G.H., Grief, C.: On solving block-structured indefinite linear systems. SIAM J. Sci. Comput. **24**, 2076–2092 (2003). https://doi.org/10.1137/S1064827500375096
9. Arioli, M.: Generalized Golub-Kahan bidiagonalization and stopping criteria. SIAM J. Matrix Anal. App. **34**, 571–592 (2013). https://doi.org/10.1137/120866543
10. Greif, C., Schotzau, D.: Preconditioners for saddle point linear systems with highly singular (1.1) blocks. Electron. Trans. Numer. Anal. **22**, 114–121 (2006). (Special Volume on Saddle Point Problems)
11. Arioli, M., Orban, D.: Iterative methods for symmetric quasi-definite linear systems Part I: Theory. Science & Tehnology, Facilities Council, RAL, Harwell Oxford, Technical report RAL-TR-2013-003 (2013)
12. Gould, N., Orban, D., Rees, T.: Projected Krylov methods for saddle-point systems. SIAM J. Matrix Anal. App. **35**(4), 329–343 (2014). https://doi.org/10.1137/130916394

13. Il'in, V.P.: Mathematical Modeling. Part I. Continuous and Discrete Models., SBRAS Publ., Novosibirsk (2017). (in Russian)
14. Greif, C., Wathen, M.: Conjugate gradient for nonsingular saddle-point systems with a maximally rank-deficient leading block. J. Comput. Appl. Math. **358**, 1–11 (2019). https://doi.org/10.1016/j.cam.2019.02.016
15. Estrin, R., Winther, R.: Preconditioning descretization of system of partial differential equations. Nuner. Linear. Algebra App. **18**, 1–40 (2011). https://doi.org/10.1002/nla.716
16. Estrin, R., Greif, C.: SPMR: a family of saddle-point minimum residual solvers. SIAM J. Sci. Comput. **40**(3), 1884–1914 (2018). https://doi.org/10.1137/16M1102410
17. Il'in, V.P., Kazantcev, G.Y.: Iterative solution of saddle-point systems of linear equations. J. Math. Sci. **249**(2), 199–208 (2020). https://doi.org/10.1007/s10958-020-04934-7
18. Notay, Y.: Convergence of some iterative methods for symmetric saddle point linear systems. SIAM J. Matrix Anal. Appl. **40**(1), 122–146 (2019). https://doi.org/10.1137/18M1208836
19. Il'in, V.P.: Two-level iterative methods for solving the saddle point problems. MSR 2020 J. Phys. Conf. Ser. **1715**, 012004 (2021). https://doi.org/10.1088/1742-6596/1715/1/012004
20. Ivanov, M.I., Kremer, I.A., Laevsky Yu.M.: On the streamline upwind scheme of solution to the filtration problem. Siberian Electron. Math. Rep. **16**, 757–776 (2019). (in Russian) https://doi.org/10.33048/semi.2019.16.051
21. Dongarra, J., et al.: Applied Mathematics Research for Exascale Computing. Technical report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA (2014)
22. Kruse, C., Sosonkina, M., Arioli, M., Tardieu, N., Rude, U.: Parallel solution of saddle point systems with nested iterative solvers based on the Golub-Kahan bidiagonalization. Concurr. Comput. Pract. Exp. (2020). https://doi.org/10.1002/cpe.5914
23. Il'in, V.P.: Problems of parallel solution of large systems of linear algebraic equations. J. Math. Sci. **216**(6), 795–804 (2016). https://doi.org/10.1007/S10958-016-2945-4
24. Il'in, V.: Parallel intelligent computing in algebraic problems. In: Sokolinsky, L., Zymbler, M. (eds.) PCT 2021. CCIS, vol. 1437, pp. 108–117. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81691-9_8
25. Il'in, V.P.: Iterative preconditioned methods in Krylov spaces: trends of the 21st century. Comput. Math. Math. Phys. **61**(11), 1750–1775 (2021). https://doi.org/10.1134/S0965542521110099