GeMA
Geo Modelling Analysis

30 ANOS
TECGRAF
PUC-RIO

ERAS

# Training Shell: Example 01 3D Modelling of Reservoir Production

***Multiphysics Modeling and Simulation Group***

**Tecgraf Institute / PUC-Rio**

**Training Meeting**

**September 2020**

# Key Example Points

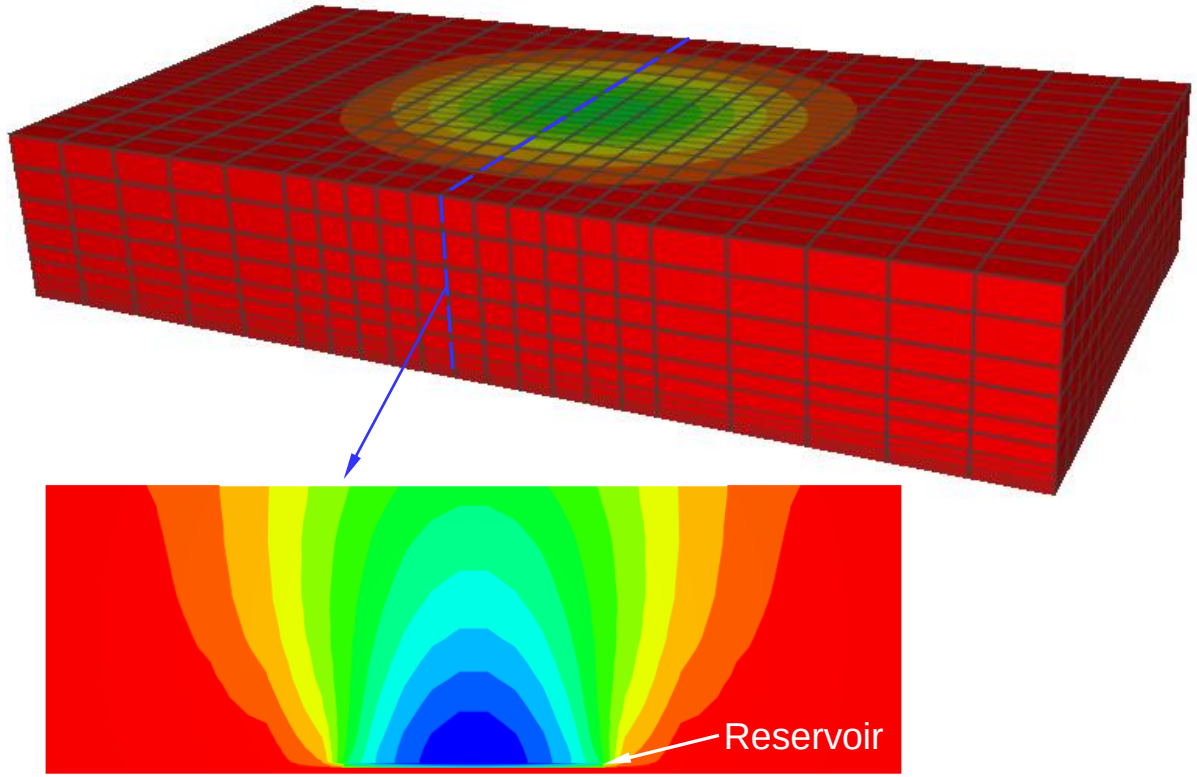
- **<u>This example shows:</u>**
  - How to setup models to analyze subsidence in reservoirs;
  - How to initialize the in-situ stresses and pore pressure in GeMA;
  - The reservoir displacements owing to the porous-flow migration.

➢ Documentation: http://www.tecgraf.puc-rio.br/gema

- ***Tutorial***
- ***Examples***
- ***Reference manual***

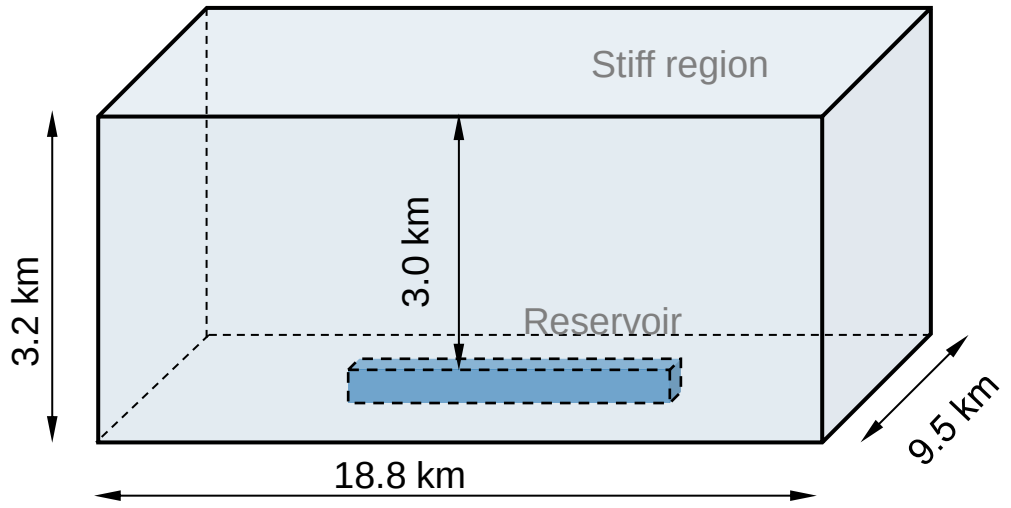
# Example Overview

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



# Problem

- This problem presents a soft reservoir (6.7 x 3.5 x 0.076 km) within a stiff region (18.8 x 9.5 x 3.2 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 0.092$m^3$/s.

Simulation file: coupledHMRes3.lua

# GeMA Input Files

*.lua

Main file that stores the analyzes description and loads the remaining model files

*_model.lua

File that gathers the model data (nodes, elements, boundary conditions, etc.).

*_solution.lua

File that gathers information regarding the execution of the analyzes (solver settings, script, orchestration details, etc)

# Running File: *.lua

```
-----------------------------------------------
-- The model description
-----------------------------------------------
Simulation
{
  name         = 'coupledHMRes3',
  modelVersion = '1.0',
  description = 'This problem shows a soft productive reservoir that is contained \n'..
                'within a stiff nonpay region.\n'
}

-- Load model and solution files for this problem
dofile('$SIMULATIONDIR/$SIMULATIONNAME_model.lua')
dofile('$SIMULATIONDIR/$SIMULATIONNAME_solution.lua')
```

**Model description and file information to run the numerical simulation.**

# Model file: State Variables

```
-----------------------------------------------------
--  State variables
-----------------------------------------------------
StateVar{id = 'u', dim = 3, description = 'Displacements in
the X and Y directions', unit = 'm', format = '8.4f',
groupName = 'mechanic'}
StateVar{id = 'P', description = 'Pore pressure', unit =
'kPa', format = '8.4f', groupName = 'hydraulic'}
```

- Variable definition for the problem:

- ***id*** refers to the names of each variable (u = displacement; P = pore pressure);
- ***dim*** = 3 refers to a three-dimensional (3D) problem;
- The ***description*** field is freely available to the user and serves to give a brief information about the variable mentioned in ***id.***
- ***unit*** provides the unit of the state variable, while format indicates the format (precision) in which this variable will be printed;
- ***groupName*** classifies the physics associated with the model you are working on (‘mechanic’, ‘hydraulic’, etc).

# 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},
  }
}
```

➢ Definition of material properties:

- ***id*** refers to the name of the set of properties to be defined by the user;
- ***typeName*** refers to the name defined in the GeMA structure for assigning material properties. This string cannot be changed by the user.
- The ***description*** field is free to fill in and describes the table to be written below.
- ***properties*** = { } is a LUA table where the user must enter the name corresponding to each of the relevant parameters for his material model. Additionally, you must enter a description of each parameter freely and the corresponding unit.
- ***values*** = { } is a LUA table where the user must enter the numeric values for each of the parameters mentioned and described in the properties table.
- ***constMap*** accesses the table for the type of material model to which the analysis will be performed.

# Model File: Nodes

```
-- Nodal coordinates
local mesh_nodes = {
   {     0.000,     0.000,     0.000},
   {     0.000,   609.600,     0.000},
   {     0.000,     0.000,  -609.600},
   {     0.000,   609.600,  -609.600},
   {     0.000,  1219.200,     0.000},
   {     0.000,     0.000, -1219.200},
   {  1219.200,     0.000,     0.000},
   {  1219.200,   609.600,     0.000},
   {     0.000,  1219.200,  -609.600},
   {  1219.200,     0.000,  -609.600},
   {     0.000,   609.600, -1219.200},
   {  1219.200,   609.600,  -609.600},
   {     0.000,     0.000, -1676.400},
                      ......
                      .....
                      .....
                      .....
                      .....
```

- Definition of Mesh Information:

local ***mesh_nodes*** = { } is the LUA table that gathers the {X, Y, Z} coordinates of the model nodes. The table definition syntax bellow must be followed:

```
mesh_nodes location = {
     {X1, Y1, Z1},
     {X2, Y2, Z2},
     {X3, Y3, Z3},
     {X4, Y4, Z4},
}
```

Note carefully that nodal coordinates must be defined between commas, including when moving from one line to another.

# Model File: Elements

Table with mesh incidence for Hex8 3D element

```
local mesh_HEX_1_elements = {
   {    503,    957,    947,    485,    491,    944,    933,    473},
   {    199,    503,    485,    191,    189,    491,    473,    183},
   {     44,    199,    191,     38,     40,    189,    183,     34},
   {     12,     44,     38,      8,     10,     40,     34,      7},
   {      4,     12,      8,      2,      3,     10,      7,      1},
   {    546,    993,    957,    503,    534,    974,    944,    491},
   {    228,    546,    503,    199,    213,    534,    491,    189},
   {     56,    228,    199,     44,     50,    213,    189,     40},
   {     19,     56,     44,     12,     15,     50,     40,     10},
   {     11,     19,     12,      4,      6,     15,     10,      3},
   {    547,    990,    977,    532,    503,    957,    947,    485},
                                  ....
                                  ....
                                  ....
                                  ....
                                  ....
```

# Model File: Mesh

```
------------------------------------------------------------
--  Mesh definition
------------------------------------------------------------
Mesh
{
  -- General mesh attributes
  id          = 'mesh',
  typeName    = 'GemaMesh.elem',
  description = 'mesh discretization',

  -- Mesh dimensions
  coordinateDim  = 3,

  -- State vars stored in this mesh (per node)
  stateVars = {'u','P'},

  -- Mesh node coordinates
  nodeData = mesh_nodes,

  -- Element property
  cellProperties = {'MatProp'},

  -- Element data
  cellData = mesh_elements,

  -- Integration Rule
  elementRules = {
    {hex8 = 2}
  },
}
```

➢ Loading Mesh Information:

- ***id*** is the identifier containing the mesh name. ***typeName*** must have the syntax respected by the user. The ***description*** field is free to be filled in by the user;
- ***coordinateDim*** indicates the size of the finite element mesh. It must accompany the same dimension previously defined for the model;
- ***stateVars*** stores the displacement and pore pressure variables per node in the defined mesh. ***nodeData*** loads the information from the model nodes;
- ***cellProperties*** and ***cellData*** load the information with respect to the previously defined solid elements.
- ***elementRules*** = { } is the LUA table that specifies the type of numerical integration to be performed during finite element analysis. It is necessary to assign the number of integration points used by the element.

# 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},
      . . .
  }
}
```

**'0' refers to a fixed displacement at the corresponding direction, while 'nil' refers to free displacement at the corresponding direction.**

```
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: Boundary Conditions

```
--BC for prescribed fixed pore pressure (called by the geostatic step)
BoundaryCondition {
  id   = 'bpm',
  type = 'node pore pressure',
  mesh = 'mesh',

  properties  = {
    {id = 'P',  description = 'Fixed node pore pressure', unit = 'kPa',
defVal = -9999, functions = true},
  },

  nodeValues = {
   {        1 , 'pore'},
   {        2 , 'pore'},
   {        3 , 'pore'},
   {        4 , 'pore'},
   {        5 , 'pore'},
   {        6 , 'pore'},
         ....
```

**Creation of a boundary condition regarding the fixed pore pressure called by the geostatic step.**

# Solution File: Numerical Solvers

```
NumericalSolver
{
  id           = 'solver',
  typeName     = 'LisSolver',
  description  = 'Iterative matrix solver',
  solverMethod = 'bicgstab',
  precond      = 'ilut',
  tol = 1e-12,
  printIterations = true,
  --printResidual = true,
  --printTimes = true,
-- printLisConsoleMessages = true,
  maxIter = 5000,
 --gmresRestart = 40,
  --threads = 1,
  -- adds = true,
}
```

- Solver Information:

- ***NumericalSolver*** = { } is the LUA structure that defines which linear solver will be used to solve the [K] {u} = {f} system. The string typeName = '*LisSolver*' defines that the Lis iterative solver plugin will be used.

# Solution File: Physical Methods

```
------------------------------------------------------------
--  Physical Methods
------------------------------------------------------------
--  Physical Methods used in Geostatic step
PhysicalMethod {
  id       = 'Geostatic',
  typeName = 'CoupledHMFemPhysics.3D',
  type     = 'fem',

  mesh               = 'mesh',
  elementGroups      = {'HUP_1', 'HUP_2'},
  materials          = {'poroElastic'},
  isoParametric      = true,
  boundaryConditions = {'bpm','bc'},
  ruleSet            = 1,
  properties         = { material = 'materialHM' }
}

--  Physical Methods used in Depletion step
PhysicalMethod {
  id       = 'depletion',
  typeName = 'CoupledHMFemPhysics.3D',
  type     = 'fem',

  mesh               = 'mesh',
  elementGroups      = {'HUP_1', 'HUP_2'},
  materials          = {'poroElastic'},
  isoParametric      = true,
  boundaryConditions = {'bfm','bc'},
  ruleSet            = 1,
  properties         = { material = 'materialHM' }
}
```

**Geostastic Step**

**Depletion Step**

- Definition of Physics Information:

- ***Physical Method*** = { } is the LUA structure that defines the basic information regarding the physics of the problem to be carried out numerically. Again it contains an ID that can be freely defined by the user. The **typeName** variable accesses the physical type (“CoupledHMFemPhysics.3D”) of GeMA. There are several physics that can be accessed, see detailed framework manual to properly access the desired physics. The type variable must also be kept as “**fem**”.

- ***mesh, materials and boundaryConditions*** must follow the information detailed above.

# Solution File: Process execution script

```
-------------------------------------------------------
Process execution script
-------------------------------------------------------

local solverOptionsGeost = {
  geostaticType = 'fixedNode',
}

local solverOptions2 = {
  type               = 'transient nonlinear',
  timeMax            = 3.8016e8, --500 days (3.8016e8 = 4000 days)
  timeInitIncrement  =  100,
  timeMinIncrement   =  0.1,
  timeMaxIncrement   = 1.0e+7,
  iterationsMax      = 15,
  tolerance          = {mechanic =1.000E-04, hydraulic =1e-4},
  geostaticType      = 'fixedNode',
}
```

# Solution 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
      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 analyses. Refer to the GeMA manual.

# Solution File: Initial Conditions

```
function ProcessScript()

 local mesh = modelData:mesh('mesh')
-- Get displacement values in mm
  local accU = assert(mesh:nodeValueAccessor('u', 'mm'))
-- Get Pressure values in MPa
  local accP = assert(mesh:nodeValueAccessor('P', 'MPa'))

  -- Initial condition
  initialCondition ()
  -------------------------------------
  print(' GEOSTATIC Step.....\n'  )
  -------------------------------------
  -- Create the solver model for geostatic step
  local solver = fem.init({'Geostatic'}, 'solver', solverOptionsGeost)
   -- history file
  local resFile =
io.open(translatePath('$SIMULATIONDIR/out/$SIMULATIONNAME.sim'), "w+")

  fem.geostatic(solver)
  io.addResultToMeshFile(file, 0)
  --save Displacement and pressure
  resFile:write("Time"," ","P(MPa)"," ","subsidenceTop(mm)","
","subsidenceReservoir(mm)","\n")

 local Pore = accP:value(2033)
  resFile:write(0," ", Pore," ", 0.0," ", 0.0,"\n")
```

***.sim file for monitoring results**

**Variables that will appear in *.sim file**

```
 -- Run transient analysis.  Solver returns a
suggested time-step for the next iteration

   dt = fem.step(solver, dt

  -- Save results
  io.addResultToMeshFile(file, (Time))
  local Pore = accP:value(2033)/1000.0
  local Ut = accU:value(3546)*1000.0
  local Ur = accU:value(3899)*1000.0
  resFile:write(Time," ", Pore," ", Ut(3)," ", Ur(3), "\n")
  resFile:flush()
```

**Space**

**resFile:flush () updates the *.sim file with the corresponding results**

```
Time  P(MPa)  subsidenceTop(mm)  subsidenceReservoir(mm)
0  29.8544839536  0  0
100  29.854484118767  -0.00065721197414466  -0.0094329967118759
300  29.854484579611  -0.0019742038029058  -0.028341251641042
700  29.854485963135  -0.0046153249417547  -0.066263722954619
1500  29.854490159585  -0.0099128796296625  -0.14229796480693
3100  29.854502143249  -0.0205306723855  -0.29445277753202
6300  29.854532624953  -0.041786873654043  -0.59786302814937
12700  29.854600064859  -0.084303745472852  -1.1997472124835
25500  29.854730957363  -0.16928945450925  -2.3833098282232
51100  29.854954131218  -0.33901739778337  -4.6741317479623
102300  29.855271922974  -0.67758234533975  -8.9889055876977
```

# Visualization

Project Root / subsidence / out

+ NEW

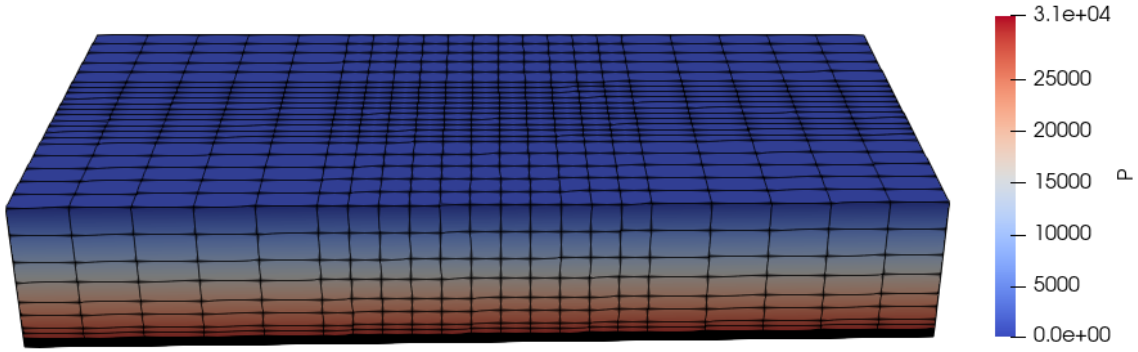
Search Files

User can download the folder files and open them using Paraview desktop version or view the results in Paraview Online Visualizer available in ERAS.

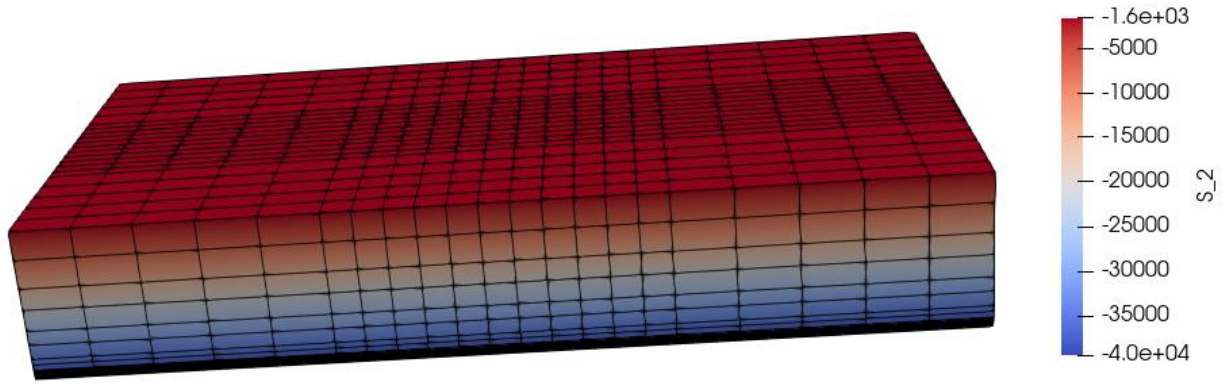
| Name | Modified | Size | Extension |
|---|---|---|---|
| coupledHMRes3.pos | 08/12/2020 17:05 | 181.5 MB | POS |
| coupledHMRes3.pvd | 08/12/2020 17:13 | 1.71 KB | PVD |
| coupledHMRes3.sim | 08/12/2020 17:05 | 1.28 KB | SIM |
| coupledHMRes30.vtu | 08/12/2020 17:12 | 1.26 MB | VTU |
| coupledHMRes30_gauss.vtp | 08/12/2020 17:12 | 6.74 MB | VTP |
| coupledHMRes31.vtu | 08/12/2020 17:12 | 1.62 MB | VTU |
| coupledHMRes310.vtu | 08/12/2020 17:12 | 1.95 MB | VTU |
| coupledHMRes310_gauss.vtp | 08/12/2020 17:12 | 8.32 MB | VTP |
| coupledHMRes311.vtu | 08/12/2020 17:12 | 1.98 MB | VTU |
| coupledHMRes311_gauss.vtp | 08/12/2020 17:12 | 8.36 MB | VTP |
| coupledHMRes312.vtu | 08/12/2020 17:12 | 2.03 MB | VTU |

.
.

# Initial Conditions

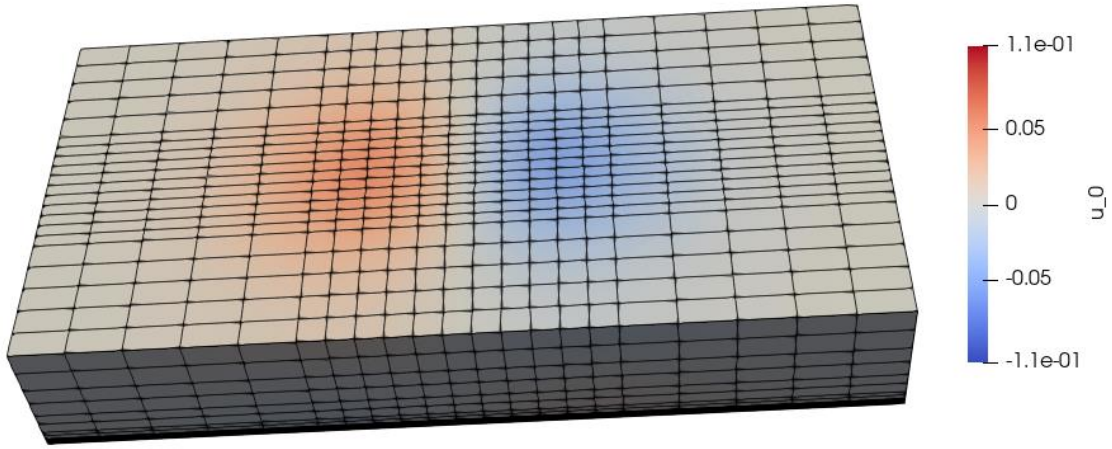



Pore Pressure (kPa)

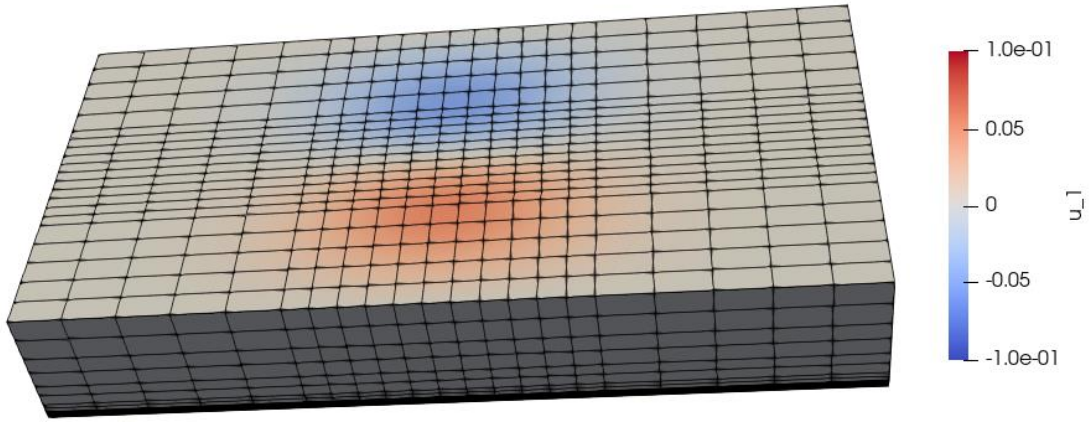



Initial Stresses (kPa)

# Results at time = 500 days

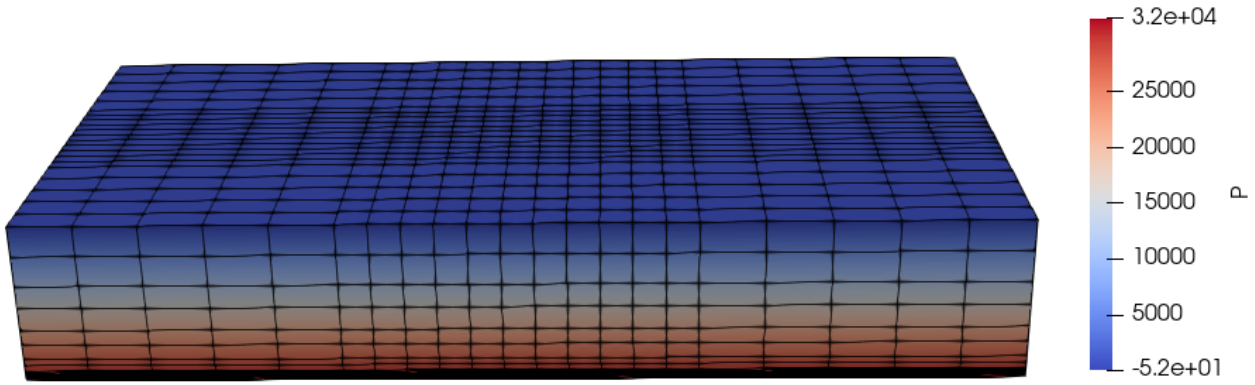



Displacement in x-direction (m)

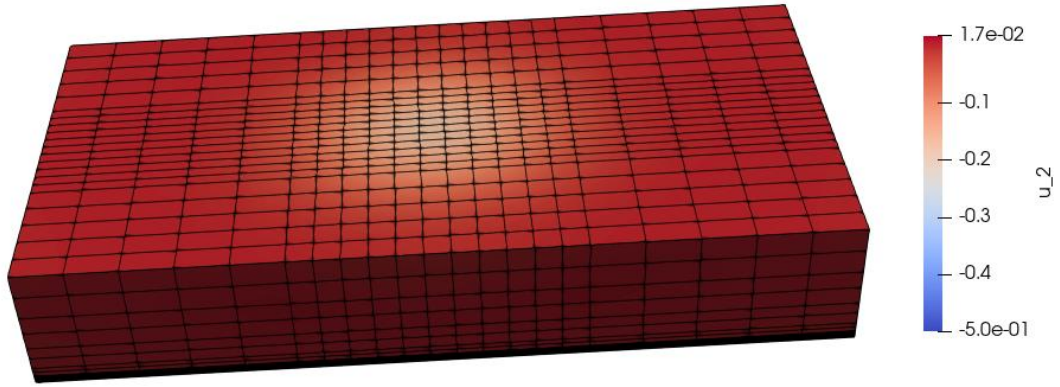



Displacement in y-direction (m)

# Results at time = 500 days

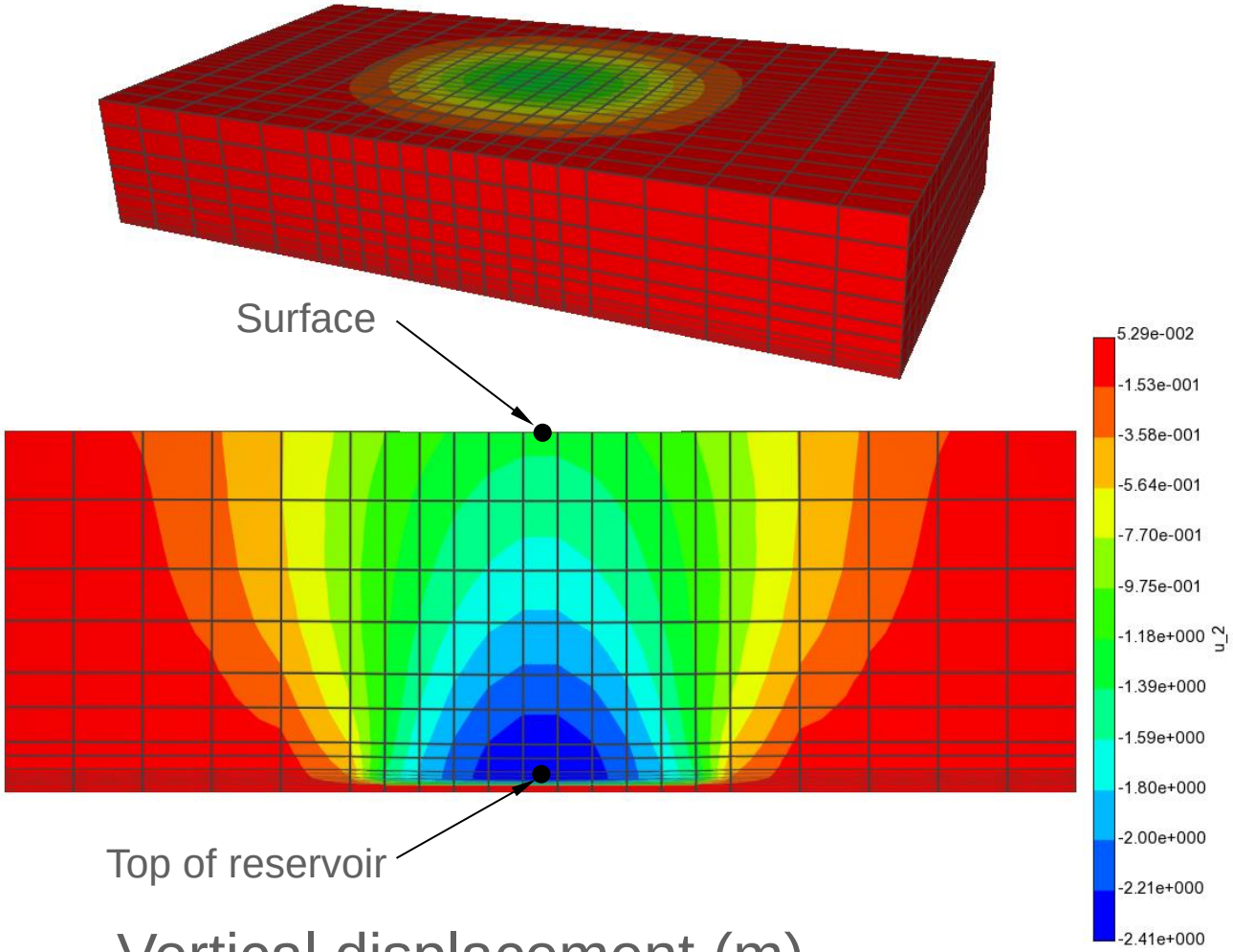



Pore Pressure (kPa)




Vertical Displacement (m)

# Results




Vertical displacement (m)