

Field-Split Iterative Solver for Quasi-Static Biot Equation

Sergey Solovyev[1(✉)], Mikhail Novikov[1], and Vadim Lisitsa[2]

[1] Institute of Mathematics SB RAS, Koptug ave. 4, Novosibirsk 630090, Russia
solovevsa@ipgg.sbras.ru
[2] Institute of Petroleum Geology and Geophysics SB RAS, Koptug ave. 3, Novosibirsk 630090, Russia
lisitsavv@ipgg.sbras.ru

Abstract. Frequency domain Biot equation in quasi-static state can be applied to model low-frequency loading of fluid-filled poroelastic materials and estimate effective frequency-dependent strain-stress relations. We propose an algorithm of solving the system of linear algebraic equations (SLAE) obtained from finite-difference approximation of Biot equations. The algorithm involves the Biconjugate Gradient Stabilized (BICGSTAB) iterative solver with a field-split preconditioner technique. In this paper, we estimate the number of floating-point operations required for both the proposed algorithm and direct approach. Performed numerical experiments demonstrate the fast convergence of the iterative process and confirm performance and memory usage benefits of the suggested approach compared to the direct method.

Keywords: Poroelasticity · wave-induced fluid flow · Biot equation · quasi-static state · finite differences · direct methods for SLAE · iterative methods · field-split preconditioner

1 Introduction

Numerical solution of the system of linear algebraic equations (SLAE) problem arises from a differential problem, and often involves computational cost problem, so it requires both knowledge base in mathematical algorithms for solving such systems and program implementation for high-performance computing systems (HPC). Biot equation in the quasi-static state in the frequency domain is approximated by finite-difference approach [10,11] on the staggered grid with partial separation of variables and equations [1]. The SLAE can be solved either by optimised implementation of the direct approach based on LU-decomposition [2,5,9] or by iterative algorithms using preconditioner techniques [14]. There are

S.S. developed the algorithm and M.N. did the numerical simulations using the supercomputer facilities of Siberian Supercomputer Center (Cluster NKS-30T) under the support of RSCF grant no. 19-77-20004. V.L. performed the analysis of the results under the support of FNI project FWZZ-2022-0022.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14106, pp. 45–58, 2023.
https://doi.org/10.1007/978-3-031-37111-0_4

various preconditioner approaches such as ILU0, multigrid [13], filed-split [6], etc. In this paper we use the field split preconditioner to separate equations and variables into two groups: related to solid deformation and related to fluid transport, and these equations can be solved separately.

First, we present the numerical upscaling algorithm for estimation of the effective properties of fractured porous isotropic media in the low-frequency range. Then we describe the finite-difference approximation of Biot equations and corresponding mesh stencils, which define the SLAE properties and so effect the details of algorithm of its numerical solution. Two approaches are considered: *Direct* is based on optimised LU matrix decomposition (we use black-box implementation of well known software) and *Iterative* preconditioned solver. We have developed our own matrix preconditioner based on the field-split idea. Floating point operations (FLOPS) estimates for both approaches coupled with memory usage and numerical experiments show the advantages of the direct approach for small problems and the iterative one for the large models.

2 Statement of the Problem

2.1 Biot Equations in Quasi-Static State

In this paper we consider Biot model equations stated in the quasi-static regime representing the pressure diffusion process in fluid-filled poroelastic media [4]:

$$\frac{\partial}{\partial x}\left[C_{11}\frac{\partial u_x}{\partial x}+C_{13}\frac{\partial u_z}{\partial z}\right]+\frac{\partial}{\partial z}\left[C_{55}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\right]+\frac{\partial}{\partial x}\left[\alpha M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]=0, \tag{1}$$

$$\frac{\partial}{\partial x}\left[C_{55}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[C_{13}\frac{\partial u_x}{\partial x}+C_{33}\frac{\partial u_z}{\partial z}\right]+\frac{\partial}{\partial z}\left[\alpha M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]=0, \tag{2}$$

$$\frac{\partial}{\partial x}\left[\alpha M\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)\right]+\frac{\partial}{\partial x}\left[M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]-i\omega\frac{\eta}{k}w_x=0, \tag{3}$$

$$\frac{\partial}{\partial z}\left[\alpha M\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)\right]+\frac{\partial}{\partial z}\left[M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]-i\omega\frac{\eta}{k}w_z=0. \tag{4}$$

In these notations $\boldsymbol{u}=(u_x,u_z)^T$ is the velocity vector, $\boldsymbol{w}=(w_x,w_z)^T$ is the relative fluid velocity vector, ω is the time frequency, C_{11}, C_{33}, C_{13}, C_{55} are the components of the fluid-saturated matrix stiffness tensor, M is the fluid storage coefficient, and α is the Biot-Willis coefficient. Note, that the stiffness tensor is the fourth-order tensor, however, due to its symmetry as well as the symmetry of the strain and stress tensors, we can use the Voight notations and write down the Hooke's law in the following form:

$$\begin{pmatrix}\sigma_{xx}\\ \sigma_{zz}\\ \sigma_{xz}\end{pmatrix}=\begin{pmatrix}C_{11} & C_{13} & 0\\ C_{13} & C_{33} & 0\\ 0 & 0 & C_{55}\end{pmatrix}\begin{pmatrix}\varepsilon_{xx}\\ \varepsilon_{zz}\\ \varepsilon_{xz}\end{pmatrix},$$

where the components of the strain tensor are defined as follows:

$$\varepsilon_{xx} = \frac{\partial u_x}{\partial x}, \varepsilon_{zz} = \frac{\partial u_z}{\partial z}, \varepsilon_{xz} = \frac{1}{2}\left(\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z}\right) \tag{5}$$

The system of Eqs. (1)–(4) should satisfy the following boundary conditions:

$$\begin{aligned} \sigma \cdot \boldsymbol{n}|_{\partial D} &= \boldsymbol{\sigma}_0 \\ \boldsymbol{w} \cdot \boldsymbol{n}|_{\partial D} &= 0. \end{aligned} \tag{6}$$

The goal of this research is to recover the frequency-dependent stiffness tensor, corresponding to orthotropic viscoelastic media so that the solution of the original problem (1)–(4) and (6) coincides with the solution of the upscaled problem:

$$\frac{\partial}{\partial x}\left[\hat{C}_{11}(\omega)\frac{\partial \hat{u}_x}{\partial x} + \hat{C}_{13}(\omega)\frac{\partial \hat{u}_z}{\partial z}\right] + \frac{\partial}{\partial z}\left[\hat{C}_{55}(\omega)\left(\frac{\partial \hat{u}_x}{\partial z} + \frac{\partial \hat{u}_z}{\partial x}\right)\right] = 0, \tag{7}$$

$$\frac{\partial}{\partial x}\left[\hat{C}_{55}(\omega)\left(\frac{\partial \hat{u}_x}{\partial z} + \frac{\partial \hat{u}_z}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\hat{C}_{13}(\omega)\frac{\partial \hat{u}_x}{\partial x} + \hat{C}_{33}(\omega)\frac{\partial \hat{u}_z}{\partial z}\right] = 0, \tag{8}$$

$$\hat{\sigma} \cdot \boldsymbol{n}|_{\partial D} = \boldsymbol{\sigma}_0 \tag{9}$$

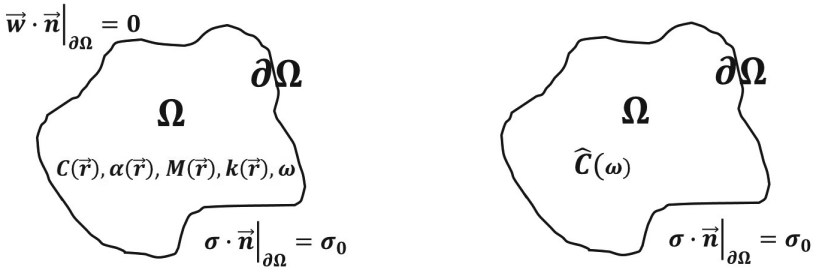

Fig. 1. Left: original fluid-saturated poroelastic media; right: reconstructed effective frequency-dependent viscoelastic media. Vector $\boldsymbol{n}$ is the outer normal, and $C(\boldsymbol{r}), \alpha(\boldsymbol{r}), M(\boldsymbol{r}), k((\boldsymbol{r}))$ are the space-dependent parameters of Eqs. (1)–(4).

To solve the upscaling problem we consider system of equations stated in a rectangular domain $D = [L_1^x, L_2^x] \times [L_1^z, L_2^z]$ with boundary conditions

$$\begin{aligned} (\sigma_{xx}, \sigma_{xz})^t|_{x=L_1^x, x=L_2^x} &= (\sigma_{01}, \sigma_{02})^t = (\phi_x, \psi) \\ (\sigma_{xz}, \sigma_{zz})^t|_{z=L_1^z, z=L_2^z} &= (\sigma_{01}, \sigma_{02})^t = (\psi, \phi_z), \end{aligned} \tag{10}$$

where ϕ_x and ϕ_z are the loads σ_{xx} and σ_{zz} at the opposite sides of the domain, respectively, and ψ is the torsion value applied to all four sides of the domain.

Resolving the problem (1)–(4) one can define the averaged stress and strain components as $\int_\Omega \varepsilon dv = \int_\Omega \hat{\varepsilon} dv = I_\varepsilon$ ($\varepsilon \in \{\varepsilon_{xx}, \varepsilon_{zz}, \varepsilon_{xz}\}$) and same stress σ_0 at the boundary $\partial\Omega$. The strains ε distributions in the original medium satisfy the solution of the Eqs. (1)–(4) with boundary conditions (Fig. 1, left). Thus for $\sigma \in \{\sigma_{xx}, \sigma_{zz}, \sigma_{xz}\}$ (components of the stress tensor in porous fluid-saturated media) $\int_\Omega \sigma dv = \int_\Omega \hat{\sigma} dv = I_\sigma$.

If the upscaled problem (7), (8) is considered in a rectangular domain with the boundary conditions (9) the following relations are valid due to constant values of the stiffness tensor:

$$I_\sigma = \int_\Omega \hat{\sigma} dv = \int_\Omega \hat{C}\hat{\varepsilon} dv = \hat{C} \int_\Omega \hat{\varepsilon} dv = \hat{C} I_\varepsilon$$

If we require the averaged stresses and strains for the two problems to coincide, we finish up with the following system of linear algebraic equations (SLAE) $I_\sigma = \hat{C} I_\varepsilon$ with three equations and nine unknown components of the tensor $\hat{C}$:

$$\begin{aligned} I_{\sigma xx} &= \hat{C}_{11} I_{\varepsilon xx} + \hat{C}_{13} I_{\varepsilon zz} + \hat{C}_{15} I_{\varepsilon xz}, \\ I_{\sigma zz} &= \hat{C}_{31} I_{\varepsilon xx} + \hat{C}_{33} I_{\varepsilon zz} + \hat{C}_{35} I_{\varepsilon xz}, \\ I_{\sigma xz} &= \hat{C}_{51} I_{\varepsilon xx} + \hat{C}_{53} I_{\varepsilon zz} + \hat{C}_{55} I_{\varepsilon xz}. \end{aligned} \tag{11}$$

It is necessary to perform three experiments to obtain a unique solution, because one experiment with one load leads to three Eqs. (11) with nine unknowns $\hat{C}$. Experiments of original media reconstruction use three sets of boundary conditions (10) with basic loads:

– x-direction compression:

$$\phi_x = 1, \phi_z = 0, \psi = 0, \tag{12}$$

– z-direction compression:

$$\phi_x = 0, \phi_z = 1, \psi = 0, \tag{13}$$

– xz-torsion:

$$\phi_x = 0, \phi_z = 0, \psi = 1. \tag{14}$$

The details of the reconstruction of the effective viscoelastic media using set of numerical experiments on different models, estimation of both velocity and quality factors are presented in the paper [12].

Below we briefly describe the discretisation of 2D Biot Eqs. (1)–(4) with the presented boundary conditions (10) and construction of the SLAE. After that we develop algorithms of the numerical solution of this linear system.

2.2 Discretisation of Biot Equations

Biot Eqs. (1)–(4) have been discretised by the rectangular grid of $N_x \times N_z$ points. The domain boundaries cross the mesh cells at the centres of the cell faces as illustrated in Fig. 2.

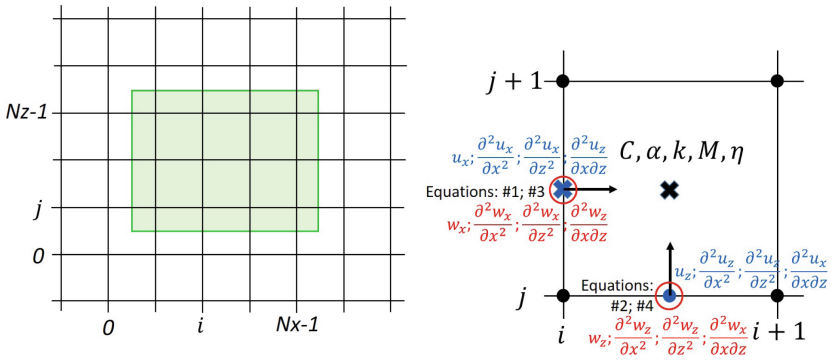

Fig. 2. Discretisation of the computational domain (left). Solution components, derivatives, location of physical parameters on the computational grid, and nodes where equations are approximated (right).

Positions of the variables on the grid are given in the Fig. 2. Horizontal components of the displacement vectors u_x and w_x are defined in the half-integer points in the x direction and at integer points in the z direction. The vertical components of the solution vector are stored at integer points in the x direction and at half-integer points in the z direction. All the coefficients of the equations are defined in integer grid points in both spatial directions. Equations (1) and (3) are approximated at the centres of the vertical edges, (2) and (4) at the centres of the horizontal edges.

Biot equations are approximated inside the computational domain (Fig. 2) and at its boundary. Thus the total number of finite-difference equations approximating the Eqs. (1) and (3) is $(N_x - 1)N_z$; and the number of equations approximating (2) and (4) is $N_x(N_z - 1)$. So, the total number of linear equations is

$$N = 2(N_x - 1)N_z + 2N_x(N_z - 1) \approx 4N_x N_z.$$

Details of the finite-difference approximation of the Biot equations are presented in [12] and will not be discussed here. However, we need to point out the main properties of the derived system of equations. First, the matrix of the system A is a sparse square matrix, which is complex, non-Hermitian and non-symmetric. Second, due to the Neumann-type boundary conditions (6) the matrix is ill-conditioned (singular). In addition, we need to consider the number of non-zero elements $NNZ(A)$ in the matrix A which depends on the finite-difference rules used to approximate the equations. The stencils used to approximate Biot's equations are shown in the Fig. (3). The approximation of the Eqs. (1) and (2) requires 16 points, so it involves 16 elements. The approximation of the Eqs. (3) and (4) requires 14 elements. So, $NNZ(A) \approx 2(16 + 14)N_x N_z = 60N_x N_z$.

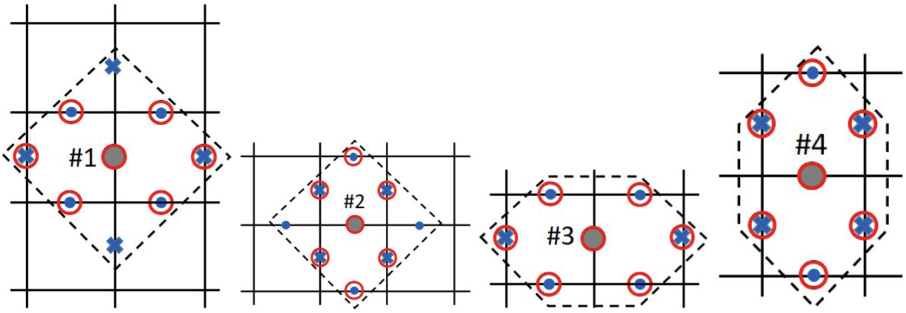

Fig. 3. Patterns for Biot equations.

3 Numerical Solution of SLAE

When developing a solver of SLAE which is derived from the discretisation of differential equations, the following points need to be considered:

- **Accuracy**, which means that the algorithm should provide a reasonable numerical solution error or relative residual.
- **Performance**, which means that the algorithm should solve the SLAE in a reasonable time. Therefore, the number of floating point operations (FLOP) must be estimated and minimised if possible. In addition, the implementation of the algorithm should use the high-performance features like parallelisation.
- **Memory usage** should be minimised if possible, to fit available computational resources.

There are two main groups of methods to solve SLAE: *Direct* methods and *Iterative* methods. Below we consider both approaches and apply them to the solution of Biot equations in the quasi-static state.

3.1 Direct Solver

Direct solvers are based on the LU factorization algorithms. They provide the best achievable accuracy, but require enormous amounts of computer memory and number of flops. For sparse SLAEs, there are special algorithms that reduce memory consumption and flops. These algorithms include reordering of columns and rows of the initial matrix, multifrontal factorization, and others. They are implemented in commercial solvers such as SuperLU [9], MUMPS [2], PARDISO[5], etc. In our previous research we proposed an algorithm to solve the Biot equations in the quasi-static state using the direct solvers [12]. However, due to severe memory limitations (512 Gb of RAM), the use of the direct solver did not allow us to proceed to problems with more than 2000^2 points.

For the sake of completeness we recall the main estimates of the RAM $Mem(Dir)$ and Flops $Flops(Dir)$ to solve the Biot equations using the direct solver.

Sparse direct solver performs three steps to solve SLAE:

1. Reordering (preliminary step to reduce the number of non-zeros in the LU factors),
2. Factorization (perform LU factorization),
3. Solving (solve the systems $Ly = b$ and $Ux = y$).

The Reordering uses graph partitioning algorithms to permute the columns and rows of initial matrix before factorization. It operates with integer numbers, faster than factorization step and can be executed just once for all time frequencies. The last is due to the fixed matrix pattern which is independent of the frequency. We do not take this step into account to estimate $Flops(Dir)$. Factorization and solving steps are performed for every frequency. Thus,

$Flops(Dir) = N_{freq}[Flops(Fct) + Flops(Slv)]$.

The number of flops at solving step $Flops(Slv)$ can be estimated by the number of non-zero elements in L and U, because a single element in these matrices leads to two operations (one summation and one multiplication). The matrix A is structurally symmetric (not self-adjoint), so the patterns of L and U coincide, and $NNZ(LU) = 2 * NNZ(L)$. We also need to consider the number of right hand sides (RHS). Three different loads provide three RHS, so $Flops(Slv) = (3\ RHS) * (2\ operation\ per\ one\ element) * (2NNZ(L))$

Finally, the total number of floating-point operations is:

$Flops(Dir) = N_{freq}[Flops(Fct) + 12NNZ(L)]$.

Below we provide the values of $Flops(Fct)$ and $NNZ(L)$ corresponding to the particular experimental set-up.

3.2 Iterative Solver

Iterative algorithms mostly use less memory than direct methods. However, the accuracy of the solution is not guaranteed if convergence of iterative process is poor. Different preconditioners are used to speed up the convergence rate [3,7,8,13]. The preconditioner B leads to the SLAE: $\hat{A}x = \hat{b}$ where $\hat{A} = B^{-1}Ax$, $\hat{b} = B^{-1}b$. To solve the complex non-hermitian matrix $\hat{A}$ the BICGSTAB algorithm [14] is used:

$$
\begin{aligned}
&set\ x_0\\
&set\ j_{restart}\\
&r_0 := \hat{b} - \hat{A}x_0\\
&p_0 := r_0\\
&For\ j = 0, 1, \ldots, until\ convergence\ Do:\\
&a_j := \tfrac{(r_j, r_0^*)}{(\hat{A}p_j, r_0^*)}\\
&s_j := r_j - a_j\hat{A}p_j\\
&\omega_j := \tfrac{(\hat{A}s_j, s_j)}{(\hat{A}s_j, \hat{A}s_j)}\\
&x_{j+1} := x_j + \alpha_j p_j + \omega_j s_j\\
&r_{j+1} := s_j + \omega_j \hat{A}s_j\\
&If\ remainder(j/j_{restart}) = 0\ Then\ r_{j+1} := \hat{b} - \hat{A}x_{j+1}\\
&\beta_j := \tfrac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)}\tfrac{\alpha_j}{\omega_j}\\
&p_{j+1} := r_{j+1} + \beta_j(p_j - \omega_j\hat{A}p_j)\\
&EndDo
\end{aligned}
\tag{15}
$$

To compute the number of floating-point operations of BCGStab we should count the "constant times a vector plus a vector" operations (**axpy**), the scalar product (**dot**) and the $\hat{A}x$ product (the most time consuming part of this algorithm). The $\hat{A}x$ product consists of the matrix-vector product of the initial matrix A (**mv**) and the inversion the preconditioner B (**inv**). Taking into account the Flops provided in the Table 1 the total number of operations for iterative process can be estimated as:

$Flops(Itr) = N_{freq}N_{iter}[80N + 2Flops(invB)]$.

The construction of the preconditioner B and the counting $Flops(invB)$ is described in the next subsection.

Table 1. Estimation of floating-point operations number for the major operations in BCGStab. We suppose that $N = 4n^2$ and $NNZ(A) = (32 + 26) * n^2 = 15N$.

Operation type	Nflops per one operation	Number of operations per one step
axpy	2N	6
dot	2N-1	4
mv	2*NNZ(A)	2
inv	TBD	2

3.3 Construction of the Preconditioner

On the one hand the preconditioner B should be easily invertible, on the other hand it should not be as close to the inverse matrix of the system as possible. The standard preconditioners like Jacoby, Zeidel, Block-Jacoby are not suitable

for Biot equations in the quasi-static state. According to numerical experiments they do not guarantee the convergence of the BCGStab.

Another approach to construct preconditioner B is called "Field-Split" [6]. The idea is to reduce the system of equations to the block-lower-triangular form; i.e., to exclude the variable w from the Eqs. 1, 2. Then the equilibrium equations for the elastic matrix can be solved, and its solution can be used to construct a right-hand side of the Eqs. 3, 4. Then the components of w can be found as the solution of the Eqs. 3, 4 with new right-hand sides (Table 2).

So, the matrix representation of the preconditioning operator can be represented as:

Table 2. Differential equation of "Field-Split" preconditioner B.

$\frac{\partial}{\partial x}\left[C_{11}\frac{\partial u_x}{\partial x}+C_{13}\frac{\partial u_z}{\partial z}\right]+\frac{\partial}{\partial z}\left[C_{55}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\right]$	$+0$		$=0$
$\frac{\partial}{\partial x}\left[C_{55}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[C_{13}\frac{\partial u_x}{\partial x}+C_{33}\frac{\partial u_z}{\partial z}\right]$	$+0$		$=0$
$\frac{\partial}{\partial x}\left[\alpha M\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)\right]$	$+\frac{\partial}{\partial x}\left[M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]$	$-i\omega\frac{\eta}{k}w_x$	$=0$
$\frac{\partial}{\partial z}\left[\alpha M\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)\right]$	$+\frac{\partial}{\partial z}\left[M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]$	$-i\omega\frac{\eta}{k}w_z$	$=0,$

Table 3. Block representation of the initial matrix A, low-triangular preconditioner B, vectors x and y.

$$\mathrm{A}=\begin{array}{|c|c|}\hline A_0 & A_1\\ \hline A_2 & A_3\\ \hline\end{array}\quad \mathrm{B}=\begin{array}{|c|c|}\hline A_0 & 0\\ \hline A_2 & A_3\\ \hline\end{array}\quad \mathrm{x}=\begin{array}{|c|}\hline x_0\\ \hline x_3\\ \hline\end{array}\quad \mathrm{y}=\begin{array}{|c|}\hline y_0\\ \hline y_3\\ \hline\end{array}$$

Therefore the algorithm of the solution SLAE $Bx = y$ is:

1. Solve $A_0 x_0 = y_0$
2. Set $y_3 = y_3 - A_2 x_0$
3. Solve $A_3 x_3 = y_3$

Note that the matrices A_0 and A_2 are real and do not depend on the frequencies ω. The matrix A_3 has complex values only on the main diagonal. Pattern of these matrices and corresponding differential operators are presented in the Fig. (4) and Table (4).

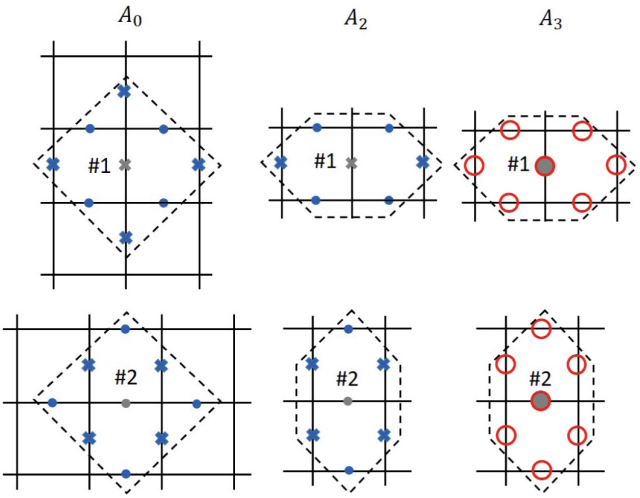

Fig. 4. Left: pattern for equation A_0; Center: pattern for equation A_2; Right: pattern for equation A_3;

Table 4. Differential equation of the "Field-Split" preconditioner B.

A_j block of B	Differential operator	number of A_j non-zeros
A_0	$\frac{\partial}{\partial x}\left[C_{11}\frac{\partial u_x}{\partial x}+C_{13}\frac{\partial u_z}{\partial z}\right]+\frac{\partial}{\partial z}\left[C_{55}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\right]$ $\frac{\partial}{\partial x}\left[C_{55}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[C_{13}\frac{\partial u_x}{\partial x}+C_{33}\frac{\partial u_z}{\partial z}\right]$	$2*9*n^2$
A_2	$\frac{\partial}{\partial x}\left[\alpha M\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)\right]$ $\frac{\partial}{\partial z}\left[\alpha M\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)\right]$	$2*7*n^2$
A_3	$\frac{\partial}{\partial x}\left[M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]-i\omega\frac{\eta}{k}w_x$ $\frac{\partial}{\partial z}\left[M\left(\frac{\partial w_x}{\partial x}+\frac{\partial w_z}{\partial z}\right)\right]-i\omega\frac{\eta}{k}w_z$	$2*7*n^2$

Let us now estimate the flops to invert the preconditioner: $Flops(invB) = Flops(invA_0) + 14n^2 + Flops(invA_3)$. The main benefit of inversion A_0 and A_3 compared to inversion of the full matrix A is that A_0 and A_3 are smaller than A, and A_0 does not depend on frequency and can be factorized only once in the case of inversion by direct solvers. The A_3 should be factorized for each frequency. Thus, the following estimation is valid:

$$Flops(invB) = [Flops(FctA_0)+Flops(SlvL_0)]+14n^2+[Flops(FctA_3)+Flops(SlvL_3)]$$

.

Estimates of $Flops(SlvL_0)$ and $Flops(SlvL_3)$ can be obtained from the number of non-zero elements of the factors L_0 and L_3: $Flops(SlvL) = 12Nnz(L)$. So, the total number of floating-point operations is:

$$Flops(Itr) = Flops(FctA_0)+$$
$$+N_{freq}[Flops(FctA_3) + N_{iter}[80N + 2 * 12Nnz(L_0) + 14n^2 + 2 * 12Nnz(L_3)]]$$

4 Numerical Experiments

As mentioned in the previous sections, the time taken by the solvers depends heavily on the number of floating-point operations. Additional functionality such as the approximation of Biot equation and the construction of the effective stiffness tensor C do not require high number of FLOPS, but can also affect the performance. The approximation requires a lot of data coping which makes this part of the algorithm "bandwidth-limited" and difficult to speed up. This operation is optimised by constructing the matrix one time for all frequencies ω_i and modifying only the diagonal for each ω_i. The construction of the stiffness tensor C is performed for each frequency, but it handles with small number of operations. The numerical experiments confirm the small time of execution of the mentioned functions in comparison with the numerical solution of SLAE. Moreover, we estimate parameters for counting FLOPS for both direct and iterative approaches (Table 3).

We considered Biot equations stated in a square domain. The model corresponds to the fractured porous media, as presented in Fig. (5). The coefficients of the Biot equations in fractures and in the background are provided in Table (5).

Table 5. Coefficients of Biot equations for numerical experiments.

Coeff	Background	Crack
C_{11}	$69.10 * 10^9$	$38.96 * 10^9$
C_{12}	$7.159 * 10^9$	$32.67 * 10^9$
C_{22}	$69.10 * 10^9$	$38.96 * 10^9$
C_{33}	$30.97 * 10^9$	$22.62 * 10^9$
M	$20.10 * 10^9$	$9.330 * 10^9$
αM	$5.953 * 10^9$	$6.854 * 10^9$
$\frac{\eta}{k}$	$1000 * 10^9$	$0.007 * 10^9$

Let us recall the estimates of the flops for the two approaches:

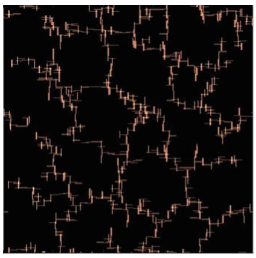

Fig. 5. Model crack.

- Direct solver: $Flops(Dir) = N_{freq}[Flops(FctA) + 12 * NNZ(L)]$,
- Iterative solver: $Flops(Itr) = Flops(FctA_0) + N_{freq}[Flops(FctA_3) + N_{iter}[80N + 2 * 12Nnz(L_0) + 14n^2 + 2 * 12Nnz(L_3)]]$.

Number of FLOPS for factorization and number of non-zero elements for various SLAE sizes N (various Biot mesh sizes n) are provided in the Table 6. The stopping criterion of BCGStab is based on achieving the relative residual $\|\hat{b} - \hat{A}x_0\|/\|\hat{b}\| < 10^{-10}$. It is enough to compute the effective tensor $\hat{C}$.

Table 6. Results of numerical experiments: Number of FLOPS, number of non-zero elements in L-factors and number of iterations.

Mesh size	$\hat{A}$ size	FLOPS number			Non-zero number*10^9			Iteration number
$n = n_x = n_z$	N	$FctA$	$FctA_0$	$FctA_3$	L	L_0	L_3	N_iter
250	$0.25 * 10^6$	67	13	3	0, 024	0, 007	0, 004	4
500	$1 * 10^6$	655	137	36	0, 116	0, 035	0, 019	4
1000	$4 * 10^6$	5312	1241	304	0, 539	0, 170	0, 091	4
2000	$16 * 10^6$	45119	11178	3445	2, 477	0, 770	0, 434	4
4000	$64 * 10^6$	N/A	93872	28881	N/A	3, 711	1, 965	4

First of all, we need to point out that BCGStab converged in only 4 iterations when using the field-split preconditioner. Number of flops to factorize the matrix A_0 is four times less than to factorize the entire matrix A. Factorization of the matrix A_3 requires even smaller number of flops and it is about 5% of that for the matrix A. Moreover, using the direct solver does not allow solving the problem of the size of 4000^2 points, while the iterative solver is suitable for such a problem. However, to estimate the total number of the flops for the iterative solver, we need to take into account the number of iterations to achieve the appropriate accuracy. Also, the necessity to solve Biot equations for a series of frequencies should also be considered. The number of floating-point operations estimation (for 30 frequencies) is shown in the Table 7. In general, the BCGStab iterative

solver with split-field preconditioner requires only 10% of the flops needed to solve the Biot equation with the direct solver.

Table 7. Total number of FLOPS for 30 frequencies.

Mesh	GFLOPS	
$n = n_x = n_z$	Direct	Iterative
250	2029	149
500	19690	1372
1000	159554	11159
2000	1354471	118158
4000	N/A	977291

5 Conclusion

We presented an iterative solver to solve the discretized Biot equation in quasi-static state. The solver is based on the BCGStab algorithms with field-split preconditioner. The idea of the approach is to use the block lower-triangular matrix as the preconditioner of the entire system. To invert the preconditioner, it is necessary to solve the static elastic equations and the Darcy equations (with respect to relative fluid displacements) independently. Solving the two independent problems is much simpler, than solving the coupled problem and requires fewer number of flops and less memory (RAM). In this research we used the direct solvers to invert the parts of the preconditioner, and compared the performance of the proposed algorithm with a purely direct solver of the entire system. We showed that the use of BCGStab reduces the computation time by the factor of ten compared to the direct solver. Indeed, using of the direct solver to invert the diagonal blocks of the preconditioner may not be the optimal choice, and our further research will focus on constructing appropriate iterative processes to deal with the preconditioner.

References

1. Alekseev, A.S., Mikhailenko, B.G.: Solution of dynamic problems of elastic wave propagation in inhomogeneous media by a combination of partial separation of variables and finite difference methods. Geophysics **48**, 161–172 (1980)
2. Amestoy, P., Duff, I.S., Koster, J., L'Excellent, J.Y.: A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM J. Matrix Anal. Appl. **23**(1), 15–41 (2001)
3. 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)
4. Biot, M.A.: Theory of propagation of elastic waves in fluid-saturated porous solid. I. low-frequency range. J. Acoust. Soc. Am. **28**, 168–178 (1956)

5. Bollhöfer, M., Schenk, O., Janalik, R., Hamm, S., Gullapalli, K.: State-of-the-art sparse direct solvers. In: Grama, A., Sameh, A.H. (eds.) Parallel Algorithms in Computational Science and Engineering. MSSET, pp. 3–33. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-43736-7_1
6. Calandrini, S., Aulisa, E., Ke, G.: A field-split preconditioning technique for fluid-structure interaction problems with applications in biomechanics. Int. J. Numer. Method Biomed. Eng. **36**(3), e3301 (2020). https://doi.org/10.1002/cnm.3301
7. 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)
8. 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)
9. Li, X.S.: An overview of SuperLU: algorithms, implementation, and user interface. ACM Trans. Math. Software **31**(3), 302–325 (2005)
10. Masson, Y.J., Pride, S.R., Nihei, K.T.: Finite difference modeling of Biot's Poroelastic equations at seismic frequencies. J. Geophys. Res.: Solid Earth **111**(B10), 305 (2006)
11. Quintal, B., Steeb, H., Frehner, M., Schmalholz, S.M.: Quasi-static finite-element modeling of seismic attenuation and dispersion due to wave-induced fluid flow in poroelastic media. J. Geophys. Res. **116**, B01201 (2011)
12. Solovyev, S., Novikov, M., Lisitsa, V.: Numerical solution of anisotropic biot equations in quasi-static state. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C. (eds.) Computational Science and Its Applications - ICCSA 2022 Workshops. Lecture Notes in Computer Science, vol. 13378, pp. 310–327. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10562-3_23
13. Stuben, K.: A review of algebraic multigrid. J. Comput. Appl. Math. **128**(1–2), 281–309 (2001)
14. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing, New York (1996)