Simulation of flows in heterogeneous porous media with Feel++

Obed SABA

Project M1
Master CSMI

August 23, 2023

Table of Contents

Main objective:

Simulation of flows in `heterogeneous` *porous media with* `Feel++`*.* `Heterogeneous`*, because the porous medium is composed of several materials with different properties.*

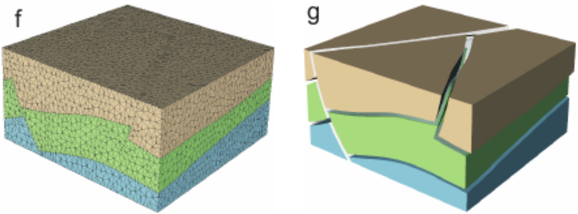


Figure: *Example of heterogeneous porous medium*

Work plan and Intermediate steps

Work plan:

- Presentation of the models
- Presentation the mesh generation in 3D with Gmsh
- Presentation of the 3D of Richards model
- Presentation of the 3D of two-phase flow : CFPDEs
- Conclusion

Intermediate steps:

- Generation of a suitable mesh for a 3D simulation.
- Explore the Richards model in 3D cases with CFPDEs
- Adapting time step in case of non-convergence when solving Richards equations with Feel++
- Investigae the Two-phase model in 3D

Tools

The work was organised and coordinated via the following tools:

- **Slack** : for internal communication.
- **Github** : for code sharing, version management and issue tracking.
- **Python** : for the implementation of the numerical model.
- **Gmsh** : for the generation of the mesh.
- **Feel++** : for the numerical simulation.
- **Jupyter notebook** : for the presentation of the results.

MESHING FRACTURED DOMAINS

Example of a fractured domain

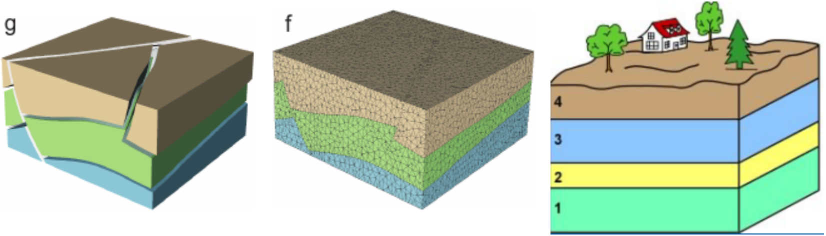


Figure: *Example of a fractured domain*

Mesh Generation

- Mesh with a one fracture

Mesh with a one fracture

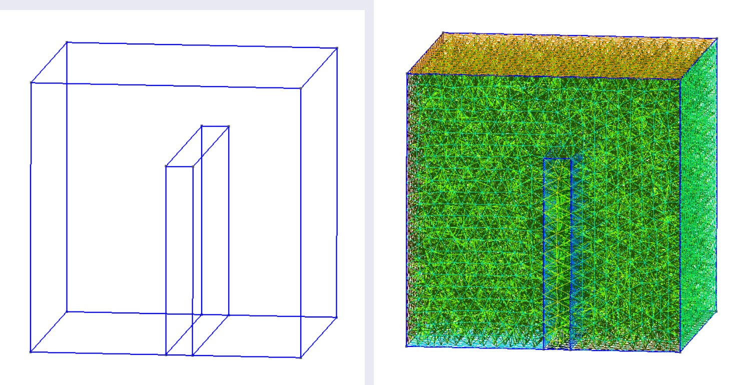
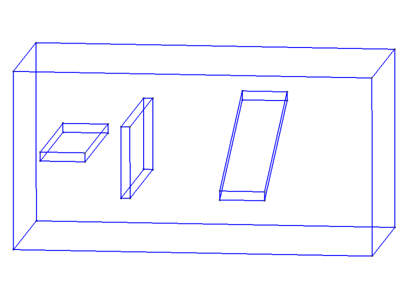


Figure: *Mesh in 3D with 1 fracture*

Mesh Generation

- Complex Mesh with three fractures

Complex Mesh with three fractures



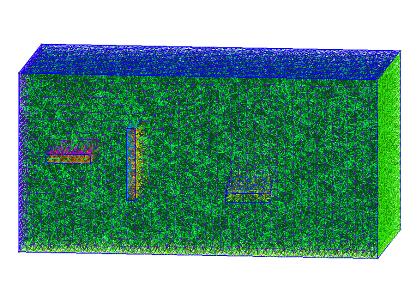
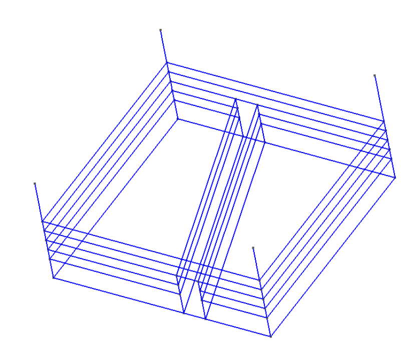


Figure: *Mesh in 3D with 3 fractures see*

Mesh Generation

- Complex terrain Mesh with fractures

Complex terrain Mesh with fractures



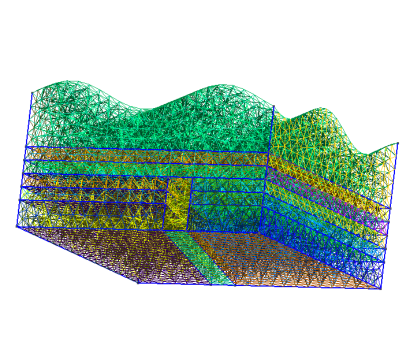


Figure: *Mesh in 3D with 6 fractures see*

SOLVING RICHARDS' PDES OVER FRACTURED DOMAINS

PDE Model:Two-Phase Model

Mass conservation equation:

$$\begin{cases} \frac{\partial(\phi\rho_\omega S_\omega)}{\partial t} + div(\rho_\omega \overrightarrow{V}_\omega) = 0 \\ \frac{\partial(\phi\rho_0 S_0)}{\partial t} + div(\rho_0 \overrightarrow{V}_0) = 0 \end{cases}$$

Two phase Darcy laws:

$$\begin{cases} \overrightarrow{V}_\omega = -\frac{k_{r,\omega}(S_\omega)}{\mu_\omega} K(\nabla P_\omega - \rho_\omega \overrightarrow{g}) \\ \overrightarrow{V}_0 = -\frac{k_{r,0}(S_0)}{\mu_0} K(\nabla P_\omega + \nabla P_c(S_\omega) - \rho_0 \overrightarrow{g}) \end{cases}$$

Pore volume conservation:

$$S_\omega + S_0 = 1$$

Where:

- ϕ : porosity.
- S_i : saturation.
- V : velocity.
- K : permeability.
- P_i : pressure.

Towards Simplification: The Richards Model

Considering the intricate nature and numerical cost of our primary model, we turn to the Richards model for a more streamlined approximation.

Assumptions:

- Air viscosity is much less than water's.
- Air is continuously connected to the atmosphere ($patm = 0$).
- Solid skeleton is rigid; water is uniform and incompressible.

Resulting in:

$$\begin{cases} \partial_t s - \nabla.(\lambda(s)(\nabla p - g)) = 0 \\ s = S(p) \end{cases} \tag{1}$$

Where:

- s: saturation.
- $\lambda(s)$: hydraulic conductivity
- g: gravity.
- $S(p)$: relation between s and p

The Brooks-Corey law S(p)

$$S_1(p) = \begin{cases} \left(\frac{p}{p_b}\right)^{\frac{1}{\beta}} & \text{if } p \leq p_b \\ 1 & \text{if } p > p_b \end{cases} \quad (2)$$

$$S_2(p) = \frac{1}{1 + e^{-\beta p - 1}} \quad (3)$$

$$S_3(p) = \tanh(ap + b) \quad (4)$$

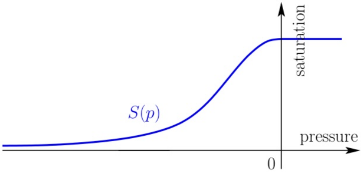


Figure: $S(p)$ function from [5]

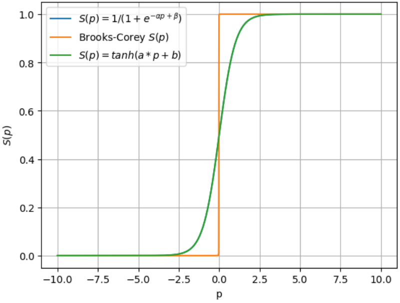


Figure: approximate saturation function

CFPDE: Coefficients For PDE

What is CFPDE?

CFPDE is a generalization of PDEs, introducing function-based coefficients to encapsulate more complex behaviors and dependencies in the equation. A typical CFPDE is:

$$d(u)\partial_t u + \nabla.(-c(u)\nabla u - \alpha(u)u + \gamma(u)) + \beta(u)\nabla u + a(u)u = f(u) \quad (5)$$

Where:

- $d(u)$: Time dependency.
- $c(u)$: Diffusion coefficient.
- $\alpha(u)$: Convection coefficient.
- $\gamma(u)$: Reaction term.
- $\beta(u)$: Gradient influence.
- $a(u)$: Source/sink term.
- $f(u)$: External force function.

Configuring CFPDE for Richards

Identifications on Equation 1:

- $u = s$
- $d(u) = 1$
- $\gamma(u) = -\lambda(s)(\nabla p - \overrightarrow{g})$
- All other coefficients (c, α, β, a, f) are zero.

JSON Configuration:

```
{
"saturation":{
    "setup":{
        "unknown":{
            "basis":"Pch1",
            "name":"S",
            "symbol":"s"
        },
        "coefficients":{
            "d": "1.0",
            "gamma": "{-saturation_s^m * materials_permkx * (pressure_grad_p_0 - gx), -saturation_s^m * materials_permky * (pressure_grad_p_1 - gy)}:saturation_s:pressure_grad_p_0:pressure_grad_p_1:gx:gy:materials_permkx:materials_permky:m"
        }
    }
}
}
```

Configuring CFPDE for Richards

Identifications on Equation 2:

- $u = p$
- $f(u) = s - S(p)$
- All other coefficients (c, α, β, a, γ) are zero.

JSON Configuration:

```
{
    "pressure":{
        "setup":{
            "unknown":{
                "basis":"Pch1",
                "name":"P",
                "symbol":"p"
            },
            "coefficients":{
                "f":"{saturation_s - materials_sigmoid}:saturation_s:
    materials_sigmoid"
            }
        }
    }
}
```

Results in 2D

With permeability

$$K = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad S(p) = \frac{1}{1 + e^{(-p-1)}} \tag{6}$$

Figure: Results of the simulation in 2D with opening

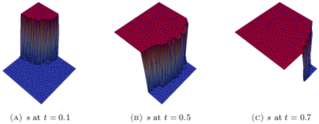
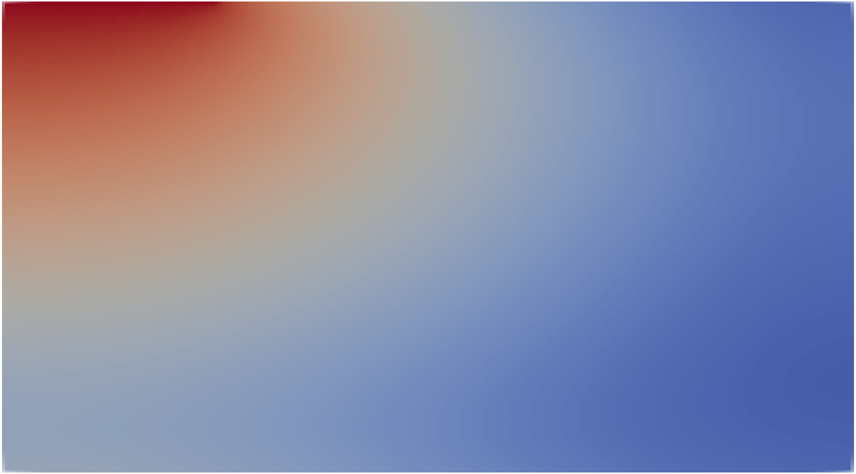


Figure: Results taken from the book

Results in 2D

2D simulation

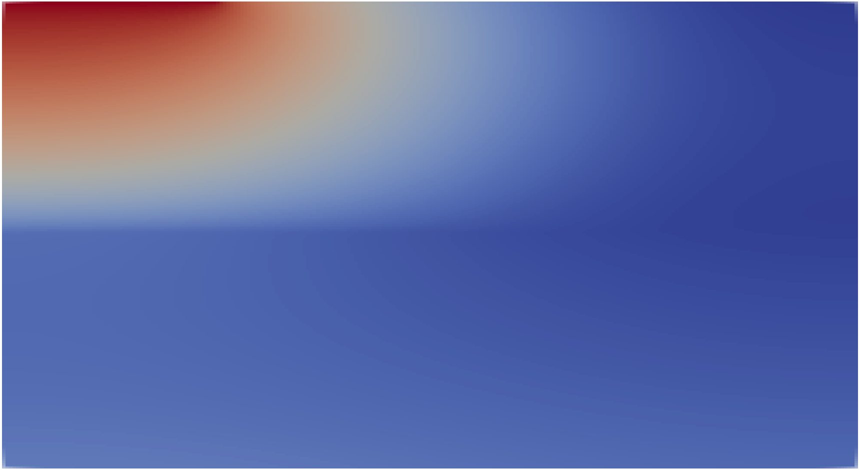


Heterogeneous permeability

$$K = \begin{bmatrix} 1 + 10(y < 0.7) & 0 \\ 0 & 1 + 10(y < 0.7) \end{bmatrix} \quad \textit{and} \quad \alpha = 1, \beta = -1 \qquad (7)$$

Heterogeneous permeability

2D simulation



Results in 3D

$$K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad S(p) = \frac{1}{1 + e^{(-p-1)}} \tag{8}$$

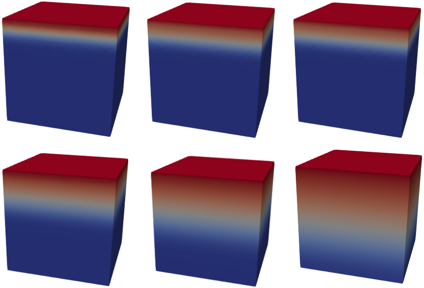


Figure: Results of the simulation in 3D

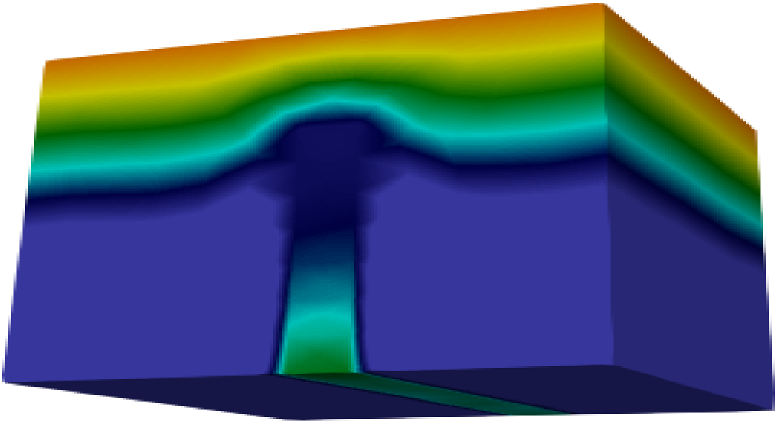
Results in 3D

We consider Ω_1 as the matrix and Ω_2 as fractures :

$$K_{\Omega_1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.2 \end{pmatrix} \quad \text{and} \quad K_{\Omega_2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{9}$$

Results in 3D

Heterogeneous permeability



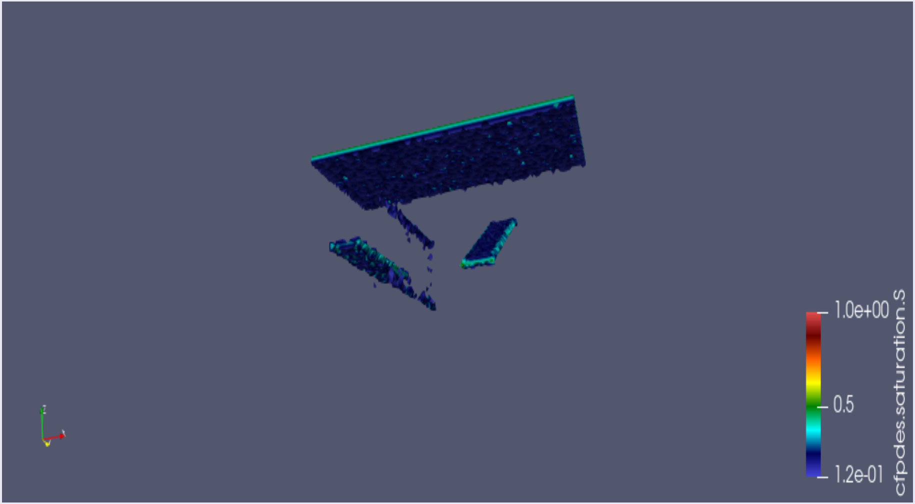
Results in 3D

We consider Ω_1 as the matrix and $\Omega_2, \Omega_3, \Omega_4$ as fractures :

$$\begin{cases} K_{\Omega_1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.2 \end{pmatrix} \quad \text{and} \quad K_{\Omega_2} = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ K_{\Omega_3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad K_{\Omega_4} = \begin{pmatrix} cos(1.11) & 0 & 0 \\ 0 & cos(0.46) & 0 \\ 0 & 0 & cos(0.57) \end{pmatrix} \end{cases} \tag{10}$$

Results in 3D

Heterogeneous permeability

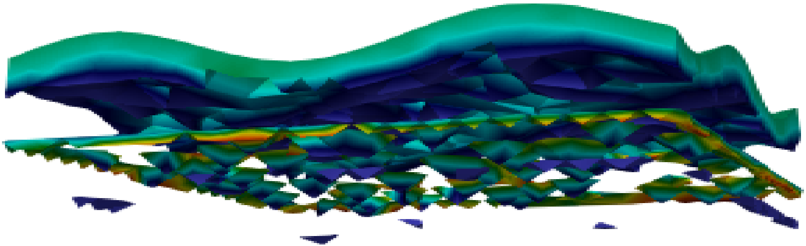


Results in 3D

$$
\begin{cases}
K_{\Omega_1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.9 \end{pmatrix} \quad \text{and} \quad K_{\Omega_2,\Omega_6} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.2 \end{pmatrix} \\
K_{\Omega_3,\Omega_5} = \begin{pmatrix} 0.9 & 0 & 0 \\ 0 & 0.9 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad K_{\Omega_4} = \begin{pmatrix} 0.85 & 0 & 0 \\ 0 & 0.85 & 0 \\ 0 & 0 & 0.2 \end{pmatrix} \\
K_{\Omega_7} = \begin{pmatrix} cos(-1.262) & 0 & 0 \\ 0 & cos(-0.291) & 0 \\ 0 & 0 & cos(-0.983) \end{pmatrix}
\end{cases}
\tag{11}
$$

Results in 3D

Heterogeneous permeability



Adapting time step in case of non-convergence with Feel++

Adapting time step in case of non-convergence with Feel++

GOAL: adapt the time step to save time on the simulations

```
{
def richards(hsize, json, dim=3, verbose=False):
    if verbose:
        print(f"Solving the richards problem for hsize = {hsize}...")
    feelpp.Environment.setConfigFile("../feel/richards.cfg")
    richards = cfpdes(dim=dim, keyword=f"cfpdes-{dim}d")
    richards.setMesh(getMesh(f"16.geo", hsize=hsize, dim=dim, verbose=verbose))
    richards.setModelProperties(json)
    richards.init(buildModelAlgebraicFactory=True)
    richards.printAndSaveInfo()
    richards.startTimeStep()
    measures = None
    if richards.isStationary():
        try:
            richards.solve()
        except Exception as e:
            print(f"Error encountered during solve: {e}")
            return None
        richards.exportResults()
    else:
        while not richards.timeStepBase().isFinished():
            if richards.worldComm().isMasterRank():
                print("
    ================================================================================\n")
                print("time simulation: ", richards.time(), "s \n")
                print("
    ================================================================================\n")
}
```

Adapting time step in case of non-convergence with Feel++

```
{
        dt = richards.timeStepBase().timeStep()
        while dt > 1e-10:  # lower limit for time step
            start_time = time.time()
            try:
                converged = richards.solve()
                solve_time = time.time() - start_time
                if solve_time < 1000.0:  # threshold for solve time
                    break  # if solve() is fast enough, exit the inner while loop
                else:
                    print(f"Solve() took too long ({solve_time} seconds), reducing time step")
                    dt *= 0.5  # reduce time step by a factor of two
                    richards.setTimeStep(dt)  # set new time step
            except Exception as e:
                print(f"Error encountered during solve: {e}")
                traceback.print_exc() # print traceback to see where the error occurred
                if not converged:
                    print(f"Warning: solve() did not converge : {converged}")
                    dt *= 0.5
                    richards.setTimeStep(dt) # set new time step
        if dt <= 1e-10:  # lower limit for time step
            print("Warning: Time step became too small even though solve() takes too long. Exiting simulation.")
            return None
        richards.exportResults()
        richards.updateTimeStep()
if measures :
    measures = richards.postProcessMeasures().values()
    return measures
}
```

Numerical simulation with the two-phase model

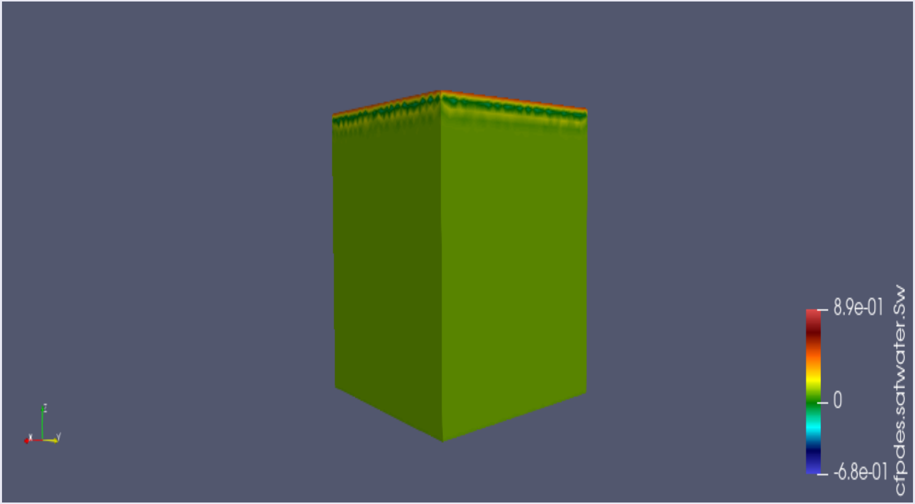
CFPDEs : two-phase model

$$\begin{cases} \frac{\partial(\phi\rho_w S_w)}{\partial t} + div(-\rho_w \frac{S_w^2}{\mu_w} K(\nabla P_w - \rho_w \overrightarrow{g})) = 0 \\ \frac{\partial(\phi\rho_o(1-S_w))}{\partial t} + div(-\rho_o \frac{(1-S_w)^2}{\mu_o} K(\nabla P_w + \nabla P_c(S_w) - \rho_o \overrightarrow{g})) = 0 \end{cases} \tag{12}$$

$$K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{for the permeability}$$

$$\text{and} \quad P_c = -((S_w - 0.5)^3 + 5) \quad \text{for the pressure}$$

Two Phase-Flow simulation



Conclusion

Conclusion

We understood how to model and simulate heterogeneous environments with `Feel++`.

Perspectives :

- Fix the two-phase simulations
- Fix the adaptive time steppings

Reflection on the Learning Experience

- Deepened understanding of flow simulation in porous media.
- Hands-on experience and application with `Feel++`.
- Acquired efficient methodologies for tackling challenges in the field.
- *Gentleness:* My instructor was remarkably kind, always approaching topics with a gentle demeanor, which made the learning environment comfortable.
- *Understanding:* More than just teaching, my instructor demonstrated a deep understanding of the topics, ensuring that complex ideas were broken down and presented in an approachable manner.
- *Patience:* No matter the challenge, there was always an aura of patience. This greatly assisted in my ability to tackle and understand even the most difficult subjects.
- *Rigour:* The patience and kindness never compromised the rigour of the coursework. Standards were maintained, ensuring a quality education and thorough understanding.

Conclusion

Thank you for your attention

References

A. Szymkiewicz.
Modeling of Water Flow in insaturated Porous Media.
2013.

https://www.researchgate.net/figure/Two.phase.flow.in.the.porous.media.fig1.324844316

Guy Chavent, Jerome Jaffre.
Mathematical models and finite element for reservoir simulation.
Single, multiphase and multicomponent flows through.

Walker, S.
"Flow in Porous Media."
John Wiley & Sons, 1986.

Roland Masson -
"Numerical simulation of two phase porous media flow models with application to oil recovery."