Preconditioning aspects of large scale micro finite element analysis of 3D bone poroelasticity

Author(s):
Turan, Erhan; Arbenz, Peter

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-007577595

Rights / License:
In Copyright - Non-Commercial Use Permitted
Preconditioning aspects of large scale micro finite element analysis of 3D bone poroelasticity

Erhan Turan, Peter Arbenz*

ETH Zürich, Department of Computer Science, CH-8092 Zürich, Switzerland

Abstract

In this paper, a solver for poroelasticity problems is discussed that is related to osteoporotic human bones. Osteoporosis is a major health problem that compromises the integrity of bones. A good understanding of the disease requires an accurate simulation of the physics. For that purpose, a finite element solver based on Biot’s consolidation has been developed. A mixed formulation is used to discretize the geometries taken from medical imaging. The resulting indefinite linear systems are solved by Krylov space methods supplemented by variants of Schur complement-based block preconditioners.

Keywords: poroelasticity, mixed finite elements, flexible GMRES, preconditioning, symmetric indefinite systems

1. Introduction

In poroelasticity, the deformation of a porous structure filled with a fluid is studied. The basics of the theory has first been investigated by Terzaghi in the late 1920’s. The foundations of poroelasticity, on the other hand, are due to Biot [1]. In his theory, a porous Hookean solid is coupled with the pressure of the containing fluid using Darcy’s law. To date, several other methodologies are proposed to investigate poroelasticity, like the mixture theory [2] and also recently the microscopic theory and multiscale approaches [3, 4]. Still, Biot’s consolidation equations are applied in different areas like groundwater flow, petroleum engineering, biomechanics, geothermal problems [5].

This study is motivated by a disease called osteoporosis. Osteoporosis is one of the major health problems affecting around 200 million people worldwide, 8.9 million of which suffer from a fracture [6]. 37.5% of all patients live in Europe, Japan, and the United States which account for only 11% of the world population. This is due to the fact that the life expectancy in these countries is high and that the disease affects mostly elderly people, especially women. As

*Corresponding author

Email addresses: turane@gmail.com (Erhan Turan), arbenz@inf.ethz.ch (Peter Arbenz)
a matter of fact, according to the WHO, the risk for an osteoporotic fracture in women above 50 years is about 50%, for men the risk is about 20%. For instance, over 2 million fractures caused by osteoporosis cost nearly $17 billion in 2005, in the United States alone [7]. This issue is acknowledged by the European Union to support a project called VPHOP that has the objective to detect patient specific osteoporosis potential [8]. To contribute to the efforts, a solver called ParFE [9, 10] has been developed to analyze linear elasticity on bone structures. Since the models are generated by CT scans, the code is adapted for voxel-based meshes. ParFE is used to assess bone remodeling which examines the changes within the bones that is exposed to external cyclic load. There are several proposals to formulate bone remodeling. For example, Adachi et al. [11] used an idea based on the calculation of wall shear stresses at the lacuno-canaliculare level of trabecular bone. However, the computations are performed using bone poroelasticity [12] hence our objective is to extend ParFE for poroelastic problems.

There are two common methodologies to analyze Biot’s consolidation numerically. In the \(u-p\) formulation, the equation set is reduced by eliminating the specific velocity. Although extending ParFE by a piecewise linear pressure \(Q_1-Q_1\) is straightforward, a standard Galerkin approach leads to oscillatory pressure fields [13]. One remedy is the use of a higher order approximation for \(u\) than for \(p\), like \(Q_2-Q_1\). One limitation for this stable pair is the memory consumption because of the second order approximation. Additionally, the number of finite elements cannot be reduced since the geometry is approximated quite inaccurately anyway using a CT-based voxel mesh. Therefore, we conclude that higher order elements do not improve the accuracy. Several strategies are proposed to stabilize the \(u-p\) formulation by complementing the original set of equations by additional terms [14, 13, 15]. The main drawback of this approach is that the stability parameters have to be adjusted which normally requires additional numerical experiments that depends on the mesh and materials.

The poroelasticity equations can alternatively be discretized by the \(u-f-p\) formulation that is employed in this study, see Section 3. This approach treats the flux, \(f\), as a primary variable. The methodology is free of instabilities and ensures continuity across element faces [16]. Additionally, it provides a basis for a smooth transition to more advanced treatments that considers Stokes flow, for instance.

<table>
<thead>
<tr>
<th>(n^3) Elements</th>
<th>(u-p)</th>
<th>(u-f-p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>local dof</td>
<td>(32 = 3 \times 8 + 1 \times 8)</td>
<td>(31 = 3 \times 8 + 1 \times 6 + 1)</td>
</tr>
<tr>
<td>total dof</td>
<td>(4(n+1)^3)</td>
<td>(3(n+1)^3 + 3n^2(n+1) + n^3)</td>
</tr>
<tr>
<td>local nonzeros</td>
<td>808</td>
<td>457</td>
</tr>
<tr>
<td>total nonzeros</td>
<td>(\approx 297n^3)</td>
<td>(\approx 229n^3)</td>
</tr>
</tbody>
</table>

Table 1: Comparison of number of degrees of freedom for \(u-p\) and \(u-f-p\) formulations for a \(n \times n \times n\) grid of a non-porous, i.e., full domain.

The \(u-f-p\) formulation requires to store more unknowns than the \(u-p\) formu-
lation, cf. Table 1. However, the total number of nonzero entries in the system matrices favor the $u-f-p$ formulation.

The presence of the specific velocity requires a mixed element formulation. The resulting saddle point problems require particular attention regarding iterative solution methods and preconditioners, see [17] for a survey. There are even preconditioners that are specially adapted to geomechanics problems [18]. In this study, we used block-diagonal and block-upper-triangular preconditioners with the same structure as the original system.

Poroelasticity is a time-dependent phenomenon requiring a transient solver. In bone remodeling, in particular, the geometry adapts to the loading to lower excessive stresses. In a voxel-based simulation areas of high stress are “fortified” by additional voxels, while voxels in areas of low stress are discarded. We do not investigate these techniques here but restrict ourselves to iterative solvers in unaltered geometries. The motivation for this study was to create a framework that allows fast and efficient solutions of a sequence of similar linear systems. Later, it will serve as a starting point for a dedicated bone remodeler.

The paper is organized as follows: In the next section, the governing equations of poroelasticity are introduced along with physical parameters. In Sections 3 and 4, the discretization of the equations and their parallel numerical solution is discussed. In section 5, a validation study is performed. In the same section, several model problems are investigated. Test cases are selected from different artificial porous structures as well as from real bone geometries. Finally, the work is summarized and some suggestions are made for additional studies.

2. Mathematical modeling

In order to model a poroelastic material, a set of three equations is needed. These are (i) the equilibrium equation to model elastic deformation, (ii) Darcy’s law to model fluid flow, and (iii) mass conservation [19]. For a linear isotropic material, the equilibrium equation can be stated as

$$\nabla \cdot (2\mu \varepsilon(u(x,t))) + \nabla (\lambda \nabla \cdot u(x,t)) - \alpha \nabla p(x,t) + F(x,t) = 0,$$

where $u$ is the displacement, $\varepsilon$ is the linearized strain tensor,

$$\varepsilon(u) = \frac{1}{2} (\nabla u + (\nabla u)^T),$$

$p$ is the pressure, $F$ is the external force, $\lambda$ and $\mu$ are the Lamé parameters, and $\alpha$ is the Biot–Willis coefficient. The Lamé parameters are related to the Young’s modulus $E$ and the Poisson ratio $\nu$ by

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} = \frac{2\mu\nu}{(1-\nu)}.$$

Darcy’s law,

$$f = -\frac{k}{\eta} \nabla p,$$

$$f = -\frac{k}{\eta} \nabla p,$$
states that the fluid velocity $f$ is proportional to the gradient of the pressure which is defined in excess of the hydrostatic pressure. Here, $k$ is the permeability and $\eta$ is the dynamic viscosity. Mass conservation can be written as

$$\alpha \frac{\partial}{\partial t} \nabla \cdot \mathbf{u} + S_\varepsilon \frac{\partial p}{\partial t} + \nabla \cdot f = S_f,$$

where the specific storage at constant strain $S_\varepsilon$ is a measure of released fluid volume per unit pressure in the control volume. $S_f$ is an external time-dependent source or sink.

The boundary conditions are given by

$$\mathbf{u}(x,t) = \mathbf{u}_D, \quad \text{on } \partial \Omega_D,$$

$$\sigma(x,t) \mathbf{n}(x) = t(x,t), \quad \text{on } \partial \Omega_N,$$

$$f(x,t) \cdot \mathbf{n}(x) = 0, \quad \text{on } \partial \Omega.$$

We approximate the temporal derivatives in (3) by the implicit Euler method with time step $\tau$,

$$\alpha \nabla \cdot \mathbf{u}(x,t) + S_\varepsilon p(x,t) + \tau \nabla \cdot f(x,t) = \tau S_f + \alpha \nabla \cdot \mathbf{u}(x,t-\tau) + S_\varepsilon p(x,t-\tau).$$

We impose the initial conditions

$$\mathbf{u}(x,0) = \mathbf{u}_0, \quad p(x,0) = p_0, \quad x \in \Omega.$$

In numerical tests, the initial conditions might be assumed to be zero which satisfy the original set of equations. Or, $\mathbf{u}_0$ and $p_0$ can be calculated from the loading like detailed in Section 5.1.

Although the system of equations is closed with (1)–(5), some additional formulae are needed to compute the material properties. The calculation of the constrained storage coefficient $S_\varepsilon$ requires special attention. We can use,

$$S_\varepsilon = \frac{1}{K'_s} \left( 1 - \frac{K}{K'_s} \right) + \phi \left( \frac{1}{K_f} - \frac{1}{K_\phi} \right)$$

where the porosity $\phi$ and the bulk moduli of the phases are used. In (6), $K'_s$ is the unjacketed bulk modulus, $K_f$ is the fluid bulk modulus, and $K_\phi$ is the unjacketed pore bulk modulus [19]. Material properties tabulated by Wang [19] are computed with the assumption\(^1\) that the solid-grain modulus $K_s$ is equal to both $K'_s$ and $K_\phi$.

3. Numerical methods

The governing equations are discretized with the $u$-$f$-$p$ formulation using mixed finite elements. Each of the main variables is treated as a primary variable.

\(^1\)Conditions for this assumption to hold are given by Detournay and Cheng [20].
of the discrete problem. In this approach, the full set of equations (1)–(3) is used. To employ the finite element method we need weak formulations of these equations. The weak formulation of Biot’s model of poroelasticity uses the displacements \( u \in (H^1(\Omega))^3 \), the fluid flux \( f \in H(\text{div}; \Omega) \), and the pressure \( p \in L^2(\Omega) \) [21] with the appropriate boundary conditions (4). The equations

\[
\int_{\Omega} [2\mu\varepsilon(u) : \varepsilon(v) + \lambda \nabla \cdot u \nabla \cdot v] \, d\Omega - \alpha \int_{\Omega} p \nabla \cdot v \, d\Omega = 0, \\
\int_{\Omega} K^{-1} f \cdot g \, d\Omega - \int_{\Omega} p \nabla \cdot g \, d\Omega = 0, \\
-\alpha \int_{\Omega} q \nabla \cdot u \, d\Omega - \int_{\Omega} q \nabla \cdot f \, d\Omega - S_e \int_{\Omega} p q \, d\Omega = - \int_{\Omega} S_f q \, d\Omega,
\]

have to hold for all \( v \in (H^1(\Omega))^3 \), \( g \in H(\text{div}; \Omega) \), and \( q \in L^2(\Omega) \) satisfying homogeneous boundary conditions (4). \( S_f \) includes both a source term and transient terms [21].

For the finite element discretization we assume that the connected domain \( \Omega \) is obtained by a 3-dimensional CT scan and is composed of voxels. In the above \( u-f-p \) formulation, we approximate each of the three displacement components by piecewise trilinear polynomials \((Q_1)\), and the pressure by piecewise constant polynomials \((P_0)\). The flux is approximated by Raviart–Thomas \((RT_0)\) elements that have a continuous normal component across element interfaces [22]. The tangential components can jump. The displacements are determined by their values at the element vertices, the pressure can be represented by its value at the element center. The \( RT_0 \) fluxes have a constant normal component on each element face. Hence, they can be represented by the normal components at the center of element faces. In a voxel, there are 24 degrees of freedom for the displacements, 6 degrees of freedom for the fluxes, and one degree of freedom for the pressure.

The advantage of this formulation over some other approaches like \( u-p \), is that the resulting discrete problem is stable and does not need additional terms to provide a stable numerical solution [16]. Because of the same reason the original set of governing equations are kept intact. Also, since there is no stability term, there is no need to perform computational experiments to find an optimized stability parameter. Even if the total number of unknowns is higher than with the \( u-p \) formulation, fewer nonzeros are generated in the global matrix. This is more decisive when subblocks of the global stiffness matrix are to be stored. Additionally, flux values are continuous across element boundaries as stated by Ferronato et al. [16] which secures mass conservation. Having available the full set of primary variables, the improvement of the model to Stokes flow can be realized with little difficulty since the fluxes are kept as primary variables. This approach avoids the calculation of the flux values in post-processing as well.

The finite element discretization of the \( u-f-p \) formulation (7)–(9) leads to
the linear system

\[
\mathcal{A} \mathbf{x} = \begin{bmatrix}
A_{uu} & O & A_{pu}^T \\
O & A_{ff} & A_{pf}^T \\
A_{pu} & A_{pf} & -A_{pp}
\end{bmatrix}
\begin{bmatrix}
u \\
f \\
p
\end{bmatrix} =
\begin{bmatrix}
0 \\
0 \\
b
\end{bmatrix}.
\]

(10)

With appropriate boundary conditions, the diagonal blocks \(A_{uu}, A_{ff},\) and \(A_{pp}\) are symmetric positive definite, such that the whole \(3 \times 3\) block matrix \(\mathcal{A}\) is symmetric indefinite. Note that \(A_{pp}\) is diagonal. Also \(A_{ff}\) has a simple structure. Since the \(RT_0\) vector basis functions on cubical finite elements have only one nonzero component, \(A_{ff}\) consists of essentially 1-dimensional, i.e., tridiagonal pieces.

There are Krylov space methods designed for solving symmetric indefinite systems, most notably MINRES and SYMMLQ [23]. In this study we use the generalized minimal residual method, GMRES [24] which is more stable than the mentioned solvers [25]. The potentially excessive memory consumption of GMRES is eliminated with restarts.

A crucial ingredient for the success of any Krylov space method is the preconditioner. In this work, we use two types of preconditioners, a block-diagonal and a block upper-triangular preconditioner [26],

\[
\mathcal{M}_0 := \begin{bmatrix}
A_{uu} & A_{ff} \\
A_{ff} & S_{pp}
\end{bmatrix},
\quad \mathcal{M}_1 := \begin{bmatrix}
A_{uu} & O & A_{pu}^T \\
O & A_{ff} & A_{pf}^T \\
O & O & S_{pp}
\end{bmatrix},
\]

(11)

with the negative Schur complement

\[
S_{pp} = A_{pp} + A_{pf} A^{-1}_{ff} A_{pf}^T + A_{pu} A^{-1}_{uu} A_{pu}^T.
\]

(12)

Both preconditioners, \(\mathcal{M}_0\) and \(\mathcal{M}_1\), are infeasible. Therefore we make the following simplifications: (i) \(A_{uu}\) is replaced by a V-cycle of an aggregation-based algebraic multigrid preconditioner; (ii) \(A_{ff}\) is replaced by its diagonal \(D_{ff}\); and (iii) \(S_{pp}\) is approximated by the spectrally-equivalent [21]

\[
\tilde{S}_{pp} = A_{pp} + A_{pf} D_{ff}^{-1} A_{pf}^T.
\]

(13)

In our particular setting, \(\tilde{S}_{pp}\) is equivalent to the 7-point Laplacian for finite differences [26] that is replaced by a V-cycle of an aggregation-based algebraic multigrid preconditioner.

4. Parallelization aspects

Our code, PorFE [27], is written in C++ using software from the Trilinos project [28, 29]. The foundation of Trilinos is the Epetra package which allows to define and build parallel matrices and vectors. Epetra hides the MPI communication behind sophisticated C++ constructs. Flexible GMRES is implemented in the package Belos which is a templated library of basic Krylov solvers. Belos
also provides Krylov subspace recycling that will be helpful in the time dependent problems. The ML package provides smoothed aggregation-based algebraic multilevel preconditioners.

As in ParFE [10] we employ the graph partitioner ParMETIS [30] to distribute the data. However, in PorFE the nodes of the graph not only relate to displacements (at grid points) but also to flux values (at face centers) and pressure values (at voxel centers). The graph partitioner should combine degrees of freedom that are geometrically close. Nodes corresponding to displacements get a weight three to reflect the number of degrees of freedom at grid points. (In our numerical experiments we observed that the weighted graph lead to slightly better performance than the unweighted graph.) An initial partitioning is obtained based on the coordinates of the degrees of freedoms. It is used in the parallel construction of the graph. ParMETIS then generates a refined partitioning for the distribution of numerical quantities.

Since the matrix-vector products are performed blocks-wise, unique partitions are extracted from the global distribution for each primitive variable. Hence, even if a segregated approach is followed for each individual block, the global partitioning is calculated by all variables combined. If the graph size, i.e. the number of indices exceed the limits of Trilinos or ParMETIS, then the partitioning follows the original recursive coordinate bisection.

In addition to the connectivity list (element-to-node table) used in the elasticity problem, two additional tables are needed to form the system matrix: an element-to-face table and an element-neighbor list. Both tables have six columns. They can be easily created from the initial CT image.

5. Results and discussion

We performed several numerical experiments on the validity of the solver as well as on the performance of different preconditioning options and the efficiency of the parallel computations. In these computations, Trilinos version 10.10.3, ParMETIS version 3.2.0 and hdf5 version 1.8.5 are used. The simulations are performed on compute nodes consisting of 2 AMD Interlagos processors each having 16 cores. A node has a memory of 32GB.

The simulations are performed on AMD Interlagos processors which have 16 cores. Each compute node has a memory of 32GB and 2 processors.

5.1. Benchmark problem

There are a number of test problems used in geomechanics to check the validity of the solver. Here, we used Terzaghi’s consolidation problem for which there exists an analytical solution. The geometry of the model is depicted in Fig. 1. The solutions for $u$ and $p$ are given by Wang [19].

![Loading on the body](Figure 1: Loading on the body)
In this test problem, the medium is assumed to be incompressible, hence $\alpha$ is set to unity. Material properties listed in Table 2 are taken similar to those used in [16] and the references therein.

As observed from Table 3, the number of unknowns scales approximately with the factor 8.

### Table 2: Material properties for benchmark problems

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
<th>parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>40.0 MPa</td>
<td>$k$</td>
<td>$1.02 \times 10^{-6}$ mm$^2$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>40.0 MPa</td>
<td>$\eta$</td>
<td>$1.0 \times 10^{-9}$ MPa s</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>1</td>
<td>$S_\epsilon$</td>
<td>$1.65 \times 10^{-4}$ MPa$^{-1}$</td>
</tr>
</tbody>
</table>

### Table 3: Test meshes for the first benchmark problem

<table>
<thead>
<tr>
<th>mesh_id</th>
<th>elements</th>
<th>nodes</th>
<th>faces</th>
<th>total dof</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>30</td>
<td>124</td>
<td>151</td>
<td>553</td>
</tr>
<tr>
<td>$b_2$</td>
<td>240</td>
<td>529</td>
<td>964</td>
<td>2 891</td>
</tr>
<tr>
<td>$b_4$</td>
<td>1 920</td>
<td>3 025</td>
<td>6 736</td>
<td>17 731</td>
</tr>
<tr>
<td>$b_8$</td>
<td>15 360</td>
<td>19 521</td>
<td>49 984</td>
<td>123 907</td>
</tr>
<tr>
<td>$b_{16}$</td>
<td>122 880</td>
<td>139 009</td>
<td>384 256</td>
<td>924 163</td>
</tr>
<tr>
<td>$b_{32}$</td>
<td>983 040</td>
<td>1 046 529</td>
<td>3 011 584</td>
<td>7 134 211</td>
</tr>
<tr>
<td>$b_{64}$</td>
<td>7 864 320</td>
<td>8 116 225</td>
<td>23 842 816</td>
<td>56 055 811</td>
</tr>
<tr>
<td>$b_{128}$</td>
<td>62 914 560</td>
<td>63 918 081</td>
<td>189 743 104</td>
<td>444 411 904</td>
</tr>
</tbody>
</table>

The tests are performed using right-preconditioned FGMRES(100) with a relative tolerance of $10^{-5}$ on the first time step. Our analysis is started employing the block-diagonal preconditioner (D). The number of iterations required for a converged solution at this initial step is summarized in Table 4. As seen from the table, there are high iteration counts (~80) for most of the meshes. In order to reduce the number of iterations, the preconditioner should be improved. For that reason, block triangular preconditioner (T) is also tested on the same model problems. This time, the inner blocks will be solved using PCG with predefined tolerances as explained in Section 3. Here, the convergence criterion for $A_{ff}$ and $S_{pp}$ are kept equal to the outer tolerance, $10^{-5}$, and for $A_{uu}$ block, $10^{-8}$ is enforced for a reasonably accurate solution. Referring to Table 4 again, we can state that there is a significant reduction in the number of iterations; nearly half of the iterations are enough for convergence. Unfortunately, when solution times are considered for each of the three cases, $b_{1,32}$, $b_{1,64}$, and $b_{1,128}$, we observe that the (D) preconditioner much faster. This is expected since we are now performing extra iterations for each block which increase the work load. Like the block diagonal preconditioner, block triangular preconditioner also scales well as depicted in Fig. 3. Still, some simplifications should be made in order to reduce the clock time for (T). In the following part, we will discuss a number of
adaptations that can be applied on the block-triangular preconditioner.

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th>b₁</th>
<th>b₂</th>
<th>b₄</th>
<th>b₈</th>
<th>b₁₆</th>
<th>b₃₂</th>
<th>b₆₄</th>
<th>b₁₂₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonal (D)</td>
<td>58</td>
<td>81</td>
<td>86</td>
<td>86</td>
<td>84</td>
<td>74</td>
<td>74</td>
<td>81</td>
</tr>
<tr>
<td>Triangular (T)</td>
<td>22</td>
<td>31</td>
<td>34</td>
<td>36</td>
<td>38</td>
<td>39</td>
<td>39</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 4: Number of outer iterations

Effect of inner iterations

One idea might be to relax the necessity to get a high accuracy solution for inner block solve of \( A_{uu} \) either by increasing the tolerance of convergence or limiting the maximum number of inner iterations per block. For that reason, we first examined the convergence behavior of PorFE for higher tolerances, namely \( 10^{-5} \) for every block including \( A_{uu} \). Table 5 compared with Table 4 will reveal the fact that now the solver requires more inner iterations to converge though in less time. Also, computations on \( A_{ff} \) shows a rise in the number of the iteration if simulated in parallel. In order to explain this behavior a short discussion on the structure of \( A_{ff} \) is necessary. Normally in our discrete model, the IC(0) preconditioner created for the \( A_{ff} \) block does not create additional fill-in’s simple because the factorization forms an exact inverse of \( A_{ff} \). That’s why convergence is achieved in one step when simulated in single core. However,
Table 5: Total number of inner iterations (with averages) for each subblock.

<table>
<thead>
<tr>
<th>mesh_id</th>
<th># cores</th>
<th>$A_{uu}$</th>
<th>$A_{ff}$</th>
<th>$S_{pp}$</th>
<th>outer</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_2</td>
<td>1</td>
<td>462 (7.0)</td>
<td>66 (1.0)</td>
<td>205 (3.1)</td>
<td>66</td>
</tr>
<tr>
<td>b_2</td>
<td>4</td>
<td>462 (7.0)</td>
<td>453 (6.9)</td>
<td>220 (3.3)</td>
<td>66</td>
</tr>
<tr>
<td>b_4</td>
<td>1</td>
<td>441 (7.0)</td>
<td>63 (1.0)</td>
<td>248 (3.9)</td>
<td>63</td>
</tr>
<tr>
<td>b_4</td>
<td>4</td>
<td>432 (7.1)</td>
<td>414 (6.8)</td>
<td>290 (4.8)</td>
<td>61</td>
</tr>
<tr>
<td>b_8</td>
<td>1</td>
<td>392 (7.0)</td>
<td>56 (1.0)</td>
<td>286 (5.1)</td>
<td>56</td>
</tr>
<tr>
<td>b_8</td>
<td>4</td>
<td>375 (7.2)</td>
<td>325 (6.3)</td>
<td>255 (4.9)</td>
<td>52</td>
</tr>
<tr>
<td>b_16</td>
<td>4</td>
<td>362 (7.4)</td>
<td>284 (5.8)</td>
<td>325 (6.6)</td>
<td>49</td>
</tr>
<tr>
<td>b_16</td>
<td>8</td>
<td>375 (7.7)</td>
<td>303 (6.2)</td>
<td>400 (8.7)</td>
<td>49</td>
</tr>
</tbody>
</table>

as the number of processors increases, a block variant of IC is used with one-level overlap that results in more inner iterations. We should note that, in parallel simulations, even though the number of average iterations for $A_{uu}$ (~7) and $A_{ff}$ (~6.5) stays nearly constant, there is an increase in the number of iterations on the Schur block, $S_{pp}$. This could be a starting point for another adaptation on the block-triangular preconditioner: the maximum number of iterations per block could be restricted to reduce the run time while keeping the outer iterations as constant as possible. This opportunity should be taken as an advantage because the block triangular preconditioner is still inferior against (D) and there is still some room for extra changes in terms of the solution parameters.

At first, we carried out additional computations to analyze the performance of the solver for a fixed number of $S_{pp}$ iterations. As stated in Table 6, for a maximum number of 3 iterations per outer iteration, we still get a similar
convergence behavior attained before. One should note that the reason for

<table>
<thead>
<tr>
<th>mesh_id</th>
<th># cores</th>
<th>$A_{uu}$</th>
<th>$A_{ff}$</th>
<th>$S_{pp}$</th>
<th>outer</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_2</td>
<td>4</td>
<td>462 (7.0)</td>
<td>453 (6.9)</td>
<td>195 (2.95)</td>
<td>66</td>
</tr>
<tr>
<td>b_4</td>
<td>4</td>
<td>432 (7.1)</td>
<td>414 (6.8)</td>
<td>180 (2.95)</td>
<td>61</td>
</tr>
<tr>
<td>b_8</td>
<td>4</td>
<td>375 (7.2)</td>
<td>325 (6.3)</td>
<td>153 (2.94)</td>
<td>52</td>
</tr>
<tr>
<td>b_16</td>
<td>4</td>
<td>364 (7.4)</td>
<td>284 (5.8)</td>
<td>144 (2.94)</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 6: Total (average) number of inner iterations for fixed $S_{pp}$ iterations

average values on $S_{pp}$ being smaller than 3 is that the first outer step requires no
iteration on Schur block. On top of this, if we perform an additional limitation
for the number of iterations (5) on $A_{ff}$ block, we get a convergence performance
given in Table 7. Even if an optimized setup of the inner tolerances and the

<table>
<thead>
<tr>
<th>mesh_id</th>
<th># cores</th>
<th>$A_{uu}$</th>
<th>$A_{ff}$</th>
<th>$S_{pp}$</th>
<th>outer</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_2</td>
<td>4</td>
<td>462 (7.0)</td>
<td>326 (4.9)</td>
<td>195 (2.95)</td>
<td>66</td>
</tr>
<tr>
<td>b_4</td>
<td>4</td>
<td>432 (7.1)</td>
<td>300 (4.9)</td>
<td>180 (2.95)</td>
<td>61</td>
</tr>
<tr>
<td>b_8</td>
<td>4</td>
<td>375 (7.2)</td>
<td>254 (4.9)</td>
<td>153 (2.94)</td>
<td>52</td>
</tr>
<tr>
<td>b_16</td>
<td>4</td>
<td>364 (7.4)</td>
<td>239 (4.9)</td>
<td>144 (2.94)</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 7: Total (average) number of inner iterations for fixed $A_{ff}$ and $S_{pp}$ iterations

number of iterations might be varying for a given problem, we are able to
propose to set the iteration limit to 10, 5, and 3 for the three blocks, respectively
and the relative inner tolerance as $10^{-5}$. With these new parameters, a fixed
block-triangular preconditioner (FT) is defined which at the end performs as in
Table 8.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonal (D)</td>
<td>25.93</td>
<td>20.98</td>
<td>19.21</td>
<td>41.34</td>
</tr>
<tr>
<td>Triangular (T)</td>
<td>82.01</td>
<td>91.7</td>
<td>117.4</td>
<td>282.8</td>
</tr>
</tbody>
</table>

Table 8: Comparison of solution times in seconds for selected number of [cores].

Previous modifications on the solver parameters improved the convergence
of the solver when the block-triangular preconditioner is applied all of which
are more evident that the set of parameters are more useful in larger problems.
Still, these changes are not enough to beat the block-diagonal preconditioner
in solution time. In fact, these results are another clue for us to suggest a new
kind of a preconditioner that lies between those two: A block-triangular variant
without PCG. In other words, a resemblance with the implementation of the
block diagonal will be achieved where upper off-diagonal blocks, namely $A_{pu}^T$
and $A_{pf}^T$ will also be included in the calculations. Although the CG solution
will be removed from the preconditioner, multilevel solver, ML will be kept in
the system and only Jacobi preconditioning will be applied for \( A_{ff} \). When we perform same tests on this simpler preconditioner, it can be stated that this new variant is slightly faster than the block-diagonal preconditioner as examined in Figure 4.

![Figure 4: Scaling comparison of block-diagonal and block-triangular preconditioners.](image)

The number of iterations averaged across different numbers of cores are given in Figure 5. As observed from the figure, the number of iterations for simpler
block-triangular preconditioner lies between the block-diagonal preconditioner and the original block-triangular preconditioner. Yet, the performance of the solver is improved in terms of solution time. Another aspect is that the average run time per outer iteration is nearly constant for preconditioners D and ST. The values for mesh b_{64} are listed in Table 9. That means the computations related to off-diagonal are also kept well-scaled, hence the proposed repartitioning is useful. All said, these two approached can be used to analyze different meshes as realized in the next sections.

<table>
<thead>
<tr>
<th># cores</th>
<th>216</th>
<th>324</th>
<th>648</th>
<th>1296</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>0.800</td>
<td>0.537</td>
<td>0.271</td>
<td>0.158</td>
</tr>
<tr>
<td>ST</td>
<td>0.800</td>
<td>0.546</td>
<td>0.274</td>
<td>0.159</td>
</tr>
</tbody>
</table>

Table 9: Solution time per iteration in seconds.

5.1.1. Transient performance

Since poroelasticity is a transient phenomenon, we should also test for unsteady performance of the solver. As mentioned before, first order backward Euler method is used for discretization purposes. In order to test PorFE, a simulation is performed in time till steady state is reached, which is defined as the time where the fluid content is completely removed and the displacement matches with pure linear elastic solution. An analytical calculation shows that the final tip displacement should be equal to $1.25 \times 10^{-3}$ mm. So, the point of time where this value is reached is announced as steady state. In Fig. 6, for each of 36000 time steps the number of iterations are given for problem b1_2. After the initial stages of the simulation the number of iteration steps varies between 1 and 2 with an average value of 1.5 steps.

5.2. Artificial domains

Before starting to deal with real bone structures, we performed studies on computer generated porous domains using the tools in ParOsol [31]. An attempt is made using Menger [32] sponges which are recursive porous structures. Secondly, we created trabecular-like bone structures where the domains are formed using Perlin noise. For comparison purposes, full domains are also generated that matches the sizes of the porous problems.

5.2.1. Full cubes

In following sections, we’ll show the results of different geometries in which a consistent pattern for porous mesh setups will be used. We start the analysis using a set of full domains i.e domains with no holes as shown in Figure 7 at which for each mesh f_{n}, a cubic volume of $n^3$ spanned. Mesh information is given in Table 10. Both preconditioners are examined in this case.

Similar to previously studied benchmark domains, the block-triangular preconditioner requires the least amount of iterations but in most amount of time. For fixed block-triangular variant, the inner iterations for various cases are given
in Table 11. For \( f_{243} \), Fixed version (FT) computes one outer iteration in 2.33 s whereas the original form (T) performs the same in 3.69 seconds leading us to the fact that original form is 50% slower. Again in this model, simpler variant (ST) and block-diagonal preconditioner (D) are too fast against the previous two. The speedup graph in Figure 11 verifies that the run times of them are comparable where (D) is a bit faster.
Table 10: Mesh information for full domains

<table>
<thead>
<tr>
<th>mesh_id</th>
<th>elements</th>
<th>nodes</th>
<th>faces</th>
<th>total dof</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_9</td>
<td>729</td>
<td>1 000</td>
<td>2 430</td>
<td>6 159</td>
</tr>
<tr>
<td>f_27</td>
<td>19 683</td>
<td>21 952</td>
<td>61 236</td>
<td>146 775</td>
</tr>
<tr>
<td>f_81</td>
<td>531 441</td>
<td>551 368</td>
<td>1 614 006</td>
<td>3 799 551</td>
</tr>
<tr>
<td>f_243</td>
<td>14 348 907</td>
<td>14 526 784</td>
<td>43 223 868</td>
<td>101 153 127</td>
</tr>
</tbody>
</table>

Table 11: Total number of inner iterations on full cubes with (FT)

<table>
<thead>
<tr>
<th>mesh_id</th>
<th># cores</th>
<th>$A_{uu}$</th>
<th>$A_{ff}$</th>
<th>$S_{pp}$</th>
<th>solve time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_27</td>
<td>2</td>
<td>91</td>
<td>47</td>
<td>36</td>
<td>13 (6.26)</td>
</tr>
<tr>
<td>f_81</td>
<td>64</td>
<td>103</td>
<td>68</td>
<td>36</td>
<td>13 (11.99)</td>
</tr>
<tr>
<td>f_243</td>
<td>2048</td>
<td>204</td>
<td>96</td>
<td>48</td>
<td>17 (39.67)</td>
</tr>
</tbody>
</table>

5.2.2. Menger sponges

A Menger sponge is a fractal structure proposed by Menger [32]. The process of creating 3D structures starts with a base image which is later mirrored recursively up to a certain level. In this example, a mesh that is composed of $3 \times 3$ elements is selected as the initial geometry and reflected up to 5 times to create domains as displayed in Fig. 8. The overall volume occupied by the geometry is the same of the full cubes analyzed before. One important property of Menger cubes is that they are both porous and geometrically symmetric. Mesh sizes are given in Table 12 with varying porosity levels. The number of faces are nearly two-times for then the nodes.

Table 12: Mesh information for Menger Sponges

<table>
<thead>
<tr>
<th>mesh_id</th>
<th>elements</th>
<th>nodes</th>
<th>faces</th>
<th>total dof</th>
<th>porosity</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_9</td>
<td>400</td>
<td>896</td>
<td>1 728</td>
<td>4 816</td>
<td>~0.55</td>
</tr>
<tr>
<td>m_27</td>
<td>8 000</td>
<td>15 616</td>
<td>33 024</td>
<td>87 872</td>
<td>~0.41</td>
</tr>
<tr>
<td>m_81</td>
<td>160 000</td>
<td>295 808</td>
<td>648 192</td>
<td>1 695 616</td>
<td>~0.30</td>
</tr>
<tr>
<td>m_243</td>
<td>3 200 000</td>
<td>5 789 440</td>
<td>12 865 536</td>
<td>33 433 856</td>
<td>~0.22</td>
</tr>
</tbody>
</table>

When we test PorFE in this example, we observe the very same trend as before. Both (D) and (ST) are faster and (FT) is faster then (T) even if the iterations are limited as summarized in Table 13. On m_243, one outer iteration requires 4.29 s and 2.6 s for (T) and (FT), respectively. The performance of (D) and (ST) is compared again in Figure 11. This time, however, simpler block-triangular is faster in every test run; 10.96 s vs. 9.14 s for m_243 on 512 cores.

5.2.3. Pseudo bones

We also created some trabecular like artificial bones using Perlin noise based generator included by ParOSol. The main advantage is that the meshes can be refined as if the medical scanning of a sample is performed in higher resolution.
Table 13: Total number of inner iterations on Menger cubes with (FT)

Consequently, a constant level of porosity can be obtained as seen in Table 14. This time, number of faces - number of nodes ratio is around 2.5 but all faces are still amounts 1/3 of all unknowns.

Table 14: Pseudo domains

Some sample results are presented on Table 15 for fixed block-triangular
preconditioner. Here we also observe a counter example where use of limited variables performs poorly for $p_{320}$ on 256 cores. One cure is to increase the maximum number of iterations for Schur block to 10 which seems to be working acceptably fast. This is a case where an end for inner iterations should be pronounced with inner tolerances rather than number of iterations.

<table>
<thead>
<tr>
<th>mesh_id</th>
<th>$A_{nn}$</th>
<th>$A_{ff}$</th>
<th>$S_{pp}$</th>
<th>outer (solve time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_{080} [4]</td>
<td>205</td>
<td>68</td>
<td>51</td>
<td>18 (42.11 s)</td>
</tr>
<tr>
<td>p_{160} [32]</td>
<td>285</td>
<td>95</td>
<td>69</td>
<td>24 (87.44)</td>
</tr>
<tr>
<td>p_{320} [256]</td>
<td>1100</td>
<td>345</td>
<td>207</td>
<td>70 (324 s)</td>
</tr>
<tr>
<td>p_{320} [256]</td>
<td>217</td>
<td>64</td>
<td>130</td>
<td>14 (66.09 s)</td>
</tr>
<tr>
<td>p_{640} [2048]</td>
<td>517</td>
<td>132</td>
<td>220</td>
<td>23 (192.7 s)</td>
</tr>
</tbody>
</table>

Table 15: Total number of inner iterations for pseudo bones using (FT)

5.3. Bone structure

After testing many domains of various sizes, we performed several computations on a bone sample. Here, we selected a wrist model depicted in Fig. 10, see [33]. This geometry, with voxels of 50 µm, is rather a complex structure which pushes the boundaries of PorFE.

<table>
<thead>
<tr>
<th>mesh_id</th>
<th>elements</th>
<th>nodes</th>
<th>faces</th>
<th>total dof</th>
</tr>
</thead>
<tbody>
<tr>
<td>w_i</td>
<td>9 013 446</td>
<td>12 178 452</td>
<td>30 063 142</td>
<td>75 611 944</td>
</tr>
</tbody>
</table>

Table 16: Wrist meshes info

There are several comments to be made for this model. Firstly, the block-diagonal preconditioner (D) is always the fastest way of preconditioning the
<table>
<thead>
<tr>
<th># cores</th>
<th>D</th>
<th>T</th>
<th>FT</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>56.1</td>
<td>95.4</td>
<td>101.3</td>
<td>62.86</td>
</tr>
<tr>
<td>750</td>
<td>40.0</td>
<td>76.5</td>
<td>88.5</td>
<td>44.45</td>
</tr>
<tr>
<td>1000</td>
<td>37.93</td>
<td>63.63</td>
<td>63.99</td>
<td>46.37</td>
</tr>
<tr>
<td>1250</td>
<td>33.12</td>
<td>53.39</td>
<td>51.92</td>
<td>35.85</td>
</tr>
<tr>
<td>1500</td>
<td>29.36</td>
<td>53.04</td>
<td>49.02</td>
<td>33.35</td>
</tr>
</tbody>
</table>

Table 17: Comparison of the preconditioners for \( v_1 \) – run time in seconds and (iteration count).

Secondly, the simple variant of the block-triangular preconditioner (ST) performs poorly compared to all previous cases and stays behind (D). The number of iterations for block triangular preconditioners with PCG spans equally across varying number of cores in terms of the iteration count. The fixed version cures itself only after 1250 cores, which suggest to use more inner iterations for smaller number of cores. Except a deviation of (ST) on 1K cores, the speedup performance is well as observed in Figure 11.

![Speedup for different meshes](image)

**Figure 11: Speedup for different meshes**

6. Conclusion

In this work, the performance of a poroelastic solver is investigated which is developed for large scale micro finite element analysis. Mixed finite element method is applied to discretize the problems using the \( u/f/p \) formulation. Displacements are approximated using trilinear voxel elements whereas flux values
are discretized with lowest order Raviart–Thomas elements. The pressure is assumed piecewise constant. Two types of preconditioners are introduced to improve convergence, a block-diagonal and a block-triangular. An approximation is made for the Schur on the pressure block at which the contributions from the displacement coupling is neglected and only the diagonal values of the flux-block is considered. Multilevel preconditioners are employed separately to each diagonal block. Furthermore, PCG is employed in the block-triangular preconditioner to improve the accuracy of the inner solves. First the solver is validated successfully in time against benchmark problems with a known analytic solution. Strong scaling tests are performed and it is observed the solver scales well. When both preconditioners are compared we can conclude that the triangular variant is optimal in the sense that the iteration count is independent on the problem size. On the contrary, the block-diagonal preconditioner, even not optimal, is faster in solution time. In order to reduce the solve time of the triangular preconditioner, several numerical experiments are performed on the limit of inner iterations. It is observed that, for full domains, smaller number of maximum iterations are enough to achieve convergence in same amount of outer iterations but in shorter time. In porous domains, however, the outer iterations are susceptible to inner iterations where a convergence mandated with inner tolerances are more favorable. Later, a simpler variant of the block-triangular preconditioner is employed where the PCG contribution is neglected. This adaptation of the preconditioner is able to cope with fast run times of the block-diagonal preconditioner and even faster in some problems. After the solver is tested on various problems, its performance is also tested on a real hand wrist sample which has a reasonable complex geometry and seen that the solver acts well.

Future work on large scale bone poroelasticity includes the implementation of bone remodeling where the adaptation of the bone structure to external loading is considered. In remodeling, the mesh geometry changes. This affects (i) the problem size and (ii) the connectivity of both nodes and faces. It will be a big challenge to update the affected data structures efficiently. Furthermore, in the context of the solver, (iii) the initial data and (iv) the preconditioner have to be modified to take into account the geometric changes. The envisaged computational overhead might be alleviated by remodeling only after some number of time steps. To simplify the geometric representation of the domain a larger than necessary domain of voxels may be generated. Individual voxel are turned on or off by varying material properties ranging from 0 to real values [34].

Finally, the poroelastic solver can be extended to cover nonlinear effects to examine large deformations and fracture and further, non-isothermal features can be implemented which is useful to analyze geothermal systems.

Acknowledgments

This project has been supported by the Swiss National Science Foundation grant #205320_125114 and by grant #223865 - VPHOP in the 7th RTD framework programme of the European Commission. The computations have been
performed in the framework of a Large User Project grant of the Swiss National Supercomputing Centre (CSCS).

References


