Injecting carbon-dioxide into deep reservoirs

06.12.2020-Modelling Project Work 2
Said Ghemam Amara
Eötvös Loránd University (ELTE)
Faculty of Science -Applied mathematics(MSC)
Supervisor: **Pr.Izsák Ferenc**

Overview

The problem of injecting carbon-dioxide into deep reservoirs is important from the practical point of view and its modeling is mathematically challenging. In the framework of the planned project, first, the corresponding one-phase flow**.** Another important task is the computer simulation of the corresponding process using the Matlab toolbox MRST**.**
In this study, CO2 is injected in the aquifer for a period of 30 years. Thereafter we simulate the migration of the CO2 in a post-injection period of 720 years. The simulation is done using the vertical average/equilibrium framework.

In this part from the project, we will explain two functions that are of great importance in the simulation of co2 injection in deep reservoirs.

First function:

Solve incompressible flow problem (fluxes/pressures) for VE equation.

SYNOPSIS:

```
   state = solveIncompFlowVE(state, G, S, rock, fluid)
   state = solveIncompFlowVE(state, G, S, rock, fluid, 'pn1', pv1, ...)
```

DESCRIPTION:

This function assembles and solves a (block) system of linear equations defining interface fluxes and cell and interface pressures at the next time step in a sequential splitting scheme for the reservoir simulation problem defined by

Darcy's law and the Vertical Equilibrium assumption and a given set of external influences (wells, sources, and boundary conditions).

NOTE:

REQUIRED PARAMETERS:

state : Reservoir and well solution structure either properly initialized from functions 'initResSol' and 'initWellSol' respectively, or the results from a previous call to functio 'solveIncompFlowVE' and,
possibly, a transport solver such a function 'explicitTransportVE'.

OPTIONAL PARAMETERS:

bc : Boundary condition structure as defined by function 'addBC'.
This structure accounts for all external boundary conditions to the reservoir flow. May be empty (i.e., bc = []) which is interpreted as all external no-flow (homogeneous Neumann) conditions.

Src : Explicit source contributions as defined by function 'addSource'. May be empty (i.e., src = []) which is interpreted as a reservoir model without explicit sources.

NOTE: This is a special purpose option for use by code which needs to modify the system of linear equations directly.

LinSolve : Handle to linear system solver software to which the fully assembled system of linear equations will be passed. Assumed to support the syntax x = LinSolve(A, b) in order to solve a system Ax=b of linear equations. Default value: LinSolve = @mldivide (backslash).

RETURNS:

state: Update reservoir and well solution structure with new values for the fields:

pressure: Pressure values for all cells in the discretised reservoir model, 'G'.

facePressure: Pressure values for all interfaces in the discretised reservoir model, 'G'.

Second function:

Explicit single point upwind solver for two-phase flow using VE equations.

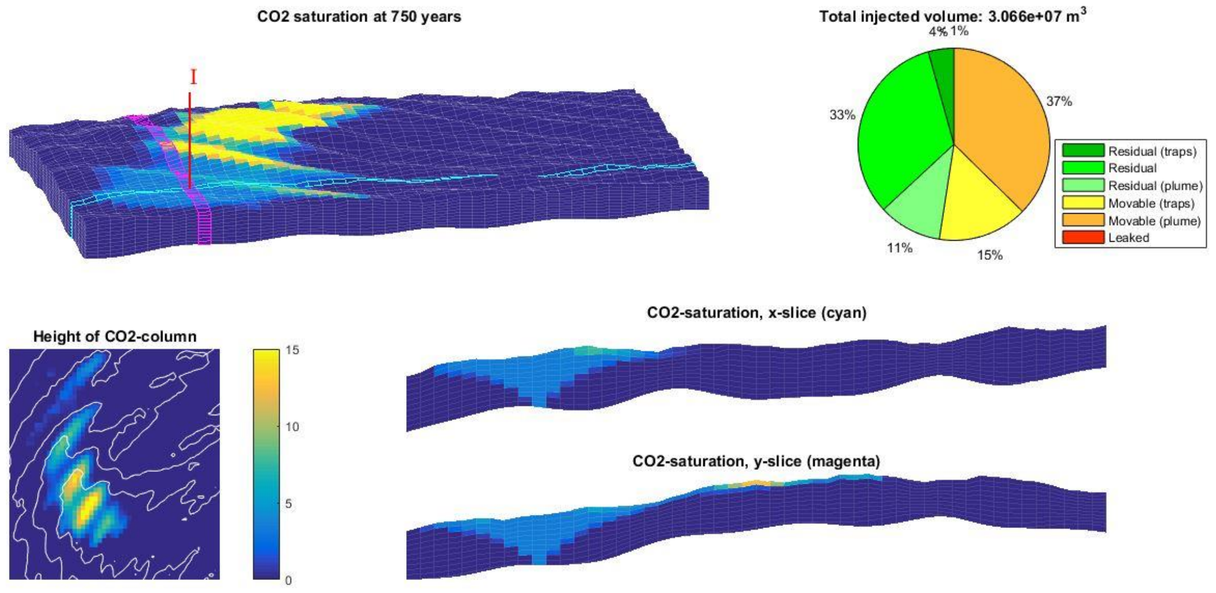
SYNOPSIS:

[state, dt_v] = explicitTransportVE(state, G_top, tf, rock, fluid)
[state, dt_v] = explicitTransportVE(state, G_top, tf, rock,… fluid, 'pn1', pv1)

DESCRIPTION:
Function explicitTransportVE solves the Buckley-Leverett transport equation
h_t + f(h)_x = q
using a first-order upwind discretisation in space and a forward Euler discretisation in time. The transport equation is solved on the time interval [0,tf].
The upwind forward Euler discretisation of the Buckley-Leverett model
for the Vertical Equilibrium model can be written as:
h^(n+1) = h^n - (dt./pv)*((H(h^n) - max(q,0) - min(q,0)*f(h^n)) where
H(h) = f_up(h)(flux + grav*lam_nw_up*(z_diff+rho_diff*h_diff(h)))
z_diff, h_diff are two point approximations to grad_x z, and grad_x h, f_up and lam_nw_up are the Buckely-Leverett fractional flow function and the mobility for the non-wetting phase, respectively, evaluated for upstream mobility:
f_up = A_w*lam_w(h)./(A_w*lam_w(h)+A_nw*lam_nw(h))
lam_nw_up = diag(A_nw*lam_nw(h), pv is the porevolume, lam_x is the mobility for phase x, while A_nw and A_w are index matrices that determine the upstream mobility. If h_diff is evaluated at h^(n+1) instead of h^n we get a semi implicit method.

PARAMETERS:
state: Reservoir solution structure containing valid water saturation state.h(:,1) with one value for each cell in the grid.
G_top: Grid data structure discretising the top surface of the reservoir model, as defined by function 'topSurfaceGrid'.
NB: The explicit scheme is only stable provided that dt satisfies a CFL time step restriction.
time_stepping: Either use a standard CFL condition ('simple'), Coats formulae ('coats'), or a heuristic bound that allows for quite optimistic time steps ('dynamic').

Now, we would like to see the result after run the code. The result is:



After seeing the result, it is worth asking the following question: What is the result means ?.

Firstly, we will see in a video how does CO2 moving in deep reservoirs after injection.

Secondly, I'm going to explain the relative circle.

Residual (traps): this phase of trapping happens very quickly as the porous rock acts like a tight, rigid sponge. As the supercritical CO2 is injected into the formation it displaces fluid as it moves through the porous rock. As the CO2 continues to move, fluid again replaces it, but some of the CO2 will be left behind as disconnected – or residual – droplets in the pore spaces which are immobile, just like water in a sponge. This is often how the oil was held for millions of years.

Residual: Residually trapped CO2 outside free plume and residual traps.

Residual (plume): CO2 still in the free-flowing plume, but destined to be left behind after imbibition.

Movable: the movable CO2 plume never grows large, and as it moves it is quickly dissolved and does not migrate far. On the other hand, since the brine

below the plume is saturated with CO2 at all times, no additional dissolution occurs in areas where the plume remains present.
Leaked: CO2 that has left the simulated domain.

Some important equations used in this simulation:

- Darcy’s law for a single-phase fluid: $\vec{v} = \frac{k}{\mu}(\nabla p - g\rho\nabla z)$ (1)
- Darcy law: $q = \frac{Q}{A} = k\left(\frac{h_1 - h_2}{L}\right)$ (2)
- Buckley-Leverett transport equation: $u_t + f(u)_x = 0$ (3)

References:
1- Knut-Andreas Lie, 2019, An Introduction to Reservoir Simulation Using MATLAB/GNU Octave User Guide for the MATLAB Reservoir Simulation Toolbox (MRST),(p115)
2- Knut-Andreas Lie, 2019, An Introduction to Reservoir Simulation Using MATLAB/GNU Octave User Guide for the MATLAB Reservoir Simulation Toolbox(MRST),(p114)
3- Knut-Andreas Lie, 2019, An Introduction to Reservoir Simulation Using MATLAB/GNU Octave User Guide for the MATLAB Reservoir Simulation Toolbox (MRST),(p172)