Parallel Algebro-Geometric Multigrid Methods for Unstructured Grids

Maxim Batalov[1(✉)], Alexey Gurin[2], and Valery Il'in[1]

[1] Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia
makcum1990@list.ru, ilin@sscc.ru

[2] Lavrentyev Institute of Hydrodynamics SB RAS, Novosibirsk, Russia
gurin_aleksei87@mail.ru

Abstract. We study iterative methods for solving large systems of linear algebraic equations (SLAEs) with sparse matrices in Krylov subspaces, applied to a three-dimensional Laplace operator problem discretized on cubic unstructured meshes. The preconditioner for the original SLAE is constructed using recursive algorithms and data structures, which generate operators for the multigrid incomplete factorization method. Here, the forward step corresponds to the traditional reduction step, while the backward step handles solution prolongation. We discuss the implementation of these recursive algorithms and data structures within the INMOST and PETSc software frameworks. Additionally, we investigate properties of node renumbering for initial unstructured meshes. Numerical results are presented for methodological applied problems with data typical of geophysical core modeling.

Keywords: Large sparse SLAE · Algebraic multigrid method · Incomplete factorization algorithm · Recursive ordering · Parallelization of algorithms · Unstructured grid

1 Introduction

Multigrid methods, which celebrated their 60th anniversary recently, have a rich developmental history. They remain the only class of algorithms for solving large systems of linear algebraic equations (SLAEs) with sparse matrices—arising from finite-difference, finite-volume, and finite-element approximations of multidimensional boundary value problems—that are asymptotically optimal in order. Specifically, their computational cost scales proportionally to the problem size as the characteristic grid step $h \to 0$. Here, we focus on node-type grid SLAEs, where each node corresponds to one component of the solution vector.

In the pioneering works by R. P. Fedorenko and N. S. Bakhvalov [1,14], the proposed approach relied on separately reducing high- and low-frequency error components. Subsequent studies [5,23,32] introduced cascade algorithms, which involved sequentially solving problems on progressively refined meshes. The most significant advances, however, emerged in algebraic multigrid methods (AMG). These methods, constructed as preconditioned iterative processes

L. Sokolinsky et al. (Eds.): PCT 2025, CCIS 2891, pp. 206–220, 2026.
https://doi.org/10.1007/978-3-032-22051-6_11

in Krylov subspaces (see [2,6–12,26,29,30,33–38] and references therein), have led to widely adopted techniques such as V-, W-, and K-cycles, smoothing procedures, coarse-grid correction, reduction, and solution prolongation. For $m > 2$, m-grid algorithms are defined recursively via two-grid applications. Implementations of these methods, such as BoomerAMG and AMGCL [13,17], are now ubiquitous in scientific computing.

In [3,4,15,22], algebro-geometric multigrid algorithms were proposed based on partitioning the initial dense mesh in three dimensions into four subsets according to their topological primitives in the sparse mesh: nodes, edges, faces, and cells. With an appropriate ordering of nodes and vector components, the SLAE matrix attains a block-tridiagonal structure, and its solution is naturally performed in Krylov subspaces. For preconditioning, either the incomplete factorization method with diagonal compensation or symmetric (or asymmetric, for asymmetric SLAEs) successive over-relaxation is used, as detailed in [19,27,31].

The software implementation of this approach comprises two distinct parts: the first is geometric-topological, involving primarily logical transformations of mesh objects, while the second is purely algebraic and consists of vector-matrix operations.

We extend this approach along three directions. First, we account for implementation specifics of multigrid algorithms on unstructured meshes, particularly for the applied problem of "digital rock" with multiscale porous media, where real data are obtained from tomographic scans of geophysical samples. Second, we investigate the parallel efficiency of the proposed iterative methods on multiprocessor computing systems. Third, we examine the technological aspects of implementation within the INMOST platform [18] using the PETSc library [28], including a comparison of our custom code with the BoomerAMG solver from the HYPRE library [16].

This paper is organized as follows. Section 2 presents the principles of mesh node classification and ordering, along with the structure of the resulting matrices, including details on the solved seven-diagonal SLAEs. Section 3 describes preconditioned iterative processes in Krylov subspaces for block-tridiagonal algebraic systems. In Sect. 4, we discuss parallelization strategies for the proposed multigrid methods. Section 5 provides an analysis of preliminary numerical experiments. Finally, the Conclusions outline generalizations of these methods to broader problem classes and future research directions.

2 Formation of Algebro-Geometric Nested Structures

For simplicity, we consider symmetric positive definite (s.p.d.) SLAEs with the following notation:

$$Au = f, \quad A = A^\top = \{a_{t,s}\} \in \mathbb{R}^{N\times N}, \quad u = \{u_t\}, \quad f = \{f_t\} \in \mathbb{R}^N. \tag{1}$$

More specifically, on a three-dimensional cubic mesh with step size h,

$$\Omega^h : x_i = ih,\ i = 1,\dots,N_x;\ y_j = jh,\ j = 1,\dots,N_y;\ z_k = kh,\ k = 1,\dots,N_z,$$

we study a seven-point stencil system in multi-index form:

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

where A is a Stieltjes matrix (see [21]). Note that all algebraic systems discussed in this paper are of nodal type, meaning each grid node corresponds to one equation and one component of the unknown vector.

To solve the SLAE (1)–(2), we employ a sequence of m nested meshes Ω^h_l, $l = 1, \ldots, m$, where each mesh Ω^h_l has N_l nodes. The meshes satisfy the inclusion relation

$$\Omega^h = \Omega^h_1 \supset \Omega^h_2 \supset \cdots \supset \Omega^h_m,$$

with the boundary Γ passing through the planes of the coarsest mesh Ω^h_m. Each finer mesh Ω^h_{l-1} $(l = 2, \ldots, m)$ is obtained from the l-level mesh Ω^h_l by bisecting its edges.

We recursively decompose the node sets and corresponding vector components into two subsets, yielding the following representations:

$$\Omega^h = \bar{\Omega}^h_1 \cup \Omega^h_2 = \bar{\Omega}^h_1 \cup \bar{\Omega}^h_2 \cup \cdots \cup \bar{\Omega}^h_{m-1} \cup \Omega^h_m,$$

$$u = u^{(1)} = \left((\bar{u}^{(1)})^\top, (u^{(2)})^\top\right)^\top = \left((\bar{u}^{(1)})^\top, (\bar{u}^{(2)})^\top, \ldots, (\bar{u}^{(m-1)})^\top, (u^{(m)})^\top\right)^\top. \tag{3}$$

Here, we refer to the subset of nodes Ω^h_{l+1} as the coarse l-level mesh and $\overline{\Omega}^h_l$ as the reduced fine l-level mesh. By ordering the vector components in (3) sequentially—first $\bar{u}^{(1)}$, then $\bar{u}^{(2)}$, and finally $u^{(m)}$—we transform the SLAE (1) into the following block form:

$$Au = A^{(1)}u^{(1)} = \begin{bmatrix} D_{1,1} & L_{1,1} \\ U_{2,1} & D_{1,2} \end{bmatrix} \begin{bmatrix} \bar{u}^{(1)} \\ u^{(2)} \end{bmatrix} = \begin{bmatrix} D_{1,1} & L_{1,2} & \cdots & L_{1,m} \\ U_{2,1} & D_{2,2} & L_{2,3} & \\ \vdots & \ddots & \ddots & \ddots \\ U_{m,1} & & U_{m,m-1} & D_{m,m} \end{bmatrix} \begin{bmatrix} \bar{u}^{(1)} \\ \bar{u}^{(2)} \\ \vdots \\ u^{(m)} \end{bmatrix} = f. \tag{4}$$

We emphasize that for an l-level mesh Ω^h_l, the subset of coarse nodes Ω^h_{l+1} can be defined arbitrarily, subject only to the natural condition $N_l > N_{l+1}$.

Next, we examine the topology-based node classification in detail. The node set Ω^h_l is partitioned into four subsets:

$$\Omega^h_l = \Omega^l_1 \cup \Omega^l_2 \cup \Omega^l_3 \cup \Omega^l_4, \tag{5}$$

based on their association with topological primitives of the coarser mesh Ω^h_{l+1}: nodes ($\otimes$), edges ($\circ$), faces ($\times$), and cells ($\bullet$). Figure 1 illustrates a mesh fragment with these elements labeled accordingly.

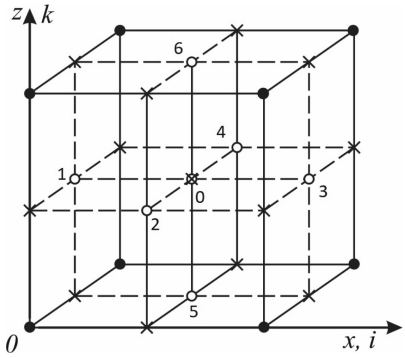

Fig. 1. Fragment of the mesh domain showing topological primitives

If we number all nodes in these subsets sequentially, along with the corresponding components of the vectors u and f, then (1) takes the following block-tridiagonal form:

$$Au = A^{(l)}u^{(l)} = \begin{bmatrix} A_{1,1}^{(l)} & A_{1,2}^{(l)} & 0 & 0 \\ A_{2,1}^{(l)} & A_{2,2}^{(l)} & A_{2,3}^{(l)} & 0 \\ 0 & A_{3,2}^{(l)} & A_{3,3}^{(l)} & A_{3,4}^{(l)} \\ 0 & 0 & A_{4,3}^{(l)} & A_{4,4}^{(l)} \end{bmatrix} \begin{bmatrix} u_1^{(l)} \\ u_2^{(l)} \\ u_3^{(l)} \\ u_4^{(l)} \end{bmatrix} = \begin{bmatrix} f_1^{(l)} \\ f_2^{(l)} \\ f_3^{(l)} \\ f_4^{(l)} \end{bmatrix}. \tag{6}$$

Here, all diagonal blocks are diagonal matrices, as evident from the nodal connectivity pattern in the SLAE (2) (Fig. 1). The matrix $A_{1,1}^{(l)}$ is hexadiagonal, while $A_{2,3}^{(l)}$ and $A_{3,2}^{(l)}$ are quadradiagonal and bidiagonal, respectively.

To construct preconditioned iterative methods for solving SLAE (6), we examine two approaches for approximate triangular decomposition of the matrix $A^{(l)}$. The first is the incomplete factorization method with diagonal compensation:

$$A^{(l)} \approx \begin{bmatrix} G_{1,1}^{(l)} & 0 & 0 & 0 \\ A_{2,1}^{(l)} & G_2^{(l)} & 0 & 0 \\ 0 & A_{3,2}^{(l)} & G_3^{(l)} & 0 \\ 0 & 0 & A_{4,3}^{(l)} & G_4^{(l)} \end{bmatrix} (G^{(l)})^{-1} \begin{bmatrix} G_1^{(l)} & A_{1,2}^{(l)} & 0 & 0 \\ 0 & G_2^{(l)} & A_{2,3}^{(l)} & 0 \\ 0 & 0 & G_3^{(l)} & A_{3,4}^{(l)} \\ 0 & 0 & 0 & G_4^{(l)} \end{bmatrix}, \tag{7}$$

$$G_1^{(l)} = A_{1,1}^{(l)},$$

$$G_2^{(l)} = A_{2,2}^{(l)} - \left(A_{2,1}^{(l)}(G_1^{(l)})^{-1}A_{1,2}^{(l)}\right)_1 - \theta_2 S_2^{(l)},$$

$$S_2^{(l)}e_2 = \left[A_{2,1}^{(l)}(G_1^{(l)})^{-1}A_{1,2}^{(l)} - \left(A_{2,1}^{(l)}(G_1^{(l)})^{-1}A_{1,2}^{(l)}\right)_1\right] e_2,$$

$$G_3^{(l)} = A_{3,3}^{(l)} - \left(A_{3,2}^{(l)}(G_2^{(l)})^{-1}A_{2,3}^{(l)}\right)_1 - \theta_3 S_3^{(l)},$$

$$S_3^{(l)}e_3 = \left[A_{3,2}^{(l)}(G_3^{(l)})^{-1}A_{2,3}^{(l)} - \left(A_{3,2}^{(l)}(G_2^{(l)})^{-1}A_{3,2}^{(l)}\right)_1\right] e_3,$$

$$G_4^{(l)} = A_{4,4}^{(l)} - A_{4,3}^{(l)}(G_3^{(l)})^{-1}A_{3,4}^{(l)}.$$

Here, $\theta \in [0,1]$ is the compensation parameter; $S_k^{(l)}$ and $G_k^{(l)}$ $(k = 1,2,3)$ are diagonal matrices; e_2 and e_3 are trial vectors of dimensions $N_2^{(l)}$ and $N_3^{(l)}$, respectively, typically with all components equal to unity. The notation $(C)_1$ denotes the diagonal part of the matrix C. The matrix $G_4^{(l)}$ is heptadiagonal and shares the same sparsity pattern as $A_{4,4}^{(l)}$. The relations (7) ensure the full compensation condition $Be = Ae$ when $\theta = 1$ and $e \in \mathbb{R}^N$.

The second matrix approximation approach applies either symmetric or asymmetric successive over-relaxation (SSOR or USSOR), depending on the properties of $A^{(l)}$. This method selects the matrix $G_k^{(l)}$ in (7) using a simple formula:

$$G_k^{(l)} = \omega^{-1} A_{k,k}^{(l)}, \tag{8}$$

where $\omega \in (0,1)$ is the relaxation parameter.

As shown below, implementing iterative processes with the preconditioning matrix $B^{(l)}$ requires solving an auxiliary SLAE of the form $B^{(l)}q = r$ at each step. This can be efficiently performed using the following formulas:

$$B^{(l)}q^n = r^n, \quad (G^{(l)} + L^{(l)})v^n = r^n, \quad (G^{(l)} + U^{(l)})q^n = G^{(l)}v^n; \tag{9}$$

$$\begin{gathered} G_1^{(l)}v_1 = r_1, \quad G_2^{(l)}v_2 = r_2 - A_{2,1}^{(l)}v_1, \quad G_3^{(l)}v_3 = r_3 - A_{3,2}^{(l)}v_2, \\ q_3 = v_3, \quad G_2^{(l)}w_2 = A_{2,3}^{(l)}q_3, \quad q_2 = v_2 - w_2, \quad q_1 = v_1 - (G_1^{(l)})^{-1}A_{1,2}^{(l)}q_2. \end{gathered} \tag{10}$$

3 Preconditioned Krylov Iterative Processes

This section briefly presents conjugate and semiconjugate direction methods for solving symmetric and nonsymmetric SLAEs using the preconditioning matrix B. All described algorithms share the common implementation of the following formulas:

$$\begin{gathered} r^0 = f - Au^0, \quad p^n = B^{-1}r^n, \quad n = 0,1,\ldots, \\ u^{n+1} = u^n + \alpha_n p^n, \quad r^{n+1} = r^n - \alpha_n Ap^n, \end{gathered} \tag{11}$$

where p^n are direction vectors, r^0 is the initial residual, while α_n and β_n are iterative parameters to be determined.

When A^γ induces orthogonality of the direction vectors,

$$(A^\gamma p^n, p^k) = \rho_k^{(\gamma)}\delta_{n,k}, \quad \rho_k^{(\gamma)} = (A^\gamma p^k, p^k) = \|p^k\|_\gamma^2, \tag{12}$$

where $\gamma = 0,1,2$ and $\delta_{n,k}$ is the Kronecker delta, relations (4) define the objective functional:

$$\Phi_\gamma(r^n) = (A^{\gamma-2}r^n, r^n) = (r^0, r^0)_{\gamma-2} - \sum_{k=0}^{n-1}\left[2\alpha_k(r^0, A^{\gamma-1}p^k) - \alpha_k^2\rho_k\right]. \tag{13}$$

Under the conditions

$$\begin{gathered} p^{n+1} = B^{-1}r^{n+1} + \beta_n p^n, \quad \alpha_n = \sigma_n/\rho_n, \\ \beta_n = \sigma_{n+1}/\sigma_n, \quad \sigma_n = (r^n, B^{-1}r^n), \quad \rho_n = (p^n, A^\gamma p^n), \end{gathered} \tag{14}$$

the functional reaches its minimum value:

$$\Phi_\gamma(r^n) = \|r^0\|^2_{\gamma-2} - \sum_{k=0}^{n-1} (r^0, p^k)^2_{\gamma-1}/\rho_k. \tag{15}$$

To satisfy the stopping criterion $\|r^n\| = (r^n, r^n)^{1/2} \le \varepsilon \|f\|$ for a given $\varepsilon \ll 1$, the required number of iterations is bounded by

$$n(\varepsilon) \le \sqrt{\kappa} \left[\log(2\varepsilon^{-1})\right]/2, \tag{16}$$

where κ is the spectral condition number of matrix $B^{-1}A$.

In this family of preconditioned conjugate direction (CD) methods, $\gamma = 0$ corresponds to the minimum error algorithm, while $\gamma = 1$ and $\gamma = 2$ yield the conjugate gradient and conjugate residual algorithms, respectively. The two-term formulas (14) define the Hestenes–Stiefel orthogonalization for the direction vectors p^n. An alternative approach uses the three-term Lanczos relations, though these are less stable against round-off errors.

For asymmetric SLAEs, semi-conjugate direction (SCD) methods with long vector recursions replace CD methods, though at a greater computational cost. We consider these algorithms in a generalized form with multi-preconditioning, where new iterative approximations are computed using a matrix P_n of direction vectors rather than a single vector p^n:

$$\begin{gathered} r^0 = f - Au^0, \quad n = 0, \ldots, \quad u^{n+1} = u^n + P_n \bar{\alpha}_n, \\ r^{n+1} = r^n - AP_n\bar{\alpha}_n = r^q - AP_q\bar{\alpha}_q - \cdots - AP_n\bar{\alpha}_n, \quad 0 \le q \le n, \\ P_n = (p^n_1, \ldots, p^n_{M_n}) \in \mathbb{R}^{N \times M_n}, \quad \bar{\alpha}_n = (\alpha_{n,1}, \ldots, \alpha_{n,M_n})^\top \in \mathbb{R}^{M_n}. \end{gathered} \tag{17}$$

Here, $\bar{\alpha}_n \in \mathbb{R}^{M_n}$ contains the iterative parameters, and the direction vectors p^n_k satisfy the semi-conjugacy condition (with $n'' \le n$ in subsequent relations):

$$\begin{gathered} (Ap^n_k, A^\gamma p^{n'}_{k'}) = \rho^{(\gamma)}_{n,k} \delta^{k,k'}_{n,n'}, \quad \rho^{(\gamma)}_{n,k} = (Ap^n_k, A^\gamma p^n_k), \\ \gamma = 0, 1, \quad n' = 0, 1, \ldots, n-1, \quad k, k' = 1, 2, \ldots, M_n. \end{gathered} \tag{18}$$

When the coefficients $\bar{\alpha}_n = \{\alpha_{n,l}\}$ in (18) are determined by

$$\alpha_{n,l} = \sigma_{n,l}/\rho^{(\gamma)}_{n,n}, \quad \sigma_{n,l} = (r^0, A^\gamma \bar{p}^n_l), \tag{19}$$

the residual functional

$$\Phi^{(\gamma)}_n(r^{n+1}) \equiv (r^{n+1}, A^{\gamma-1} r^{n+1}) = (r^q, A^{\gamma-1} r^q) - \sum_{k=q}^{n} \sum_{l=1}^{M_n} \frac{(r^q, A^\gamma p^k_l)^2}{\rho^{(\gamma)}_{k,l}}, \tag{20}$$

for $q = 0, 1, \ldots, n$, attains its minimum in the block Krylov subspaces:

$$\mathcal{K}_M = \mathrm{Span}\{p^0_1, \ldots, p^0_{M_0}, Ap^1_1, \ldots, Ap^1_{M_1}, \ldots, Ap^n_1, \ldots, Ap^n_{M_n}\}, \tag{21}$$

where $M = M_0 + M_1 + \cdots + M_n$.

The orthogonal properties of the direction vectors p_j^n are generally determined using various preconditioning matrices $B_{n,l}$, leading to

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

$$B_{n,l} \in \mathbb{R}^{N \times N}, \quad l = 1, \ldots, M_n; \quad \gamma = 0, 1, 2,$$

$$\bar{\beta}_{n,k}^{(\gamma)} = \{\beta_{n,k,l}^{(\gamma)}\} = (\beta_{n,k,1}^{(\gamma)} \ldots \beta_{n,k,M_n}^{(\gamma)})^\top \in \mathbb{R}^{M_n}, \tag{22}$$

$$\beta_{n,k,l}^{(\gamma)} = -\frac{(A^\gamma p_l^k, A B_{n+1,l}^{-1} r^{n+1})}{\rho_{n,l}^{(\gamma)}}, \quad n = 0, 1, \ldots, \quad k = 0, \ldots, n, \quad l = 1, \ldots, M_n.$$

Note that specific variants of these methods for asymmetric SLAEs exhibit convergence rates equivalent to GMRES-type algorithms based on Arnoldi orthogonalization, including those with dynamic or flexible preconditioning (FGMRES). For details, see [24,31].

4 Performance Issues of Algebro-Geometric Multigrid Methods

Based on preliminary studies of mathematical efficiency, iteration counts, and computational complexity estimates for algebraic multigrid methods, we identify two primary variants whose numerical results are presented in the next section. For conciseness, we focus mainly on symmetric SLAEs.

Algorithm 1. Conjugate Gradient method with m-recursive Incomplete Factorization (CGRIF(m)). This algorithm operates on a sequence of m nested meshes. The preconditioning matrix $B^{(1)}$ has its lower diagonal block $G_4^{(1)} = A^{(2)}$ approximately factorized. The process is recursive: $A^{(2)} \approx B^{(2)}$, $G_4^{(2)} = A^{(3)} \approx B^{(3)}$, etc. On the coarsest grid ($l = m$), the block $G_4^{(m)}$ is factorized exactly using LU decomposition, resulting in a single-level preconditioned iterative process.

Algorithm 2. Recursive Preconditioned Conjugate Gradient method with K-cycles (RPCG-KIF(m)). This multilevel iterative process applies the preconditioned CG method recursively. For each auxiliary SLAE $G_4^{(l)} v_4^{(l)} = \psi^{(l)}$, several CG iterations are performed, except on the coarsest grid ($l = m$), where the system is solved exactly. This procedure resembles the K-cycles proposed by I. Notay and can be interpreted as preliminary smoothing during the SLAE reduction stage. The second stage of incomplete factorization (prolongation) involves several unpreconditioned Krylov iterations when transitioning between grid levels, also referred to as K-cycle smoothing.

For Algorithm 1, the convergence analysis is straightforward when A is a Stieltjes matrix. In this case, the preconditioner B remains symmetric positive definite, and the iteration count estimate follows from (16). Algorithm 2, however, requires empirical analysis because its dynamic preconditioning produces

an asymmetric matrix B, necessitating Krylov methods with long recurrences (e.g., FGMRES or FCG—flexible generalized minimal residual or conjugate gradient methods) instead of standard PCG at the top iteration level.

Let us now examine parallel performance characteristics of these iterative processes on multiprocessor systems. The key metrics are parallel speedup and computational efficiency:

$$S_p = \frac{T_1(A)}{T_p(A)}, \quad E_p = \frac{S_p}{p}, \tag{23}$$

where $T_p(A)$ represents the solution time for problem (A) using p processors.

The preconditioned conjugate gradient methods primarily involve vector-matrix operations, suggesting near-linear speedup potential for large-scale SLAEs (10^7–10^{10} unknowns) running on hundreds to thousands of processors. However, a practical challenge emerges when handling real-world problems on unstructured grids: matrices stored in compressed sparse formats complicate memory access patterns.

Also, library constraints affect implementation flexibility. Our implementation uses the INMOST platform with PETSc and HYPRE libraries, which impose specific requirements on data structures and inter-module interfaces. While PETSc's extensive functionality handles core operations (grid management, variable renumbering, block-tridiagonal matrix assembly), we employ its black-box implementations for CG and GMRES methods. These library constraints can sometimes limit methodological extensions for novel applications.

5 Numerical Experiments

We present preliminary experimental results comparing the efficiency and performance of our developed methods with the well-known BoomerAMG algorithm from the HYPRE library. The analysis considers model SLAEs for the Laplace equation in a cubic domain with $N = 64^3$, 128^3, and 256^3 nodes. Boundary conditions were set as $u = 1, 2, 3$ on opposite cube faces, with initial approximation $u^0 = 0$ and stopping criterion $\epsilon = 10^{-7}$ for the relative residual. All computations were performed on Yandex Cloud [39] using Intel Ice Lake vCPU (64 cores) with 128 GB RAM.

Tables 1, 2 and 3 compare Algorithm 2 (with varying numbers of embedded meshes m) against BoomerAMG (which automatically determines its mesh count). We focus on an experimentally optimized configuration performing two PCG iterations per grid level during reduction and one GMRES iteration during prolongation. This K-cycle variant is denoted $K_{2,1}$.

The tables present:

- iteration counts (n);
- iteration time (t_1, seconds);
- LU decomposition time on coarsest grid (t_2, seconds);

– SLAE preparation time (t_3, seconds);
– total runtime ($t = t_1 + t_2 + t_3$, seconds).

Right columns (labeled H) show corresponding BoomerAMG results. Table 4 specifically compares results for $N = 128^3$ across the following variants:

– $K_{2,0}$: two PCG iterations per grid level (reduction) and no prolongation cycles;
– $K_{0,1}$: no PCG iterations (reduction) and one GMRES iteration (prolongation);
– $K_{0,0}$: essentially Algorithm 1 without K-cycles.

Table 1. Performance of Algorithm 2 ($K_{2,1}$ cycles) versus BoomerAMG on a 64^3 grid. Times are in seconds.

64^3	Algorithm 2 ($K_{2,1}$)						H
m	2	3	4	5	6	7	—
n	16	17	17	17	17	17	10
t_1	0.341	0.263	0.256	0.249	0.276	0.336	0.403
t_2	0.501	0.490	0.001	0	0	0	0
t_3	0.161	0.181	0.184	0.178	0.182	0.175	0.495
t	1.003	0.934	0.441	0.427	0.458	0.511	0.898

Tables 5 and 6 present the runtime performance of the algorithms for grids of size $N = 64^3$ and 128^3, respectively. The computations were performed using $p = 1, 2, 3, 4, 6, 8, 12, 16, 24, 32$ processors with a two-grid version of the multigrid (MG) algorithm. The reported times are as follows:

– t_1: PCG iteration time;
– t_2: LU decomposition time;

Table 2. Performance of Algorithm 2 ($K_{2,1}$ cycles) versus BoomerAMG (H) on a 128^3 grid. Times are in seconds.

128^3	Algorithm 2 ($K_{2,1}$)							H
m	2	3	4	5	6	7	8	—
n	15	16	17	17	17	17	17	10
t_1	3.856	1.581	1.801	1.701	1.727	2.086	2.188	4.066
t_2	16.935	1.170	0.117	0.002	0.001	0	0	0
t_3	1.993	1.513	1.398	1.374	1.366	1.573	1.578	4.903
t	22.784	4.264	3.316	3.077	3.094	3.659	3.766	8.969

Table 3. Performance of Algorithm 2 ($K_{2,1}$ cycles) versus BoomerAMG on a 256^3 grid. Times are in seconds.

256^3	Algorithm 2 ($K_{2,1}$)								H
m	2	3	4	5	6	7	8	9	—
n	15	16	16	17	17	17	17	17	10
t_1	37.296	14.703	16.414	18.140	17.747	17.263	16.590	18.263	33.887
t_2	879.616	29.089	2.526	0.370	0.001	0.001	0.001	0	0
t_3	12.366	13.856	14.146	14.022	13.788	13.261	13.673	16.644	46.840
t	929.278	57.648	33.086	32.532	31.536	30.525	30.264	34.907	80.727

Table 4. Comparison of different K-cycle variants ($K_{0,0}$, $K_{0,1}$, $K_{2,0}$, and $K_{2,1}$) on a 128^3 grid, showing iteration counts (n) and computation times in seconds (t)

m	Number of meshes				
	3	4	5	6	7
$n(K_{0,0})$	35	74	147	221	234
$t(K_{0,0})$	6.646	6.293	10.945	15.565	16.532
$n(K_{0,1})$	32	71	144	470	211
$t(K_{0,1})$	6.888	6.854	12.459	36.170	17.429
$n(K_{2,0})$	21	25	29	31	31
$t(K_{2,0})$	4.672	3.745	3.920	4.077	4.207
$n(K_{2,1})$	16	17	17	17	17
$t(K_{2,1})$	4.264	3.316	3.077	3.094	3.659

- t_3: SLAE preparation time;
- t: total solution time ($t = t_1 + t_2 + t_3$).

Each table cell shows two values: the upper value corresponds to Algorithm 2 with $K_{2,1}$ cycles, while the lower value represents BoomerAMG results. All computations achieved good accuracy, with absolute residual norms around 10^{-5} and relative residuals approximately 10^{-7} (detailed accuracy data omitted for brevity). We observed that experimental results remain largely insensitive to changes in the initial error $\delta_0 = u - u^0$.

Figure 2 compares the parallel speedup and efficiency between the $K_{2,1}$ two-grid variant and BoomerAMG for the 128^3 grid case.

Based on the presented results, we draw the following conclusions regarding the mathematical efficiency of the studied algorithms and the performance of the INMOST-PETSc platform implementation:

- The K-cycles, serving as iterative smoothing operators across grid levels, significantly reduce the iteration count in the algebro-geometric method. This count remains nearly independent of the initial grid step h. While the two-

Table 5. Solution times (in seconds) for different processor counts (p) on a 64^3 grid. Upper rows show Algorithm 2 results, lower rows show HYPRE results.

p	Number of processors									
	1	2	3	4	6	8	12	16	24	32
t_1	0.341	0.239	0.159	0.131	0.112	0.106	0.100	0.090	0.093	0.099
	0.403	0.760	0.214	0.175	0.144	0.112	0.100	0.091	0.092	0.086
$t_2 + t_3$	0.662	0.774	0.619	0.544	0.478	0.468	0.442	0.427	0.448	1.206
	0.495	0.228	0.861	0.767	0.704	0.555	0.660	0.753	0.880	1.021
t	1.003	1.013	0.778	0.675	0.590	0.574	0.542	0.517	0.541	1.305
	0.898	0.988	1.075	0.942	0.848	0.667	0.760	0.844	0.972	1.107

Table 6. Solution times (in seconds) for different processor counts (p) on a 128^3 grid. Upper rows show Algorithm 2 results, lower rows show HYPRE results.

p	Number of processors									
	1	2	3	4	6	8	12	16	24	32
t_1	3.856	2.357	1.798	1.656	1.313	1.003	0.874	0.729	0.733	0.723
	4.066	2.199	1.999	1.153	1.505	1.135	1.110	1.038	0.873	0.785
$t_2 + t_3$	18.928	11.680	8.735	7.439	6.255	5.194	4.535	4.409	4.340	4.393
	4.903	10.272	10.158	6.439	7.247	5.711	7.062	7.293	7.218	7.551
t	22.784	14.037	10.533	9.095	7.568	6.197	5.409	5.138	5.073	5.116
	8.969	12.471	12.157	7.592	8.752	6.846	8.172	8.331	8.091	8.336

grid version achieves the minimal iteration count, the optimal SLAE solution time occurs with $m = 5$–7 grid levels.

– The developed Algorithm 2 (RPCG-KIF(m)) demonstrates approximately 50% faster computation time in single-processor mode compared to Boomer-AMG, despite requiring about 1.5× more iterations.
– Parallel performance of the RPCG-KIF(m) implementation shows limited speedup ($S_p \approx 4.5$), peaking at $p = 24$ processors. This limitation arises because parallelization is managed entirely by PETSc via MPI, with only the external interface accessible to our implementation. Notably, native parallelization of vector-matrix operations within the B preconditioner is currently unattainable on this platform.

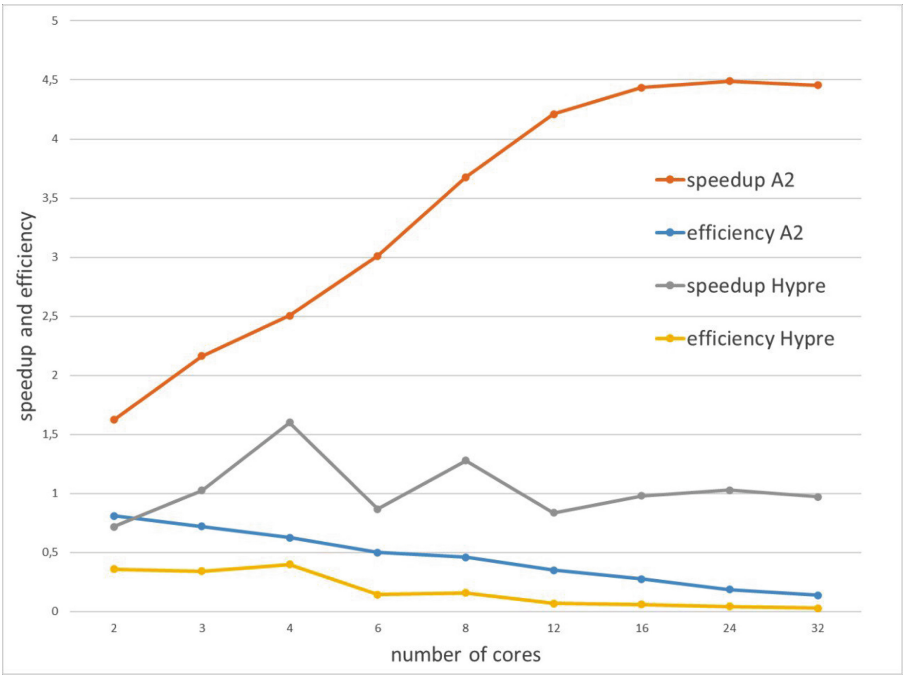

Fig. 2. Parallel performance comparison: speedup and efficiency of the $K_{2,1}$ two-grid variant versus BoomerAMG, on a 128^3 grid

6 Conclusions

The key contribution of this work is demonstrating that the algebro-geometric multigrid method, previously introduced by the authors, achieves significantly improved efficiency when enhanced with K-cycles as iterative smoothing operators during both reduction and prolongation stages. Specifically, the optimized version runs approximately twice as fast as the popular BoomerAMG solver from the HYPRE library.

Regarding parallel performance on multiprocessor systems, the current speedup results remain modest—our method shows about 1.5× greater parallel efficiency compared to BoomerAMG when increasing the processor count (p). This limitation stems primarily from implementation constraints within the INMOST-PETSc platform. The algorithm's distinctive requirement for node reordering according to topological primitives (nodes, edges, faces, and cells) of the sparse mesh presents an interesting graph problem. While theoretically solvable using only the CSR matrix format, the computational complexity makes this approach impractical. The existing mesh-aware implementation in INMOST-PETSc provides an efficient solution, though at the cost of losing direct parallelization capabilities for vector-matrix operations fundamental to Krylov subspace preconditioning.

The experimental results suggest three key directions for future development. First, enhancing convergence rates and reducing iteration counts, particularly through symmetric preconditioner design for symmetric SLAEs. Second, improving parallel implementation efficiency, where substantial gains appear achievable. Third, extending the method's applicability to broader SLAE classes, including those with complex matrix structures and block patterns arising in high-accuracy PDE systems. We note in closing that artificial intelligence approaches hold significant promise for next-generation mathematical software development, including for the problem classes and algorithms considered here, though detailed exploration of this direction falls beyond our current scope.

Acknowledgments. This work was supported by the Russian Science Foundation (Project № 24-21-00402).

References

1. Bakhvalov, N.S.: On the convergence of a relaxation method with natural constraints on the elliptic operator. USSR Comput. Math. Math. Phys. **6**(5), 101–135 (1966). https://doi.org/10.1016/0041-5553(66)90118-2. https://www.sciencedirect.com/science/article/pii/0041555366901182
2. Bank, R., et al.: Algebraic multigrid domain and range decomposition (AMG-DD/AMG-RD). SIAM J. Sci. Comput. **37**(5), 113–136 (2015). https://doi.org/10.1137/140974717
3. Batalov, M., Il'In, V.: Multigrid incomplete factorization methods in Krylov subspaces on unstructured grids. In: Sokolinsky, L., et al. (eds.) Parallel Computational Technologies. CCIS, pp. 163–176. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-73372-7_12
4. Batalov, M., et al.: On parallel multigrid methods for solving systems of linear algebraic equations. In: Sokolinsky, L., Zymbler, M., (eds.) PCT 2023. CCIS, vol. 1868, pp. 93–109. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-38864-4_7
5. Bornemann, F.A., Deuhard, P.: The cascadic multigrid method for elliptic problems. Numer. Math. **75**(2), 135–152 (1996). https://doi.org/10.1007/s002110050234
6. Brandt, A.: Algebraic multigrid theory: the symmetric case. Appl. Math. Comput. **19**(1), 23–56 (1986). https://doi.org/10.1016/0096-3003(86)90095-0. https://www.sciencedirect.com/science/article/pii/0096300386900950
7. Brandt, A., McCormick, S., Ruge, J.: Algebraic multigrid (AMG) for sparse matrix equations. In: Evans, D.J. (ed.) Sparsity and its Applications, pp. 257–284. Cambridge University Press, Cambridge (1985)
8. Brezina, M., et al.: Adaptive smoothed aggregation (ASA) multigrid. SIAM Rev. **47**(2), 317–346 (2005). https://doi.org/10.1137/050626272
9. Brezina, M., et al.: Algebraic multigrid based on element interpolation (AMGe). SIAM J. Sci. Comput. **22**(5), 1570–1592 (2001). https://doi.org/10.1137/S1064827598344303
10. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial. 2nd edn. SIAM (2000)

11. Cleary, A.J., Falgout, R.D., Henson, V.E., Jones, J.E.: Coarse-grid selection for parallel algebraic multigrid. In: Ferreira, A., Rolim, J., Simon, H., Teng, S.-H. (eds.) IRREGULAR 1998. LNCS, vol. 1457, pp. 104–115. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0018531
12. Cleary, A.J., et al.: Robustness and scalability of algebraic multigrid. SIAM J. Sci. Comput. **21**(5), 1886–1908 (2000). https://doi.org/10.1137/S1064827598339402
13. Demidov, D.: AMGCL: an efficient, flexible, and extensible algebraic multigrid implementation. Lobachevskii J. Math. **40**(5), 535–546 (2019). https://doi.org/10.1134/S1995080219050056
14. Fedorenko, R.P.: The speed of convergence of one iterative process. USSR Comput. Math. Math. Phys. **4**(3), 227–235 (1964). https://doi.org/10.1016/0041-5553(64)90253-8. https://www.sciencedirect.com/science/article/pii/0041555364902538
15. Gurieva, Y.L., Il'in, V.P., Petukhov, A.V.: On multigrid methods for solving two-dimensional boundary-value problems. J. Math. Sci. **249**(2), 118–127 (2020). https://doi.org/10.1007/s10958-020-04926-7
16. HYPRE: Scalable Linear Solvers and Multigrid Methods. https://computing.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods/
17. Henson, V.E., Yang, U.M.: Boomeramg: a parallel algebraic multigrid solver and preconditioner. Appl. Numer. Math. **41**(1), 155–177 (2002). https://doi.org/10.1016/S0168-9274(01)00115-5. https://www.sciencedirect.com/science/article/pii/S0168927401001155. Developments and Trends in Iterative Methods for Large Systems of Equations - in memorium Rudiger Weiss
18. INMOST: A Toolkit for Distributed Mathematical Modeling. https://www.inmost.org/
19. Il'in, V.P..: Finite element methods and techniques. ICM&MG SB RAS, Novosibirsk (2007). https://elibrary.ru/qjtctj
20. 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
21. Il'in, V.P.: Mathematical Modeling Part 1. Continuous and discrete models. Publisher of the Siberian Branch of the Russian Academy of Sciences, p. 428 (2017)
22. Il'in, V.P.: Multigrid incomplete factorization methods in Krylov subspaces. J. Math. Sci. **272**, 1–10 (2023). https://doi.org/10.1007/s10958-023-06446-6
23. Il'in, V.P.: One version of the multigrid method. Sib. Math. J. **26**(2), 240–244 (1985). https://doi.org/10.1007/BF00968767
24. Knizhnerman, L.: Error bounds for the arnoldi method: a set of extreme eigenpairs. Linear Algebra Appl. **296**(1), 191–211 (1999). https://doi.org/10.1016/S0024-3795(99)00122-6. https://www.sciencedirect.com/science/article/pii/S0024379599001226
25. Notay, Y.: Flexible conjugate gradients. SIAM J. Sci. Comput. **22**(4), 1444–1460 (2000). https://doi.org/10.1137/S1064827599362314
26. Notay, Y., Napov, A.: A massively parallel solver for discrete poisson-like problems. J. Comput. Phys. **281**, 237–250 (2015). https://doi.org/10.1016/j.jcp.2014.10.043. https://www.sciencedirect.com/science/article/pii/S0021999114007256
27. Olshanskii, M.A., Tyrtyshnikov, E.E.: Iterative Methods for Linear Systems Theory and Applications. SIAM (2014)
28. PETSc: Portable Extensible Toolkit for Scientific Computation. https://petsc.org/release/

29. Reusken, A.: A multigrid method based on incomplete gaussian elimination. Numer. Linear Algebra Appl. **3**(5), 369–390 (1996). https://doi.org/10.1002/(SICI)1099-1506(199609/10)3:5<369::AID-NLA89>3.0.CO;2-M. https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-1506%28199609/10%293%3A5%3C369%3A%3AAID-NLA89%3E3.0.CO%3B2-M
30. Ruge, J.W., Stüben, K.: 4. algebraic multigrid. Multigrid Methods, pp. 73–130 (1987). https://doi.org/10.1137/1.9781611971057.ch4. https://epubs.siam.org/doi/abs/10.1137/1.9781611971057.ch4
31. Saad, Y.: Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics (2003). https://doi.org/10.1137/1.9780898718003. https://epubs.siam.org/doi/abs/10.1137/1.9780898718003
32. Shaidurov, V.V.: Some estimates of the rate of convergence for the cascadic conjugate-gradient method. Comput. Math. Appl. **31**(4), 161–171 (1996). https://doi.org/10.1016/0898-1221(95)00228-6. https://www.sciencedirect.com/science/article/pii/0898122195002286. Selected Topics in Numerical Methods
33. Stüben, K.: Algebraic Multigrid (AMG): An Introduction with Applications ; Updated Version of GMD Report No 53, March 1999. GMD-Report. GMD-Forschungszentrum Informationstechnik (1999). https://books.google.ru/books?id=rzLiGwAACAAJ
34. Stüben, K.: Algebraic multigrid (AMG): experiences and comparisons. Appl. Math. Comput. **13**(3), 419–451 (1983). https://doi.org/10.1016/0096-3003(83)90023-1. https://www.sciencedirect.com/science/article/pii/0096300383900231
35. Vaněk, P.: Smoothed prolongation multigrid with rapid coarsening and massive smoothing. Appl. Math. **57**(1), 1–10 (2012). https://doi.org/10.1007/s10492-012-0001-3
36. Vaněk, P., Mandel, J., Brezina, M.: Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. Computing (Vienna/New York), **56**(3), 179–196 (1996). https://doi.org/10.1007/bf02238511
37. Vasilevsky, Y.V., Olshansky, M.A.: Quick Course on Multigrid Methods and Domain Decomposition Methods: textbook. MAKS Press (2007)
38. Jinchao, X., Zikatanov, L.: Algebraic multigrid methods. Acta Numer **26**, 591–721 (2017). https://doi.org/10.1017/S0962492917000083
39. Yandex Cloud: A full-fledged cloud platform providing scalable infrastructure, storage, machine learning and development tools to build and enhance digital services and applications. https://yandex.cloud/ru