01/08/2018

GeMA – Surface subsidence analysis and reservoir compaction

Version 1.0

Example set purpose

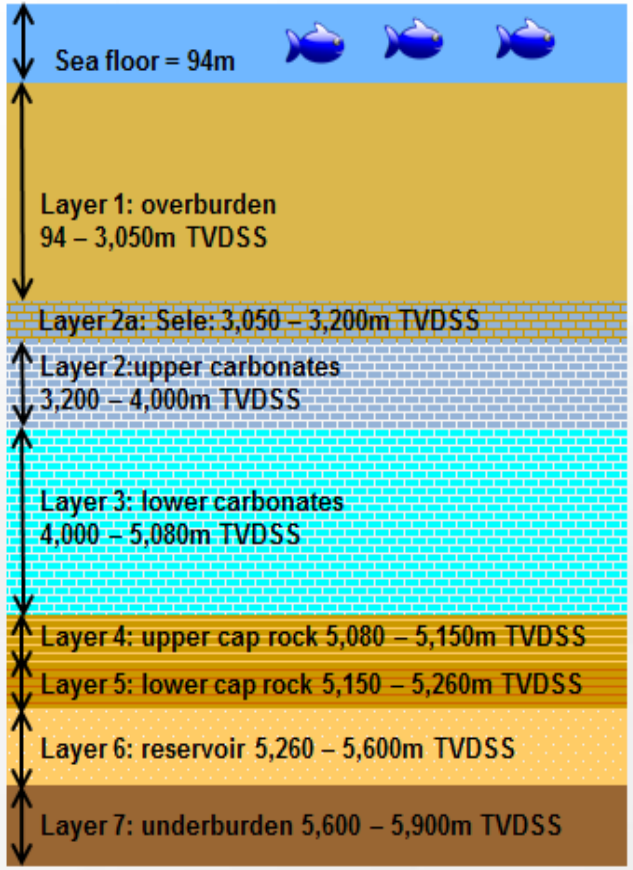
This examples show:

- How to setup the surface subsidence analysis for uncoupled and HM fully coupled models
- How to setup boundary conditions (displacement and pore pressure)
- How to replace a physics unit system
- Several different orchestration techniques for schemes with a geostatic step for external and internal forces equilibrium before the subsequent depletion step

1 – 2D UNCOUPLED SUBSIDENCE ANALYSIS

The Problem

- This example is the analysis of surface subsidence and compaction of an oil reservoir, of the North Sea field. The formations and properties are:



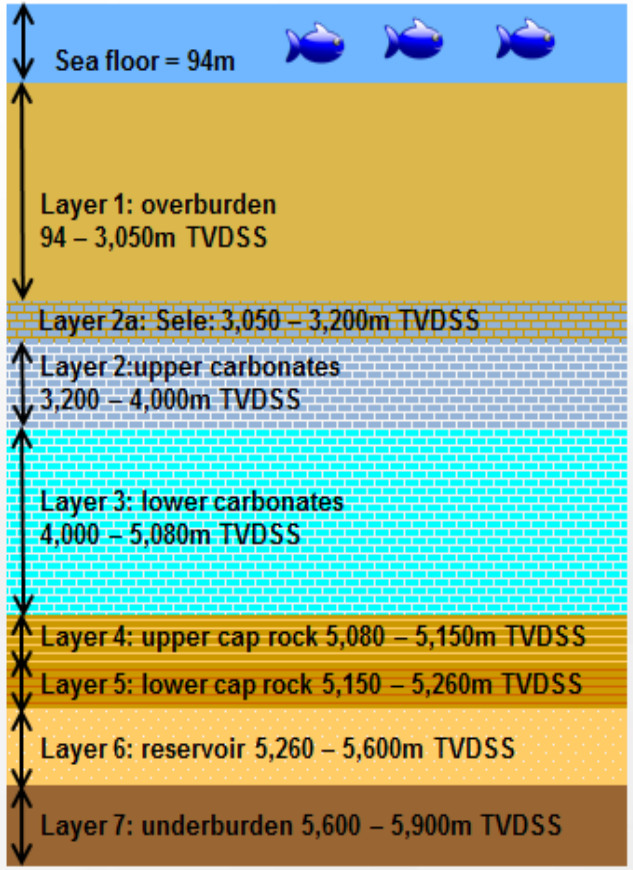
Formation layers

Elastic properties	Young modulus E (MPa)	Poisson ratio ν	Hydraulic conductivity K (m/year)	porosity n	Top layer m
Overburden	2400.0	0.39	3.154E-05	0.20	-92.0
Sele	6400.0	0.32	3.154E-01	0.20	-3050.0
Upper Carbonates	19000.0	0.28	3.154E-03	0.15	-3200.0
Lower Carbonates	18200.0	0.26	3.154E-04	0.15	-4000.0
Upper Cap Rock	5800.0	0.35	1.419E-02	0.10	-5080.0
Lower Cap Rock	8400.0	0.30	1.419E-02	0.10	-5150.0
Reservoir	12600.0	0.23	1.419E+02	0.20	-5260.0
Sideburden	12600.0	0.23	1.419E-02	0.20	-5260.0
Underburden	29000.0	0.27	7.096E-04	0.10	-5600.0

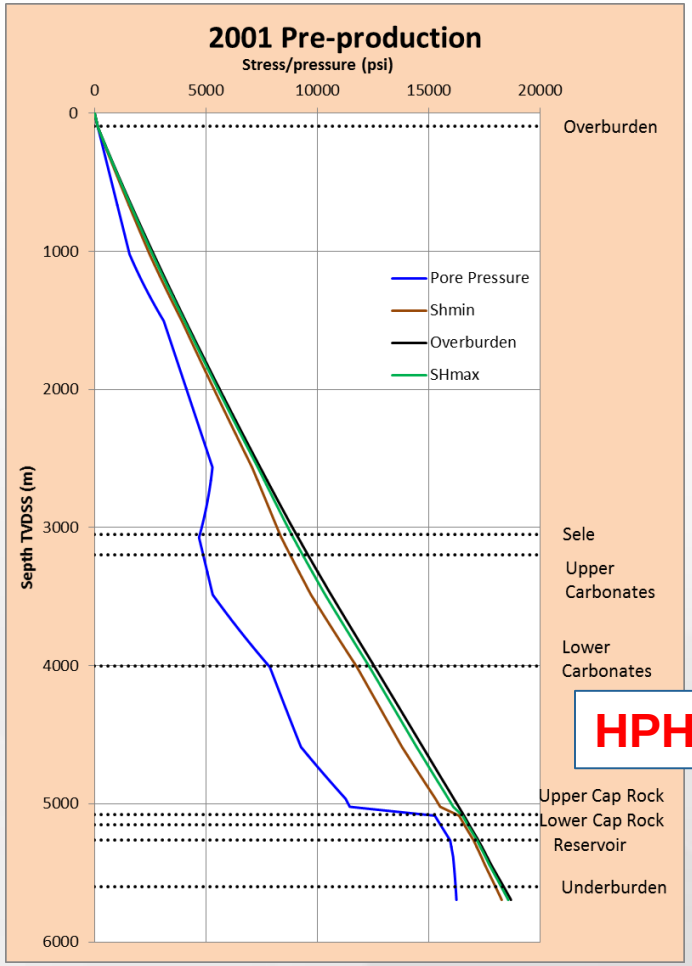
Material properties

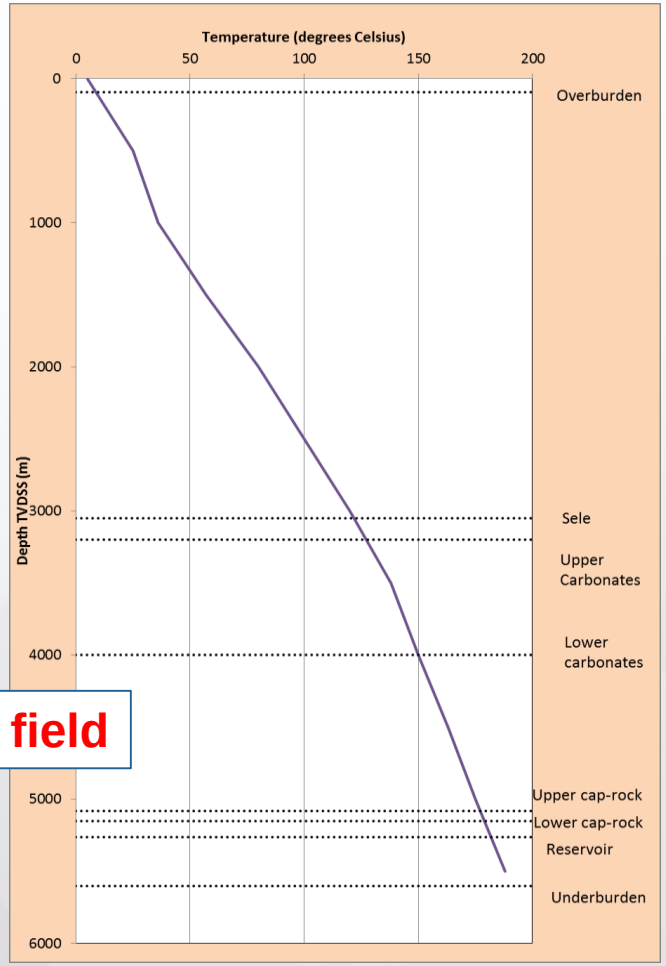
Stresses/pressures

- Stratigraphy, stresses/pressures and temperatures at pre-production



Formation layers





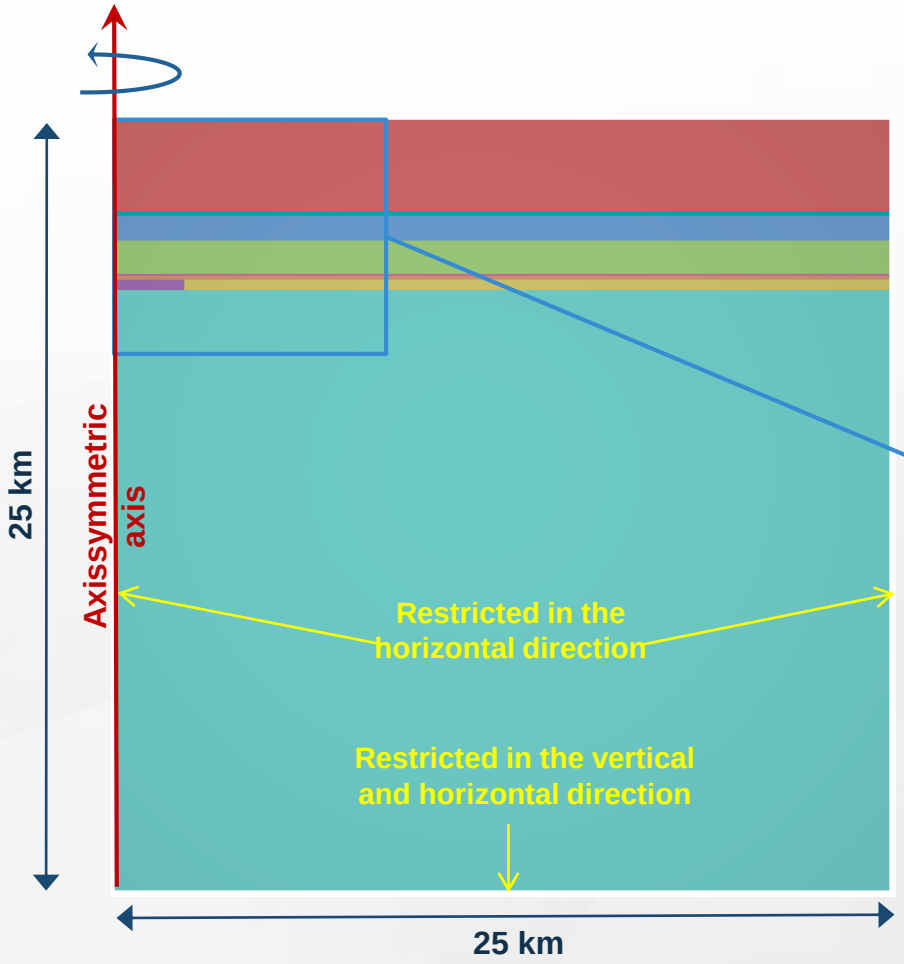
HPHT field

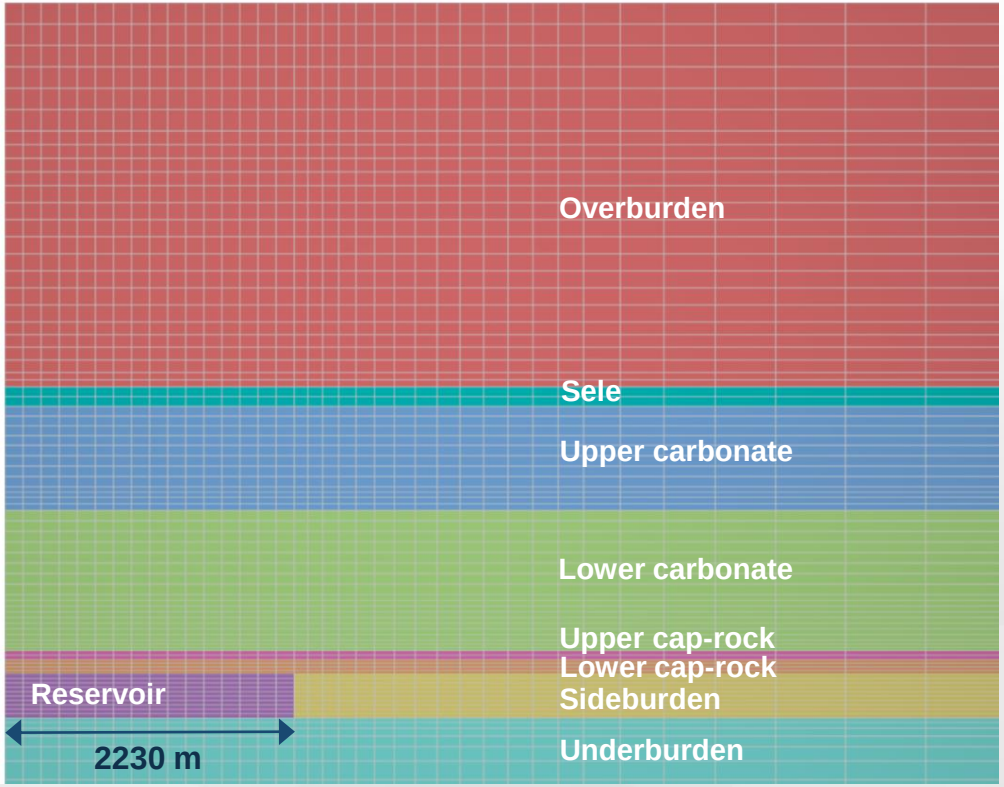
Data “apparently’ across wells G11 and G12

Spatial discretization

- Spatial discretization of model

Model:
Number of quadrilateral elements = 18876
Number of nodes = 57183
Quadratic elements





Model file: Material properties

```
PropertySet
{ id           = 'MatProp',
  typeName     = 'GemaPropertySet',
  description  = 'Material properties',
  properties   = {
     {id = 'E',          description = 'Elasticity modulus',           unit = 'MPa'},
     {id = 'nu',         description = 'Poisson ratio'},
     {id = 'K',          description = 'Hydraulic permeability in x',  unit = 'm/year'},
     {id = 'gw',         description = 'Specific weight of water',     unit = 'MN/m^3'},
     {id = 'Kww',        description = 'Bulk modulus of water',        unit = 'MPa'},
     {id = 'Pht',        description = 'Porosity'},
     {id = 'materialHM', description = 'Hydro-mechanics material type',
                         constMap = constants.CoupledHMFemPhysics.materialModels},
     {id = 'h',          description = 'Element thickness',            unit = 'm'},
  },
  values = {
     {E = 1.26e+04,   nu = 0.23,     h =  1.0,        materialHM = 'poroElastic',
      K =141.912,     gw = 4.50e-03, Pht =2.00e-01,   Kww =2.20e+03},
     ...
  }
}
```

A map with the supported transfer function names

These properties correspond to the first layer, repeat for any other layer

Material type related to the constitutive model.

Model file: Boundary condition

```
-- Constrained displacements. Sides are constrained in the x direction
-- while the base is constrained in both x and y.
BoundaryCondition {
  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},
  },
  nodeValues = {
  -- Node fixed in x and y
      {       1,  0.0000e+00,  0.0000e+00},
      {       2,  0.0000e+00,         nil},
      ...
  }
}
```

Prescribed displacement in x

Node free in y

A default value different from 0.0 is needed so that free nodes in one direction are not misinterpreted as a 0.0 displacement.

Model file: Boundary condition

```
-- Prescribed Pore pressure - Geostatic
BoundaryCondition {
  id   = 'bpmc',
  type = 'node pore pressure',
  mesh = 'mesh',
  properties  = {
    {id = 'P',  description = 'Fixed node pore pressure', unit = 'MPa',  defVal = -9999},
  },
  nodeValues = {
         {        1        ,        111.9    },
         {        2        ,        111.9    },
          ...
  }
}
-- Prescribed pore pressure - Depletion
BoundaryCondition {
  id   = 'bcDep',
  type = 'node pore pressure',
  mesh = 'mesh',
  properties  = {
    {id = 'P',  description = 'Fixed node pore pressure', unit = 'MPa',  defVal = -9999},
  },
   nodeValues = {
         {        34704    ,        65.732   },
         {        34708    ,        48.11    },
          ...
  }
}
```

Solution file: Replacing unit systems

- In our model data, pressure is being expressed in MPa and the time in years. Since the Hydro and the Mechanical physics operate by default in kPa and seconds, a conversion will be automatically done. This works fine for most models, but in this simulation it can create ill-conditioned matrices with a big magnitude difference between numbers. One solution to this problem can be to leave the unit fields on the model empty, so that they are expressed in Mpa but the physics thinks they are in kPa. A better solution is to exchange the unit system that will be used by the physics to do its calculations, allowing us to keep the correct units in the model. This is done by providing physics objects with the unitSystem tag.

```
local MPaYearUnitSystem = {
  time  = 'year',
  coord = 'm',
  u     = 'm',
  ux    = 'm',
  uy    = 'm',
  P     = 'MPa',
  E     = 'MPa',
  S     = 'MPa',
  nu    = '',
  K     = 'm/year',
  gw    = 'MN/m3',
  Kww   = 'MPa',
  Pht   = '',
  h     = 'm',
}
```

Replaces the internal unit used by the physics for state variables, properties, Gauss and boundary condition attributes. Keyed by the internal names used by the physics.

```
PhysicalMethod {
  id                  = 'Depletion_step',
  typeName            = 'CoupledHMFemPhysics.Axissymetric',
  type                = 'fem',
  mesh                = 'mesh',
  elementGroups       = { 'QUP_1', 'QUP_2', ... },
  materials           = {'poroElastic'},
  boundaryConditions  = {'bc','bcDep'},
  ruleSet             = 1,
  properties          = { material = 'materialHM' },
  isoParametric       = false,
  useConsistentMatrix = true,

  unitSystem = MPaYearUnitSystem,
}
```

Replaces the physics unit system

Solution file: Initial condition

```
function initCond ()
   local mesh = modelData:mesh('mesh')
   -- Stress accessor
   local accS = mesh:gaussAttributeAccessor('S',1, true)
   -- Node coordinate accessors
   local coordAc = mesh:nodeCoordAccessor()

   -- Initial stress
   for i=1, mesh:numCells() do      -- For each element
      local e    = mesh:cell(i)     -- element
      -- integration rule
      local ir = mesh:elementIntegrationRule(e:type(),1)
      -- element shape
      local shp = assert(e:shape())
      -- elemental node coordinates
      local Xnode = e:nodeMatrix(coordAc, true)

      -- 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)
            -- get vertical and horizontal stress to
               vertical coordinate
            local St = interpStress(coord(2))

            -- fill stress vector according
            local v = {St[1], St[2], St[3], 0.0}
            -- set stress accessor
            accS:setValue(e, j, v)
      end
   end

   -- Initial pore pressure
   local accPore =
               assert(mesh:nodeValueAccessor('P'))

   for ii=1, mesh:numNodes() do
        local xy = coordAc:value(ii)
        -- get the pore pressure by interpolation
        -- to vertical point reference
        local Pp = interpPpressure(xy(2))
        -- set pore pressure accessor
        accPore:setValue(ii, Pp)
   end
end
```

Solution file: Stress interpolation

- This function interpolates the in-situ stresses, to define them as initial conditions. This step is executed before the geostatic step.

```
function interpStress(z)
    local Sxc
    local Syc
    local Szc
    for i = 1, #initData-1  do
        local _data0 = initData[i]
        local _data2 = initData[i+1]
        if (z <= _data0[1] and z >= _data2[1]) then
            -- interpolation of stress
            -- calculate the horizontal minimum, vertical and horizontal maximum stress.
            Sxc = _data0[2] + (z - _data0[1])*(_data2[2]-_data0[2])/(_data2[1]-_data0[1])   --sh
            Syc = _data0[3] + (z - _data0[1])*(_data2[3]-_data0[3])/(_data2[1]-_data0[1])   --sv
            Szc = _data0[4] + (z - _data0[1])*(_data2[4]-_data0[4])/(_data2[1]-_data0[1])   --sH
        end
    end
    return {Sxc, Syc, Szc}
end
```

Solution file: Pore pressure interpolation

- Similar to the stresses, the pore pressure distribution also must be initialized. This function defines the pore pressure as a function of depth.

```
function interpPpressure(z)
    local Ppc
    for i = 1, #initData-1  do
        local _data0 = initData[i]
        local _data2 = initData[i+1]
        if (z <= _data0[1] and z >= _data2[1]) then
            -- interpolation of pore pressure
            Ppc = _data0[5] + (z - _data0[1])*(_data2[5]-_data0[5])/(_data2[1]-_data0[1])
        end
    end
    return Ppc
end
```

Solution file: Solver options

```
-- Solver options to geostatic step
local solverOptionsGeost = {
  geostaticType      = 'fixedNode',
}


-- Solver options to depletion step
local solverOptions = {
  type               = 'transient nonlinear',
  timeMax            = 1.0,
  timeInitIncrement  = 1.0,
  timeMinIncrement   = 0.01,
  timeMaxIncrement   = 1.0,
  iterationsMax      = 15,
  tolerance          = {mechanic =1.000E-05, hydraulic =1e-5},
  geostaticType      = 'fixedNode',
}
```

- **Variable time step, transient, non-linear solver**
- **Maximum time for the simulation in years**
- **Size of the increment in years**
- **Minimum increment in years**
- **Maximum increment in years**
- **Maximum number of interactions**

Solution file: Orchestration

```
function ProcessScript()
  ------------------------------------------------------------------------------
  -- Initial condition
  ------------------------------------------------------------------------------
  print(' ==================='  )
  print(' Initializing stress and pore pressure.....\n')
  print(' ==================='  )
  -- Call of initial Condition function to initializing of stress and pore pressure
  initCond()

  ------------------------------------------------------------------------------
  -- Geostatic
  ------------------------------------------------------------------------------
  print(' ==================='  )
  print(' Geostatic Step.....\n')
  print(' ==================='  )
  local solver = fem.init({'Geostatic_step'}, 'solver', solverOptionsGeost)
  local file = io.prepareMeshFile('mesh', '$SIMULATIONDIR/out/$SIMULATIONNAME.nf ', 'nf',
                                  {'u','P',},{'S', 'E',},{split = true, saveDisplacements = true})

  -- run Geostatic step
  io.addResultToMeshFile(file, 0)
  fem.geostatic(solver)
  io.addResultToMeshFile(file, 1)
```

Save displacement, pore pressure stresses, and strains

… continues on the next slide

Solution file: Orchestration

```
-------------------------------------------------------------------------------
-- Depletion
-------------------------------------------------------------------------------
print(' ==================='  )
print(' Depletion Step.....\n')
print(' ==================='  )
-------------------------------------------------------------------------------
local solver = fem.init({'Depletion_step',}, 'solver', solverOptions)
local dt        = solverOptions.timeInitIncrement
local FinalTime = solverOptions.timeMax
local Time      = dt
local LastStep  = false

setCurrentTimeUnit('year')
```

Tell GeMA that the time (dt) is being given in years

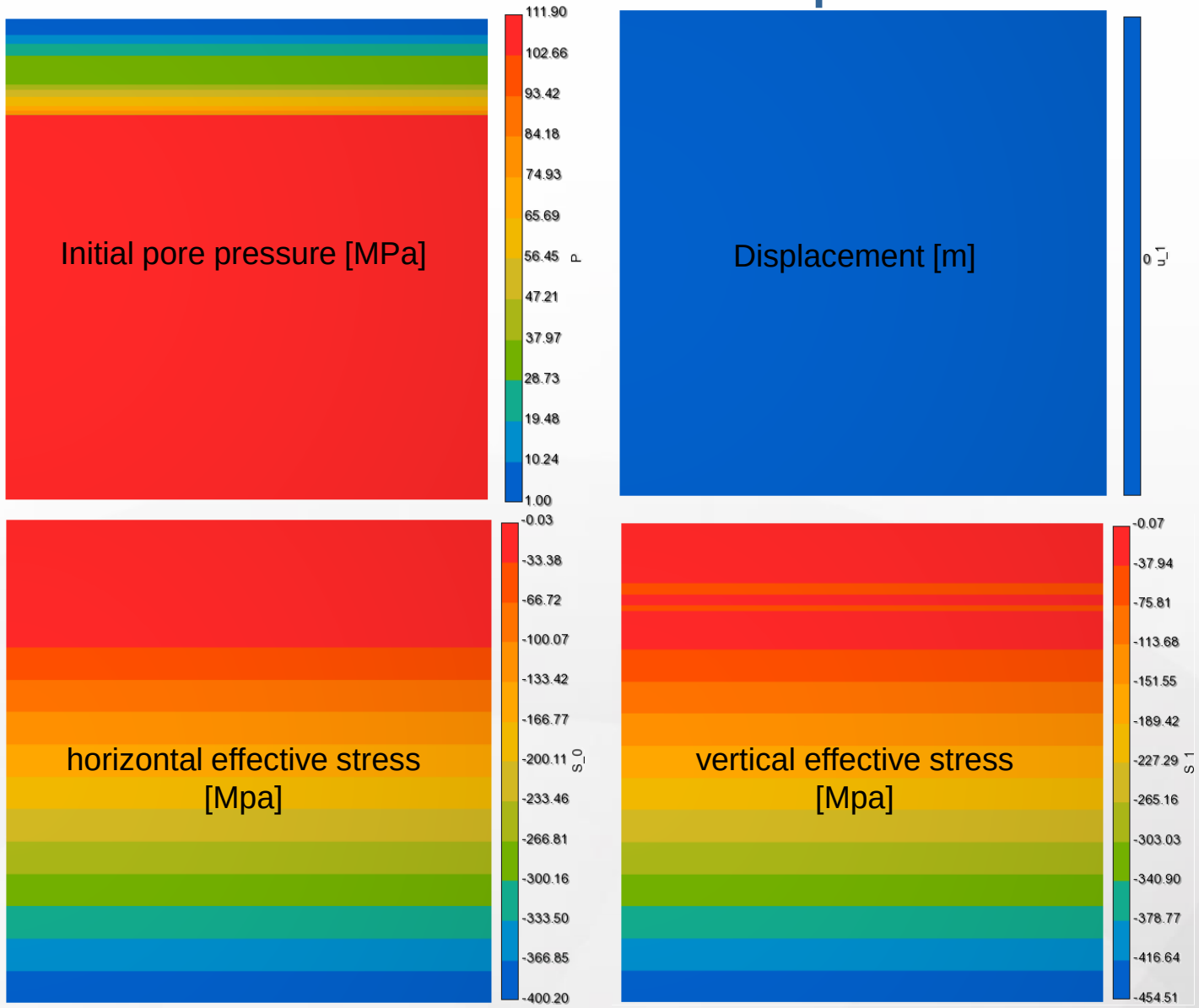
```
while (Time <= FinalTime) do
     print('------------------')
     print(('Depletion step - time = %1 yr'):num(Time))
     print('------------------')

     -- Run fully coupled analysis.  Solver returns a suggested time-step
     -- for the next iteration
     dt = fem.step(solver, dt)
```

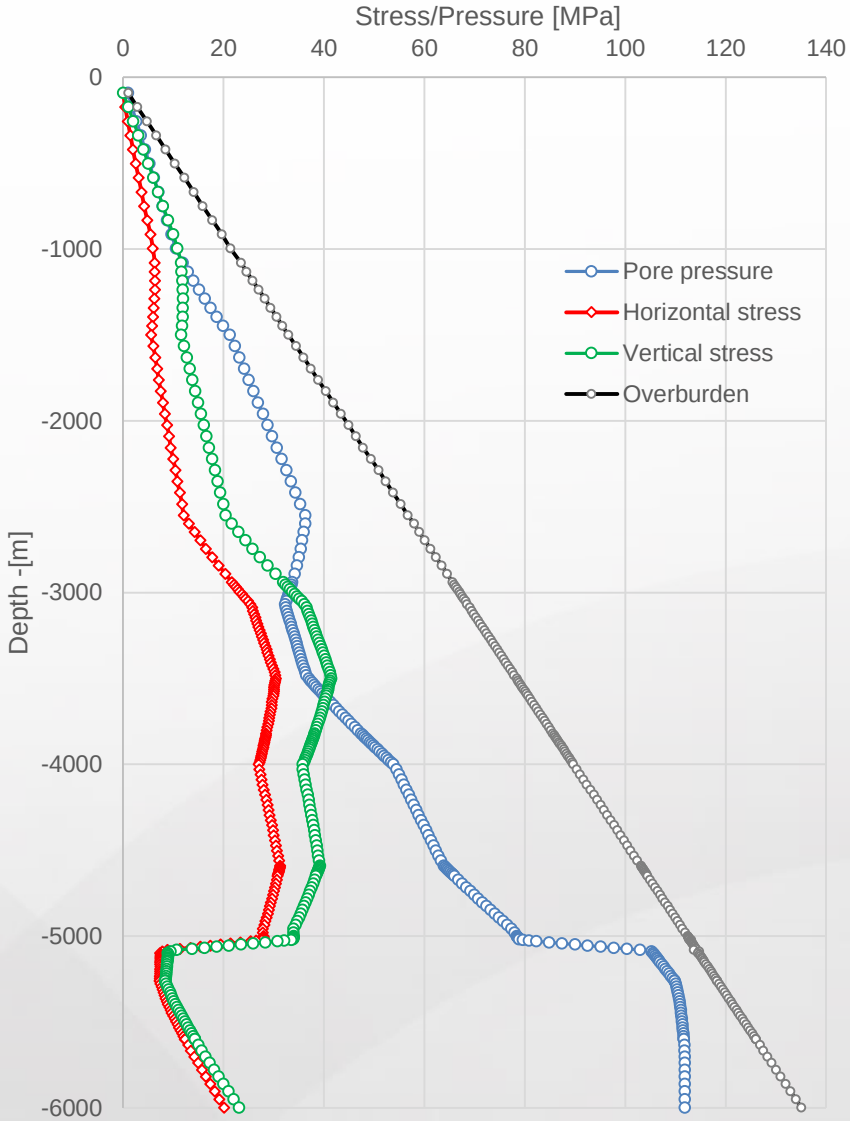
… continues on the next slide

Solution file: Orchestration

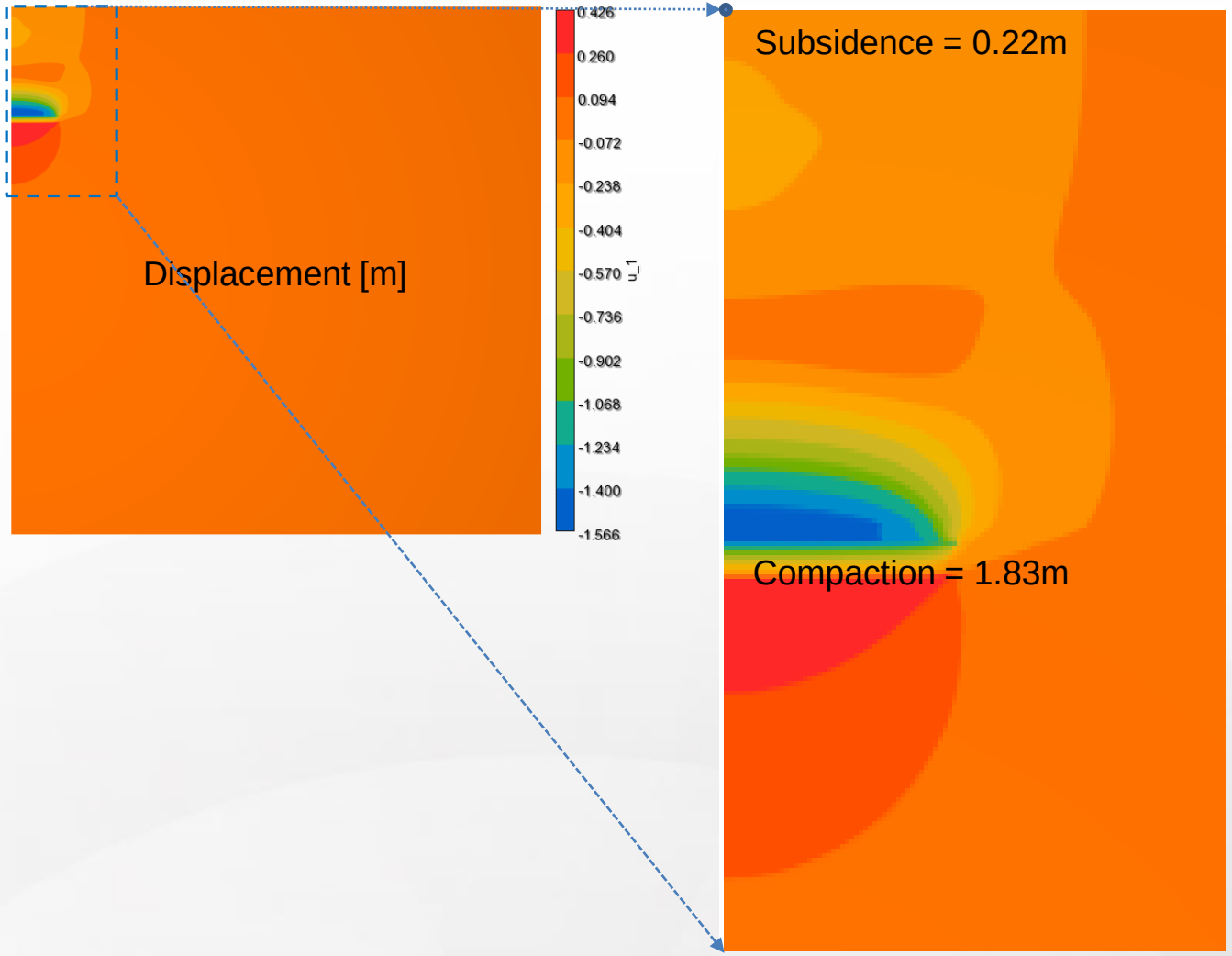
```
        io.addResultToMeshFile(file, Time)

        -- Adjust time to guarantee that the last iteration will be on the
        -- requested final simulation time
        if (Time + dt > FinalTime and not LastStep) then
            dt   = FinalTime - Time
            Time = FinalTime
            if equal(dt, 0.0) then break end
            LastStep = true
        else
            Time = Time +  dt
        end
        print('Time = ' .. Time)
    end

    io.closeMeshFile(file)

end
```

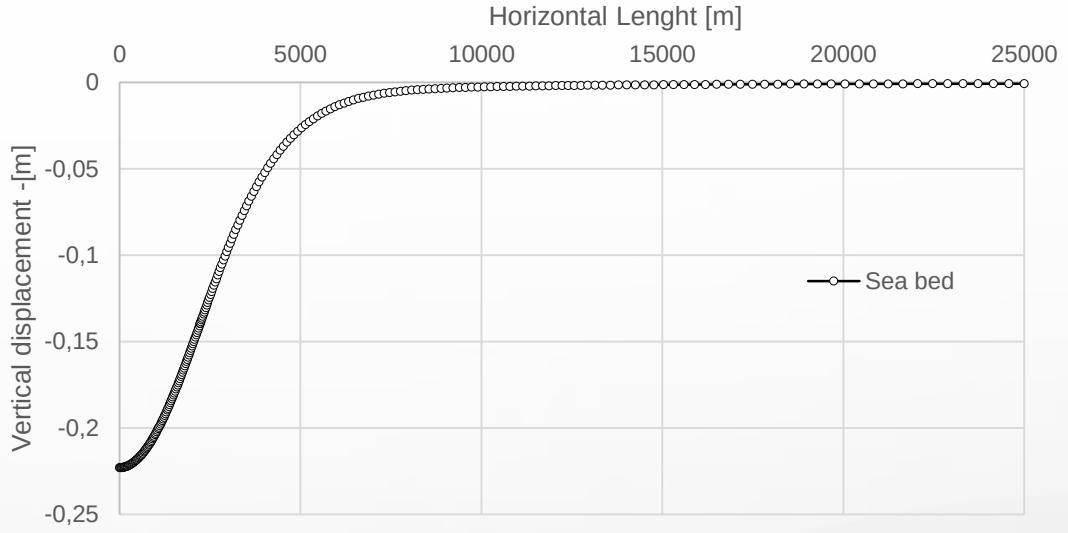
Results: Geostatic - step

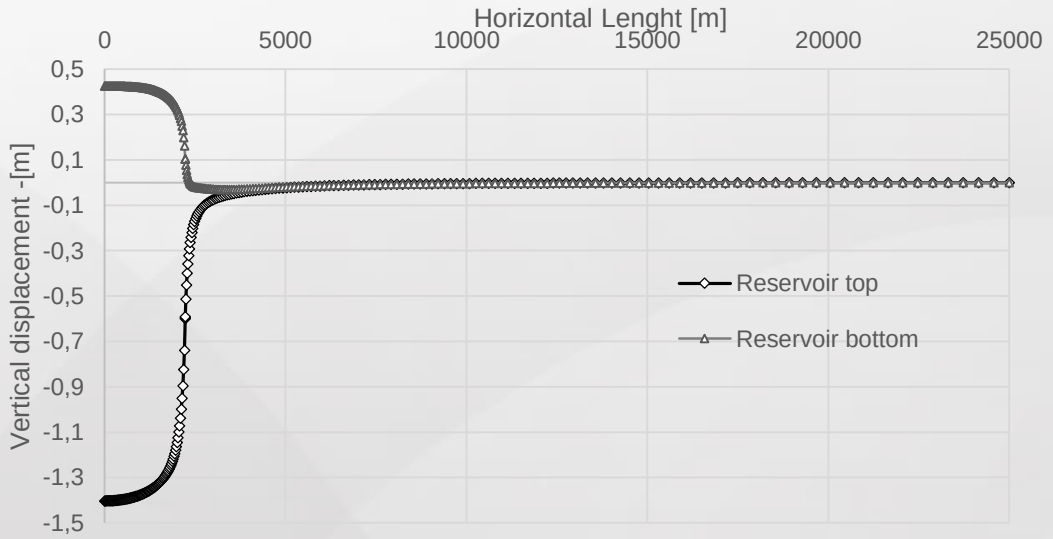




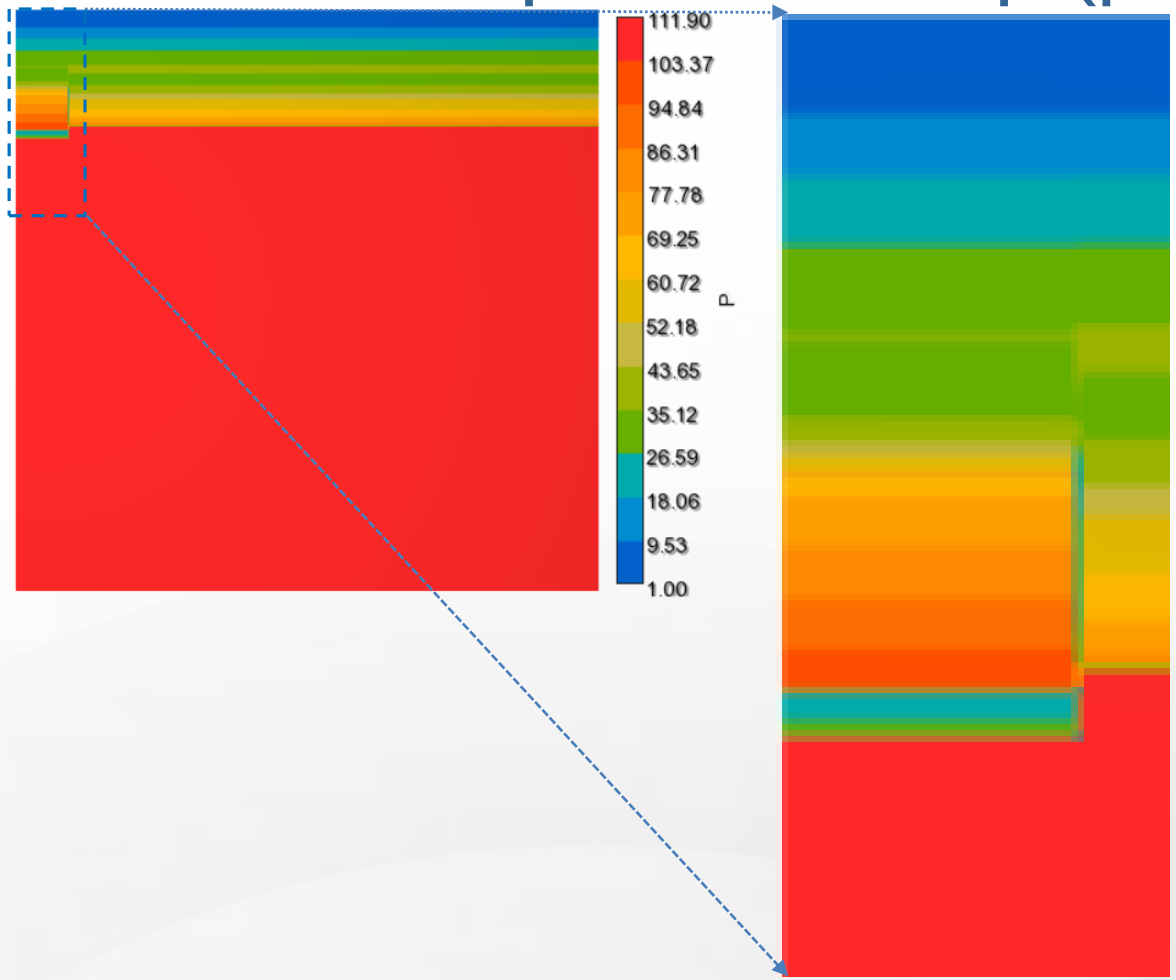
Results: Depletion – step (displacement)

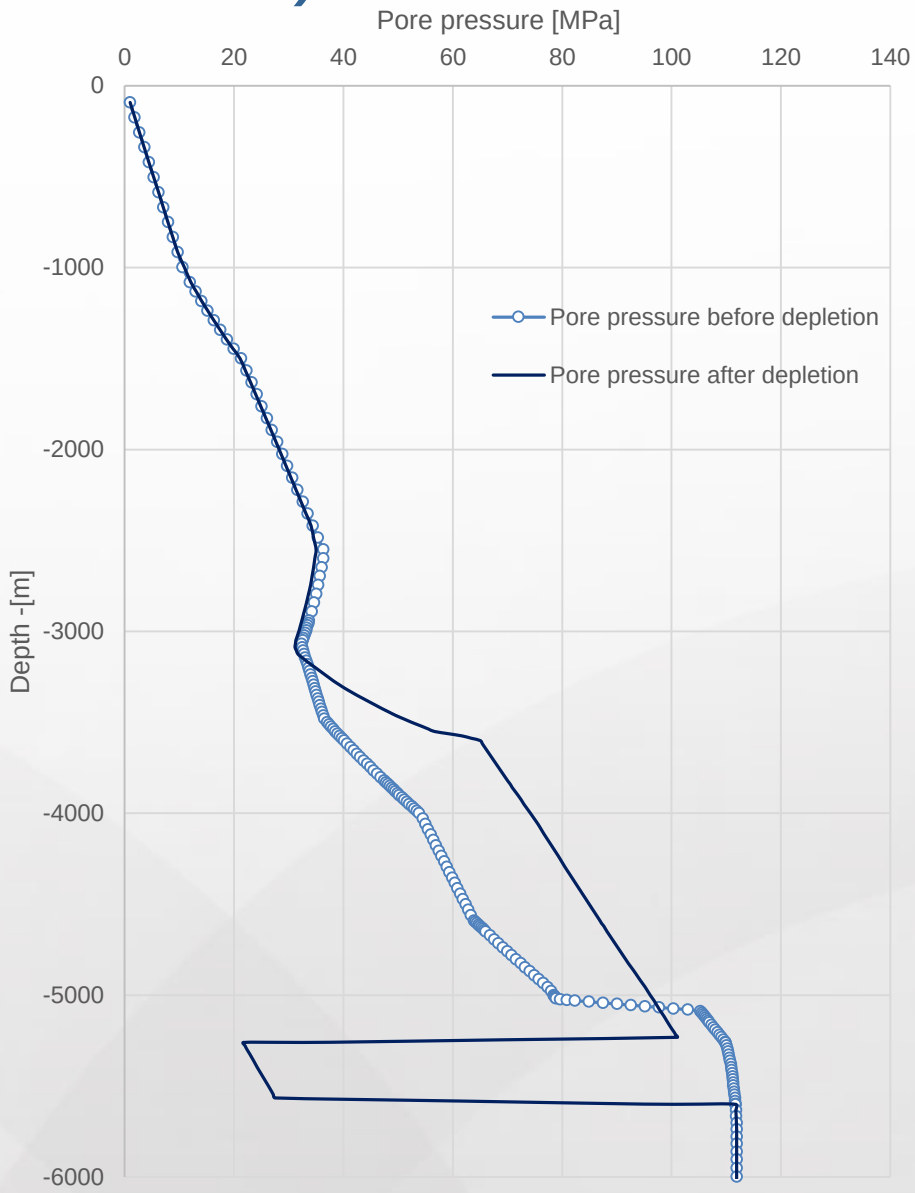






Results: Depletion – step (pore pressure)

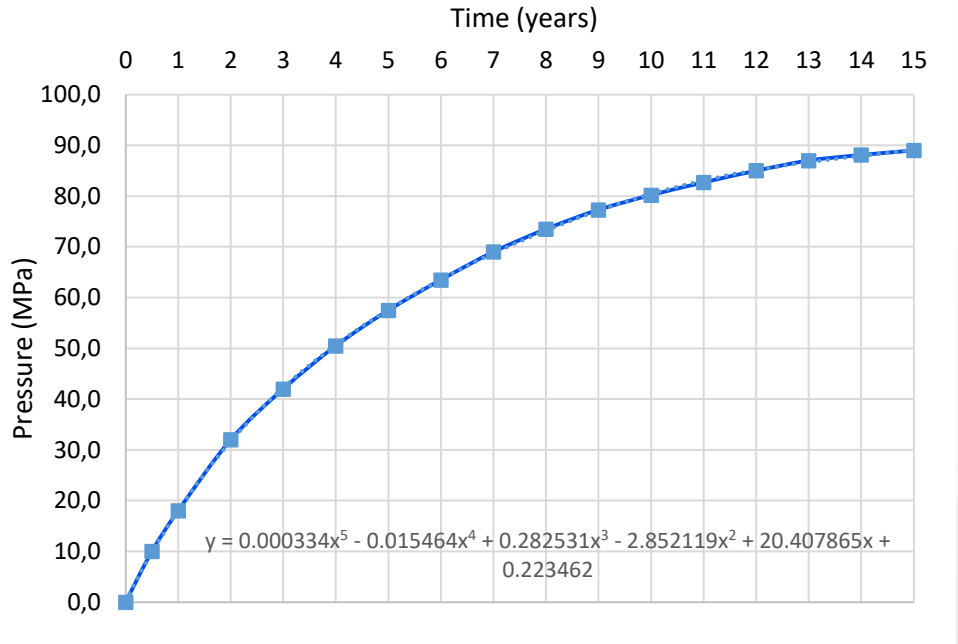




2 – 2D FULLY COUPLED SUBSIDENCE ANALYSIS

Model file: Boundary condition

```
--  Depletion Loading functions
NodeFunction { id = 'depletFunct',
   parameters = { {src = 'time', unit = 'year'},
                },
   method = function(t)
       return = 110.7 - (0.000334*t^5 - 0.01546*t^4
              + 0.28253*t^3-2.8521*t^2+20.408*t)
   end
}

--Depletion pore pressure
BoundaryCondition {
  id    = 'bcDep',
  type = 'node pore pressure',
  mesh = 'mesh',
  properties  = {
    {id = 'P',  description = 'Prescribed node pore pressure', unit = 'kPa',
     defVal = -9999, functions = true},
  },
   nodeValues = {
        {33156      , 'depletFunct'},
        {33222      , 'depletFunct'},
        ...
   }
}
```

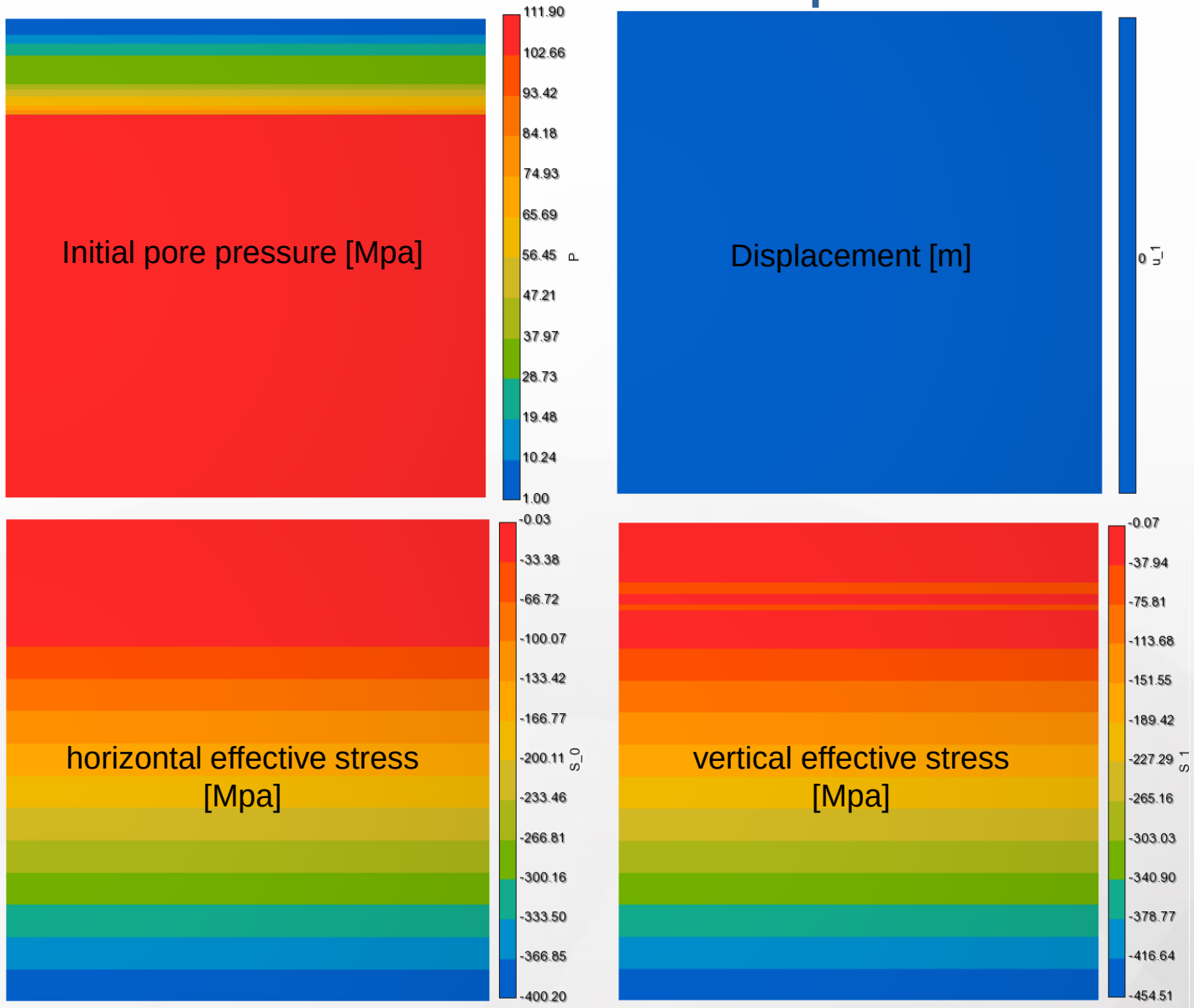


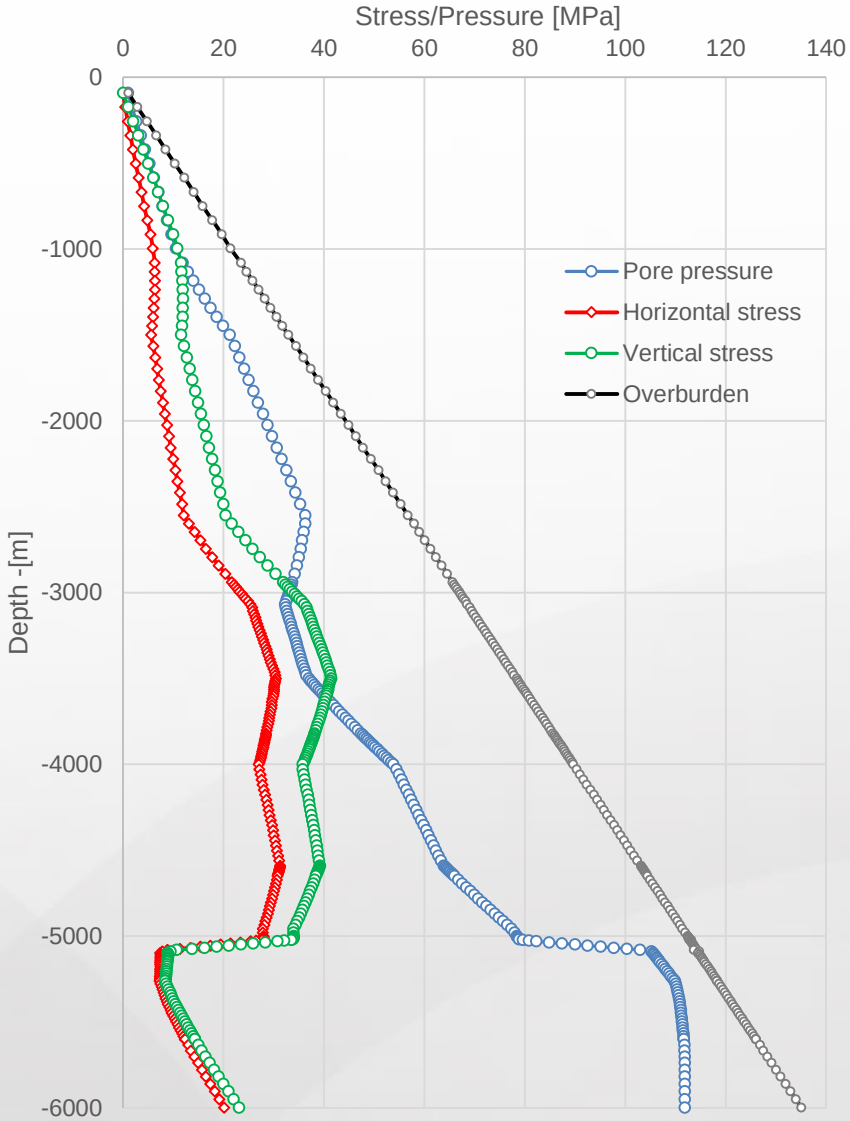
Solution file: Solver options

```
-- Solver options to geostatic step
local solverOptionsGeost = {
  geostaticType      = 'fixedNode',
}


-- Solver options to depletion step
local solverOptions = {
  type               = 'transient nonlinear',
  timeMax            = 15.0,
  timeInitIncrement  = 0.01,
  timeMinIncrement   = 0.001,
  timeMaxIncrement   = 1.0,
  iterationsMax      = 15,
  tolerance          = {mechanic =1.000E-05,hydraulic =1e-5,},
  geostaticType      = 'fixedNode',

}
```

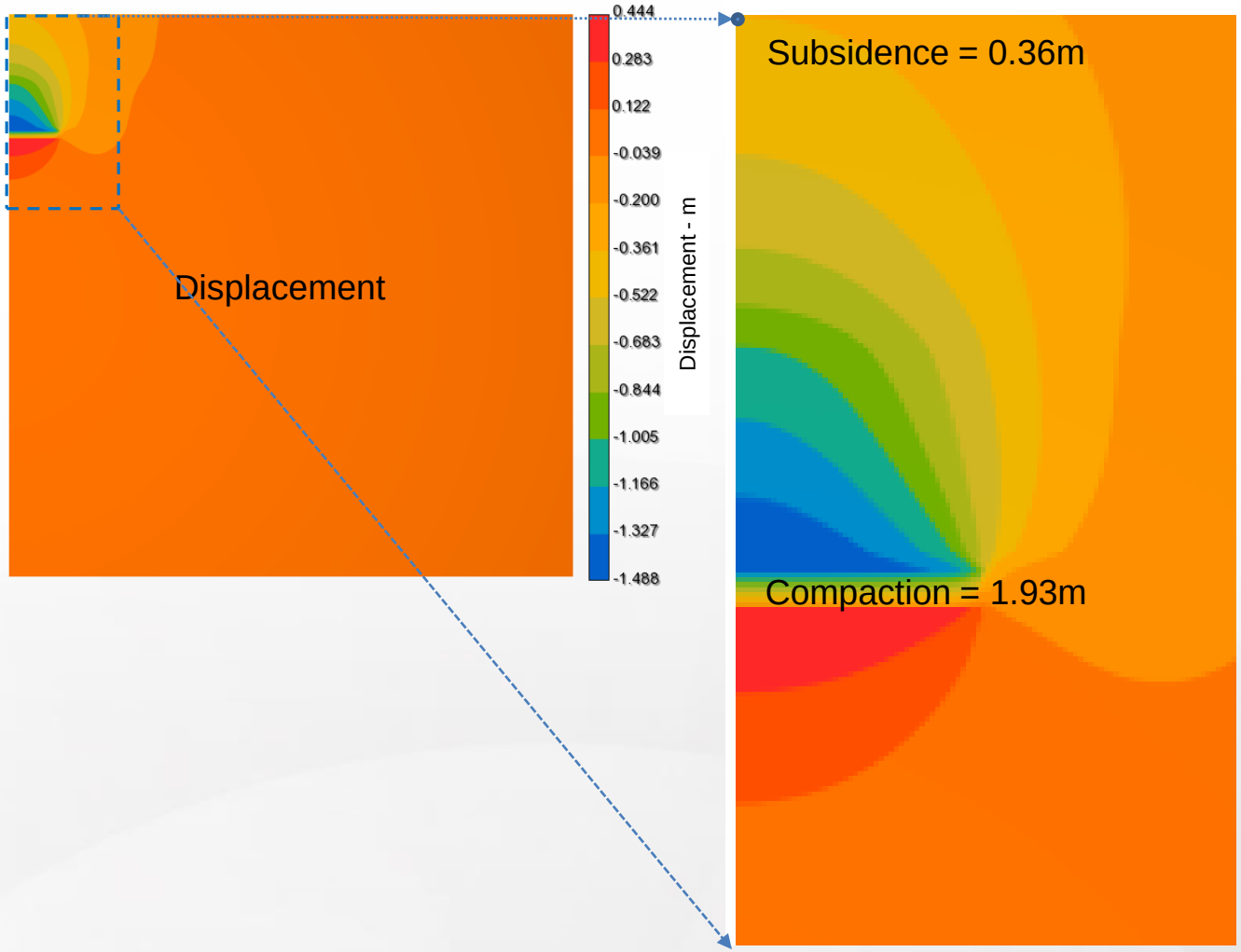
- **Variable time step, transient, non-linear solver**
- **Maximum time for the simulation in years**
- **Size of the increment in years**
- **Minimum increment in years**
- **Maximum increment in years**
- **Maximum number of interactions**

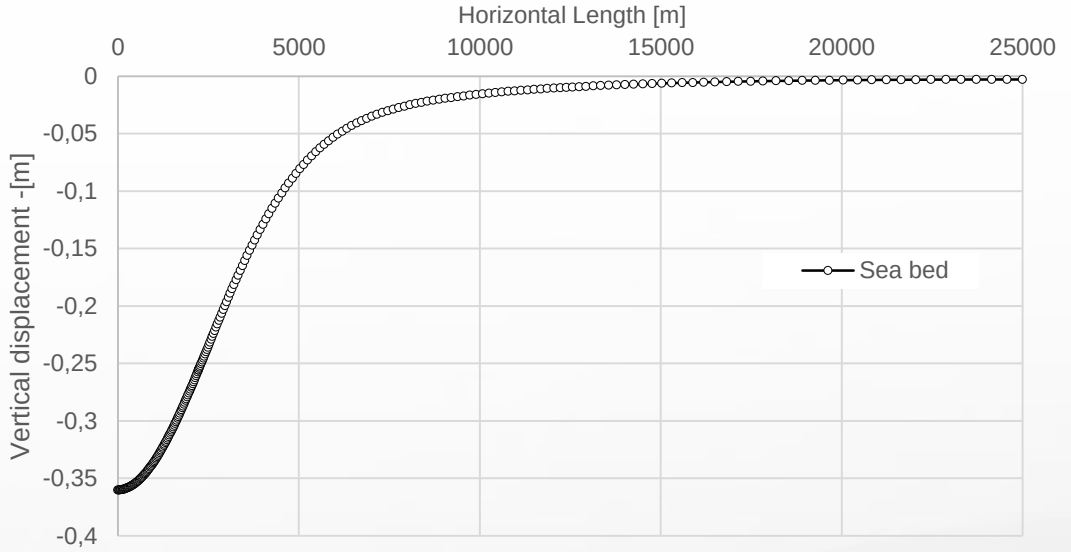
Results: Geostatic - step

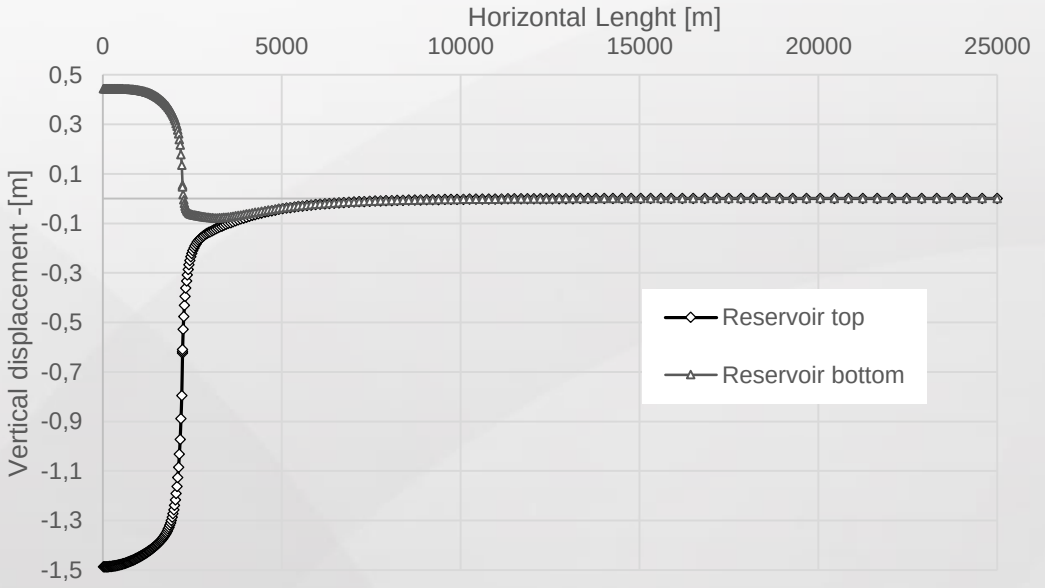




Results: Depletion – step (displacement)







Results: Depletion – step (pore pressure)