Field Scale Application Case: Steam Injection into Saturated Soil

Kilian Weishaupt, LH2

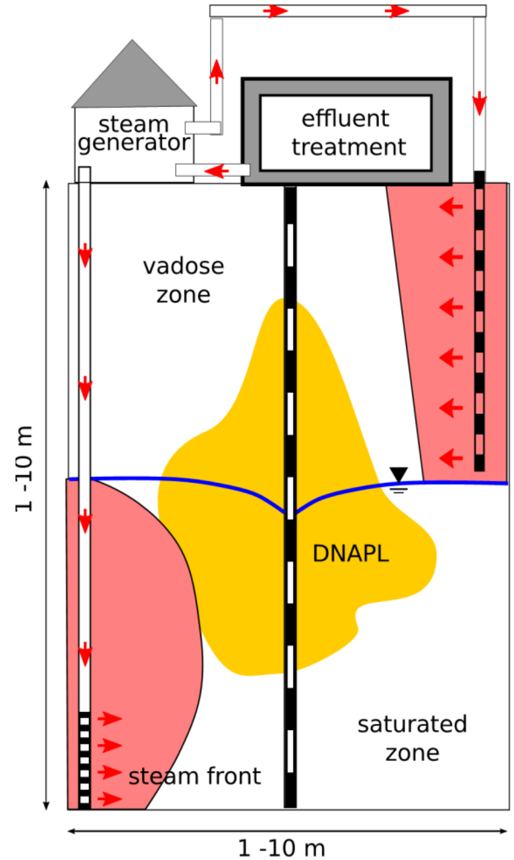
University of Stuttgart
Germany

Hypothetical Field Scale Application:

- Steam injection into fully water-saturated soil

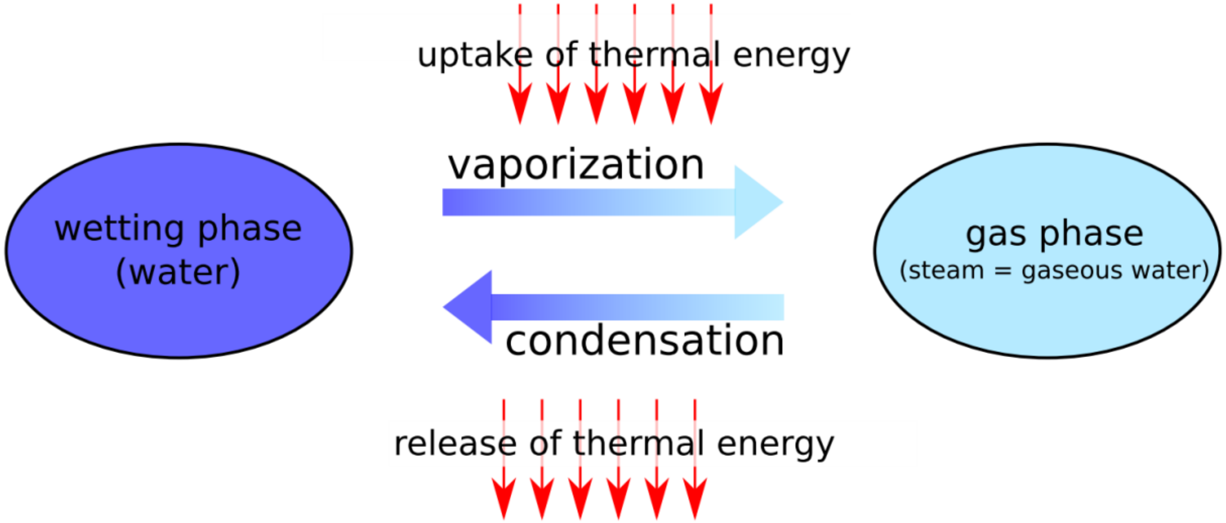
- Problem: Effective range of injection well (thermal radius of influence) limited by buoyancy effects

- Approach: Soil preheating by hot water pre-injection →increase TRI

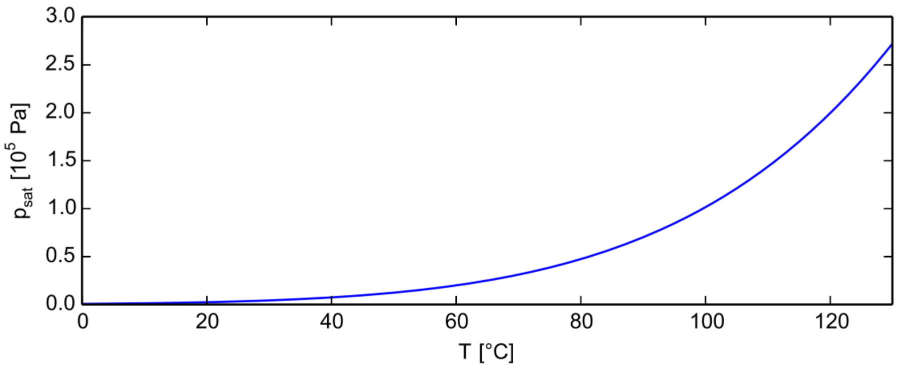


- Model used: 2p1cni (currently in *dumux-devel*)

2p1cni model



phase state dependent primary variables



present phases	primary variables
water	p_g, T
gas	p_g, T
water + gas	$p_g(T)$, S_w

2p1cni model

balance equations

storage term

diffusive fluxes

advective fluxes

mass

$$\phi \frac{\partial \sum_\alpha (\rho_\alpha S_\alpha)}{\partial t} - \sum_\alpha \operatorname{div} \left\{ \rho_\alpha \frac{k_{r\alpha}}{\mu_\alpha} \mathbf{K} (\mathbf{grad} p_\alpha - \rho_\alpha \mathbf{g}) \right\} - q^w = 0$$

energy

$$\phi \frac{\partial \sum_\alpha (\rho_\alpha u_\alpha S_\alpha)}{\partial t} + (1-\phi) \frac{\partial \rho_s c_s T}{\partial t} - \operatorname{div}(\lambda_{pm} \, \mathbf{grad} T) - \sum_\alpha \operatorname{div} \left\{ \rho_\alpha h_\alpha \frac{k_{r\alpha}}{\mu_\alpha} \mathbf{K} (\mathbf{grad} p_\alpha - \rho_\alpha \mathbf{g}) \right\} - q^h = 0$$

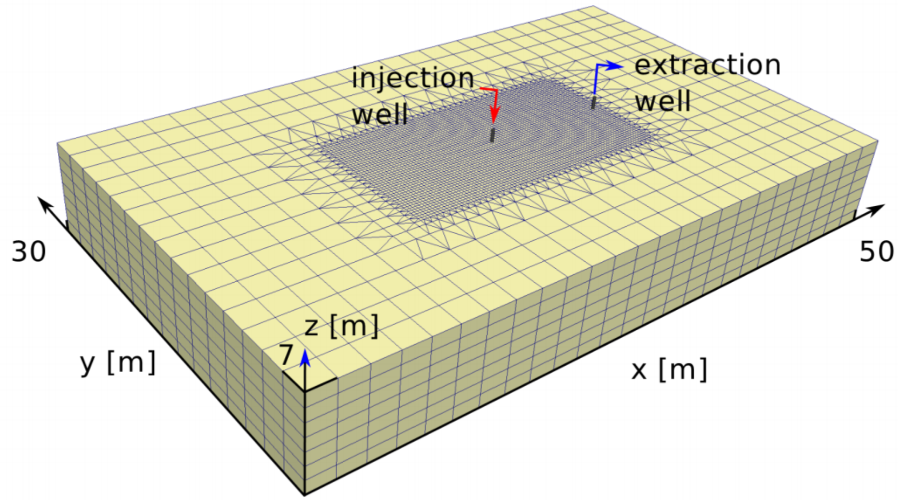
$$S_w + S_g = 1$$

$$p_w = p_g - p_c$$

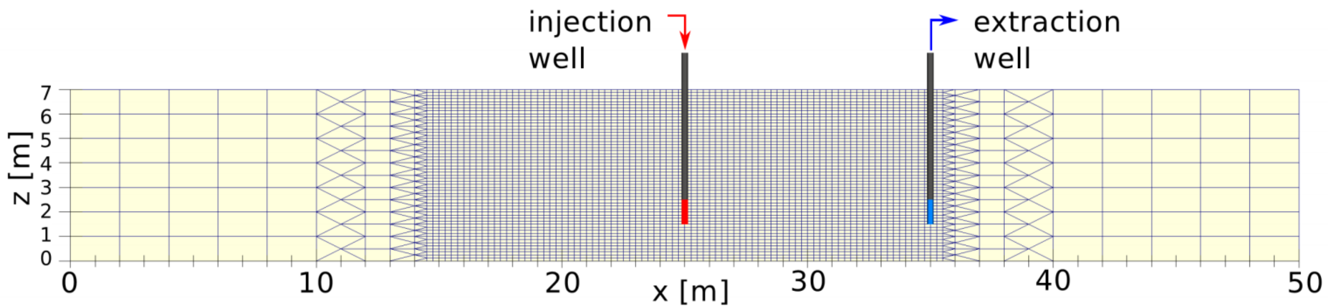
$$p_c = p_c(S_w)$$

$$T = T(p_g)$$

Problem Setup



$$K_{xx} = K_{yy} = 1 \times 10^{-11} \text{ m}^2$$
$$K_{zz} = 1 \times 10^{-12} \text{ m}^2$$
$$\Phi = 0.4$$
$$Q_s = 180 \text{ kg/h}$$



Boundary conditions:

- No flow Neumann BC on bottom
- Dirichlet BC for pressure and temperature on top and lateral sides

$$dx = dy = 2\text{m}$$
$$dz = 1\text{m}$$

$$dx = dy = 0.25\text{m}$$
$$dz = 0.125\text{m}$$

Local grid refinement (3x):

$$15 \text{ m} < x < 35 \text{ m}$$
$$10 \text{ m} < y < 20 \text{ m}$$

Local Grid Refinement

- Simple modification of class *DgfGridCreator (dumux/dumux/io/dgfgridcreator.hh)*

- Original function:

```
static void makeGrid(const std::string& dgfFileName)
    {
        gridPtr_ = GridPointer(dgfFileName.c_str(), Dune::MPIHelper::getCommunicator());
    };
```

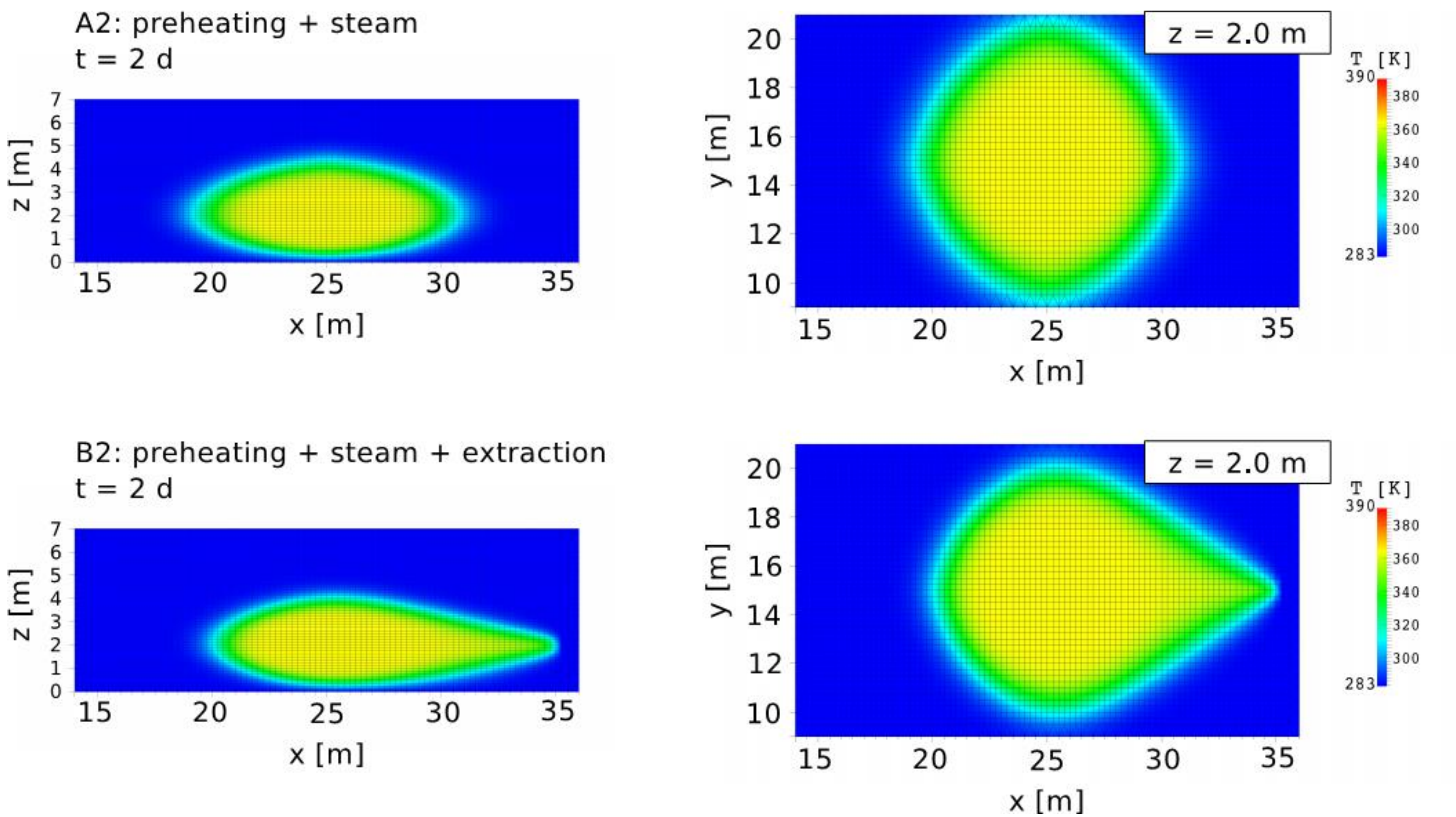
modified function:

```
static void makeGrid(const std::string& dgfFileName) {

 gridPtr_ = GridPointer(dgfFileName.c_str(), Dune::MPIHelper::getCommunicator());
 GridView gridView = gridPtr_->leafGridView();

 // loop for multiple refinement
 for(int i = 0; i < numRefinements; ++i) {

     // iterate over all elements
     for(ElementIterator eIt = gridView.template begin<0>(); eIt != gridView.template end<0>(); ++eIt) {

         // iterate over all element corners (nodes)
         for (int i = 0; i < eIt->geometry().corners(); ++i) {

             // get the node's global position
             GlobalPosition globalPos = eIt->geometry().corner(i);
             Scalar x = globalPos[0];
             Scalar y = globalPos[1];
             Scalar z = globalPos[2];

             // check whether the nodes lies within the region of interest
             if(x > lowerLeft[0]-eps && x < upperRight[0]+eps &&
                y > lowerLeft[1]-eps && y < upperRight[1]+eps &&
                z > lowerLeft[2]-eps && z < upperRight[2]+eps)
                {
                  gridPtr_->mark( 1, *eIt ); // mark the node for refinement
                }
         }
     }

  // adaption and refinement procedure
  gridPtr_->preAdapt();
  gridPtr_->adapt();
  gridPtr_->postAdapt();

 }
}
```

given by input file
→ adaptable on runtime

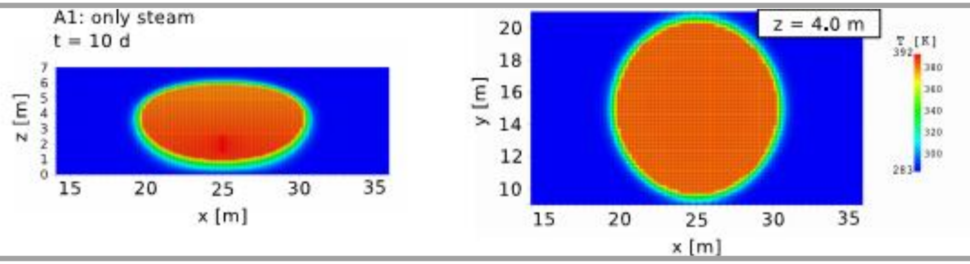
Simulation Results

- Result of preheating for 48 h:

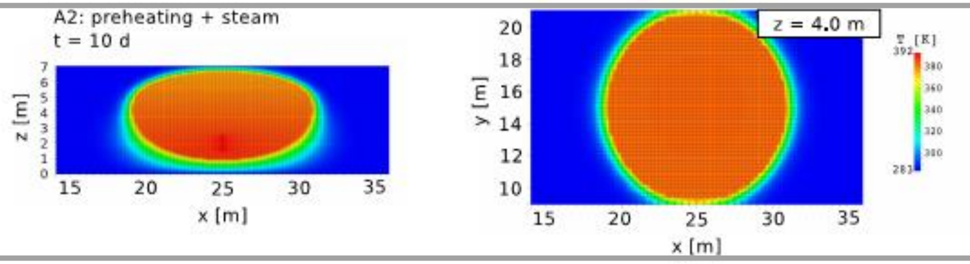


Simulation Results

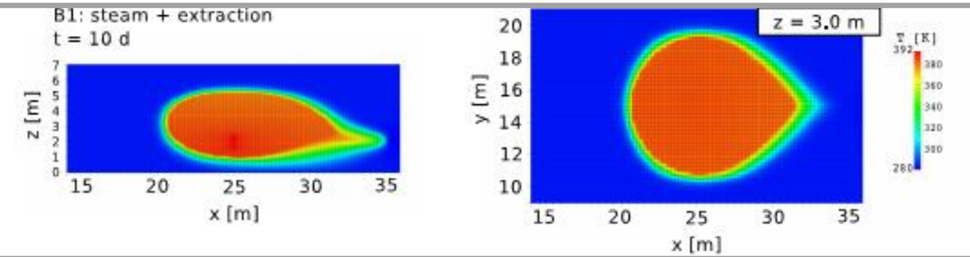
only steam injection



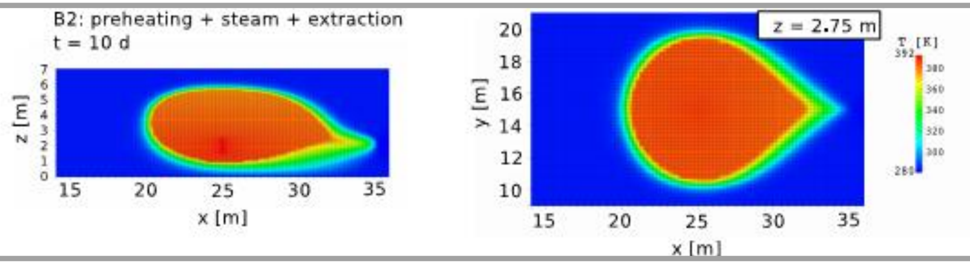
preheating + steam injection



steam injection + extraction well



preheating + extraction well + steam injection



Summary:

- runtime: 50 h on 4 processes
- 231003 nodes

- (static) local grid refinement with UG is easy to implement and can be adapted on runtime
- also works in parallel execution

- soil preheating can increase the TRI by about 10 %

Implementation of a robust model

- blocking of spurious fluxes: