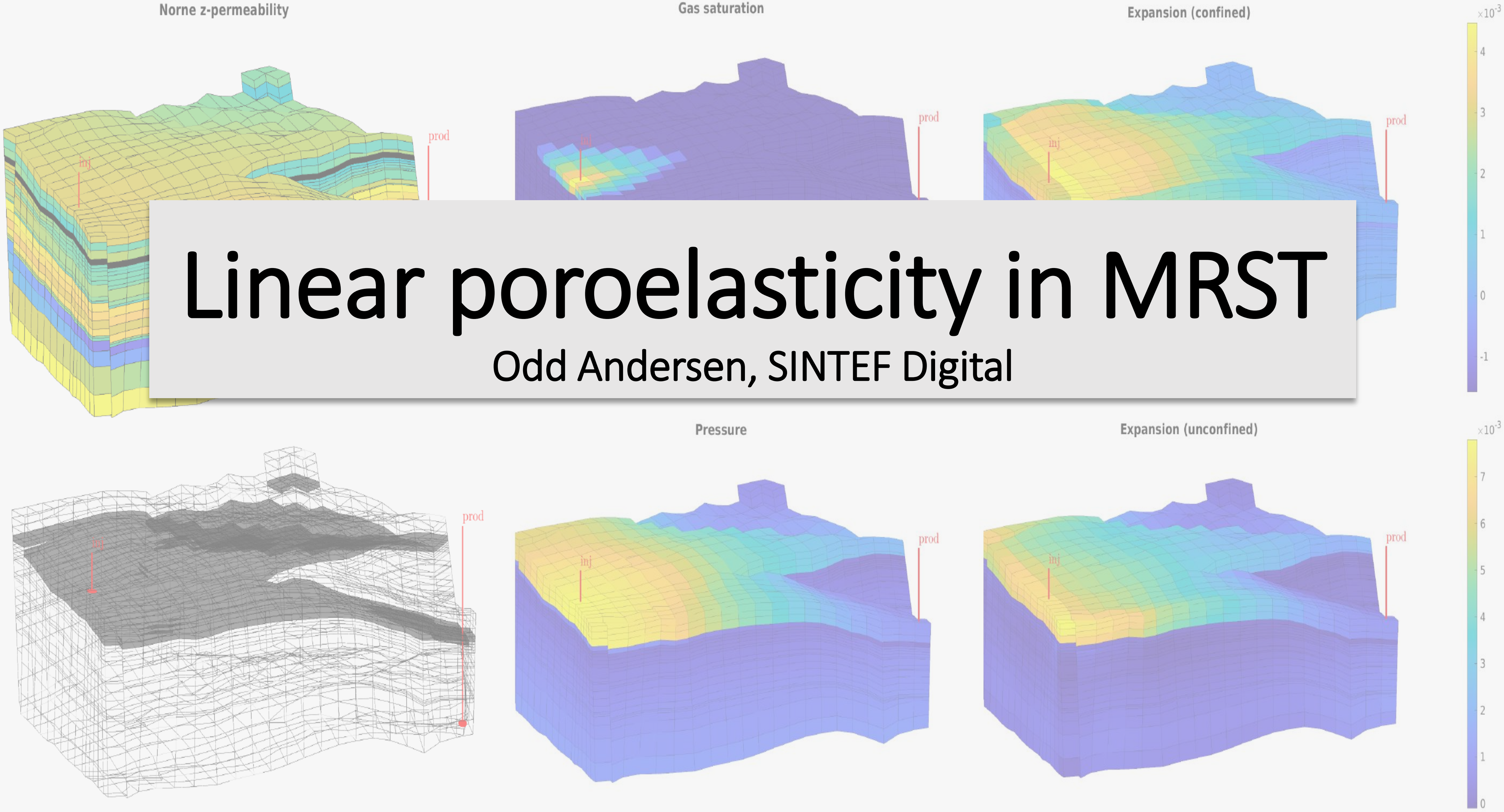
Linear poroelasticity in MRST

Odd Andersen, SINTEF Digital



Linear elasticity: the MRST `vemmech` module

- The linear theory allows for the computation of (small) displacements, stresses and strains as a response to external loads and body forces.
- In MRST, the function `VEM_linElast` computes the mechanical deformation.
- The problem is specified in terms of the spatial MRST grid, boundary conditions (forces/constraints) and body forces.
- The **virtual element method** is used in the discretization -> allows use on "difficult" grids.

The linear elastic equations:

$$\mathbf{F} = \nabla \vec{u}$$

'u' represents the spatial (infinitesimal) displacement field from an initial configuration. F is the deformation gradient tensor.

$$\epsilon = \frac{1}{2}(\mathbf{F} + \mathbf{F}^T) = \frac{1}{2}\left(\nabla \vec{u} + (\nabla \vec{u})^T\right)$$

The infinitesimal strain tensor ϵ is defined as the symmetric part of F.

$$\boldsymbol{\sigma}(x) = \mathbf{C}(x) \cdot \boldsymbol{\epsilon}(x)$$

Generalized Hooke's law specifies a linear relationship between the strain (ϵ) and stress (σ) tensor. The linear relationship is described by the rank-4 stiffness tensor C. For "simple" materials, C can be uniquely specified in terms of two elastic parameters, such as Young's modulus and Poisson's ratio.

$$\nabla \cdot \boldsymbol{\sigma} + \vec{b} = 0$$

The static balance of forces is expressed in terms of the divergence of the strain tensor, and the body force vector 'b'. In addition, boundary conditions must be specified.

The `VEM_linElast` function:

Stiffness tensor (six coefs. per cell in grid).

```
u = VEM_linElast(G, C, el_bc, load);
```

computed displacement field

MRST spatial grid to define shape and subdivision of domain.

Boundary conditions and body force data structures.

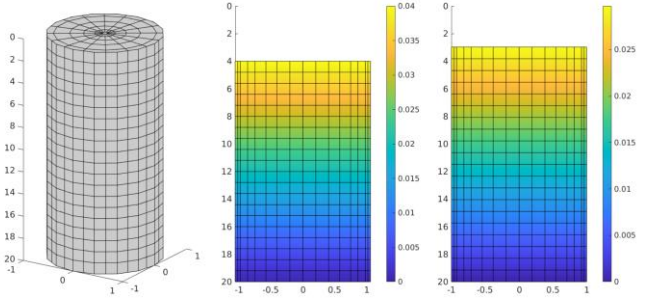
Example: uniaxial compression of cylinder

- We apply a compressional load along the axis of a cylinder
- The amount of compression depends on whether lateral boundary conditions are free or laterally constrained ("roller")
- The ratio of lateral expansion to axial compression is defined by Poisson's parameter. This simple exercise may thus be used as a basic validation of the numerical scheme.

The cylinder geometry is represented by the grid 'G', and boundary conditions (uniaxial compression) by the 'el_bc' object. In the code snippet on the right, we define the stiffness matrix using the 'Enu2C' command, then call 'VEM_linElast' to compute displacements.

```
% solve the linear elastic system
C = Enu2C(E * ones(Nc, 1), nu * ones(Nc, 1), G);
uu = VEM_linElast(G, C, el_bc, load);
```

The cylinder grid and the resulting deformation is here shown (side view) for two different choices lateral boundary conditions. In the first case (middle), lateral boundaries are free to expand. In the second case (right), lateral boundaries are constrained, leading to less vertical compression for a given vertical load.



With unconstrained lateral boundaries, we compute the amount of vertical compression and lateral expansion, as well as the ratio. We verify that this corresponds well with the value of Poisson's ratio (nu) used in the specification of the stiffness tensor C. (The small discrepancy mainly due to our grid not being a perfect cylinder).

```
L = max(G.nodes.coords(:,3)) - min(G.nodes.coords(:,3)); % length
R = max(G.nodes.coords(:,1)); % radius - should equal 1
axial_strain  = uu(top_nodes(1), 3) / L;
radial_strain = max(uu(:,1)) / R;
measured_nu   = radial_strain / axial_strain;
relative_err  = (measured_nu - nu)/nu;
fprintf('Real nu: %1.5f.  Measured nu: %1.5f\n', nu, measured_nu);
fprintf('Relative error: %1.2e\n', relative_err);
```

```
Real nu: 0.30000.  Measured nu: 0.30007
Relative error: 2.20e-04
```

Full example with source code available in Chapter 14 of "Advanced Modelling with the MATLAB Reservoir Simulation Toolbox (MRST), Cambridge Core (forthcoming)

Coupled mechanics and flow: the `ad-mechanics` module

- The theory of poroelasticity couples the linear elastic system with the equations of fluid flow in a porous medium.
- Two-way coupled system: mechanical deformation affect pore volume, and changes in the pore pressure field affects mechanical deformation.
- In MRST, coupled mechanics and flow can be modeled using the ad-mechanics module, combining the linear elasticity computations of VEM_linElast with one-phase or multi-phase flow equations.
- The resulting equations can be solved as a single, fully-coupled system, or in an iterative manner using operator splitting.

The basic, poroelastic equations:

- Modify equations of linear elasticity by adding pressure term to the stress tensor.
- Modify equations of fluid flow by adding a strain term to the accumulation term.
- In both cases, Biot's parameter α is used as a coupling multiplier.

$$\boldsymbol{\sigma} = \mathbf{C} \cdot \boldsymbol{\epsilon} - \alpha p \mathbf{I}$$

Stress tensor from linear elasticity modified by adding a pressure term, p. Biot's parameter is represented by α.

$$\nabla \cdot [\mathbf{C} \cdot \boldsymbol{\epsilon}] - \nabla(\alpha\, p) + \vec{b} = 0$$

The modified balance-of-forces equation. Compare with its linear elastic counterpart on the previous page.

$$\alpha\dot{\epsilon} + S_{\epsilon}\dot{p} - \frac{1}{\mu}\mathbf{K}\left(\nabla p - \rho_f \vec{g}\right) = Q$$

Darcy-based flow equation for one-phase flow in porous media. (Note that ad-mechanics also supports multi-phase, nonlinear flow). The equation is coupled to the mechanics system though it's first term. Time derivatives are written using the dot notation.

Models for coupled mechanics/flow in `ad-mechanics`:

Name	Solver	Flow model
`MechWaterModel`	fully coupled	1-phase flow
`MechOilWaterModel`	fully coupled	2-phase flow
`MechBlackOilModel`	fully coupled	3-phase black-oil flow
`MechFluidFixedStressSplitModel`	fixed stress split	1-phase, 2-phase, or 3-phase black-oil (chosen when setting up the model)

Example: Mandel's problem

- Archetypical example of a poroelastic problem exhibiting non-monotonic pressure behavior that cannot be accounted for without considering the two-way coupling between mechanics and flow.
- Infinitely long, rectangular slab of poroelastic material interposed between two parallel, rigid an impermeable plates.
- Free lateral movement, and free flow of fluid across lateral boundaries.
- Normal load applied along the y-axis. We measure the evolution of fluid pressure.

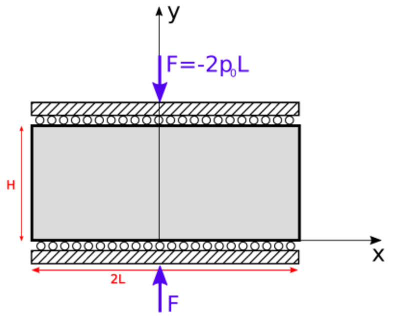
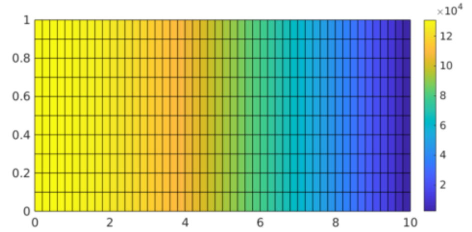
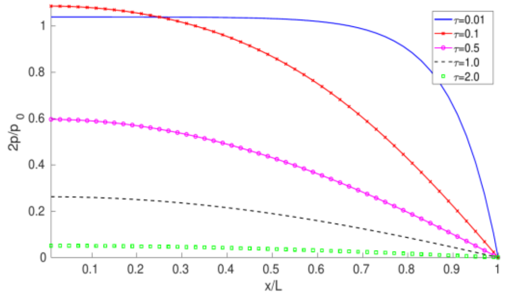


Illustration of the setup of the problem. The fluid-filled porous medium is drawn in grey, and interposed between two rigid plates. The medium extends infinitely in the z-direction. A force F is applied normal to the plates at t=0.



Pressure profile inside the (right half of) poroelastic slab after simulating Mandel's problem for a period of one characteristic time. Note that in order to compute characteristic time and analytical solution, we need to compute several poroelastic constants (see next page) derived from those used to specify the original problem.



Pressure profile inside (right half of) slab for different multiples of characteristic time. Note that between time τ=0.01 and τ=0.1, we observe a pressure increase at the middle of the slab, before dropping again at later times. This non-monotonous pressure evolution is referred to as the Mandel-Cryer effect.

Full example with source code available in Chapter 14 of "Advanced Modelling with the MATLAB Reservoir Simulation Toolbox (MRST), Cambridge Core (forthcoming)

Computing poroelastic parameters

Poroelastic parameters:

- A large number of poroelastic parameters have been defined in literature.
- They are interrelated; a small subset is enough to specify all the others. (Chosen parameter values must be internally consistent).
- Which ones are required or useful is highly problem-dependent.
- It can be hard to know how to compute one selection of parameter from a given different selection.

Symbol	Name
Poroelastic constants from Biot's basic constitutive relationships	
K	drained bulk modulus
H	inverse of poroelastic expansion coefficient
R	inverse of unjacketed specific storage coefficient
M	inverse of constrained specific storage coefficient
Compressibilities (other than K)	
K_s	unjacketed bulk modulus
K_p	inverse of drained pore compressibility
K_f	inverse of fluid compressibility
K_ϕ	inverse of unjacketed pore compressibility
K_v	uniaxial drained bulk modulus
Storativities	
S	uniaxial specific storage coefficient
S_σ	unconstrained specific storage coefficient
S_ϵ	constrained specific storage coefficient
S_γ	unjacketed specific storage
Other parameters from linear elasticity, drained or undrained	
K_u	undrained bulk modulus
$K_v^{(u)}$	uniaxial undrained bulk modulus
E	Young's modulus (drained)
E_u	Young's modulus (undrained)
λ	Lamé's parameter (drained)
λ_u	Lamé's parameter (undrained)
ν	Poisson's ratio (drained)
ν_u	Poisson's ratio (undrained)
G	Shear modulus (drained *and* undrained)
Other parameters	
α	Biot–Willis coefficient
β	effective stress coefficient for pore volume
γ	loading efficiency
η	poroelastic stress coefficient
c_m	Geertsma's parameter
B	Skempton's coefficient

Table: List of various poroelastic parameters one may encounter in literature, roughly sorted by category

`poroParams` utility function:

- `ad-mechanics` utility function that computes all poroelastic parameters from an arbitrary set of (compatible) input parameter values.
- User inputs porosity, as well as any chosen set of parameter values.
- Function computes all other values fully defined by the provided values.

porosity

known poroelastic parameters specified by user

```
poroParams(0.25, true, 'K', 1e9, 'H', 1.2e9, 'R', 1.2e9)
```

Other parameter values computed by function

```
ans =
  struct with fields:
            K: 1.0000e+09
            H: 1.2000e+09
            R: 1.1000e+09
            M: 4.6588e+09
          K_s: 6.0000e+09
          K_p: 300000000
          K_f: 2.1290e+09
        K_phi: 6.0000e+09
          K_v: NaN
            S: NaN
      S_sigma: 9.0909e-10
    S_epsilon: 2.1465e-10
      S_gamma: 7.5758e-11
        alpha: 0.8333

         beta: 0.9500
        gamma: NaN
          eta: NaN
          c_m: NaN
            B: 0.9167
          K_u: 4.2353e+09
         K_vu: NaN
            E: NaN
          E_u: NaN
       lambda: NaN
     lambda_u: NaN
           nu: NaN
         nu_u: NaN
            G: NaN
```

These values could not be computed from the given input

```
poroParams(0.25, true, 'K', 1e9, 'H', 1.1e9, 'R', 1.2e9, 'G', 0.9e9)
```

```
ans =
  struct with fields:
            K: 1.0000e+09
            H: 1.2000e+09
            R: 1.1000e+09
            M: 4.6588e+09
          K_s: 6.0000e+09
          K_p: 300000000
          K_f: 2.1290e+09
        K_phi: 6.0000e+09
          K_v: 2.2000e+09
            S: 5.3030e-10
      S_sigma: 9.0909e-10
    S_epsilon: 2.1465e-10
      S_gamma: 7.5758e-11
        alpha: 0.8333

         beta: 0.9500
        gamma: 0.7143
          eta: 0.3409
          c_m: 3.7879e-10
            B: 0.9167
          K_u: 4.2353e+09
         K_vu: 5.4353e+09
            E: 2.0769e+09
          E_u: 2.5214e+09
       lambda: 4.0000e+08
     lambda_u: 3.6353e+09
           nu: 0.1538
         nu_u: 0.4008
            G: 900000000
```

By also specifying shear modulus (G), all poroelastic parameters could now be computed