Multi-Point Stress Approximation Module in MRST

September 13, 2021

Multi-Point Stress Approximation (MPSA)

- The MPSA method applies for linear elasticity:

$$-\nabla \cdot .\pi = f \qquad \text{momentum balance}$$
$$\pi = C\varepsilon \qquad \text{stress constitutive relation}$$
$$\varepsilon = \frac{1}{2}(\nabla u + \nabla u^T) \qquad \text{strain}$$
$$u \qquad \text{displacement}$$

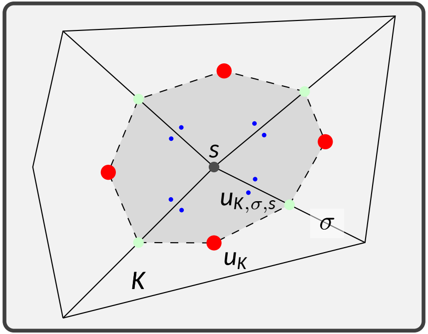
- The method was proposed and analysed in Nordbotten: *"Convergence of a Cell-Centered Finite Volume Discretization for Linear Elasticity"* (2015)
- We the version imposing symmetry weakly, MPSAW, as proposed in Keilegavlen and Nordbotten: *"Finite volume methods for elasticity with weak symmetry"* (2017)

Multi-Point Stress Approximation

- The advantage of the method is that it is a **finite volume** type of method. In particular,
  - The method is **cell-centered**
  - The method ensures a **discrete momentum conservation** equation. We have continuous forces at the faces. (similar to discrete mass conservation equation for finite volume methods).
- The method can be seen as an extension of MPFA-O to linear elasticity.
- The module is available at

  `https://bitbucket.org/mrst/mpsaw`
- Joint work with the University of Bergen.

Basic ideas behind MPFA/MPSA



K: cell
σ: face
s : vertex

u_K : value at cell-center
$u_{K,\sigma,s}$: value at cell-face-node

Gradient reconstruction in a *cell-node* region.

$$\nabla_d u_{K,s} = \sum_{\sigma} (u_{K,\sigma,s} - u_K) g_{K,\sigma,s}.$$

Here, $g_{K,\sigma,s}$ are chosen such that reconstruction is **consistent**

Nodal reduction

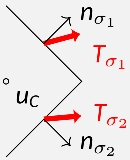
For MPFA (then u is a pressure), We can express $u_{K,\sigma,s}$ as a linear combination of u_K , using

- pressure continuity at the faces
- flux continuity at the faces

Symmetry requirement

- Imposing in the same way
  - displacement continuity at the faces
  - force continuity at the faces

  does not work in the case of linear elasticity because we have a symmetry constraint.
- To illustrate that, note that we cannot find a **affine** displacement u with matches a given displacement value and forces at the faces in a triangle



Find affine u such that

$$C\varepsilon(u)n_{\sigma_i} = T_{\sigma_i}$$
$$u(x_C) = u_C$$

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

because $\varepsilon(u)$ is symmetric (3 free variables) and we have two vectors T_{σ_i} (4 given variables).

SINTEF

Weak symmetry

- We relax the symmetry condition by replacing $\varepsilon(u)$ with

$$\varepsilon_{\text{weak},d}(u)_{K,s} = \frac{1}{2}(\nabla_d u_{K,s} + < \nabla_d u >_s^T)$$

  where $< \nabla_d u >_s$ is an average of the reconstructed gradient around the node.
- The nodal reconstruction is now possible (assuming displacement and *force* continuity)
- The symmetry of the strain $\varepsilon_{weak,d}$ holds weakly in the sense that its average is, by construction, symmetric.

Overview of the test cases in the module

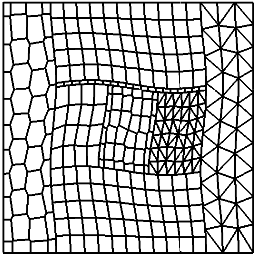
- The examples are presented here:

  `https://bitbucket.org/mrst/mpsaw/src/master/examples/`
- `assemblyMpfaExample.m` : Assembly of the basic MPSA matrices.
- `mpsaExample` : Set of examples with basic geometries and boundary conditions.
- `tiltedExample.m` : Example with non-cartesian directions sliding constraints.
- `convergencetests` : Directory with convergence tests that validate the implementation. The test cases for MPSA are taken from seminal paper, those for Biot system are new.
- `assemblyBiotExample` : Assembly of the basic MPSA-MPFA matrices for coupled flow-geomechanics systems.
- `biotBlackoilExample.m` : Example of coupled blackoil-geomechanics system.
- `biotCompositionalExample.m` : Example of coupled compositional-geomechanics system.

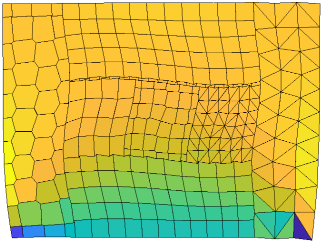
Compaction case on a complex grid

- Example of a complex grid (different types of cell)
- Compaction test : fixed displacement at bottom, constant force at the top.

grid



Divergence of diplacement
deformed grid



Some details on assembly

- We denote $\boldsymbol{u}_{\mathrm{nfd}}$: displacement at the node-face dofs in $\mathbb{R}^{\mathrm{nfd}}$
  $\boldsymbol{u}_{\mathrm{cd}}$: displacement at the cell dofs in $\mathbb{R}^{\mathrm{cd}}$
- The Dirichlet boundary conditions are imposed as Lagrange multipliers (see next slide). The system is assembled as

$$\begin{pmatrix} A_{11} & A_{12} & -D \\ A_{21} & A_{22} & 0 \\ D^T & A_{22} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_{\mathrm{nfd}} \\ \boldsymbol{u}_{\mathrm{cd}} \\ \boldsymbol{\lambda}_{\mathrm{bc}} \end{pmatrix} = \begin{bmatrix} \texttt{extforce} \\ \texttt{force} \\ \texttt{bcvals} \end{bmatrix}$$

  where `extforce`: Neumann forces at boundary in the node-face ($\mathbb{R}^{\mathrm{nfd}}$)
  `force` : volumetric forces in the cell dofs ($\mathbb{R}^{\mathrm{cd}}$)
  `bcvals` : value of the Dirichlet linear forms (see next slide) ($\mathbb{R}^{\mathrm{bc}}$)
- Function signature:

```
function assembly = assembleMPSA(G, prop, loadstruct, eta, tbls, mappings, varargin)
```

  where `prop` provides the material properties (see `setupStiffnessTensor`) and `loadstruct` the Dirichlet boundary condition and `extforce` and `force`. The value `eta` determines the location of the continuity point.

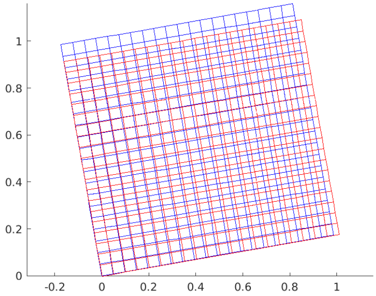
General Boundary Conditions

- Basic illustration given in `tiltedexample` of a rotated grid ($\theta = 10$ degrees).
- We impose rolling condition at the bottom and on the left-hand side, by the two following linear forms (encoded in *D*)

$$\sin(\theta)u_1 - \cos(\theta)u_2 = 0 \quad \text{for nodes at bottom}$$
$$\cos(\theta)u_1 + \sin(\theta)u_2 = 0 \quad \text{for nodes at left}$$

  and we impose a normal force at the top.

- We recover the exact linear solution



;

Poroelasticity

- Single phase

$$-\nabla \cdot \pi - \alpha \nabla p + = f \qquad \text{Momentum balance}$$

$$\frac{\partial}{\partial t}(\alpha \nabla \cdot u + S_\epsilon p) - \nabla \cdot \frac{k}{\mu} \nabla p = q \qquad \text{Mass conservation for fluid:}$$

- Biot system is the linearised version. The second equation is replaced with

$$\alpha \nabla \cdot \boldsymbol{u} + \rho p - \tau \nabla \cdot K \boldsymbol{q} = q$$

SINTEF

Poroelasticity - assembly

- For poroelasticity, we have, **before nodal reduction**,

$$
\begin{pmatrix}
A_{11} & A_{12} & 0 & \color{red}{A_{14}} & A_{15} & 0 \\
A_{21} & A_{22} & 0 & 0 & 0 & 0 \\
0 & 0 & A_{33} & A_{34} & 0 & A_{36} \\
\color{red}{A_{41}} & \color{red}{A_{42}} & A_{43} & A_{44} & 0 & 0 \\
A_{51} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & A_{63} & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
\boldsymbol{u}_{\text{nfd}} \\
\boldsymbol{u}_{\text{cd}} \\
\boldsymbol{p}_{\text{nf}} \\
\boldsymbol{p}_{\text{c}} \\
\boldsymbol{\lambda}_{\text{bcmech}} \\
\boldsymbol{\lambda}_{\text{bcfluid}}
\end{pmatrix}
=
\begin{bmatrix}
\texttt{extforce} \\
\texttt{force} \\
\texttt{extflux} \\
\texttt{src} \\
\texttt{bcvalsmech} \\
\texttt{bcvalsfluid}
\end{bmatrix}
$$

- The coupling terms are in red. Note that they are not symmetric, because we use two different discrete gradient operators.
- Function signature

```
function assembly = assembleBiot(G, props, drivingforces, eta, tbls, mappings, varargin)
```

- The variable `props` gather the material properties (mechanical, fluid and coupling parts) and `drivingforces` the boundary conditions and source terms.

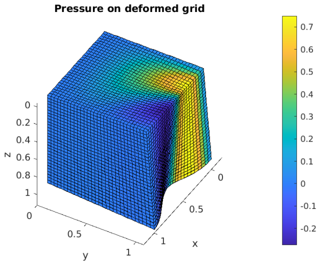
Convergence test for Biot system

- The script `biotConvergenceFunc` verifies the convergence of the implementation, as follows

- We choose a displacement and a pressure fields

$$u_1 = y(1-x)sin(2\pi xy)$$
$$u_2 = zy^2\cos(x)$$
$$u_3 = xyz$$
$$p = u_1$$

- We compute **analytically** the stimulation terms (source terms in Biot) and the boundary conditions (use different types on different sides). Those are used in as input in `biotConvergenceFunc`



Coupled simulations compositional - geomechanics

- The scripts `biotBlackoilExample` and `biotCompositionalExample` illustrate couplings that integrate MRST reservoir solvers.
- The equation for the black-oil model
  - Momentum equation : $\nabla \cdot \pi - \nabla(bp_b) + \rho_b g = 0$
  - Mass conservation equations for black-oil:

$$
\begin{aligned}
\partial_t(\phi b_o s_o) + \nabla \cdot (b_o v_o) &= b_o q_o, \\
\partial_t(\phi b_w s_w) + \nabla \cdot (b_w v_w) &= b_w q_w, \\
\partial_t[\phi(b_g s_g + b_o r_s s_o)] + \nabla \cdot (b_g v_g + b_o r_s v_o) &= b_g q_g + b_o r_s q_o.
\end{aligned}
$$

  - Darcy's law : $v_\alpha = -(k_{r\alpha}/\mu_\alpha)K(\nabla p_\alpha - \rho_\alpha g \nabla z)$
  - Pore volume change : $\phi - \phi_0 = b\nabla \cdot u + \frac{1}{N}(p_b - p_0)$
  - Constitutive relations for Biot pressure $p_b = p_b(\boldsymbol{p}, \boldsymbol{s})$ and also p_α (through capillary pressures)

Implementation using state functions

- We introduce two `StateFunctionGrouping`
  - `MechPropertyFunctions` : for the evaluation of mechanical variables
  - `BiotPropertyFunctions` : for the evaluation of the terms related to the couplings
- In MechPropertyFunctions, we find use
  - `FaceNodeDisplacement` : computes the displacement at the face node location from the cell values
  - `ConsistentDiv` : computes the *consistent* divergence (obtained from the consistent gradient reconstruction)
  - `Strain` : computes the strain
  - `Stress` : computes the stress
- In BiotPropertyFunctions, we find use
  - `Dilatation` : computes the dilation term ($\div u$ term)
  - `BasePoreVolume` : capture the pore volume from the reservoir model

Compositional model

- We implement `BiotCompositionalModel` using the `statefunction` mechanism.
- We recover the state function of the generic compositional model

```
model = setupStateFunctionGroupings@GenericBlackOilModel(model, varargin{:});
```

- We augment or modify them as follows

```
fluxprops = model.FlowDiscretization;
biotprops = model.BiotPropertyFunctions;
pvtprops  = model.PVTPropertyFunctions;
mprops    = model.MechPropertyFunctions;

pv = pvtprops.getStateFunction('PoreVolume');

biotprops = biotprops.setStateFunction('BasePoreVolume'                   , pv);
biotprops = biotprops.setStateFunction('Dilatation'                       , BiotBlackOilDilatation(model));
pvtprops  =  pvtprops.setStateFunction('PoreVolume'                       , BiotPoreVolume(model));
mprops    =    mprops.setStateFunction('FaceNodeDisplacement'             , BiotFaceNodeDisplacement(model));
fluxprops = fluxprops.setStateFunction('PermeabilityPotentialGradient', MpfaKgrad(model));
fluxprops = fluxprops.setStateFunction('PhaseUpwindFlag'                  , MpfaPhaseUpwindFlag(model));
```

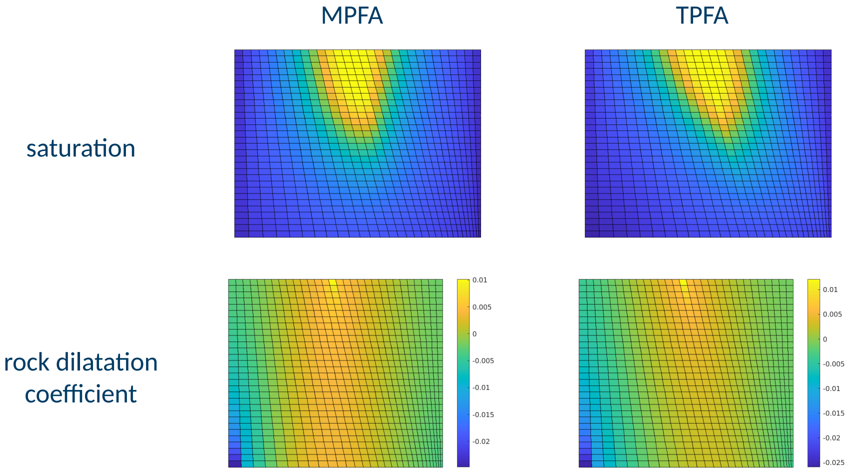
- In this way, we overwrite the following `statefunction` of the generic blackoil model

  `PoreVolume`, `PermeabilityPotentialGradient`, `PhaseUpwindFlag`

  but otherwise inherit all its functionalities - with minimal code intrusion.

coupled compositional simulation

- We consider a *skewed* grid (see MRST book) and a standard injection scenario.
- We consider coupling with both MPFA and TPFA.
- We can observe the consistency error on the mechanical part of the solution



SINTEF