

Implementation of the Spectral Preconditioner to Solve Poisson Equation

Vadim Lisitsa(✉), Aleksei Manaev, and Sergey Solovyev

Institute of Mathematics SB RAS, Koptug avenue 4, Novosibirsk 630090, Russia
lisitsavv@yandex.ru

Abstract. In this paper, we present an original preconditioner to solve Poisson equation for strongly heterogeneous media. We suggest using the Conjugate Gradient method with the preconditioner based on the solution of the Poisson equation for homogeneous media. Corresponding operator is easy to invert by spectral method, where spectral decomposition is applied in two spatial directions and the Gauss elimination method is applied to solve a series of 1D problems. We illustrate that use of such precondtioner strongly decreases the number of iterations to solve the original Poisson equation, moreover the number of iterations weakly depends on the problem size. Implementation of the suggested approach using modern GPUs allows solving problems of up to the size of 1000^3 voxels.

Keywords: Poisson equation · Conjugate gradient · preconditioner

1 Introduction

Numerical solution of Poisson equation in strongly heterogeneous media has numerous applications. In particular, it describes electric potential distribution in heterogeneous materials [1,9,23], steady state temperature distribution [5], and steady state species distribution due to diffusion. Also, one needs solving Poisson equation as a part of fluid flow simulation either multi-phase flows in open channels if the projection-based methods are applied [10,11] or multi-phase flows in porous media if IMPES schemes are used [13,17].

Then discretized, Poisson equation leads to the system of linear algebraic equations with self-adjoin positive definite matrix (if the original differential operator had the same properties). Such systems of equations can be efficiently solved using Krylov-type iterative methods, and in particular the Conjugate Gradient method is widely used [18]. However, the convergence rate of CG method strongly depends on the matrix condition number, which increases with the problem size and inhomogeneity contrast in Laplace operator. To speed-up the convergence different types of preconditioners were designed. The most universal preconditioners are based on purely algebraic consideration of the problem.

The research was supported by the Russian Science Foundation grant no. 19-77-20004-П.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15888, pp. 388–401, 2026.
https://doi.org/10.1007/978-3-031-97596-7_26

They include the Jacobi and incomplete LU factorization of the matrix [7,12,18], low-rank approximation of the matrix or its LU factors [22]. The other set of approaches are originated on the geometrical point of view with further evolution to pure algebraic versions, such as domain decomposition methods [15], and multigrid methods [6,20]. The third type of preconditioners are based on the approximation of the physical process; i.e., use the inverse Laplace operator corresponding to a simpler model as a preconditioner. The main issue of this approach is the necessity to invert the operator at each iteration, thus it should be done fast and efficient. In particular, we suggest using the Laplace operator corresponding to a homogeneous media as a repconditioner [9]. It can be easily inverted using pseudo-spectral method as discussed in applications to fluid flow simulations [3,4,8]. In our previous research [9] we suggested using similar technique to solve Poisson equation for strongly hetrerogeneous media. In this paper, we present the improvement of the approach. We applied the Conjugate Gradient method instead of BiCGStab, we use the spectral decomposition of the discretized operator instead of the continuous case, and the cuBLAS functionality was utilized. As the result, the suggested approach allows solving Poisson equation inside the domain of the size of up 1000^3 voxels with a single GPU in 5 to 30 min depending on the model heterogeneity.

2 Mathematical Problem Formulation

Consider Poisson equation

$$\nabla \cdot (\sigma(\vec{x}) \nabla \varphi) = f(\vec{x}), \tag{1}$$

stated inside a domain $\Omega = [X_1^L, X_1^R] \times [X_2^L, X_2^R] \times [X_3^L, X_3^R]$. In general, any correct boundary conditions can be stated on $S = \partial\Omega$. However, in this research we focus on the statements typical for the digital rock physics, where Neumann boundary conditions are applied at the sides of the sample, which corresponds to a sample placed inside an insulator. The potential difference is applied at the two sides, that is the Dirichlet conditions. Formally it can be written as

$$\begin{array}{l} \varphi|_{X_2=X_2^L} = \varphi^L,\ \nabla\varphi \cdot \vec{n} = 0|_{X_2=X_2^L},\ \nabla\varphi \cdot \vec{n} = 0|_{X_3=X_3^L}, \\ \varphi|_{X_2=X_2^R} = \varphi^R,\ \nabla\varphi \cdot \vec{n} = 0|_{X_2=X_2^R},\ \nabla\varphi \cdot \vec{n} = 0|_{X_3=X_3^R}. \end{array} \tag{2}$$

In these notations, φ is the unknown function, which can be considered as the electric potential, $\sigma(\vec{x})$ is the electric conductivity, which is rapidly varying and strongly heterogeneous, but bounded function, $\vec{x}$ is the vector of spatial variables, $f(\vec{x})$ is the right-hand side, φ^L and φ^R are boundary conditions (applied potentials).

2.1 Finite-Difference Approximation

To approximate the Eq. (1) in the internal points we suggest using the second order finite difference scheme:

$$\begin{gathered}\frac{1}{h_1}\left(\tilde{\sigma}_{i_1+1/2,i_2,i_3}\frac{\varphi_{i_1+1,i_2,i_3}-\varphi_{i_1,i_2,i_3}}{h_1}-\tilde{\sigma}_{i_1-1/2,i_2,i_3}\frac{\varphi_{i_1,i_2,i_3}-\varphi_{i_1-1,i_2,i_3}}{h_1}\right)\\ +\frac{1}{h_2}\left(\tilde{\sigma}_{i_1,i_2+1/2,i_3}\frac{\varphi_{i_1,i_2+1,i_3}-\varphi_{i_1,i_2,i_3}}{h_2}-\tilde{\sigma}_{i_1,i_2-1/2,i_3}\frac{\varphi_{i_1,i_2,i_3}-\varphi_{i_1,i_2-1,i_3}}{h_2}\right)+\\ +\frac{1}{h_3}\left(\tilde{\sigma}_{i_1,i_2,i_3+1/2}\frac{\varphi_{i_1,i_2,i_3+1}-\varphi_{i_1,i_2,i_3}}{h_3}-\tilde{\sigma}_{i_1,i_2,i_3-1/2}\frac{\varphi_{i_1,i_2,i_3}-\varphi_{i_1,i_2,i_3-1}}{h_3}\right)=f_{i_1,i_2,i_3},\\ i_1=2,...,I_1-1,\ i_2=2,...,I_2-1,\ i_3=2,...,I_3-1.\end{gathered} \tag{3}$$

In these notations functions φ_{i_1,i_2,i_3} and f_{i_1,i_2,i_3} are the grid functions defined at the integer grid points $\varphi_{i_1,i_2,i_3} \approx \varphi((x_1)_{i_1},(x_2)_{i_2},(x_3)_{i_3})$ and $f_{i_1,i_2,i_3} \approx f((x_1)_{i_1},(x_2)_{i_2},(x_3)_{i_3})$. The conductivity σ is also defined at the integer grid points; i.e., however, it is used in the half-integer points, where is it modified to preserve the second order of convergence at the discontinuous coefficients:

$$\begin{aligned}\tilde{\sigma}_{i_1+1/2,i_2,i_3}&=2\left(\frac{1}{\sigma_{i_1+1,i_2,i_3}}+\frac{1}{\sigma_{i_1,i_2,i_3}}\right)^{-1},\\ \tilde{\sigma}_{i_1,i_2+1/2,i_3}&=2\left(\frac{1}{\sigma_{i_1,i_2+1,i_3}}+\frac{1}{\sigma_{i_1,i_2,i_3}}\right)^{-1},\\ \tilde{\sigma}_{i_1,i_2,i_3+1/2}&=2\left(\frac{1}{\sigma_{i_1,i_2,i_3+1}}+\frac{1}{\sigma_{i_1,i_2,i_3}}\right)^{-1}.\end{aligned} \tag{4}$$

The mathematical background of this type of coefficients averaging can be found in [9,14,19,21].

Approximation of the Eq. (1) in the vicinity of the boundaries involve the boundary conditions explicitly to keep the operator self-adjoin. We provide the equations for $i_1 = 1$, $i_2 = 1$, and $i_3 = 1$, the equations at the opposite sides can be derived in the same manner:

$$\begin{gathered}\frac{1}{h_1}\left(\tilde{\sigma}_{3/2,i_2,i_3}\frac{\varphi_{2,i_2,i_3}-\varphi_{1,i_2,i_3}}{h_1}-\tilde{\sigma}_{1/2,i_2,i_3}\frac{\varphi_{1,i_2,i_3}}{h_1}\right)\\ +\frac{1}{h_2}\left(\tilde{\sigma}_{1,i_2+1/2,i_3}\frac{\varphi_{1,i_2+1,i_3}-\varphi_{1,i_2,i_3}}{h_2}-\tilde{\sigma}_{1,i_2-1/2,i_3}\frac{\varphi_{1,i_2,i_3}-\varphi_{1,i_2-1,i_3}}{h_2}\right)\\ +\frac{1}{h_3}\left(\tilde{\sigma}_{1,i_2,i_3+1/2}\frac{\varphi_{1,i_2,i_3+1}-\varphi_{1,i_2,i_3}}{h_3}-\tilde{\sigma}_{1,i_2,i_3-1/2}\frac{\varphi_{1,i_2,i_3}-\varphi_{1,i_2,i_3-1}}{h_3}\right)\\ =f_{i_1,i_2,i_3}-\varphi^L_{i_2,i_3},\\ i_2=2,...,I_2-1,\ i_3=2,...,I_3-1.\end{gathered} \tag{5}$$

$$\begin{gathered}\frac{1}{h_1}\left(\tilde{\sigma}_{i_1+1/2,1,i_3}\frac{\varphi_{i_1+1,1,i_3}-\varphi_{i_1,1,i_3}}{h_1}-\tilde{\sigma}_{i_1-1/2,1,i_3}\frac{\varphi_{i_1,1,i_3}-\varphi_{i_1-1,1,i_3}}{h_1}\right)\\ +\frac{1}{h_2}\left(\tilde{\sigma}_{i_1,3/2,i_3}\frac{\varphi_{i_1,2,i_3}-\varphi_{i_1,1,i_3}}{h_2}\right)+\\ +\frac{1}{h_3}\left(\tilde{\sigma}_{i_1,1,i_3+1/2}\frac{\varphi_{i_1,1,i_3+1}-\varphi_{i_1,1,i_3}}{h_3}-\tilde{\sigma}_{i_1,1,i_3-1/2}\frac{\varphi_{i_1,1,i_3}-\varphi_{i_1,1,i_3-1}}{h_3}\right)=f_{i_1,1,i_3},\\ i_1=2,...,I_1-1,\ \ i_3=2,...,I_3-1.\end{gathered} \tag{6}$$

$$\begin{gathered}\frac{1}{h_1}\left(\tilde{\sigma}_{i_1+1/2,i_2,1}\frac{\varphi_{i_1+1,i_2,1}-\varphi_{i_1,i_2,1}}{h_1}-\tilde{\sigma}_{i_1-1/2,i_2,1}\frac{\varphi_{i_1,i_2,1}-\varphi_{i_1-1,i_2,1}}{h_1}\right)\\ +\frac{1}{h_2}\left(\tilde{\sigma}_{i_1,i_2+1/2,1}\frac{\varphi_{i_1,i_2+1,1}-\varphi_{i_1,i_2,1}}{h_2}-\tilde{\sigma}_{i_1,i_2-1/2,1}\frac{\varphi_{i_1,i_2,1}-\varphi_{i_1,i_2-1,1}}{h_2}\right)+\\ +\frac{1}{h_3}\left(\tilde{\sigma}_{i_1,i_2,3/2}\frac{\varphi_{i_1,i_2,2}-\varphi_{i_1,i_2,1}}{h_3}\right)=f_{i_1,i_2,1},\\ i_1=2,...,I_1-1,\, i_2=2,...,I_2-1.\end{gathered} \tag{7}$$

The resulting system can be written in a short matrix-vector form as

$$A\vec{\varphi}=\vec{g}, \tag{8}$$

where $\vec{\varphi}$ is the vector representation of the unknown function with the following indexing $j=i_1+(i_2-1)I_1+(i_3-1)I_1I_2$ (the order of the indices can be permuted without the loss of generality). The matrix A is sparse, self-adjoin, and positive definite. Vector $\vec{g}$ is formed from the right-hand sides f_{i_1,i_2,i_3} and the boundary conditions $\varphi^L_{i_2,i_3}$ and $\varphi^R_{i_2,i_3}$.

2.2 Conjugate Gradient Method

To solve system (8) we suggest using the Conjugate Gradient method with preconditioner. The algorithm can be represented as follows [18]:

1. $\vec{r}_0=\vec{g}-A\vec{\varphi}_0$, $\vec{p}_0=B^{-1}\vec{r}_0$, $\vec{q}_0=\vec{p}_0$;
2. $\alpha_j=\frac{(\vec{q}_j,\vec{r}_j)}{(A\vec{p}_j,\vec{p}_j)}$;
3. $\vec{\varphi}_{j+1}=\vec{\varphi}_j+\alpha_j\vec{p}_j$
4. $\vec{r}_{j+1}=\vec{r}_j-\alpha_jA\vec{p}_j$, $\vec{q}_{j+1}=B^{-1}\vec{r}_{j+1}$;
5. $\beta_j=\frac{(\vec{q}_{j+1},\vec{r}_{j+1})}{(\vec{q}_j,\vec{r}_j)}$;
6. $\vec{p}_{j+1}=\vec{q}_{j+1}+\beta_j\vec{p}_j$

This algorithm requires the storage of four vectors: $\vec{\varphi}_j$, $\vec{r}_j$, $\vec{q}_j$, and $\vec{p}_j$. Also, one more vector is needed which is $A\vec{p}_j$. Two matrix-vector multiplications should be performed at each iteration, they are $A\vec{p}_j$ and $B^{-1}\vec{r}_{j+1}$. Matrix B is the preconditioner which is constructed to reduce the condition number of the matrix $B^{-1}A$. Among the approaches to construct the preconditioners one may mention algebraic approaches such as incomplete LU, algebraic multigrid [18,20], also geometric multigrid preconditioner is efficient for smoothly varying coefficient $\sigma(\vec{x})$. The other way around is to construct the preconditioner based on physical properties of the simulated process. In fact, it is preferred matrix B to be as close to the original matrix A as possible but easy to invert. In [2,9] a spectral preconditionner was suggested, where matrix B corresponds to the Laplace operator with constant coefficient $\sigma=const$. In this case, the inversion of operator can be done using pseudo-spectral method [16]. It means that the Fast Fourier Transform (FFT) is applied along two spatial directions (in the considered case they are x_2 and x_3) and after that the series of 1D problems along

x_1 direction can be solved. A detailed description of this approach is presented in [9], where it was implemented to precondition BiConjugate Gradient method. However, implementation of the FFT assumes the periodic boundary conditions, whereas use of Neumann boundary conditions requires the sine transform [8]. It can be implemented either by the even continuation of the original array or by the proper permutation of the arrays elements [8]. However, use of different combinations of the boundary conditions at the sides of the computational domain leads to different transofrms. To overcome this difficulty, we suggest using explicit eigen-decomposition of 1D operators along x_2 and x_3 directions.

3 Spectral Preconditioner

Consider the Poisson equation with constant coefficient σ and construct its finite-difference approximation (3). Its matrix representation can be written as

$$B\vec{\psi} = \vec{w},$$

where vectors ψ and $\vec{w}$ have the same order of elements as $\vec{\varphi}$ in Eq. 8. Matrix B can be represented as the sum of three simple tri-diagonal matrices:

$$B = B_1 + B_2 + B_3,$$

each of these matrices represent approximation of the derivative with respect to the corresponding direction. They have the following form:

$$B_1 = \frac{\sigma}{h_1^2}\left(\begin{array}{ccc|c|ccc} T & & 0 & & 0 & & \\ & \ddots & & & & \ddots & \\ 0 & & T & & & & 0 \\ \hline & & & \ddots & & & \\ \hline 0 & & & & T & & 0 \\ & \ddots & & & & \ddots & \\ & & 0 & & 0 & & T \end{array}\right),$$

where matrix T is the tridiagonal matrix

$$T = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix},$$

$$B_2 = \frac{\sigma}{h_2^2}\left(\begin{array}{ccc|c|ccc} I & -I & & & 0 & & \\ -I & 2I & -I & & & & \\ & \ddots & \ddots & \ddots & & \ddots & \\ & & -I & 2I & -I & & \\ & & & -I & I & & 0 \\ \hline & & & & & & \\ & & & \ddots & & & \\ & & & & & & \\ \hline 0 & & & & I & -I & \\ & & & & -I & 2I & -I \\ & \ddots & & & & \ddots & \ddots \\ & & & & & -I & 2I \\ & & 0 & & & & -I \end{array}\right),$$

where I is the identity matrix of the size I_1, and

$$B_3 = \frac{\sigma}{h_3^2}\left(\begin{array}{ccc|ccc|ccc|ccc|ccc} I & & & -I & & & & & & & & & & & \\ & \ddots & & & \ddots & & & & & & & & & & \\ & & I & & & -I & & & & & & & & & \\ \hline -I & & & 2I & & & -I & & & & & & & & \\ & \ddots & & & \ddots & & & \ddots & & & & & & & \\ & & -I & & & 2I & & & -I & & & & & & \\ \hline & & & & \ddots & & & \ddots & & & \ddots & & & & \\ \hline & & & & & & -I & & & 2I & & & -I & & \\ & & & & & & & \ddots & & & \ddots & & & \ddots & \\ & & & & & & & & -I & & & 2I & & & -I \\ \hline & & & & & & & & & -I & & & I & & \\ & & & & & & & & & & \ddots & & & \ddots & \\ & & & & & & & & & & & -I & & & I \end{array}\right).$$

It is clear that the eigenvalues of the matrices B_2 and B_3 coincide with those of matrices

$$C_2 = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}, C_3 = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix},$$

but the size of the matrix C_2 is I_2, and size of C_3 is I_3. If we denote by V_2 the matrix of the eigenvectors of C_2 (same for C_3):

$$V_2 = \begin{pmatrix} v^2_{11} & v^2_{12} & \dots & v^2_{1I_2-1} & v^2_{1I_2} \\ v^2_{21} & v^2_{22} & \dots & v^2_{2I_2-1} & v^2_{2I_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v^2_{I_2-11} & v^2_{I_2-12} & \dots & v^2_{I_2-1I_2-1} & v^2_{I_2-1I_2} \\ v^2_{I_21} & v^2_{I_22} & \dots & v^2_{I_2I_2-1} & v^2_{I_2I_2} \end{pmatrix}.$$

then the matrices of the eigenvectors of $B_2 = U_2\Lambda_2 U_2^T$ and $B_3 = U_3\Lambda_3 U_3^T$ can be represented as:

$$U_2 = \left(\begin{array}{ccc|c|ccc} v^2_{11}I & \dots & v^2_{1I_2}I & & & & \\ \vdots & \ddots & \vdots & & & & \\ v^2_{I_21}I & \dots & v^2_{I_2I_2}I & & & & \\ \hline & & & \ddots & & & \\ \hline & & & & v^2_{11}I & \dots & v^2_{1I_2}I \\ & & & & \vdots & \ddots & \vdots \\ & & & & v^2_{I_21}I & \dots & v^2_{I_2I_2}I \end{array}\right),$$

$$U_3 = \left(\begin{array}{ccc|c|ccc} v^3_{11}I & & & \dots & v^3_{1I_3}I & & \\ & \ddots & & & & \ddots & \\ & & v^3_{11}I & & & & v^3_{1I_3}I \\ \hline & \vdots & & \ddots & & \vdots & \\ \hline v^3_{I_31}I & & & & v^3_{I_3I_3}I & & \\ & \ddots & & \dots & & \ddots & \\ & & v^3_{I_31}I & & & & v^3_{I_3I_3}I \end{array}\right).$$

The main point is that application of the eigenvalue decomposition to any of the considered matrices B_2 or B_3 does not change the other two matrices; thus, the following change of variables can applied

$$\vec{\gamma} = U_2^T U_3^T \vec{\psi},$$

and the system transforms to

$$U_2^T U_3^T (B_1 + B_2 + B_3) U_3 U_2 \vec{\gamma} = (B_1 + \Lambda_2 + \Lambda_3)\vec{\gamma} = U_3^T U_2^T \vec{w} = \vec{s}. \qquad (9)$$

Note, that matrices Λ_2 and Λ_3 are diagonal, thus the problem was reduced to a series of 1D problems which may be soled with direct Gauss elimination process with $O(I_1)$ floating point operations. Eigenvectors and eigenvalues of matrices C_2 and C_3 can be found analytically.

It means, that to invert the preconditioner, one needs to apply two series of matrix-vector multiplications with the matrix size coinciding with 1D size of the computational domain, and after that solve a series of the systems with tridiagonal matrices. Below we discuss the practical aspects of the algorithm implementation.

4 GPU-Based Implementation

Implementation of the described algorithm to GPU-based architecture is closely related to that presented previously in [9]. The main feature of the suggested implementation is that all variables are stored in GPU memory during the simulation, and only the final result $\vec{\varphi}$ is copied to the RAM. Thus, the algorithm is almost free from device-to-host or host-to-device memory copy. The array of the electric conductivity σ_{i_1,i_2,i_3} is read from file and copied to the GPU memory at the beginning. After that, memory is allocated on GPU for the solution $\vec{\varphi}$ and for the auxiliary arrays used in the CG: $\vec{r}$, $\vec{p}$, $\vec{q}$.

The preconditioned CG algorithm requires execution of four main types of the operations: action of the Laplace operator on a vector; i.e. $A\vec{p}_j$; computation of the inner products, computation of the linear combinations of vectors, and action of the preconditioner $B^{-1}\vec{r}_{j+1}$. The action $A\vec{p}_j$ is implemented directly, using formula (3). This is the stensil computation, thus it is free from data dependencies and it is easy to parallelize on the multi-core architecture. To execute the dot product and compute linear combinations we used the cuBLAS "dot" and "axpy" functions respectively.

The most time consuming procedure is the computation of action of the preconditioner $B^{-1}\vec{r}_{j+1}$. To invert the operator, we need to apply the spectral decomposition, as described above. The approach is based on the construction of the eigenvectors and eigenvalues of 1D operators. We assume the model is homogeneous, thus the matrices of the eigenvectors V_2 and V_3 can be computed using analytical formulae, same for the eigenvalues. Note, that apart from the theoretical constructions, we do not need to store matrices U_2 and U_3, but only the compact matrices V_2 and V_3 are required. The size of these matrices is $I_2 \times I_2$ and $I_3 \times I_3$, respectively. It means, that they require only a few Mb of memory to store them. The action of matrices U_2 and U_3 on a vector can be computed using their compact representation V_2 and V_3 using the cuBLAS functionality, in particular, "gemm" functionality is used, which allows to represent series of matrix-vector multiplications as a matrix-matrix multiplication with high optimization. The resulting series of systems (9) is solved then using Gauss elimination. Note, that the resulting 1D problems are independent and their size is relatively small equal to I_1. The matrices of the 1D systems are tridiagonal, thus the systems can be solved independently. Moreover, due to the small size of each problem, a single system can be solved using a single core. Note, that the total number of eigenvalues is $I_2 I_3 \approx 10^6$, thus, all available GPU cores are involved in embarrassingly parallel computations. Then the series of 1D solutions is obtained, the inverse transoms $\gamma = U_3^T U_2^T \vec{w}$ are computed using "gemm" function from cuBLAS. Thus, all computationally intense procedures in the algorithm are executed with the highly optimized cuBLAS functionality.

5 Numerical Experiments

We omit the experiments on the algorithm verification. We reproduced the experiments, presented in [9], estimating the effective electric conductivity of layered

media. Moreover, for all experiments described below we performed simulation using the presented algorithm and compared the results with those obtained using the algorithm from [9]. For our experiments we used the CT-scan of the Betheimer sandstone. The resolution of the image was 2 mkm per voxel. The size of the image was 750 voxels in each direction. In our simulations we used either parts of the image; i.e., we cropped the images to get a smaller one. To perform simulations for the sample of the higher size than 750^3 we extended the sample periodically. The images of the considered samples are provided in Figs. 1, 2 and 3. The filled regions in the figures correspond to the mineral grains and the empty space represent the pore space. We assumed the pore space filled with conductive fluid with electric conductivity equal to one; i.e., $\sigma(\vec{x}) = 1$ if $\vec{x}$ belongs to the pore space. We varied the electric conductivity of the grains $\sigma_g \in \{10^{-1}, 10^{-2}, ..., 10^{-5}\}$. Formally, the insulators like calcite, quarts, feldspar and others may have much lower conductivity. However, in this case, one may consider them absolutely insulating and solve the Poisson equation only in the pore space, instead of the whole sample. So, we omit extremely high contrasts. The case, when the conductivities of the fluid and matrix coincide; i.e., $\sigma_g = 1$ is also excluded, because it corresponds to the homogeneous material, and the suggested preconditioner will be the same as the original operator; thus, the algorithm converges in one iteration.

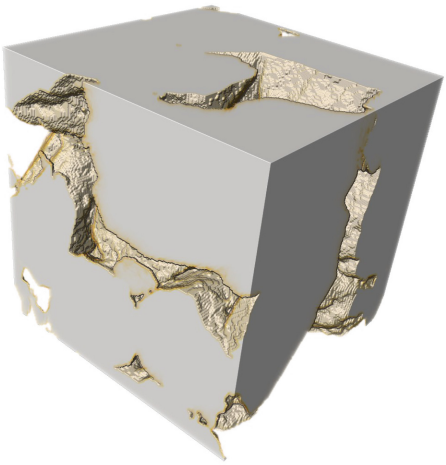

Fig. 1. The example of the model of the size 200^3 voxels

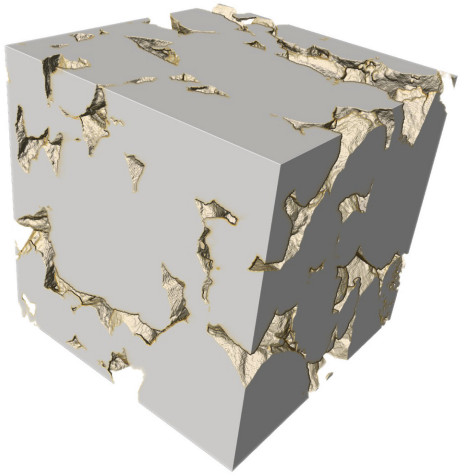

Fig. 2. The example of the model of the size 400^3 voxels

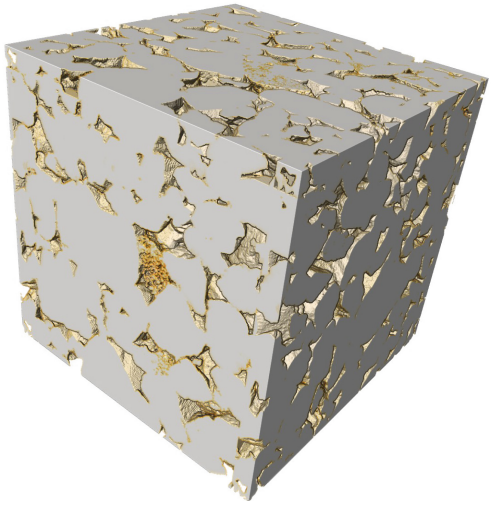

Fig. 3. The example of the model of the size 800^3 voxels

First, we performed simulations for the model of the size of 200^3 voxels. We varied the conductivity of the core matrix. Also, the simulations were done using two GPUs: nVidia A100 GPU with 80 Gb of memory and RTX 2080Ti GPU with 12 Gb of memory. These simulations were done to make the timing measurements consistent with those provided in [9]. The results are presented in Table 1. Note, that A100 GPU allows solving the problem ten times faster

Table 1. Number of iterations and timing of solving Poisson equation inside the domains of 200^3 voxels.

σ_g	no. iterations	time (s) RTX 2080	time (s) A100
10^{-1}	64	3.51	0.34
10^{-2}	157	8.49	0.87
10^{-3}	379	20.54	2.12
10^{-4}	781	42.39	4.06
10^{-5}	887	47.27	4.82

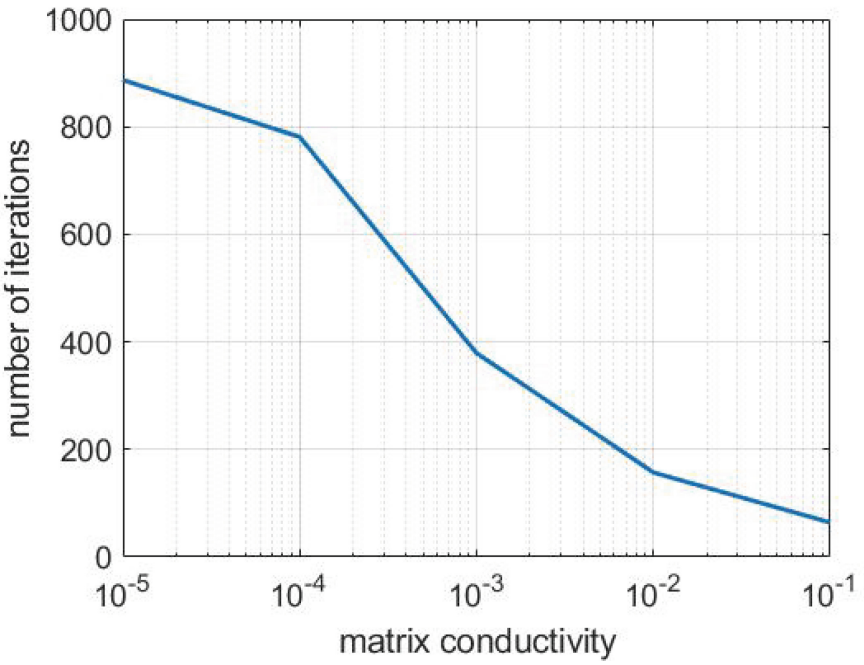

Fig. 4. Number of iteration in dependence on the conductivity contrast

than RTX2080. Number of iterations was, as expected, the same, because it does not suppose to depend on the computing architecture. The number of iterations increases with increase of the conductivity contrast. However, as it may be noted in Fig. 4, the number of iterations depends almost linearly on the $\ln \sigma_g$.

The second series of experiments was performed on the A100 GPUs only and were oriented on the large-scale simulations. We dealt with samples of the size of 200^3, 400^3, 800^3, and 1000^3 voxels, but restricted the considerations with only two values of matrix conductivity $\sigma_g = \{10^{-2}, 10^{-4}\}$. Number of iterations and timings are provided in Tables 2 and 3. Provided numbers of the iterations are

Table 2. Number of iterations and timing of solving Poisson equation in dependence on the problem size for $\sigma_g = 10^{-2}$.

size	no. iterations	time (s) A100	time per it. (s)
200^3	157	0.87	0.0055
400^3	215	9.36	0.0435
800^3	271	121.02	0.4466
1000^3	289	341.00	1.1799

Table 3. Number of iterations and timing of solving Poisson equation in dependence on the problem size for $\sigma_g = 10^{-4}$.

size	no. iterations	time (s) A100	time per it. (s)
200^3	783	4.06	0.0052
400^3	1094	46.68	0.0427
800^3	1387	812.34	0.5857
1000^3	1514	1812.78	1.1973

required to achieve the relative residual norm equal to 10^{-6}. Note, that in both cases the number of iterations are weakly dependent on the size of the computational domain. The number of iterations increases with the domain size, but the growth rate is lower than linear. In particular, increase of the computational domain from 200 to 800 points lead to the increase of the iterations number by the factor of only 1.75. We need to point out that we compare the number of iterations with respect to the size of the computational domain, whereas the actual size of the system of linear equations is 200^3 to 1000^3. It means that the increase of the number of iterations with the growth of the problem size is negligible. We also figured out, that time per a single iteration (which includes matrix-vector multiplication $A\vec{p}_j$ and action of the preconditioner $B^{-1}\vec{r}_{j+1}$) is proportional to the problem size. It means, that the action of the perconditioner does not increase the computational complexity of the algorithm; i.e., it requires the number of flops proportional to the problem size.

6 Conclusion

We presented an original approach to construct preconditioner for numerical solution of Poisson equation for strongly heterogeneous media. The idea of the preconditioner is based on that, presented in [9], where the inverted Laplace operator corresponding to the homogeneous model is used to precondition the problem when solved with Conjugate Gradient method. Apart from the previous approaches we suggested solving Poisson equation for homogeneous media using eigenvalue decomposition of two 1D matrices, approximating Laplace operator with given boundary conditions. After that the change of variables is applied by

the action of the eigenvectors matrices and the resulting series of 1D systems with tridiagonal matrices are solved using direct methods. All the matrix-vector and matrix-matrix multiplications are executed with the help of cuBLAS functionality. We applied the approach to compute the electric potential in digital rock samples with different rock matrix conductivity. We illustrated that number of iterations of the preconditioned CG algorithm is almost independent on the problem size, however, iterations linearly depend on the logarithm of the conductivity contrast. Finally, we illustrated that use of modern GPUs (A100 series) allows solving Poisson equation in a domain of the size up to 1000^3 in 5 to 30 min depending on the model heterogeneity with a single GPU.

References

1. Andra, H., et al.: Digital rock physics benchmarks - part ii: Computing effective properties. Comput. Geosci. **50**, 33–43 (2013)
2. Belonosov, M., Kostin, V., Neklyudov, D., Tcheverda, V.: 3D numerical simulation of elastic waves with a frequency-domain iterative solver. Geophysics **83**(6), T333–T344 (2018)
3. Costa, P.: A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows. Comput. Math. Appl. **76**(8), 1853–1862 (2018). https://doi.org/10.1016/j.camwa.2018.07.034
4. Crialesi-Esposito, M., et al.: Flutas: a GPU-accelerated finite difference code for multiphase flows. Comput. Phys. Commun. **284**, 108602 (2023). https://doi.org/10.1016/j.cpc.2022.108602
5. Dorn, C., Schneider, M.: Lippmann-schwinger solvers for the explicit jump discretization for thermal computational homogenization problems. Int. J. Numer. Meth. Eng. **118**(11), 631–653 (2019)
6. Evstigneev, N.M., Ryabkov, O.I., Gerke, K.M.: Stationary stokes solver for single-phase flow in porous media: a blastingly fast solution based on algebraic multigrid method using GPU. Adv. Water Resour. **171**, 104340 (2023)
7. Haber, E., Ascher, U.M., Aruliah, D.A., Oldenburg, D.W.: Fast simulation of 3D electromagnetic problems using potentials. J. Comput. Phys. **163**(1), 150–171 (2000)
8. Hasbestan, J.J., Xiao, C.N., Senocak, I.: Pittpack: an open-source poisson's equation solver for extreme-scale computing with accelerators. Comput. Phys. Commun. **254**, 107272 (2020). https://doi.org/10.1016/j.cpc.2020.107272
9. Khachkova, T., Lisitsa, V., Reshetova, G., Tcheverda, V.: GPU-based algorithm for evaluating the electrical resistivity of digital rocks. Comput. Math. Appl. **82**, 200–211 (2021)
10. Khachkova, T.S., Lisitsa, V.V., Gondul, E.A., Prokhorov, D.I., Kostin, V.I.: Two-phase flow simulation algorithm for numerical estimation of relative phase permeability curves of porous materials. Russ. J. Numer. Anal. Math. Model. **39**(4), 209–221 (2024). https://doi.org/10.1515/rnam-2024-0020
11. Kim, J.: Phase-field models for multi-component fluid flows. Commun. Comput. Phys. **12**(3), 613–661 (2012)
12. Lee, B., Min, C.: Optimal preconditioners on solving the poisson equation with neumann boundary conditions. J. Comput. Phys. **433**, 110189 (2021)

13. Lee, S., Wheeler, M.F.: Enriched galerkin methods for two-phase flow in porous media with capillary pressure. J. Comput. Phys. **367**, 65–86 (2018). https://doi.org/10.1016/j.jcp.2018.03.031
14. Lisitsa, V., Podgornova, O., Tcheverda, V.: On the interface error analysis for finite difference wave simulation. Comput. Geosci. **14**(4), 769–778 (2010)
15. Loisel, S., Szyld, D.: On the geometric convergence of optimized schwarz methods with applications to elliptic problems. Numer. Math. **114**(4), 697–728 (2010)
16. Pleshkevich, A., Vishnevskiy, D., Lisitsa, V.: Sixth-order accurate pseudo-spectral method for solving one-way wave equation. Appl. Math. Comput. **359**, 34–51 (2019)
17. Redondo, C., Rubio, G., Valero, E.: On the efficiency of the impes method for two phase flow problems in porous media. J. Petrol. Sci. Eng. **164**, 427–436 (2018). https://doi.org/10.1016/j.petrol.2018.01.066
18. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM (2003)
19. Samarskii, A.A.: The Theory of Difference Schemes, Pure and Applied Mathematics, vol. 240. CRC Press (2001)
20. Stuben, K.: A review of algebraic multigrid. J. Comput. Appl. Math. **128**(1–2), 281–309 (2001)
21. Vishnevsky, D., Lisitsa, V., Tcheverda, V., Reshetova, G.: Numerical study of the interface errors of finite-difference simulations of seismic waves. Geophysics **79**(4), T219–T232 (2014)
22. Xia, J.: A robust inner–outer hierarchically semi-separable preconditioner. Numer. Linear Algebra Appl. **19**(6), 992–1016 (2012). https://doi.org/10.1002/nla.1850
23. Zhan, X., Schwartz, L.M., Toksöz, M.N., Smith, W.C., Morgan, F.D.: Pore-scale modeling of electrical and fluid transport in berea sandstone. Geophysics **75**(5), F135–F142 (2010)