Solve the Quasi-static Biot Problem on Shared Memory Systems with Various Processors

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

[1] Institute of Mathematics SB RAS, Koptug Avenue 4, Novosibirsk 630090, Russia
vevsa@ipgg.sbras.ru

[2] Institute of Petroleum Geology and Geophysics SB RAS, Koptug Avenue 3, Novosibirsk 630090, Russia
lisitsavv@ipgg.sbras.ru

Abstract. The quasi-static frequency domain Biot equation can be used to model loading of fluid-filled poroelastic materials in low-frequency case and to estimate effective frequency-dependent strain and stress relations. To numerically solve the Biot equation, we proposed an algorithm based on finite-difference approximation, iterative solution of the obtained system of linear algebraic equations (SLAE) and using field-split preconditioner technique. The basis of the iterative solver is the Biconjugate Gradient Stabilized (BCGStab); inverting the preconditioner is based on using the direct solver. In this paper, we present the results of implementation and testing the proposed algorithm on various shared memory systems. The experiments were performed on servers with Intel, AMD and Elbrus processors. The accuracy, memory and performance characteristics were measured. The results demonstrate robustness of the proposed algorithm on these architectures.

Keywords: Poroelasticity · Biot equation · quasi-static state · finite differences · direct methods for SLAE · iterative methods · field-split preconditioner · shared memory systems

1 Introduction

The numerical investigation of the effect of complex fracture systems on seismic waves in poroelastic fluid-saturated media is an actual problem, although studies on the theory of poroelasticity and the numerical solution of the Biot equations began in the 1950 s [4].

The object of research is the modelling of the Biot equation in the quasi-static state in the porous domain with fractures. When considering a percolating system of fractures, the size of the representative sample may exceed a hundred typical fracture sizes. It leads to grids with more than several thousand grid points

S. Solovyev— Developed the algorithm and 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-*П*. AMD and Elbrus hardware were provided by the USPARS development team.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
O. Gervasi et al. (Eds.): ICCSA 2024 Workshops, LNCS 14817, pp. 305–320, 2024.
https://doi.org/10.1007/978-3-031-65238-7_21

in each direction. Several numerical algorithms were presented [3,7,8]. They are based on the finite-element approximation of the quasi-static approximation of Biot equations with the mesh refinement within the fractures. However, these researhes were done for models containing only a few tens of fractures and requiring several thousand degrees of freedom for the simulation. In the study [11] we propose an algorithm for numerically upscaling of the poroelastic model for the samples containing thousands of fractures. The approach is based on the finite difference with the use of a direct solver to resolve the obtained system of linear algebraic equations with multiple righthand sides. This study of seismic attenuation was devoted to investigate the effect of fracture connectivity on WIFF and seismic attenuation. The problem of solving the system of linear algebraic equations (SLAE) is quite actual in Biot modeling. It involves computational cost problem; requires both knowledge base in mathematical algorithms for solving such systems and program implementation for high performance computing systems (HPC). The details of implementation the SLAE solution by both direct and iterative solvers with preconditioner are described in the paper [9].

The paper is structured as follows. In Sect. 2 we recall the statement of Biot equation, domain discretisation, stensils of finite-difference approximations. Section 3 is devoted to the description of the BCGStab-based iterative solver with Field-Split preconditioner. The short software implementation Sect. 4 contains information about the third-party software used. The main Sect. 5 deals with numerical experiments: three different hardware descriptions (Intel, AMD and Elbrus), visualisation of the results and characteristics of the computational process on selected architectures.

2 Poroelastic Problem in Terms of Biot Equations in Quasi-static State

In this section we describe the problem statement: differential Biot equations with boundary conditions, reconstructed effective frequency-dependent viscoelastic media, disretization and SLAE properties.

The Biot model equations stated in the quasi-static regime representing the pressure diffusion process in fluid-filled poroelastic media [2]. Table 1 presents the equations with Biot equations.

Table 1. Differential Biot equations.

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

The velocity and relative fluid velocity vectors are $\vec{u} = (u_x, u_z)^T$ and $\vec{w} = (w_x, w_z)^T$. Parameter ω is the time frequency, the fluid-saturated matrix stiffness tensor — are the coefficients $C_{11}, C_{33}, C_{13}, C_{55}$. The bone topology permeability k; the fluid-flow parametrs are M – the fluid storage coefficient, α is the Biot-Willis coefficient, η is the fluid viscousity.

Note, that the stress tensor is defined by the fourth-order stiffness tensor, the velocity vector derivatives and by αM combined with the divergence fluid velocity vector:

$$\begin{aligned}
\sigma_{xx} &= C_{11}\frac{\partial u_x}{\partial x} + C_{13}\frac{\partial u_z}{\partial z} + \alpha M\left(\frac{\partial w_x}{\partial x} + \frac{\partial w_z}{\partial z}\right) \\
\sigma_{zz} &= C_{13}\frac{\partial u_x}{\partial x} + C_{33}\frac{\partial u_z}{\partial z} + \alpha M\left(\frac{\partial w_x}{\partial x} + \frac{\partial w_z}{\partial z}\right) \\
\sigma_{xz} &= C_{55}\left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right).
\end{aligned} \tag{1}$$

The components of the strain tensor are defined as:

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

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

$$\begin{aligned}
\sigma \cdot \vec{n}|_{\partial D} &= \vec{\sigma}_0 \\
\vec{w} \cdot \vec{n}|_{\partial D} &= 0.
\end{aligned} \tag{3}$$

The statement of the poroelastic problem is the recovery of the frequency dependent stiffness tensor of orthotropic viscoelastic media (Fig. 1, right). The solution of the initial problem (1) and (3) in poroelastic media (Fig. 1, left) should match the solution of the upscaled problem:

$$\frac{\partial}{\partial x}\left[\hat{C}_{11}(\omega)\frac{\partial \hat{u}_x}{\partial x} + \hat{C}_{13}(\omega)\frac{\partial \hat{u}_z}{\partial z}\right] + \frac{\partial}{\partial z}\left[\hat{C}_{55}(\omega)\left(\frac{\partial \hat{u}_x}{\partial z} + \frac{\partial \hat{u}_z}{\partial x}\right)\right] = 0, \tag{4}$$

$$\frac{\partial}{\partial x}\left[\hat{C}_{55}(\omega)\left(\frac{\partial \hat{u}_x}{\partial z} + \frac{\partial \hat{u}_z}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\hat{C}_{13}(\omega)\frac{\partial \hat{u}_x}{\partial x} + \hat{C}_{33}(\omega)\frac{\partial \hat{u}_z}{\partial z}\right] = 0, \tag{5}$$

$$\hat{\sigma} \cdot \vec{n}|_{\partial D} = \vec{\sigma}_0 \tag{6}$$

Note, that the stress tensor of upscaled problem is defined by the fourth-order stiffness tensor and velocity vector derivatives:

$$\begin{aligned}\hat{\sigma}_{xx} &= \hat{C}_{11}\frac{\partial \hat{u}_x}{\partial x} + \hat{C}_{13}\frac{\partial \hat{u}_z}{\partial z}\\ \hat{\sigma}_{zz} &= \hat{C}_{13}\frac{\partial \hat{u}_x}{\partial x} + \hat{C}_{33}\frac{\partial \hat{u}_z}{\partial z}\\ \hat{\sigma}_{xz} &= \hat{C}_{55}(\frac{\partial \hat{u}_x}{\partial z} + \frac{\partial \hat{u}_z}{\partial x}).\end{aligned} \tag{7}$$

The strain tensor components of such problems are defined by the same formulae as for Biot (see 2).

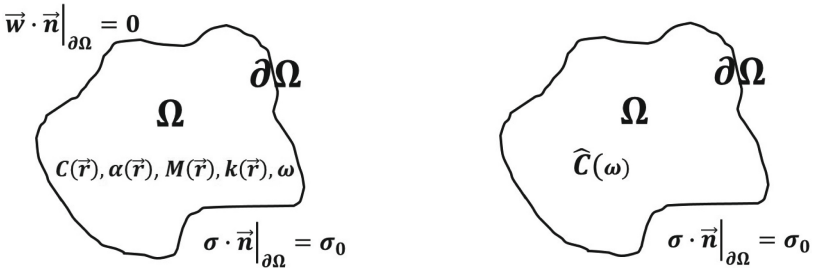


Fig. 1. Left: original fluid-saturated poroelastic media. Right: reconstructed effective frequency-dependent viscoelastic media. Vector $\vec{n}$ is the outer normal, and $C(\vec{r}), \alpha(\vec{r}), M(\vec{r}), k((\vec{r}))$ are the space-dependent parameters of Eqs. (1).

To solve the upscaling problem we need to solve the SLAE with the three right-hand sides generated by a system of equations in a square domain with three separate boundary conditions:

1. Horisontal loading: $\vec{\sigma}_{0_x x} = 1$ at left and right faces,
2. Vertical loading: $\vec{\sigma}_{0_z z} = 1$ at down and top faces,
3. Torsion loading: $\vec{\sigma}_{0_x z} = 1$ at all faces.

Details of the reconstruction algorithm are described in [12,13].

Note that apart from the solid and relative fluid velocity vectors ($\vec{u}$ and $\vec{w}$), the pressure p is useful property of the elastic problem. In poroelastic media it is:

$$-p = \alpha M\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_z}{\partial z}\right) + M\left(\frac{\partial w_x}{\partial x} + \frac{\partial w_z}{\partial z}\right). \tag{8}$$

A few remarks on homogeneous media. If the coefficients η and k don't depend on spatial coordinates, then Biot equations (1) in terms of two vector unknowns ($\vec{u}$ and $\vec{w}$) lead to equations with vector $\vec{u}$ and scalar p unknowns:

$$\frac{\partial}{\partial x}\left[C_{11}\frac{\partial u_x}{\partial x} + C_{13}\frac{\partial u_z}{\partial z} - \alpha^{\mathbf{2}}\mathbf{M}(\frac{\partial \mathbf{u_x}}{\partial \mathbf{z}} + \frac{\partial \mathbf{u_z}}{\partial \mathbf{x}}) - \alpha\mathbf{p}\right] + \frac{\partial}{\partial z}\left[C_{55}(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x})\right] = 0, \tag{9}$$

$$\frac{\partial}{\partial x}\left[C_{55}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[C_{13}\frac{\partial u_x}{\partial x}+C_{33}\frac{\partial u_z}{\partial z}-\alpha\mathbf{^2M}\left(\frac{\partial \mathbf{u_x}}{\partial \mathbf{z}}+\frac{\partial \mathbf{u_z}}{\partial \mathbf{x}}\right)-\alpha\mathbf{p}\right]=0, \tag{10}$$

$$i\omega\frac{\eta}{k}\alpha\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_z}{\partial z}\right)+\left[\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial z^2}+i\omega\frac{\eta}{k}\frac{1}{M}\right]\mathbf{p}=0, \tag{11}$$

The statement (9)–(11) in comparison with (1) leads to the SLAE with smaller size. But we are interested in the general case where all poroelastic coefficients C_{11}, C_{33}, C_{13}, C_{55}, M, α, η and k depend on spatial variables.

We therefore describe the approximation of the Biot equations (1), the properties of the SLAE and ways of numerical solutions.

2.1 Discretisation of Biot Equations

Biot equations (1) 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. The positions of the variables on the staged grid are given in the Fig. 2. The horizontal components of the displacement vectors u_x and w_x are defined at halfinteger points in the z direction and at integer points in the x direction. Visewersa, the vertical components of the solution vector are stored at integer points in the z direction and at halfinteger points in the x direction. The coefficients of the equations are defined at the centres

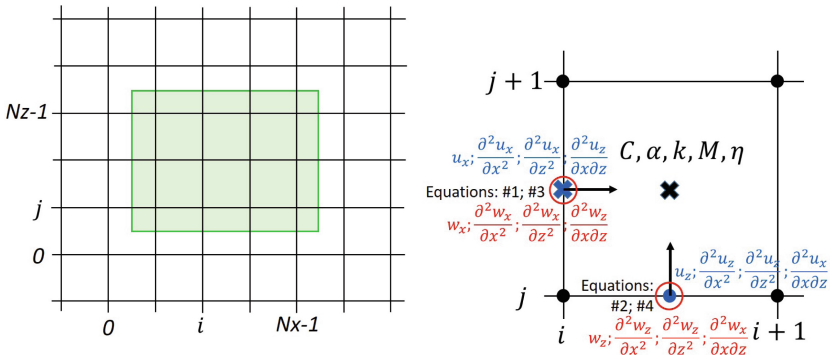


Fig. 2. Discretisation of the computational domain (left). Solution components, derivatives, location of physical parameters on the computational grid, and nodes where equations are approximated (right).

of the mesh cells (halfinteger points in both directions). The stencils used to approximate Biot's equations are shown in the Fig. (3). The details the finite difference approximation of Biot equations inside the computational domain and at its boundary (Fig. 2) are described in [10].

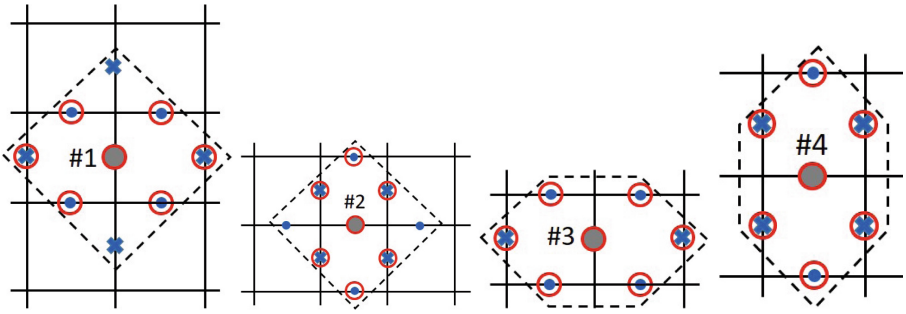


Fig. 3. Patterns of Biot equations.

Here, 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 (3), the matrix is ill-conditioned (singular). The matrix of SLAE can be represented in "blocked form"

$$A = \begin{array}{|c|c|}\hline A_0 & A_1 \\ \hline A_2 & A_3 \\ \hline \end{array}$$

where each block corresponds to the block of Biot equations (1). The diagonal block A_0 is the symmetric semi-positive definite with the real coefficient. The diagonal block A_3 is not symmetric because of the "mixed" boundary condition (3), i.e.

$\sigma_{xx} = C_{11}\frac{\partial u_x}{\partial x} + C_{13}\frac{\partial u_z}{\partial z} + \alpha M\left(\frac{\partial w_x}{\partial x} + \frac{\partial w_z}{\partial z}\right) = \phi_x$ at the left and rigth faces;

$\sigma_{zz} = C_{13}\frac{\partial u_x}{\partial x} + C_{33}\frac{\partial u_z}{\partial z} + \alpha M\left(\frac{\partial w_x}{\partial x} + \frac{\partial w_z}{\partial z}\right) = \phi_z$ at the top and bottom faces. Such properties are derived from matrix construction and are beyond the scope of this paper. In the next section we will construct a solver based on these properties.

3 Mixed Iterative-Direct SLAE Solver

During the development of numerical solvers for SLAE, several aspects need to be considered, such as **Accuracy**, i.e. the algorithm provides a suitable solution error and residual, **Performance** – speed property, usually it involves reducing the number of floating point operations (FLOP), which should be minimised. The

details of the implementation of the developed algorithms is also an important aspect (parallelization, use of HPC third party soft, etc.). The third requirement for the solver is **Memory usage** – ability to slightly increase memory usage during the solution process. This allows for a more precise solution of the Biot problem.

There are two *Direct* and *Iterative* approaches to numerically solve SLAE. The first provides the exact solution, but requires a lot of memory to store temporary data. The second one is not so demanding on memory consumption, but the quality of solution is vary sensitive to the matrix properties. Here we use the mixed approach - develop the Iterative solver with preconditioner that uses the direct solver to inverse its submatrices. Details of implementation and compare with the direct solver is presented in the paper [9].

3.1 BCGStab Based Iterative Solver

The main idea of iterative solvers is to refine of the rough solution step by step until the specified accuracy is reached. Often this process is diverges, so the preconditioner needs to be created for the initial SLAE.

Different preconditioners [1,5,6,14] are used to speed up the convergence rate. The preconditioner B gives the SLAE: $\hat{A}x = \hat{b}$ where $\hat{A} = B^{-1}Ax$, $\hat{b} = B^{-1}b$. The BICGSTAB algorithm [15] is used to solve the complex non-hermitian matrix $\hat{A}$:

$$
\begin{aligned}
&set\ x_0 \\
&set\ j_{restart} \\
&r_0 := \hat{b} - \hat{A}x_0 \\
&p_0 := r_0 \\
&For\ j = 0, 1, \ldots, until\ convergence\ Do: \\
&a_j := \frac{(r_j, r_0^*)}{(\hat{A}p_j, r_0^*)} \\
&s_j := r_j - a_j \hat{A} p_j \\
&\omega_j := \frac{(\hat{A}s_j, s_j)}{(\hat{A}s_j, \hat{A}s_j)} \\
&x_{j+1} := x_j + \alpha_j p_j + \omega_j s_j \\
&r_{j+1} := s_j + \omega_j \hat{A} s_j \\
&If\ remainder(j/j_{restart}) = 0\ Then\ r_{j+1} := \hat{b} - \hat{A}x_{j+1} \\
&\beta_j := \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \frac{\alpha_j}{\omega_j} \\
&p_{j+1} := r_{j+1} + \beta_j (p_j - \omega_j \hat{A} p_j) \\
&EndDo
\end{aligned}
\tag{12}
$$

The maximum computational complexity of the specified algorithm, i.e. the number of floating-point operations contained in the matrix $\hat{A}$ vector product (twice per step of BCGStab: $\hat{A}p_j$ and $\hat{A}s_j$). Each $\hat{A}$ vector product contains the matrix-vector product of the initial sparse matrix A (**mv**) and the inversion of the preconditioner B (**inv**). The B inversion is more expensive than the A vector product routine and is based on the use of the Field-Split preconditioning technique.

3.2 Field-Split Preconditioner

The preconditioner B should be invertible and as close as possible to the matrix of the SLAE as possible.

The reason for using Field-Split, the details of building such a preconditioner and the estimation of floating-point operations are described in the paper [9].

Here we just recall the block-low-triagonal representation of the preconditioner $B = \begin{array}{|c|c|}\hline A_0 & 0 \\ \hline A_2 & A_3 \\ \hline \end{array}$ where matrices A_0, A_2, A_3 raised from approximation the differential operators (Table 2) with boundary conditions (3).

Table 2. Differential operators of the preconditioner B.

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

The algorithm of invertion B consists of two matrix inversions A_0, A_3 and matrix A_2-vector product:

1. $x_0 = A_0^{-1} y_0$
2. $y_3 = y_3 - A_2 x_0$
3. $x_3 = A_3^{-1} y_3$

Matrix A_0 is a self-adjoint matrix with real coefficients. The A_3 has complex values on the main diagonal. The boundary conditions (3) lead to a non-symmetric of A_3.

4 Software Implementation

The Biot solver was developed in C language and used only one third party component — the direct sparse solver which is implemented in mathematical software. There are various optimized software on various architectures. The

well known highly optimised math library on Intel processors is MKL (Mathematical Kernel Library). It also shows excellent performance results on AMD architectures. The Intel MKL Pardiso component is the direct sparse solver used in Biot programme for both Intel and AMD. Special attention should be paid to the Elbrus architecture which is not supported by Intel MKL. However, like Intel MKL PARDISO, there is the USPARS direct sparse solver which has the similar functionality and is optimised for all these architectures Intel, AMD and Elbrus. Therefore, the use of USPARS was also considered in Biot programme. Some remarks on compilers: on Intel and AMD the Intel compiler was used. It takes full advantage of both Intel and AMD processors. On Elbrus system the own LCC compiler is used, specially developed for such processors. To develop and test the Biot programme, we use different servers with different modern processors, which are listed in the Table (3).

Table 3. Hardware for the numerical experiments.

	Intel(R) Xeon(R) Gold 6248R	AMD EPYC™ 7763	Elbrus 16S
web-site	ark.intel.com	www.amd.com	mcst.ru
Base Clock	3.00 GHz	2.45 GHz	2.00 GHz
1 core peak performance			
# of CPU Cores	24	64	16
RAM,G	768	1024	256

5 Numerical Experiments

There are two aspects to numerical experiments. The first is the visualisation of the physical fields resulting from the solution of the Biot equations. The second is the characteristics of the computational process on the mentioned architectures.

For the numerical experiments, we use the fracture model with interconnected cracks. The properties of the background and the cracks are given in the Table (4). The crack visualisation (material density) is shown in Fig. (4).

One of the main results is that the quality of the solution does not depend on the architecture used. Therefore, we do not clarify the architecture in the visualisation subsection.

Table 4. Properties of materials.

Coeff	Background	Crack
rho	$2.485 * 10^3$	$2.318 * 10^3$
C_{11}	$69.10 * 10^9$	$38.96 * 10^9$
C_{13}	$7.159 * 10^9$	$16.34 * 10^9$
C_{33}	$69.10 * 10^9$	$38.96 * 10^9$
C_{55}	$30.97 * 10^9$	$11.31 * 10^9$
M	$20.10 * 10^9$	$9.330 * 10^9$
αM	$5.953 * 10^9$	$6.854 * 10^9$
$\frac{\eta}{k}$	$1000 * 10^9$	$0.007 * 10^9$

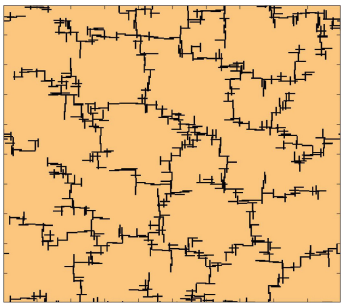


Fig. 4. Model crack.

5.1 Stress and Pressure Visualisation

The visualisation of the stress distribution for the frequency 10 Hz is shown in the Figs. (5), (6) and (7). In all of them, the lighter - the larger, the darker - the smaller value of the stress components.

The modelling results σ_{xx} and σ_{zz} of the **horizontal** loading are shown in Fig. (5) (σ_{xz} is many times smaller than the others, so it is not shown).

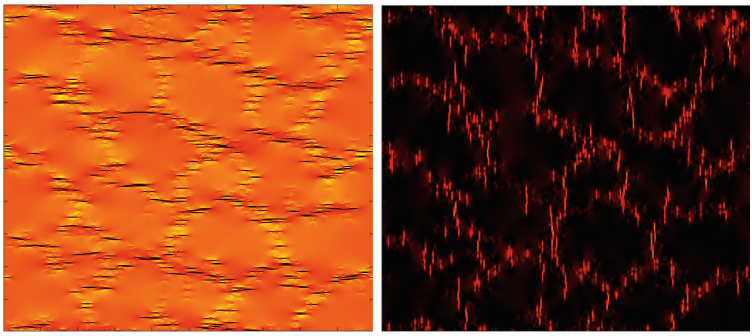


Fig. 5. Horizonal loading, stress component distribution: σ_{xx} on the left, σ_{zz} on the right.

The stress components of **vertical** loading are presented in Fig. (6) (σ_{xz} is also many times less than other, so it is not presented).

For the **torsion** loading the σ_{xz} is more than σ_{xx} and σ_{zz}. It is presented in Fig. (7).

The pressure p (see 8) is shown for different frequencies. For zero frequency, p is the same constant in the full domain for both horizontal and vertical loading. For torsional loading it is significantly less than others and is not considered. The inhomogeneous behaviour of p is observed as the frequency increases. The

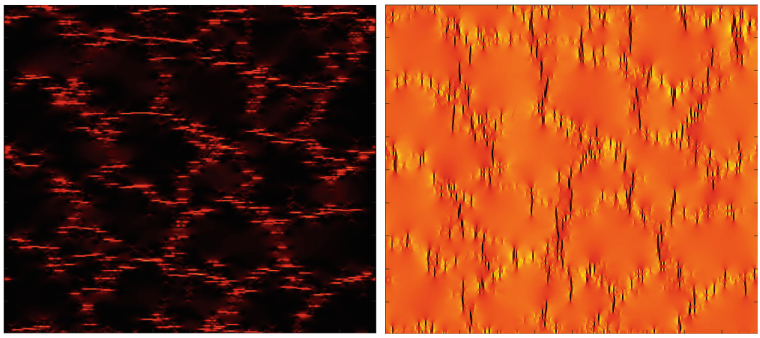


Fig. 6. Vertical loading, stress component distribution: σ_{xx} on the left, σ_{zz} on the right.

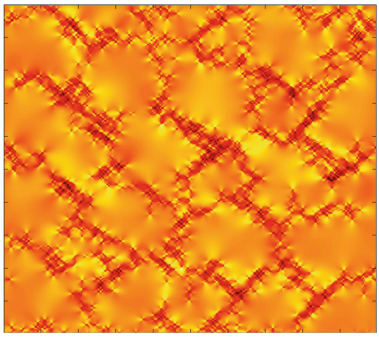


Fig. 7. Torsion loading, stress component σ_{xz} distribution.

Fig. 8 shows the p distribution for different frequencies. The average value of the pressure is the same as for the zero frequency. As the frequency increases, the pressure deviation gets closer and closer to the cracks.

5.2 Characteristics of the Computational Process

Here we are interested in two main characteristics: time and memory consumption. The implementation of Biot modelling contains main routines:

1. Read model parameters from file;
2. Approximation, i.e. construct matrix A, submatrices A_0,A_1,A_2,A_3 in CSR (Compress Sparse Row) format;
3. Solve SLAE $Ax = b$ with BCGStab including matrix-matrix multiplication and inversion preconditioner B;
4. Postprocessing: compute residual of SLAE, recover stiffness tensor.

The most expensive in terms of memory and performance is solving the SLAEs $A_0x_0 = b_0$ and $A_3x_3 = b_3$ with the inversion precondition B. As

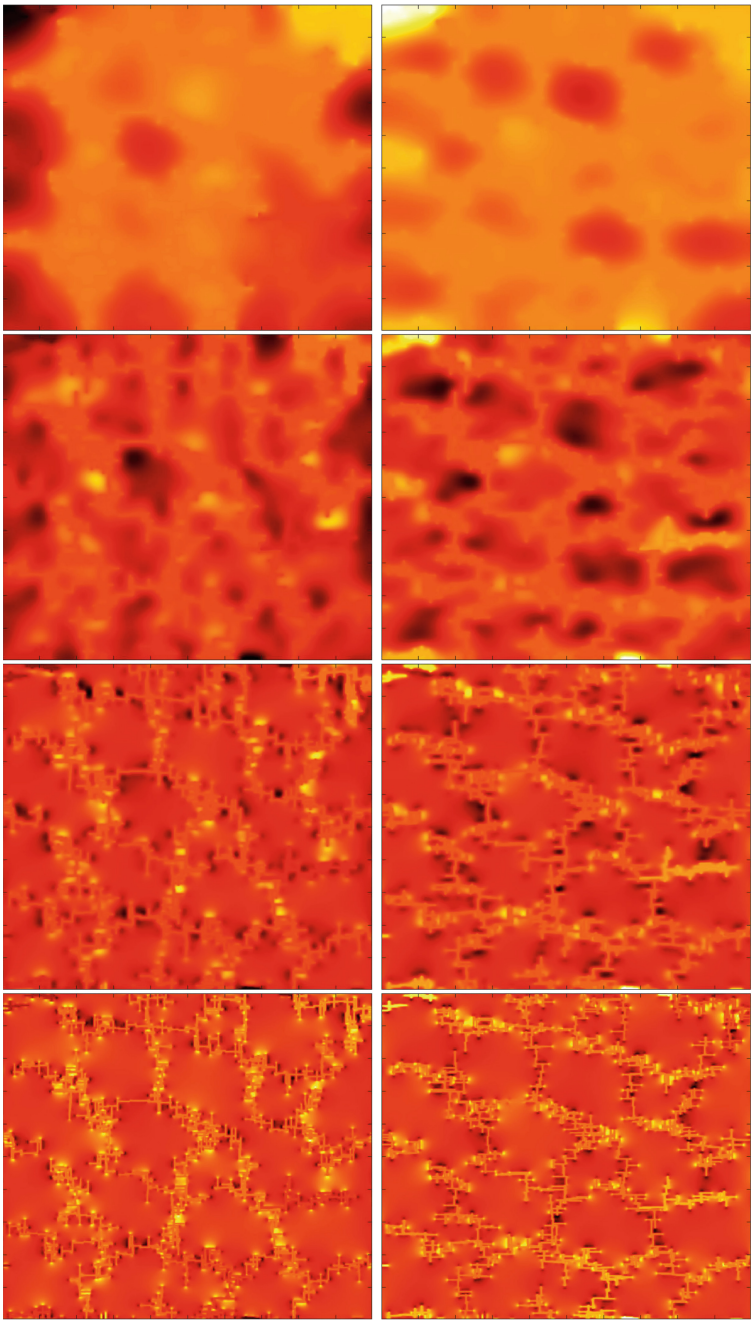


Fig. 8. Pressure for 1 Hz (top line), 10 Hz (2nd line), 100 Hz (3rd line) and 1000 Hz (bottom line); the **horizontal** load on the left, the **vertical** one on the right. The pressure for the **torsional** load is many times less than the others and is not shown.

mentioned above, this is done using highly optimised third party soft: sparse direct solvers.

First we are interested in the maximum size of the model that can be solved on systems (3). In the Table (5) we present the memory consumption of the Biot program for models of different sizes.

Table 5. Biot solver memory usage for various model sizes.

id	Mesh $n = n_x = n_z$	Problem size (# unknowns)	Peak memory usage, G
1	500	$1 * 10^6$	6
2	1000	$4 * 10^6$	25
3	2000	$16 * 10^6$	105
4	4000	$64 * 10^6$	452

The memory consumption is related to the problem size. It is predictable because the size of matrices A_0,A_1,A_2,A_3 grows linearly and the peak memory of sparse direct solvers for matrices from 2D problems has similar behaviour. This behaviour is confirmed numerically for both MKL PARDISO and USPARS solvers.

Another area of interest is timing. We measure the timing for 4 frequencies of each mesh. The Table (6) shows the total time, the percentage distribution between the treatment A_0, A_3 by the direct solver and other functions on different architectures (3). To avoid the effects of parallelization, the first results are obtained on one core of each processor.

Table 6. One core total time (sec.) for 4 frequencies, percentage of direct solvers and other functionality (**Direct/Other**).

Mesh	Intel		AMD		Elbrus
id	PARDISO	USPARS	PARDIO	USPARS	USPARS
1	116,3 (64%/36%)	122,1 (69%/31%)	79,4 (72%/28%)	85,5 (73%/27%)	261,3 (80%/20%)
2	464,1 (70%/30%)	474,3 (73%/27%)	355,4 (80%/20%)	354,0 (80%/20%)	1043,1 (82%/18%)
3	2 323,0 (78%/22%)	2 219,5 (77%/23%)	2 191,7 (87%/13%)	1 877,5 (85%/15%)	4986,0 (86%/14%)
4	15 156,6 (80%/20%)	11 431,9 (81%/19%)	14 925,0 (89%/11%)	11 670,8 (88%/12%)	N/A

Although the direct solvers are very expensive, the table shows that they are compatible with other routines. This can be explained by their high degree of optimisation. On the other hand, the proportion of direct solvers increases with the size of the problem.

The next Table (7) shows the multi-core potential of Biot solvers. The processors have different numbers of cores, so we measure the scalability on the same number of cores — 16.

Table 7. Many core (16) total time (sec.) for 4 frequencies, percentage of direct solvers and other functionality (**Direct/Other**).

Mesh	Intel		AMD		Elbrus
id	PARDISO	USPARS	PARDIO	USPARS	USPARS
1	71,8 (58%/42%)	74,3 (60%/40%)	42,9 (53%/47%)	50,8 (59%/41%)	184,4 (72%/28%)
2	249,8 (60%/40%)	252,1 (61%/39%)	144,4 (59%/41%)	173,2 (64%/36%)	686,0 (73%/27%)
3	1 066,1 (65%/35%)	1 037,7 (65%/35%)	647,7 (66%/34%)	712,2 (67%/33%)	2 805,0 (74%/26%)
4	6 422,8 (65%/35%)	4 546,6 (65%/35%)	3 676,2 (66%/34%)	3 515,7 (67%/33%)	N/A

Computing time for 16 cores is less than for one, but scalability is poor. The time contribution of the direct solvers has become smaller due to their good parallelization. The poor scalability of other functionalities can be explained by

Table 8. Maximum core total time (sec.) for 4 frequencies, percentage of direct solvers and other functionality (**Direct/Other**).

Mesh	Intel		AMD		Elbrus
Mesh	24 cores		64 cores		16 cores
id	PARDISO	USPARS	PARDIO	USPARS	USPARS
1	69,0 (58%/42%)	72,1 (60%/40%)	43,7 (52%/48%)	50,9 (59%/41%)	see Table 7
2	247,7 (60%/40%)	255,8 (61%/39%)	140,3 (57%/43%)	168,3 (63%/37%)	see Table 7
3	1 063,1 (61%/39%)	1 034,3 (62%/38%)	580,8 (60%/40%)	671,3 (65%/35%)	see Table 7
4	5 110,3 (66%/34%)	5 514,4 (62%/38%)	3 305,7 (62%/38%)	3 172,1 (64%/36%)	see Table 7

the existence of a large number of different functionalities, which makes their parallelization very difficult. On the other hand, it is the Biot solver that has the potential for improvement.

The third measurement was made on the maximum number of cores for each processor. The Table (8) shows that there is no significant reduction in computing time. This is due to poor scalability. However, the absolute times for all processors and for each of the direct solvers (PARDISO and USPARS) are reasonable.

6 Conclusion

We presented results of the implementation of the algorithm for solving the discretised Biot equation in quasi-static state based on the BCGStab iterative solver with field-split preconditioner. The aim of the research was to investigate the behaviour of the algorithm on different architectures such as Intel, AMD and Elbrus. In addition to our own implementation of the main functionality, the third party libraries were used to solve SLAE with preconditioner blocks. Similar results with good solution quality were obtained on different architectures. The characteristics of the computational process such as number of iterations, memory consumption were also similar. Although the solution time varies on different hardware, it does not increase dramatically with model size. The scalability is far from ideal and the algorithm has potential for improvement. However, this is not critical as the largest Biot problem can be solved in a reasonable time. The main conclusion is that the proposed Biot solver can be successfully used for poroelasticity modelling on hardware systems with different processors.

References

1. 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)
2. Biot, M.A.: Theory of propagation of elastic waves in fluid-saturated porous solid. i. low-frequency range. J. Acoustical Soc. America **28**, 168–178 (1956)
3. Carcione, J.M., Santos, J.E., Picotti, S.: Anisotropic poroelasticity and wave-induced fluid flow: harmonic finite-element simulations. Geophys. J. Int. **186**(3), 1245–1254 (2011)
4. Carcione, J.M., Morency, C., Santos, J.E.: Computational poroelasticity – a review. Geophysics **75**(5), 75A229-75A243 (2010)
5. 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)
6. 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)
7. 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)

8. Rubino, J.G., Caspari, E., Müller, T.M., Milani, M., Barbosa, N.D., Holliger, K.: Numerical upscaling in 2-d heterogeneous poroelastic rocks: anisotropic attenuation and dispersion of seismic waves. J. Geophys. Res. Solid Earth **121**(9), 6698–6721 (2016)
9. Solovyev, S., Lisitsa, V.: Field-split iterative solver vs direct one forquasi-static biot equation. In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds.) Supercomputing, pp. 86–99. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-49432-1_7
10. 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, pp. 310–327. Springer International Publishing (2022). https://doi.org/10.1007/978-3-031-10562-3_23
11. Solovyev, S., Novikov, M., Lisitsa, V.: A numerical investigation of wave-induced fluid flows in anisotropic fractured porous media. Comput. Math. Appli. **140**, 78–88 (2023). https://doi.org/10.1016/j.camwa.2023.03.013, https://www.sciencedirect.com/science/article/pii/S0898122123001098
12. Solovyev, S., Novikov, M., Lisitsa, V.V.: Numerical solution of anisotropic biot equations in quasi-static state. In: Computational Science and Its Applications - ICCSA 2022 Workshops - Malaga, Spain, July 4-7, 2022, Proceedings, Part II, pp. 310–327 (2022). https://doi.org/10.1007/978-3-031-10562-3_23
13. Solovyev, S., Novikov, M., Lisitsa, V.: Numerical solution of anisotropic biot equations of poroelastic fluid-saturated media in quasi-static state for numerical upscaling. Numerical Methods Program.(Vychislitelnye Metody i Programmirovanie) **24**(75), 67–88 (2023). https://doi.org/10.26089/NumMet.v24r106, https://en.num-meth.ru/index.php/journal/article/view/1257
14. Stuben, K.: A review of algebraic multigrid. J. Comput. Appl. Math. **128**(1–2), 281–309 (2001)
15. Saad, Y.: Iterative Methods for Sparse Linear Systems. PWS Publishing, New York (1996)