DOI 10.1007/s10958-020-04934-7
Journal of Mathematical Sciences, Vol. 249, No. 2, August, 2020

ITERATIVE SOLUTION OF SADDLE-POINT SYSTEMS OF LINEAR EQUATIONS

V. P. Il'in* **and G. Y. Kazantcev†** UDC 519.6

The paper considers preconditioned iterative methods in Krylov subspaces for solving systems of linear algebraic equations (SLAEs) with a saddle point arising from grid approximations of three-dimensional boundary-value problems of various types describing filtration flows of a two-phase incompressible fluid. A comparative analysis of up-to-date approaches to block preconditioning of SLAEs under consideration, including issues of scalable parallelization of algorithms on multiprocessor computing systems with distributed and hierarchical shared memory using hybrid programming tools, is presented. A regularized Uzawa algorithm using a two-level iterative process is proposed. Results of numerical experiments for the Dirichlet and Neumann model boundary-value problems are provided and discussed. Bibliography: 15 *titles.*

1. Introduction

The paper considers the problem of solving a saddle-point system of linear algebraic equations (SLAE) of the form

$$Ku \equiv \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} u_a \\ u_c \end{bmatrix} = \begin{bmatrix} f_a \\ f_c \end{bmatrix} \equiv f, \tag{1}$$

where A and C are a symmetric positive definite (s.p.d.) and a symmetric positive semidefinite square matrices of orders N_1 and N_2, respectively; $K \in \mathcal{R}^{N,N}$, $N = N_1 + N_2$. The matrix K is nonsingular if C is positive definite on the kernel of B^T, and it is singular if C and B^T share a common null space. These issues are important for many electrophysical, hydro-gas-dynamic, and other applications, especially, those arising in approximating multidimensional mixed formulations for boundary-value problems by finite difference, finite volume, finite element methods, and discontinuous Galerkin algorithms [1]. In particular, of special interest are the so-called saddle-point systems of the form (1) with $C = 0$, originating from optimization problems.

More specifically, we consider SLAEs of the form (1) that arise from a finite element approximation of a two-dimensional or a three-dimensional Neumann problem for the two-phase filtration in a mixed formulation using the Raviart–Thomas basis functions on a regular rectangular grid [2]. In this case, the matrix A is tridiagonal, and the matrix B^T has one-dimensional kernel spanned by the vector $e = \{1\}$, consisting of unit components. The matrix C can be zero or have a special form (see below). If a SLAE of the form (1) is singular, then the vector of the right-hand side f is assumed to be orthogonal to the kernel for the algebraic system to be consistent.

Methods for solving the saddle-point SLAEs form a separate part of computational algebra and have a fairly long history. An analysis of the literature concerning these issues can be found in the survey [3] and in the monograph [4], as well as in more recent publications [5–8]. The main approaches consist in constructing an easily invertible preconditioning matrix M, providing for a significant reduction of the condition number of the product $M^{-1}K$ as compared

*Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk State University, e-mail: ilin@sscc.ru.

†Novosibirsk State University, e-mail: Kazancev.grigorij@gmail.com.

Translated from *Zapiski Nauchnykh Seminarov POMI*, Vol. 482, 2019, pp. 135–150. Original article submitted October 14, 2019.

to the original matrix K, and also in using an iterative process in the Krylov subspaces

$$\mathcal{K}_{n+1}(K, M, r^0) = \mathrm{Span}\{r^0, M^{-1}Kr^0, \ldots, (M^{-1}K)^n r^0\}, \quad r^0 = f - Ku^0.$$

When implementing such algorithms on heterogeneous multiprocessor computing systems (MCS) with cluster-type architecture, it is of great practical importance to achieve a high performance of computational experiments, including scalable parallelization of methods and their mapping onto computer platforms with distributed and hierarchical shared memory. To this end, in general, hybrid programming tools are used for transmitting inter-node messages and multi-threaded computations on multi-core CPU devices (MPI and OpenMP systems, respectively), as well as vectorization of operations based on a system of instructions such as AVX, and application of ultrafast graphics accelerators (GPGPU or Intel Phi). The main purpose is to reduce losses of communication operations, which are not only slow, but also highly energy-consuming. In this connection, data structures for large sparse matrices (of orders up to 10^{10} and more) that arise in real-life applications are of importance. These and other technological problems have been investigated in numerous papers considering software for problems of numerical linear algebra (see the surveys [9, 10]).

This paper presents a comparative experimental investigation of some algorithms for solving three-dimensional filtration problems with model initial data. In Sec. 2, we consider a number of modern preconditioned block methods for solving saddle-point problems. Section 3 deals with algorithmic features of solution of the boundary-value problems in question, including issues of parallel efficiency of iterative methods. Section 4 contains results of numerical experiments, demonstrating the efficiency of the methods proposed. In conclusion, some practical issues and prospects of improving the performance of algorithms for solving saddle-point problems and also directions for further research are discussed.

2. Block preconditioned methods in Krylov subspaces

Using the Schur complement

$$S = C + BA^{-1}B^T, \tag{2}$$

the coefficient matrix of system (1) is factorized as

$$K = \begin{bmatrix} A & 0 \\ B & -S \end{bmatrix} \begin{bmatrix} I & A^{-1}B^{-1} \\ 0 & I \end{bmatrix}. \tag{3}$$

If we replace the matrices A and S in (3) by their approximations (preconditioners) M_a and M_s, then we obtain a preconditioner for the matrix K in the form of its incomplete block factorization

$$M_1 = \begin{bmatrix} M_a & 0 \\ B & -M_s \end{bmatrix} \begin{bmatrix} I & M_a^{-1}B^T \\ 0 & I \end{bmatrix}. \tag{4}$$

The less accurate approximation of K using only the left or right factor from (4) yields the incomplete block triangular preconditioner

$$M_2 = \begin{bmatrix} M_a & B^T \\ 0 & -M_s \end{bmatrix} \tag{5}$$

or the incomplete Uzawa preconditioner

$$M_3 = \begin{bmatrix} M_a & 0 \\ B & -M_s \end{bmatrix}. \tag{6}$$

Every step of the corresponding iterative processes can be represented as a sequence of stages, at which only one block component of the solution sought is recomputed (for this reason, these methods are sometimes referred to as segregation techniques). In a somewhat more general

form, such a stationary algorithm (to this point, without Krylov acceleration) is represented by the following three stages (see [5]):

$$\begin{aligned} \widehat{u}_a^{n+1} &= u_a^n + Q_a^{(1)}(f_a - Au_a^n - B^T u_c^n), \\ u_c^{n+1} &= u_c^n - M_s^{-1}(f_c - B\widehat{u}_a^{n+1} + Cu_c^n), \\ u_a^{n+1} &= \widehat{u}_a^{n+1} + Q_a^{(2)}(f_a - A\widehat{u}_a^{n+1} - B^T u_c^{n+1}). \end{aligned} \tag{7}$$

Here, $Q_a^{(1)}$ and $Q_a^{(2)}$ are some approximate inverses or generalized inverses of the above preconditioner M_a. In particular, if $Q_a^{(1)} = M_a^{-1}$, $Q_a^{(2)} = 0$ or $Q_a^{(1)} = 0$, $Q_a^{(2)} = M_a^{-1}$, then from (7) we obtain either the Uzawa algorithm with the preconditioner M_3 from (6) (the third stage being omitted, i.e., $u_a^{n+1} = \widehat{u}_a^{n+1}$), or the incomplete block triangular preconditioner M_2 from (5) (the first stage in (7) being omitted, i.e., $u_a^{n+1} = u_a^n$).

If the matrix $Q_a = Q_a^{(1)} + Q_a^{(2)} - Q_a^{(2)} A Q_a^{(1)}$ is nonsingular, then to the iterative process (7) there corresponds the preconditioner

$$M = \begin{bmatrix} I & 0 \\ BQ_a^{(1)} & I \end{bmatrix} \begin{bmatrix} Q_a^{-1} & 0 \\ 0 & -M_s \end{bmatrix} \begin{bmatrix} I & Q_a^{(2)} B^T \\ 0 & I \end{bmatrix}. \tag{8}$$

In the special case where $Q_a^{(1)} = Q_a^{(2)} = M_a^{-1}$, from (8) we obtain the so-called symmetrized incomplete Uzawa method with the preconditioner

$$M = \begin{bmatrix} I & 0 \\ BM_a^{-1} & I \end{bmatrix} \begin{bmatrix} M_a(2M_a - A)^{-1} M_a & 0 \\ 0 & -M_s \end{bmatrix} \begin{bmatrix} I & M_a^{-1} B^T \\ 0 & I \end{bmatrix}. \tag{9}$$

Now consider the case where the matrices B^T and C share a common null space. Let $V \in \mathcal{R}^{N_2, N_k}$, $N_k \le N_2$, be a rectangular matrix whose N_k columns form an orthogonal basis of the kernel $\mathcal{N}(B^T)$. In this case, for Eqs. (1) to be compatible, the subvector f_c must be orthogonal to the kernel in question (i.e., $V^T f_c = 0$), and the matrix C is positive definite on the set $\mathcal{N}(B^T) \backslash \mathcal{R}(V)$ only.

In such a situation, we consider an iterative process of the form (7) in which the matrices C and M_s^{-1} are replaced by

$$\bar{C} = C + VV^T \quad \text{and} \quad \bar{M}_s^{-1} = (I - VV^T) + VV^T, \tag{10}$$

respectively, and the initial guesses for the resulting successive approximations $\bar{u}_a^n$, $\bar{u}_c^n$ are chosen as follows:

$$\bar{u}_a^0 = u_a^0, \quad \bar{u}_c^0 = (I - VV^T) u_c^0. \tag{11}$$

The resulting iterative method is said to be regularized. The matrix

$$\bar{S} = \bar{C} + BA^{-1}B^T = S + VV^T \tag{12}$$

is symmetric positive definite, $\bar{C}$ is an s.p.d. matrix on the null space of B^T, and the matrix $\bar{M}_s$ is s.p.d. simultaneously with M_s.

Using the symmetry of the matrix C and the relations

$$CV = 0, \quad V^T C = 0, \quad B^T V = 0, \quad V^T B = 0,$$

it is nondifficult to prove, by induction, that for all n,

$$\bar{u}_a^n = u_a^n, \quad \bar{u}_c^n = (I - VV^T) \bar{u}_a^n.$$

Thus, in the exact arithmetic, the original iterative algorithm (7) and its regularized version converge (or diverge) simultaneously.

A wide class of iterative algorithms for solving saddle-point SLAEs is obtained by using block diagonal preconditioners. In [7], in particular, it is shown that if a preconditioner of the form

$$M = \begin{bmatrix} A + B^T L^{-1} B & 0 \\ 0 & L \end{bmatrix} \tag{13}$$

is used, where $L \in \mathcal{R}^{N_2,N_2}$ is an arbitrary nonsingular matrix, then the spectrum of the product $M^{-1}K$ consists of only two distinct values $\lambda_+ = 1$ and $\lambda_- = -1$, where the first one is of multiplicity N_1, and the second one is of multiplicity N_2.

Note also that for solving a SLAE of the form (1) with $C = 0$ one can propose a method, which is sufficiently close (in algorithmic terms) to using a preconditioner of the form (13). This method consists in constructing a linear combination of the block rows of system (1),

$$\begin{bmatrix} A + B^T L^{-1} B & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u_a \\ u_c \end{bmatrix} = \begin{bmatrix} f_a + B^T L^{-1} f_c \\ f_c \end{bmatrix}, \tag{14}$$

where L, as in (13), is an arbitrary s.p.d. matrix. In this way, we obtain the following algorithm, which will be called the regularized Uzawa method:

$$\begin{aligned} \bar{K} u_c &\equiv B \bar{A}^{-1} B^T u_c = B \bar{A}^{-1} (f_0 + B^T L^{-1} f_c) - f_c, \\ \bar{A} &= A + B^T L^{-1} B. \end{aligned} \tag{15}$$

Formulas (14) and (15) cover a wide range of different algorithms corresponding to different choices of the matrix L. This approach is sometimes referred to as the gradient regularization because the matrix B^T is usually obtained from an approximation of the gradient operator, see (25) below.

In the case where both the initial matrix K and the preconditioner M are symmetric, the iterative conjugate direction method for solving Eqs. (1) is as follows [11]:

$$\begin{aligned} & r^0 = f - K u^0, && p^0 = M^{-1} r^0; \\ n = 0, 1, \cdots : & \\ & u^{n+1} = u^n + \alpha_n p^n, && \alpha_n = \sigma_n / \rho_n, \quad \sigma_n = (K^\gamma M^{-1} r^n, r^n), \\ & r^{n+1} = r^n - \alpha_n K p^n, && \rho_n = (K p^n, K^\gamma p^n), \\ & p^{n+1} = M^{-1} r^{n+1} + \beta_n p^n, && \beta_n = \sigma_{n+1} / \sigma_n. \end{aligned} \tag{16}$$

Here, $\gamma = 0$ and $\gamma = 1$ correspond to the conjugate gradient and conjugate residual methods, respectively, which minimize the functional

$$\Phi_n^{(\gamma)} = (K^{\gamma-1} r^n, r^n)$$

in the Krylov subspaces

$$\mathcal{K}_{n+1}(K, M, r^0) = \mathrm{Span}\{p^0, M^{-1} K p^0, \ldots, (M^{-1} K)^n p^0\}.$$

3. Implementation of the algorithms on a rectangular grid

First consider, in more detail, the matrix structures and some features of implementation of the algorithms in the case where C = 0 in (1) and a rectangular computational domain is used. If one uses a uniform grid with step sizes h_x, h_y,

$$x_i = x_0 + i h_x, \quad i = 1, 2, \ldots, N_x, \quad y_j = y_0 + j h_y, \quad j = 1, 2, \ldots, N_y, \tag{17}$$

and lower-order Raviart–Thomas basis functions, then A is a block diagonal matrix of the form

$$A = \begin{bmatrix} A_x & 0 \\ 0 & A_y \end{bmatrix}, \quad A_x = \text{block-diag}\{A_j^x\}, \quad A_y = \text{block-diag}\{A_i^y\}.$$

Here, N_x and N_y are the numbers of grid cells along the corresponding coordinates, i.e.,

$$A_x \in \mathcal{R}^{\widehat{N}_1,\widehat{N}_1}, \quad \widehat{N}_1 = (N_x - 1)N_y, \quad \widehat{A}_y \in \mathcal{R}^{\widehat{N}_2,\widehat{N}_2}, \quad \widehat{N}_2 = N_x(N_y - 1),$$

where A_j^x and A_i^y are strictly diagonally dominant tridiagonal matrices of orders $N_x - 1$ and $N_y - 1$, respectively (in this version, the matrix dimensions are indicated for the Dirichlet problem). In the simplest model case, their nonzero entries, up to symmetric diagonal scaling, are of the form $\{1, 4, 1\}$. The matrix B^T is bidiagonal and can be represented as

$$B^T = \begin{bmatrix} B_x^T \\ B_y^T \end{bmatrix}, \quad B_x^T \in \mathcal{R}^{\widehat{N}_1,N}, \quad B_y^T \in \mathcal{R}^{\widehat{N}_2,N}, \quad N = N_x N_y,$$

where, up to scaling, the nonzero entries of $B_x = \{B_i^x\}$ and $B_y = \{B_j^y\}$ are equal to $\{-1, 1\}$. The components of the subvectors u_a and u_c in the algebraic system (1) can be correlated with the edge and volume types of grid points, which are indicated in Fig. 1 by "$\times$" and "$\circ$," respectively. Physically, they correspond to the substance flows and pressure. In Fig. 1, we also indicate the links among nodes of different types.

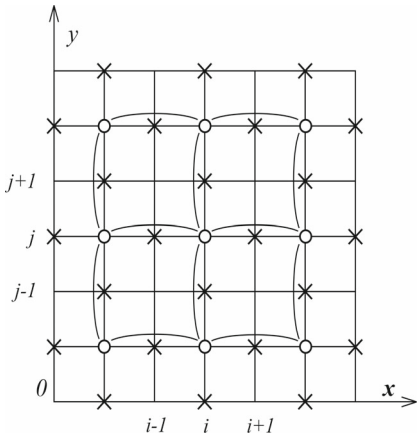

Fig. 1. The structure of internode links for a two-dimensional rectangular grid.

In the three-dimensional case, for a regular grid with parallelepiped finite elements the matrices A and B have the following forms:

$$A = \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix}, \quad B^T = \begin{bmatrix} B_x^T \\ B_y^T \\ B_z^T \end{bmatrix}, \quad A_z = \text{block-diag}\{A_k^z\}. \tag{18}$$

Here, it is natural to assume that the definition of the two-dimensional grid (17) is supplemented by discretizing the third coordinate, $z_k = z_0 + kh_z$, $k = 1, 2, \ldots, N_z$.

By eliminating the subvector

$$u_a = A^{-1}(f_a - B^T u_c)$$

from the first block row of system (1), one can reduce the Uzawa algorithm to solving the reduced system of order N_1

$$Su_c = f, \quad S = BA^{-1}B^T \in \mathcal{R}^{N_1,N_1}, \quad f = BA^{-1}f_a - f_c.$$

If the matrix A is of the form (18) and the diagonal blocks A_i^x, A_j^y, A_k^z are tridiagonal, then the implementation of this method based on the formulas of the conjugate direction method (16) (in this case it is necessary to replace A with S and set $M = I$) is trivially carried out

by solving systems with bidiagonal matrices. However, if we pass to the regularized Uzawa method (17), then the multiplication by the matrix $\bar{A}^{-1}$ (even in the simplest case where the matrix L is diagonal) implies the necessity of developing two-level iterative processes. The same also concerns algorithms using preconditioners of the form (13).

In the case of a singular system (15), we consider an iterative algorithm using a deflated version of the conjugate gradient method. For generality, let $V \in \mathcal{R}^{N,m}$ be a rectangular matrix whose columns form a basis of a certain m-dimensional deflation space. In the simplest case where $m = 1$, the vector $V = e$ spans the one-dimensional null space of the matrix B^T.

At the first step of the deflation algorithm, an initial guess u^0 is chosen from some orthogonality conditions. For an arbitrary vector u_c^{-1}, set

$$\begin{aligned} u_c^0 &= u_c^{-1} + V\bar{c}, \qquad \bar{c} \in \mathcal{R}^m, \\ r^0 &= r^{-1} - AV\bar{c}, \quad r^{-1} = f - Au^{-1}. \end{aligned} \tag{19}$$

The coefficient vector $\bar{c}$ in (19) will be determined from the overdetermined system of linear algebraic equations

$$AV\bar{c} = r^{-1}, \tag{20}$$

which is formally obtained from (19) by setting $r^0 = 0$. Upon multiplying both sides of Eq. (20) by the transposed matrix V^T, we obtain

$$\bar{c} = (V^T AV)^+ V^T r^{-1}. \tag{21}$$

Here, $(V^T AV)^+ \in \mathcal{R}^{m,m}$ is the generalized inverse matrix, which can be computed, for example, by using the Greville formula [12] or by applying the QR-decomposition (if the matrix A is nonsingular, then $(V^T AV)^+ = (V^T AV)^{-1}$). The initial direction vector p^0 is determined by the formula

$$p^0 = r^0 - V(V^T AV)^+ V^T Ar^0 = B_d r^0, \quad \text{where} \quad B_d = I - V(V^T AV)^+ V^T A. \tag{22}$$

In this case, we obtain the following deflated orthogonality relations:

$$\begin{aligned} V^T r^0 = 0, \quad V^T Ap^0 = 0, \\ V^T AB_d = 0, \qquad B_d V = 0, \end{aligned} \tag{23}$$

where B_d is a deflated preconditioning matrix.

In order to ensure that the subsequent approximations possess necessary orthogonality properties, the iterative process is carried out in accordance with the following formulas of the deflated conjugate gradient method, see [13]:

$$\begin{aligned} u^{n+1} &= u^n + \alpha_n p^n, & \alpha_n &= \sigma_n/\rho_n, \\ r^{n+1} &= r^n - \alpha_n Ap^n, & \rho_n &= (Ap^n, p^n), \\ p^{n+1} &= B_d^{-1} r^{n+1} + \beta_n p^n, & \beta_n &= \sigma_{n+1}/\sigma_n, \quad \sigma_n = (B_d^{-1} r^n, r^n). \end{aligned} \tag{24}$$

In this case, the following conditions are fulfilled:

$$V^T r^n = 0, \quad V^T Ap^n = 0.$$

Note that if two preconditioners M and B_d from (16) and (24) are used, then one must pass to the methodology of multi-preconditioned iterative processes in Krylov subspaces (see [14]).

Scalable parallelization of the iterative processes in question on a multiprocessor computing system (MCS) of cluster architecture with distributed and hierarchical shared memory is carried out by applying hybrid programming. The latter includes tools for transmitting interprocessor messages (MPI system), multi-thread computations on multi-core CPUs using the OpenMP system, and vectorization of operations using an AVX type instruction system.

For simplicity, the usage of graphics accelerators (such as GPGPU or INTEL Phi) is not considered.

Quantitatively, the performance of parallel computing on p arithmetic devices, or cores when solving a problem or implementing an algorithm A is characterized by the two coefficients: the speed up and the efficiency of a processor used:

$$\mathcal{S}_p(A) = T_1(A)/T_p(A), \quad E_p(A) = \mathcal{S}_p(A)/p.$$

Here, $T_p(A)$ is the operating time of an MCS with p processors, which includes the times for performing arithmetic operations and data transmission, i.e.,

$$T_p(A) = T_p^a(A) + T_p^c(A).$$

Roughly, the latter times for $p = 1$ can be evaluated by the formulas

$$T_1^a(A) = \tau_a Q_a, \quad T_1^c = \tau_0 + \tau_c Q_c,$$

where τ_a is the average time of performing an arithmetic operation; τ_0 is the delay time (set up) of one interprocessor communication; τ_c is the time of transmission of a real number, and Q_a and Q_c are the total numbers of arithmetic and communication operations performed. Since, for the existing computing systems, $\tau_a \ll \tau_c \ll \tau_0$, from qualitative considerations it follows that in order to improve the performance of computations, one must reduce, first of all, the number of data transmissions and the total amount of data to be sent. Also it is necessary to bear in mind that the communication operations are more time- and energy-consuming than the arithmetic operations. This factor is very important from the standpoint of operating costs of supercomputers. In order to evaluate the execution time of programs more accurately, it is necessary to take into account that the values of τ_a for different arithmetic operations and the times of data exchanges among different memory levels (CPU registers, different level caches, the total RAM of a computing node) vary considerably. In view of the complexity of real architectures, numerical experiments are the main tool used in a comparative study of the performance of software implementations of different algorithms.

Technological principles of parallelization of numerical solution of large SLAEs are considered in [9]. The main tool for achieving high performance is the additive domain decomposition method [14], both in its geometric and algebraic versions. This method is based on a balanced partitioning of a grid computational domain into subdomains, on each of which an auxiliary algebraic subsystem is considered. For these subsystems, the corresponding MPI-processes (Message Passing Interface) at node clusters with shared memory are organized. Typically, the general computational scheme is the two-level block Schwarz–Jacobi method in Krylov subspaces. The algebraic subsystems in the subdomains are solved concurrently on multi-core CPUs (Central Processor Units) with shared memory, and the "inner" parallelization is effected by means of multi-thread computations (using systems such as Open MP). If one takes into account fine features of the data structure, then an additional acceleration can be achieved by optimizing exchanges between the registers of the arithmetic device and the cache memory of a lower level by using an instruction system such as AVX (Advanced Vector Extension). It should also be mentioned that a significant improvement of the performance of a software code can be achieved by using ready-made software operations from SPARSE BLAS, efficiently applied in INTEL MKL [15].

4. Numerical results

In order to illustrate the efficiency of the iterative algorithms under study, we focus on the mathematical model of filtration of a two-dimensional incompressible fluid [2], in which the

total flow velocity $\vec{w}$ and pressure p satisfy (the gravity being neglected), in a bounded domain Ω, the relations

$$\vec{w} + \varkappa \operatorname{grad} p = 0, \quad \operatorname{div} \vec{w} = 0. \tag{25}$$

Here, the coefficient $\varkappa$ characterizes the material properties of a medium, associated with its permeability, porosity and other parameters, which can be discontinuous and highly contrasting. We assume that on the boundary of the computational domain, some boundary conditions for the velocity are given (for details, see [2]), and the scalar function of pressure is determined up to a constant. Upon reducing relations (25) to the variational formulation with the lower-order Raviart–Thomas basis functions, we obtain a system of the form (1), in which, for the model initial data, the matrices A and B have the simple block form (18). The diagonal blocks A_i^x and B_j^x of these matrices vary in dependence of the type of boundary conditions. For example, for the Neumann problem these blocks are as follows:

$$A_i^x = \frac{1}{6}\begin{bmatrix} 4 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & 4 \end{bmatrix} \in \mathcal{R}^{N_x+1,N_x+1}, \quad B_j^x = \frac{1}{h}\begin{bmatrix} -1 & & 0 \\ 1 & \ddots & \\ & \ddots & -1 \\ 0 & & 1 \end{bmatrix} \in \mathcal{R}^{N_x+1,N_x}. \tag{26}$$

In the case where the Dirichlet conditions are given on one or two sides along a certain coordinate, the numbers of columns in these matrices decrease by one or two, respectively, and the value of the corresponding angular entries of the matrices A_i^x changes from 4 to 2. The matrices A_j^y, A_k^z, B_j^y, and B_k^z are defined similarly.

In the tables below, we present results of numerical solution of the Dirichlet and Neumann boundary-value problems with model initial data that have a simple analytical solution. In the cubic computational domain, the input equations (25) were approximated on cubic grids with the numbers of nodes $N_x = N_y = N_z = 16, 32, 64$. The coefficients and normalization of the equations were chosen in such a way that the coefficient matrices of the SLAEs to be solved were of the form (26). The computations were carried out in the standard double-precision arithmetic. For both outer and inner iterations, we used the stopping criterion

$$(r^n, r^n) \le \varepsilon^2 (f, f), \quad \varepsilon = 10^{-7}.$$

The computations were carried out for the regularized Uzawa method (15) with the matrix $L^{-1} = \theta I$ and different values of the parameter θ. Actually, we used the formulas of the unpreconditioned conjugate gradient method (16), in which $\gamma = 0$, $M = I$, and the matrix K was changed for $\bar{K}$ defined in (15).

In Table 1, we present numerical results for the Neumann problem with the values $\theta = 0.0, 0.1, 0.2, 0.3, 0.6$ on the grids with cell numbers $N = 16^3, 32^3$, and 64^3. In every cell of the table, the top value is the number of outer conjugate gradient iterations, and the bottom one is the total number of inner iterations.

As can be seen from Table 1, in the Uzawa method without regularization the numbers of both outer and inner iterations grow approximately linearly as the value of N_x increases. Regularization significantly reduces the computation time, however, only up to a certain value of the parameter θ (for $\theta > 0.6$ the process behaves unpredictably).

Similar data for the Dirichlet problem on the same grids are presented in Table 2.

For both the Dirichlet and Neumann problems, the numbers of iterations of the classical (nonregularized) Uzawa method grow approximately linearly as N_x increases. On all the grids, the optimal value of the parameter is $\theta \approx 0.9$. In general, the number of outer iterations for the Dirichlet problem is less than for the Neumann problem. Note that in both tables, the numbers of inner iterations in the columns corresponding to $\theta = 0$ are provided for pure

Table 1. Numerical results for the Neumann problem.

$N \setminus \theta$	0	0.1	0.2	0.3	0.6
16^3	23	5	4	4	3
	258	80	64	64	48
32^3	47	5	5	4	36
	565	160	160	128	96
64^3	100	4	4	4	3
	1228	256	256	256	192

Table 2. Numerical results for the Dirichlet problem.

$N \setminus \theta$	0	0.1	0.2	0.9
16^3	21	5	4	3
	282	80	64	47
32^3	45	5	4	3
	618	160	128	95
64^3	96	5	4	3
	1318	320	256	191

theoretic interest only because, in this case, the matrix $\bar{A}$ is tridiagonal, whence its inversion can be effected by a direct method.

5. Conclusion

This research was aimed at developing numerical algorithms for solving saddle-point SLAEs and at applying them in modeling real-life filtration processes on modern MCS. The present paper compares some promising approaches to constructing preconditioners in Krylov subspaces, discusses issues of scalable parallelization of algorithms for solving multidimensional filtration problems, and also provides preliminary experimental results for model problems. Our road map includes increasing the convergence rate of iterative methods and also rising the performance of the software developed in the KRYLOV library [10], providing an integrated environment for solving linear algebraic problems (as part of the basic modeling system BSM).

Within the integrated environment proposed, we are planning to carry out a comparative analysis and testing of different methods on real-life problems with singularities.

This work was supported by the Russian Foundation for Basic Research (project No. 19-11-00048.)

Translated by the authors.

REFERENCES

1. V. P. Il'in, *Mathematical Modeling.* Part 1. *Continuous and Discrete Models* [in Russian], SB RAS, Novosibirsk (2017).
2. M. M. Ivanov, I. A. Kremer, and Yu. M. Laevsky, "On an upflow scheme for solving filtration problem," *Sib. Electron. Mat. Izv.*, **16**, 757–776 (2019).
3. M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta Numer.*, **14**, 1–137 (2005).
4. Yu. V. Bychenkov and E. V. Chizhonkov, *Iterative Methods for Solving Saddle-Point Problems* [in Russian], BINOM, Moscow (2010).

5. Y. Notay, "Convergence of some iterative methods for symmetric saddle point linear systems," *Matrix Anal. Appl.*, SIAM, **40**, No. 1, 122–146 (2018).
6. R. Estrin and C. Greif, "SPMR: A family of saddle-point minimum residual solvers," *SIAM J. Sci. Comp.*, **40**, No. 3, 1884–1914 (2018).
7. C. Greif and M. Wathen, "Conjugate gradient for nonsingular saddle-point systems with a maximally rank-deficient leading block," *J. Comput. Appl. Math.*, **358**, 1–11 (2019).
8. P. E. Popov and A. A. Kalinkin, "The method of separation of variables in a problem with a saddle point," *Russ. J. Numer. Anal. Math. Model.*, **23**, No. 1, 97–106 (2008).
9. V. P. Il'in, "Problems of parallel solution of large systems of linear algebraic equations," *Zap. Nauchn. Semin. POMI*, **439**, No. 6, 112–127 (2015).
10. V. P. Il'in, "On an integrated computational environment for numerical algebra," *Commun. Comput. Inf. Sci.*, **1063**, 91–106 (2019).
11. V. P. Il'in, *Methods and Technologies of Finite Elements* [in Russian], ICMMG SB RAS, Novosibirsk (2007).
12. F. R. Gantmakher, *Theory of Matrices* [in Russian], Nauka, Moscow (1958).
13. Y. L. Gurieva and V. P. Il'in, "On coarse grid correction methods in Krylov subspaces," *Zap. Nauchn. Semin. POMI*, **463**, 44–57 (2017).
14. V. P. Il'in, "Multi-preconditioned domain decomposition methods in the Krylov subspaces," *LNCS*, **10187**, 95–106 (2017).
15. Intel Math Kernel Library. Reference Manual, `http://software.intel.com/sites/products/documentation /hpc/composerxe/enus/mklxe/mk-manual-win-mac/index.html]`