Geocritical Reservoir Flow Simulation and Display using Open Porous Medium Code

John Rugis, Peter Leary, Peter Malin and Justin Pogacnik
Outline

- Geocriticality
- Modelling Porosity and Permeability
- Open Porous Media (OPM) Initiative
- Simulation with OPM
- Visualisation of Results
Geocriticality (References)

- **NZGW 2013**

  Peter Leary et al, *Prospects for Enhanced Single-Well Heat Extraction*


Exploring Geocriticality

Software toolset and workflow:

MatLab (modelling) → OPM (simulation) → ParaView (visualisation)

Voreen (visualisation)
**Modelling Porosity and Permeability**

Create synthetic porosity cube (MatLab, Voreen).

\[ \varphi - \text{porosity field} \]

\[ N - \text{power spectrum} \]

\[ k - \text{spatial frequency} \]

\[ N(k) \propto \frac{1}{k^\beta} \]

**Porosity**

- cube: 64x64x64
- value: 0.1 – 0.3
- spatial distribution: \( \beta = 1 \)
- population distribution: normal
Modelling Porosity and Permeability

Derive synthetic permeability cube (*MatLab*, *Voreen*).

\[ \kappa - \text{permeability field} \]
\[ \varphi - \text{porosity field} \]

\[ \kappa_d \propto e^{\alpha \varphi} \]

**Permeability**

cube: 64x64x64
population distribution: lognormal
proportionality constant: 10
\[ \alpha = 3 \]
Modelling Porosity and Permeability
Collection of permeability cubes (*MatLab*, *Voreen*).

population distribution: lognormal
proportionality constant: 10
\( \alpha = 3 \)

population distribution: lognormal
proportionality constant: 10
\( \alpha = 10 \)
Modelling Porosity and Permeability
Collection of permeability cubes (*MatLab*, *Voreen*).

population distribution: lognormal
proportionality constant: 10
\( \alpha = 20 \)

population distribution: lognormal
proportionality constant: 10
\( \alpha = 30 \)
Open Porous Media (OPM) Initiative

• Launched in June 2009 at Statoil Research Center, Norway.
• Supported by six research groups and several industry partners in Norway and Germany.
• Current officially funded OPM development is focused on oil reservoir engineering, enhanced oil recovery and CO₂ sequestration.
• Contributions aimed at different fields are encouraged

• The entire software suite is open-source, available under the terms of the GNU General Public License (GPL) version 3.
• All of the OPM source code is hosted in GitHub public repositories.
Simulation with OPM

• Simulate steady-state pressure and velocity fields:
  – Start with given synthetic porosity and permeability fields.
  – 64x64x64 cubic metres simulation space.
  – “Point” injection and extraction.
  – Inject and extract incompressible fluid (water) at equal rates of ten litres per second.
Simulation with OPM
Code development and modifications.

- **OPM** is an extension module to the Distributed Unified Numerics Environment (**DUNE**), a software toolbox for solving partial differential equations using grid based methods.

- Both **OPM** and **DUNE** are written in the C++ language and make use of an object oriented programming style.

- The extensive **OPM** application programming interface (API) documentation takes the form of formatted HTML pages suitable for viewing in a web browser.
Simulation with *OPM*

Code development and modifications.

- **OPM / DUNE Code Modifications:**
  1) Input model data directly from *MatLab* files.
  2) Extend basic simulation class with additional features.
  3) Add simulation output values to *VTK* files.
Simulation with *OPM*

Code development and modifications.

class IncompPropertiesBasic : public IncompPropertiesInterface
{

public:
  IncompPropertiesBasic(const int num_phases,
                        const SaturationPropsBasic::RelPermFunc& relpermfunc,
                        const std::vector<double>& rho,
                        const std::vector<double>& mu,
                        const double porosity,
                        const double permeability,
                        const int dim,
                        const int num_cells);

  ...

  virtual void setPorosity(const std::vector<double>& porosity);
  virtual void setPermeability(const std::vector<double>& permeability);

  ...
}
Simulation with OPM
Code development and modifications.

// read variable phi from MatLab file
matvar_t *mat_phi;
mat_phi = Mat_VarRead(matfp,"phi");
if(mat_phi == NULL) {
    cout << "Variable phi not in MAT file." << endl;
    exit(1);
}
.
.
// set porosity from MatLab variable
std::vector<double> phi(num_cells, 0.5);
double *ptr = (double *) mat_phi->data;
unsigned n = 0;
for(int l = 0; l < nz; l++) {
    for(int j = 0; j < ny; j++) {
        for(int i = 0; i < nx; i++) {
            phi[n++] = *(ptr++);
        }
    }
}
Mat_VarFree(mat_phi);
props.setPorosity(phi);
Simulation with OPM

Steady state solution

Velocity Field cube: 64x64x64
$\alpha = 3$

Velocity Field: cross-section (ParaView)
Simulation with *OPM*

Steady state solution

**Velocity Field**
cube: 64x64x64
\( \alpha = 10 \)

**Velocity Field:** cross-section (*ParaView*)
Simulation with *OPM*

Steady state solution

**Velocity Field**

cube: 64x64x64

$\alpha = 20$

**Velocity Field: cross-section** (*ParaView*)
Simulation with *OPM*

Steady state solution

**Velocity Field**

cube: 64x64x64

\( \alpha = 30 \)

Velocity Field: cross-section (*ParaView*)
Data Processing with *Paraview*

Not just for visualisation. Computation and analysis “filters” as well.

Example:

- **Streamlines**
  - A family of curves tangent to the velocity vector field.
  - Show fluid element flow paths.
  - *ParaView* stream tracer:
    - seed type, size and count
    - integrator type
    - etc
Simulation with **OPM & ParaView**

Steady state solution

**Velocity Field**
cube: 64x64x64
\( \alpha = 3 \)

**Velocity Field**: streamlines (**ParaView**)
Simulation with *OPM & ParaView*

Steady state solution

**Velocity Field**
cube: 64x64x64
\( \alpha = 10 \)

Velocity Field: streamlines (*ParaView*)
Simulation with **OPM & ParaView**

Steady state solution

**Velocity Field**
cube: 64x64x64
\( \alpha = 20 \)

**Velocity Field**: streamlines (*ParaView*)
Simulation with *OPM* & *ParaView*

Steady state solution

**Velocity Field**
cube: 64x64x64
\( \alpha = 30 \)

**Velocity Field:** streamlines (*ParaView*)
Questions, Further Work, Discussion?