GeMA – Modelling of Reservoir production

Version 1.0

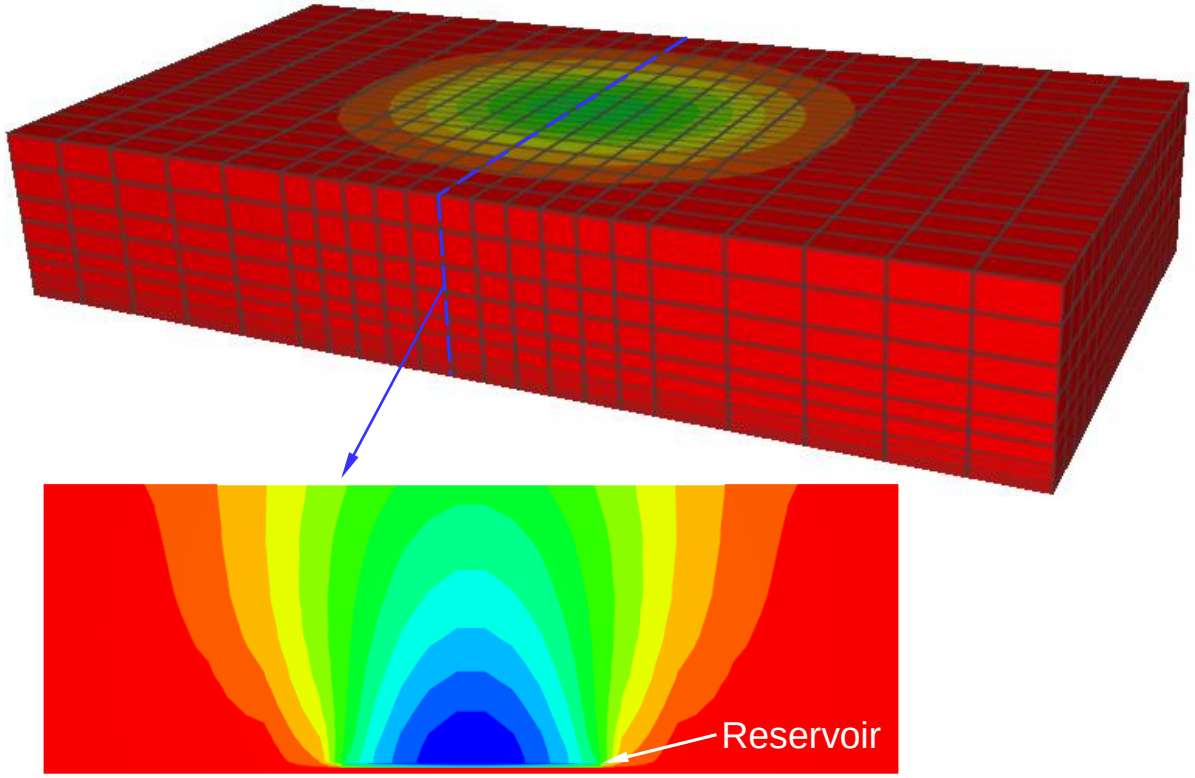
Key Example Points

This example shows:

- How to setup models to analyze subsidence in reservoirs.
- How to initialize the in-situ stresses and pore pressure in GeMA.
- The problem shows the reservoir displacements owing to the porous-flow migration.

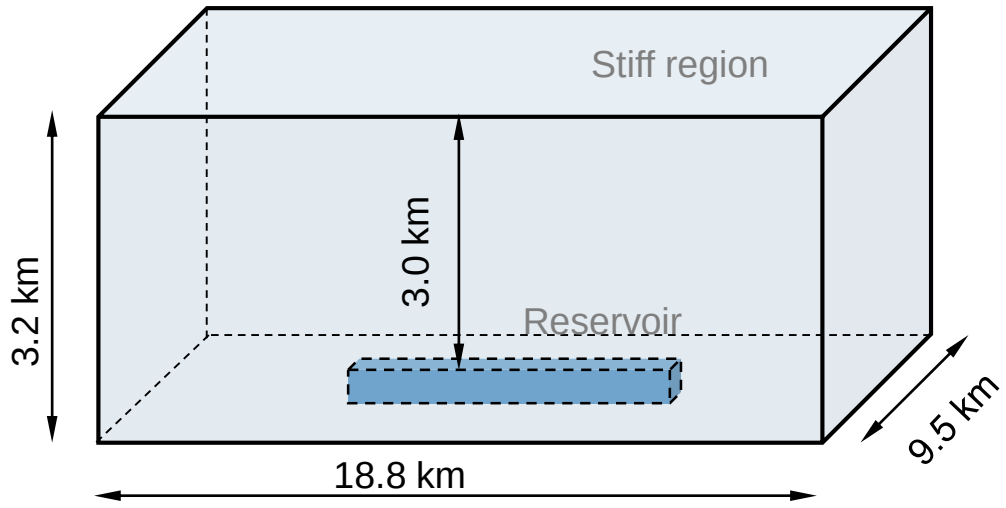
The examples

The example illustrates the fluid flow and geomechanical behavior of a reservoir after 12-years of production. The final displacement field is detailed below.



The problem

This problem presents a soft reservoir (6.7x3.5x0.076 km) within a stiff region (18.8x9.5x3.2 km). The reservoir is near 3 km below the seabed.



- Stress gradient of 22 kPa/m and pore pressure gradient of 9.78 kPa/m.
- Vertical wellbore in the center of the reservoir with production rate of 0.092m^3/s.

Simulation file: coupledHMRes3.lua

Model file: Hydromechanical properties

```
PropertySet
{
  id          = 'MatProp',
  typeName    = 'GemaPropertySet',
  description = 'Material properties',
  properties  = {
    {id = 'E',   description = 'Elasticity modulus',           unit = 'kPa'},
    {id = 'nu',  description = 'Poisson ratio'},
    {id = 'K',   description = 'Hydraulic permeability in x',  unit = 'm/s'},
    {id = 'gw',  description = 'Specific weight of water',     unit = 'kN/m^3'},
    {id = 'Kww', description = 'Bulk modulus of water',        unit = 'kPa'},
    {id = 'Pht', description = 'Porosity'},
    {id = 'materialHM', description = 'Hydro-mechanics material type',
                        constMap=constants.CoupledHMFemPhysics.materialModels},
  },
  values = {
    {E = 6.8948e+06, nu =0.25, materialHM = 'poroElastic', K = 9.87e-16, gw = 1.00e+01, Pht = 0.25,
     Kww = 2.30e+06},
    {E = 68948, nu = 0.25, materialHM = 'poroElastic', K = 9.87e-07, gw = 1.00e+01, Pht = 0.25,
     Kww = 2.30e+06},
  }
}
```

Table with material properties required by coupled solid elements

Model file: Initial pore pressure

The initial pore pressure is computed as a function of the height, considering a fluid pressure gradient of 9.78 kPa/m .

```
NodeFunction {
  id = 'pore',
  parameters = { {src = 'coordinate', unit = 'm', dim = 3} },
  method = function(z)
     local poro = 0.0
     if ( z < 0 and z >= -3300) then
         poro = - 9.7947782*z
     end
     return poro
  end
}
```

Model file: Boundary conditions

```
BoundaryCondition {  -- Prescribed displacement
  id   = 'bc',
  type = 'node displacements',
  mesh = 'mesh',
  properties  = {
    {id = 'ux',  description = 'Fixed node displacement in the X direction', unit = 'm',  defVal = -9999},
    {id = 'uy',  description = 'Fixed node displacement in the Y direction', unit = 'm',  defVal = -9999},
    {id = 'uz',  description = 'Fixed node displacement in the z direction', unit = 'm',  defVal = -9999},
  },
  nodeValues = {
  --  {    node,       x,            y,                 z}
      {       1,   0.0000e+00,  0.0000e+00 ,        nil},
      . . .
  }
}
```

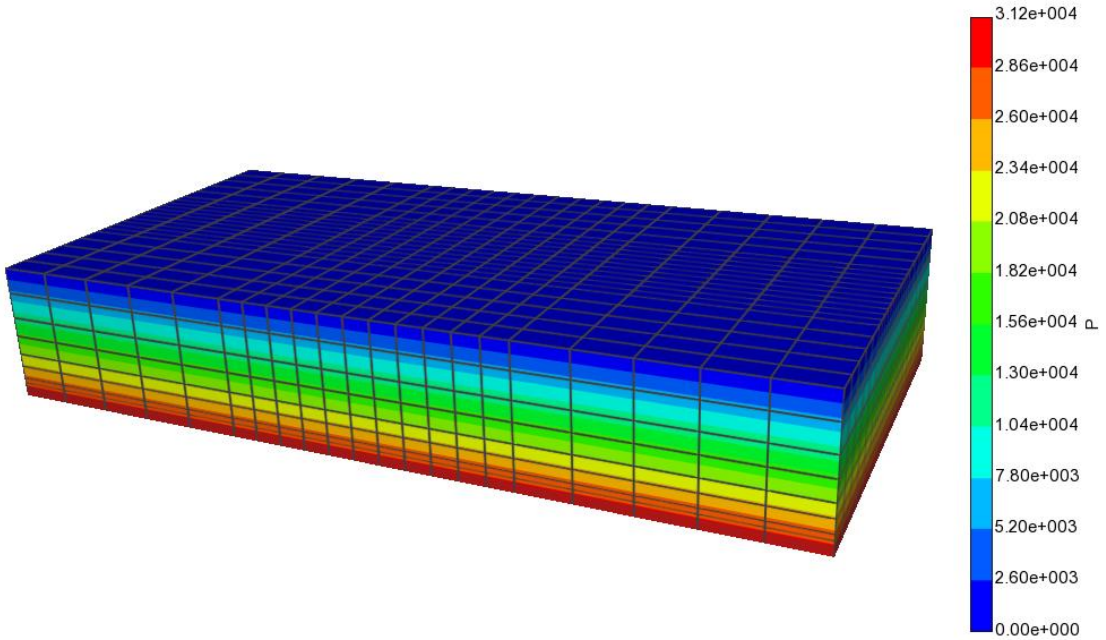
```
BoundaryCondition {   -- Fluid flow definitions
  id   = 'bfm',
  type = 'node pore flow',
  mesh = 'mesh',
  properties  = {
    {id = 'qw',  description = 'concentrated pore flow', unit = 'm^3/s', defVal = -9999},
  },
  nodeValues = {
   -- { node, qw } - fluid flow (-) injection and (+) depletion
      { 3899 , 0.0038333},
      . . .
  }
}
```

Model file: Sets initial stresses for each element

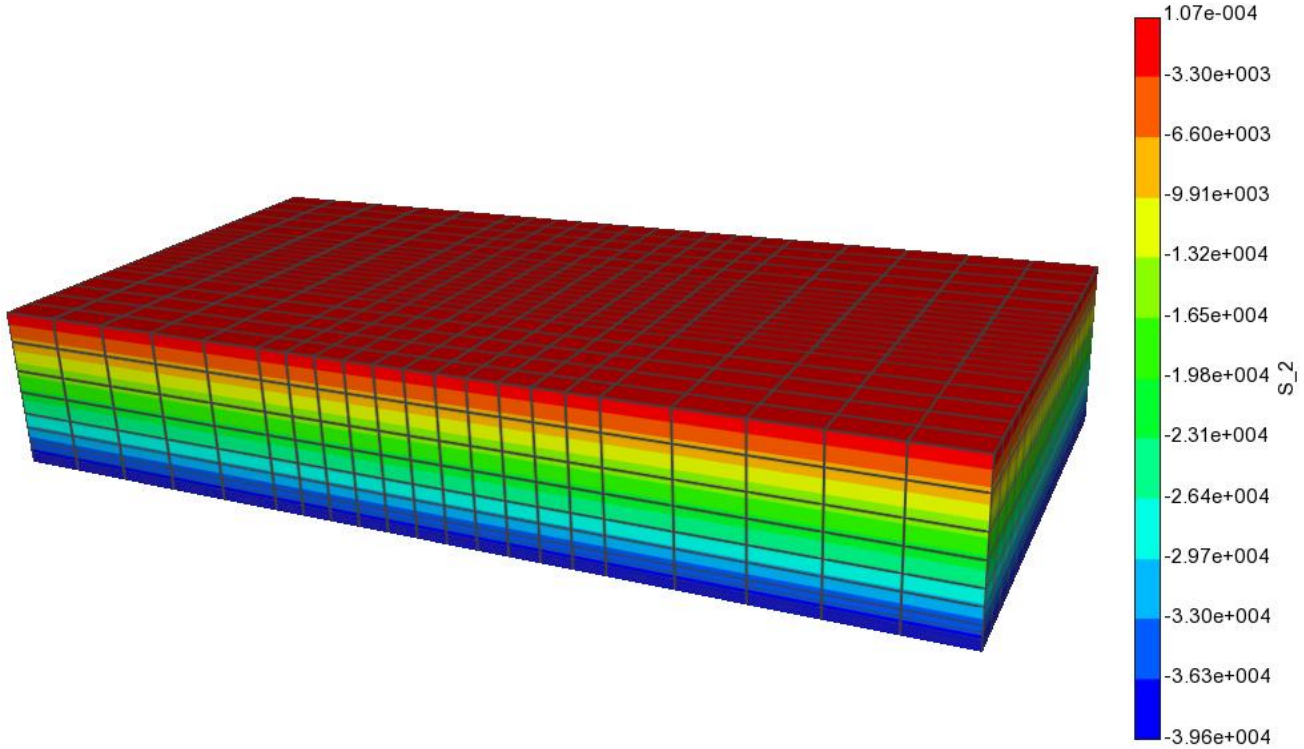
```
function initialCondition ()
  print('Initial condition...\n'  )
  -- Gets mesh object
  local mesh = assert(modelData:mesh('mesh'))

  -- Gets stress accessor for the old state
  local accS = assert(mesh:gaussAttributeAccessor('S', 1, true))

  -- Gets node coordinate accessor
  local coordAc = mesh:nodeCoordAccessor()
  --local Ko1 = 0.20/(1.0 - 0.20)   -- Ko value for nu = 0.20
  local Ko = 0.5
  -- For each element
  for i=1, mesh:numCells() do
    -- Gets element object
    local e = mesh:cell(i)

    -- Get integration rule object
    local ir  = mesh:elementIntegrationRule(e:type(), 1)
    -- Get element shape object
    local shp = e:shape()

    -- Fill node matrix
    local Xnode = e:nodeMatrix(coordAc, true)
    -- check number of integration rule
    assert(ir:numPoints() == 8)
     -- For each integration point
    for j=1, ir:numPoints() do
      -- Get integration point coordinates
      local ip, w = ir:integrationPoint(j)
      -- Integration point in cartesian coordinates
      local coord = shp:naturalToCartesian(ip, Xnode)
      -- fill stress acording
      local Sv = 0.0
      local Sh = 0.0
      if (coord(3) < 0.0 and coord(3) >= -3300) then
        Sv = 12.43914213*coord(3)
        Sh = Ko*Sv
      end

      local v = {Sh, Sh, Sv, 0, 0, 0} -- stress vector
      accS:setValue(e, j, v)
    end
  end
end
```

The process execution script is very similar to that presented for other analysis. Refer to the solution file.

Initial conditions

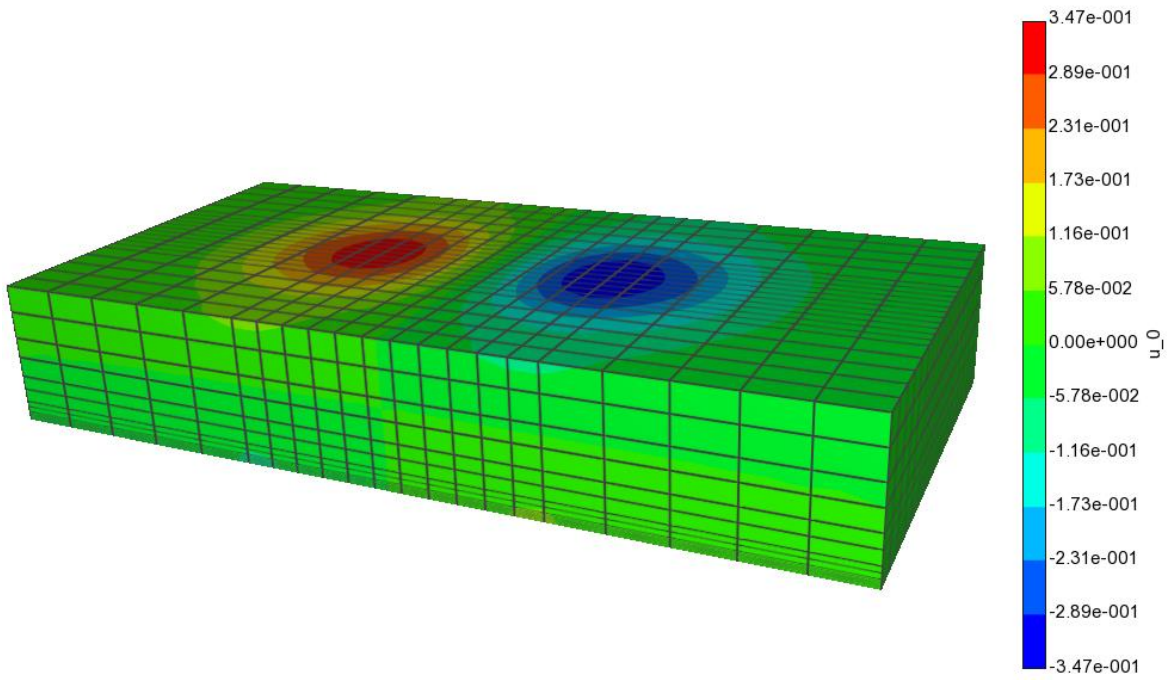


Pore pressure (kPa)

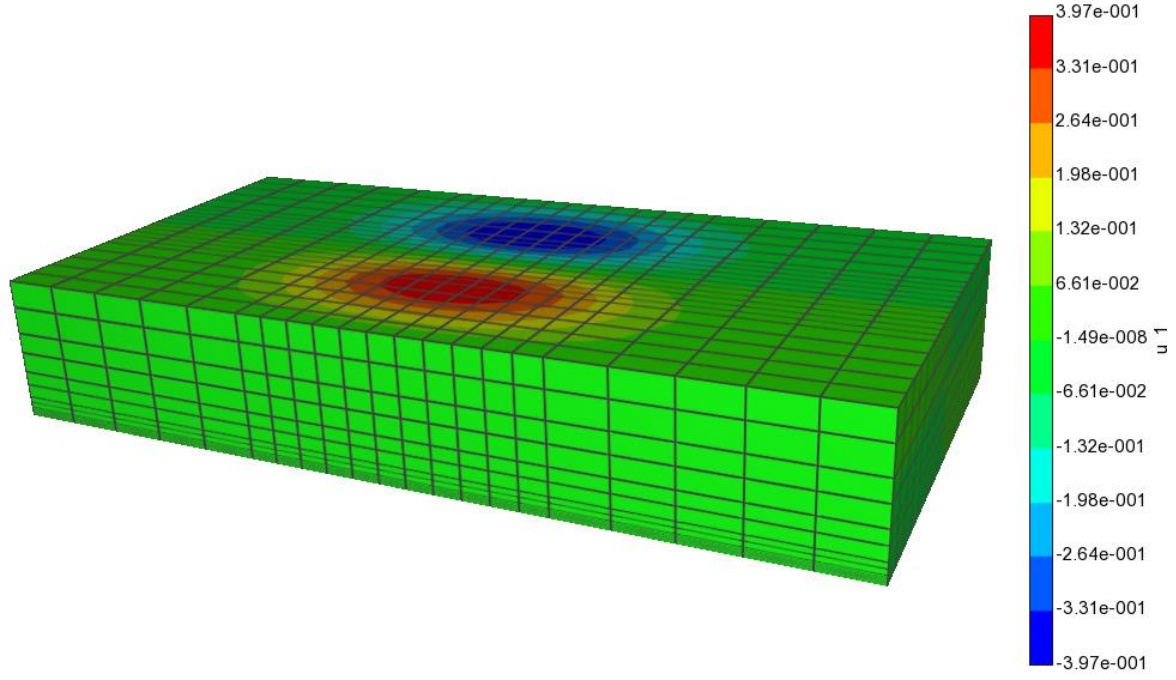


Initial stresses (kPa)

Results at time = 4400 days

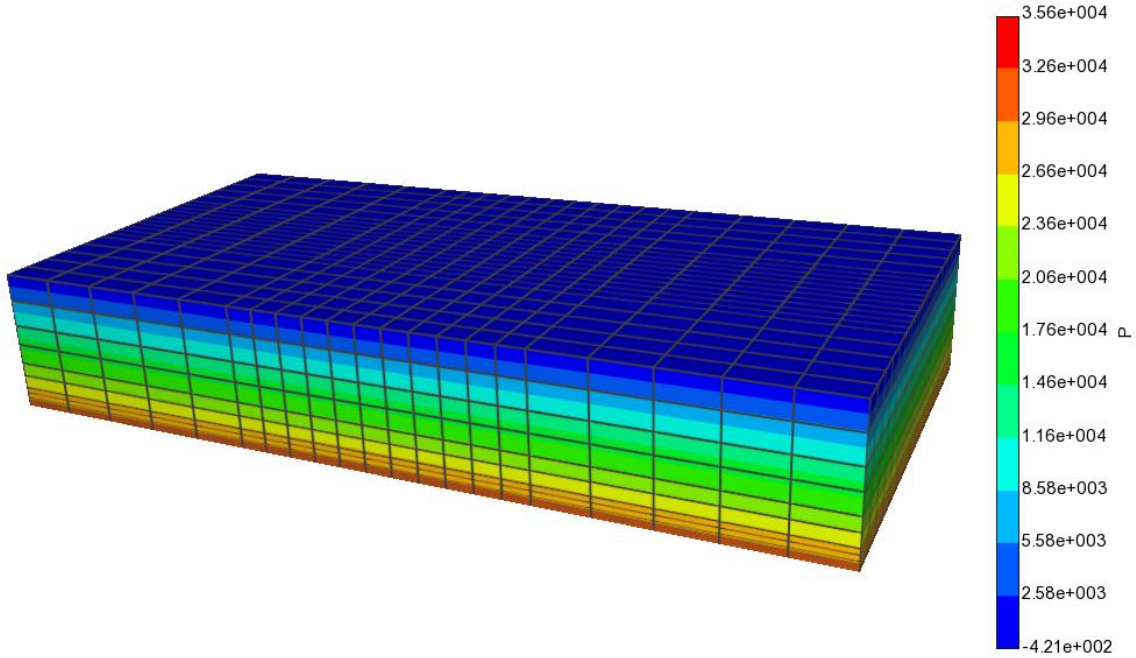


Displacement in x-direction

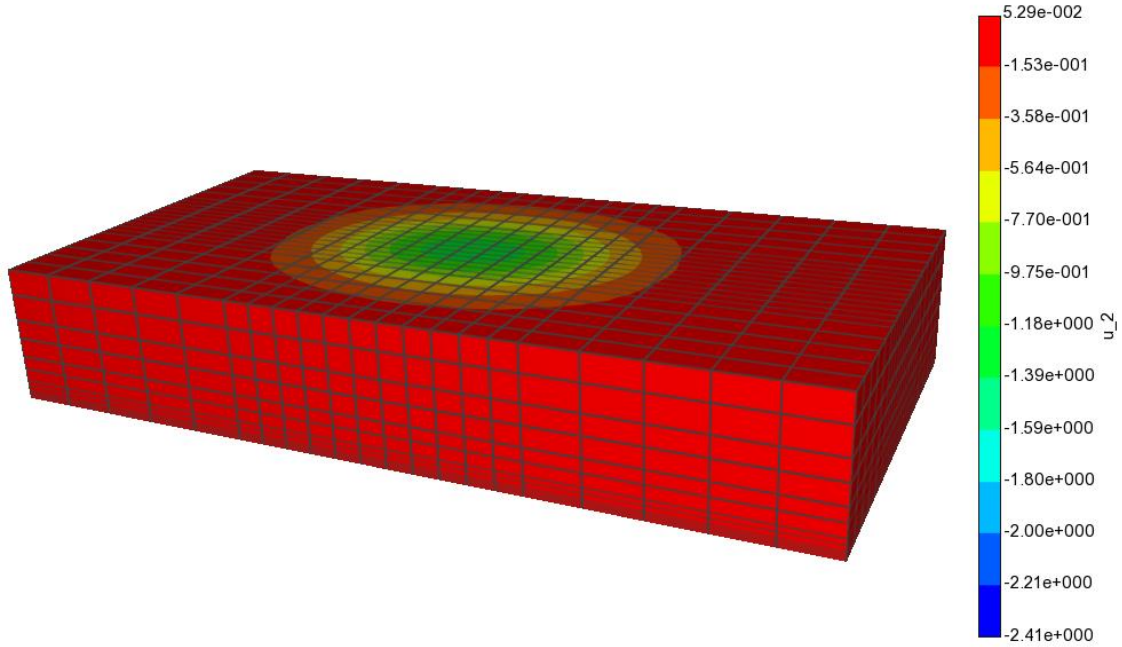


Displacement in y-direction

Results at time = 4400 days

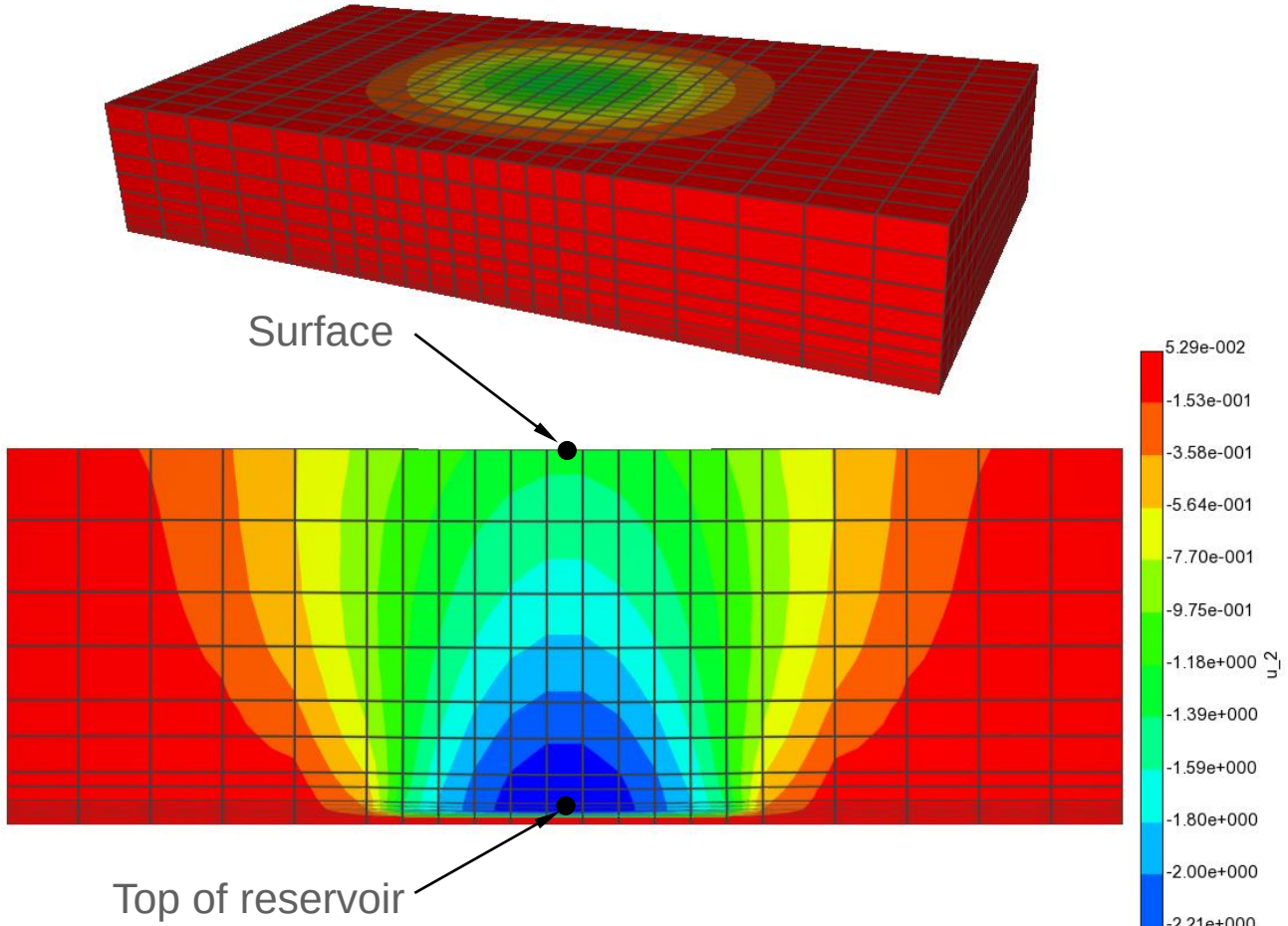


Pore pressure (kPa)



Vertical displacement (m)

Results at time = 4400 days



Vertical displacement (m)