GeMA – XFEM Hydraulic Fracturing Examples

16/07/2018 – Version 1.0

Tecgraf
PUC-RIO

Example set purpose

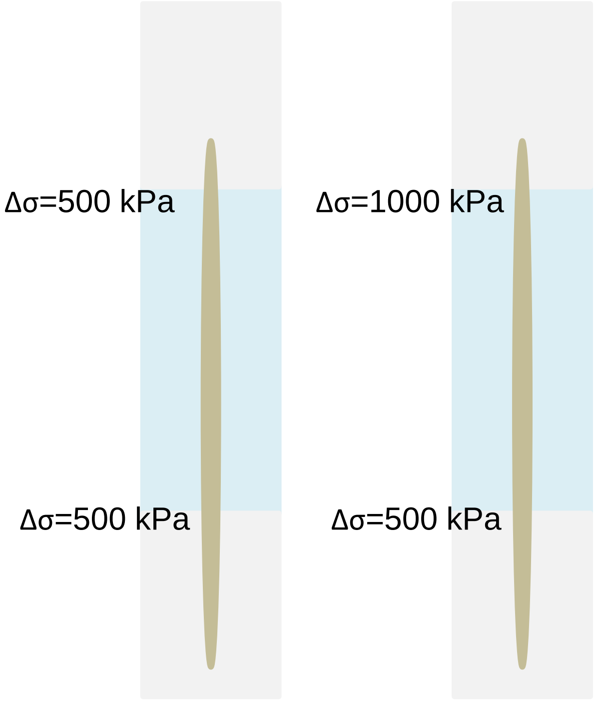
This examples show:

- How to setup natural (initial) fractures on the model
- How to setup Injection Flow Rate in the model as a boundary condition
- How to setup boundary conditions on enriched degrees of freedom (displacement and porepressure)
- How to setup XFEM specific options
- Several different orchestration techniques for simultaneous or sequential hydraulic fracturing schemes with a geostatic step for external and internal forces equilibrium before the subsequent injection step

The examples

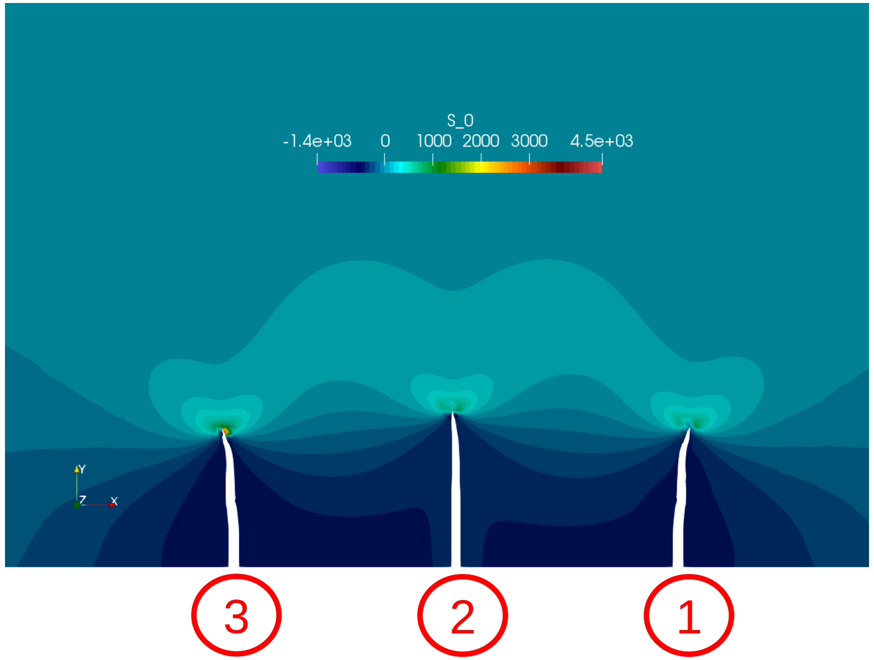
- Example 1 presents a transient analysis of a single hydraulic fracture on a square plate to be compared with KGD analytical solution

- Example 2 and 3 present a transient analysis of a single hydraulic fracture entering another layer with symmetric and asymmetric stress contrast to be compared with Simonson and Fung analytical solutions, respectively.

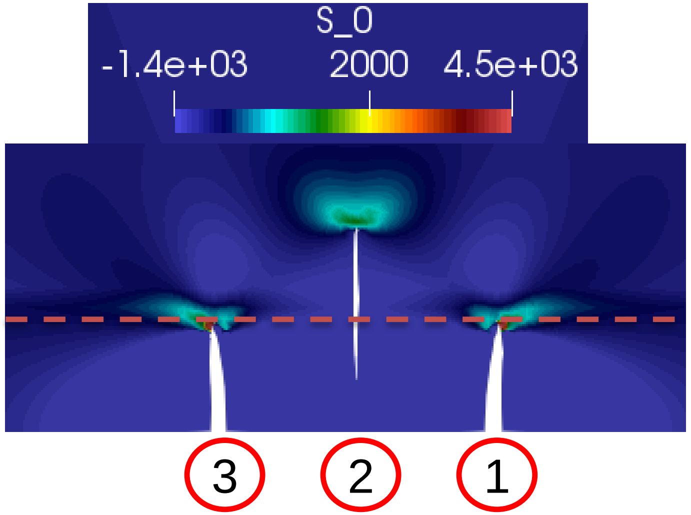


The examples

- Example 4 and 5 present a transient analysis of three simultaneous and sequential hydraulic fractures, respectively with a homogeneous stress state.



- Example 6 presents a transient analysis of three simultaneous hydraulic fractures entering another layers with symmetric stress contrast varying fracture spacings of 7 and 20 m.



The examples

On this presentation, the first example will present and analyze the complete model source. Other examples will present only relevant parts. The complete source for all models are available at the example files.

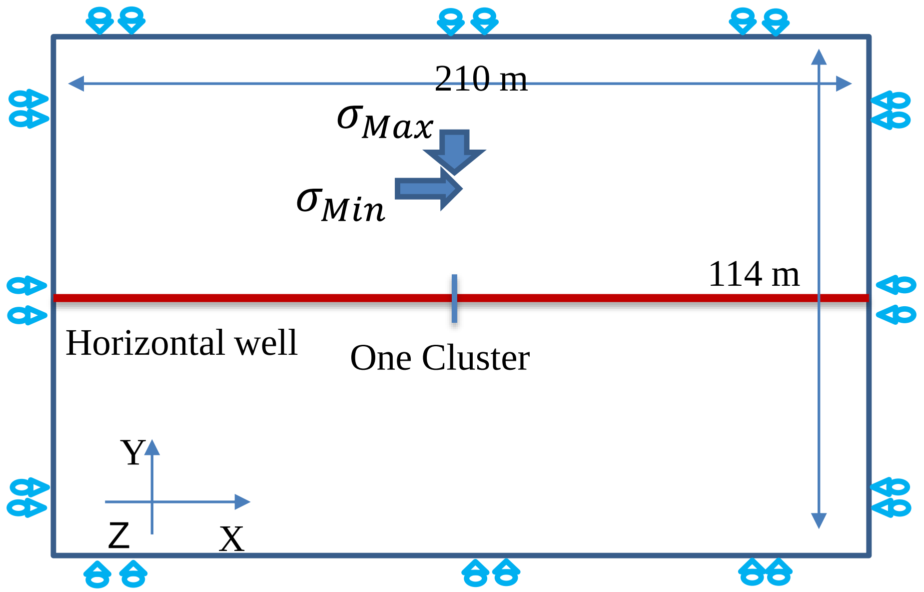
Although this examples try to explain all involved concepts and syntaxes, they are not a substitute for reading the GeMA tutorial and additional documentation.

As a convention, all the given examples will generate result files on the “out” directory.

1 – KGD ANALYTICAL MODEL IN K-VERTEX PROPAGATION REGIME

The Problem

Injection on a square plate: It is considered a constant injection rate of a Newtonian, incompressible fluid in plane strain conditions in an infinite permeable or impermeable, homogeneous elastic medium.



Reference: "*Propagation Regimes of Fluid-Driven Fractures in Impermeable Rocks*", Detournay, E. (2004)

Key Example Points

- This example presents the basic structure for a hydraulic fracturing simulation in GeMA

- In particular it shows how to:
  - Structure a GeMA model
  - Define natural (initial) fractures in the model
  - Setup material values
  - Setup a boundary condition for the injection process
  - Set up the XFEM specific options
  - Get pressures at the injection point for post-processing analysis
  - Fill empty mesh for results visualization

Simulation file: KGD.lua

Simulation file: KGD.lua

This file is the main simulation file. It stores a model description and loads the two auxiliary files storing the **simulation model** and the **simulation solution**. Splitting the simulation on those files is a convention to separate the description of **what** will be simulated (the model file) from the description of **how** it will be simulated (the solution file).

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

Loads the file “KGD_solution.lua” from the same directory as the simulation file

Macro expanding to the directory of this simulation

Macro expanding to the name of this simulation file, without extension

Model file: State variable

Next, the simulation state variable is defined. State variables are the nodal values calculated by the simulation and represent the model degrees of freedom. For a hydraulic fracturing simulation, state variables are:

```
-- State variables
StateVar{id = 'u', dim = 2, description = 'Displacements in the X and Y directions', unit = 'm',
         format = '8.4f', groupName = 'mechanic'}

StateVar{id = 'a', dim = 2, description = 'Enriched displacement in the X and Y directions', unit = 'm',
         format = '8.4f', groupName = 'mechanic'}

StateVar{id = 'P', description = 'Pore pressure degree-of-freedom', unit = 'kPa',
         format = '8.4f', groupName = 'hydraulic'}

StateVar{id = 'Pa', description = 'Enriched pore pressure degree-of-freedom', unit = 'kPa',
         format = '8.4f', groupName = 'hydraulic'}

StateVar{id = 'Pf', description = 'Fracture pressure degree-of-freedom', storage = 'both', unit = 'kPa',
         format = '8.4f', groupName = 'hydraulic'}
```

State variables can be associated with different group names. Those groups are used for prescribing different numeric tolerances for different kinds of degrees of freedom at the solution file

This state variable will have values for both geometry nodes and ghost nodes (internal nodes created by the XFEM simulation)

Model file: Material properties

For a hydraulic fracturing simulation, material properties are:

```
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',  unit = 'm/s'},
    {id = 'gw',    description = 'Specific weight of water', unit = 'kN/m3'},
    {id = 'Pht',   description = 'Porosity'},
    {id = 'SPMax', description = 'Maximum principal stress',  unit = 'kPa'},
    {id = 'Gap',   description = 'Initial gap opening',  unit = 'm'},
    {id = 'Ufw',   description = 'Dynamic fluid viscosity',  unit = 'kPa*s'},
    {id = 'Lkt',   description = 'Leakoff at top',  unit = 'm/(kPa*s)'},
    {id = 'Lkb',   description = 'Leakoff at bottom',  unit = 'm/(kPa*s)'},
    {id = 'material',  description = 'Mechanical XFEM material type', constMap = constants.Xfem.materialModels},
  },
  values = {
    {E = 17e+6, nu = 0.2,  K = 9.8e-9, gw = 9.81, Pht = 0.2, SPMax = 1.25e3,
     Gap = 0.002, Ufw = 1e-7, Lkt = 2e-10, Lkb = 2e-10,  material = 'poroElastic'},
  }
}
```

A map published by the Xfem plugin with its supported material model names

A material type from `constants.Xfem.materialModels`

Model file: Mesh

For a hydraulic fracturing simulation, Natural (Initial) fractures on the model are defined following the convention below:

For one crack:

```
--Natural (Initial) fractures on the model
local meshInitialFractures = {
   --{{x1,y1}, {x2,y2}}      initial and final crack points coordinates of Natural (Initial) fractures
  { {0.0, -0.5},  {0.0, 0.5}},
}
```

For multiple cracks:

```
--Natural (Initial) fractures on the model
local meshInitialFractures = {
   --{{x1,y1}, {x2,y2}}      initial and final crack points coordinates of Natural (Initial) fractures
  { {-10.0, -0.5},  {-10.0, 0.5}},
  { {0.0, -0.5},  {0.0, 0.5}},
  { {10.0, -0.5},  {10.0, 0.5}},
}
```

Three cracks

Initial and final crack points coordinates of natural (Initial) fractures must be coincide with an element border

Initial crack point coordinate

Final crack point coordinate

Model file: Mesh

```
Mesh
{
  -- General mesh attributes
  id          = 'mesh',
  typeName    = 'Xfem.mesh',
  description = 'Plate mesh discretization',

  -- Mesh dimensions
  coordinateDim  = 2,
  coordinateUnit = 'cm',

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

  -- Mesh node coordinates
  nodeData = nodes,

  -- Natural fractures
  naturalFractures = meshInitialFractures,
```

- The mesh name
- Xfem problems MUST use a specific mesh type provided by the Xfem plugin
- The mesh dimension (2D)
- The unit in which node coordinates are given
- Associates this mesh with state variables
- Sets the table with node coordinates
- Associates this mesh with Initial Fractures

… continues on the next slide

Model file: Mesh (continued)

… continued from previous slide

```
  -- Element data
  cellProperties = {'MatProp', 'SecProp'},
  cellData       = elements,

  --IntegrationRules
  elementRules = {
    {quad4 = 2, tri3 = 3}, --Rule set 1
    {quad4 = 3, tri3 = 3}, --Rule set 2
  },

  -- Node attributes
  cellAttributes = {
    {id = 'pl', description = 'pressure loading applied at a border', dim = 2, unit = 'kPa'},
  },

  -- Boundary data
  boundaryEdgeData = mesh_edges,
}
```

- Associates this mesh with property set MatProp and SecProp
- Sets the table with element definitions
- Sets the available rule sets for xfem elements
- Sets the table with mesh border definitions

Model file: Boundary conditions

To complete the model file, boundary conditions for prescribing injection flow rates on plate borders are needed. Injection Flow Rate definition are in m3/s, negative signal means that flow is entering into the model. Here this condition is defined in a mesh node, then in the solution file is changed to the crack mouth node or injection point. This is done because initially no crack nodes are defined in the mesh, they only appear after the preprocessing, then crack nodes are added to the mesh.

```
BoundaryCondition {
  id   = 'bc5',
  type = 'node fracture flow',
  mesh = 'mesh',

  properties  = {
    {id = 'qfw',  description = 'concentrated fracture flow', defVal = -9999},
  },

  nodeValues = {
  --  {node, flow rate value}
      {1  ,  -0.001},      -- this node will be changed to ghost node in solution file : ProcessScript()
  }
}
```

Boundary condition name

Boundary condition type for prescribed Injection Flow Rate values

Associates this boundary condition with the mesh named ‘mesh’

B.C. column name

Boundary conditions. Each table line associates a node set to a flow rate value

Solution file: Numerical solver and Physics

The first section of the solution file defines which numerical solver will be used to solve the equation system created by the XFEM method. On this example, we will use a direct matrix solver provided by the ArmadilloSolver plugin.

```
NumericalSolver {
  id          = 'solver',
  typeName    = 'ArmadilloSolver',
  description = 'Simple matrix solver',
}
```

The solver name

The plugin name used to create the numerical solver

Physics are the objects that provide the set of mathematical equations used to solve the simulation. For solving an injection problem, this example will use the Xfem plugin.

```
PhysicalMethod {
  id       = 'HMCoupledXfem',
  typeName = 'Xfem.HydroMechanic',
  type     = 'fem',

  mesh     = 'mesh',
  ruleSet  = 1, -- The integration rule set that will be used on the simulation
  boundaryConditions = {'bc1','bc3','bc4','bc5','bc6'},
  materials          = {'poroElastic'},
}
```

The physics name

The plugin name used to create the physics

Associates this physics with the mesh named ‘mesh’

Associates this physics with a set of boundary conditions

The set of material types used by the simulation

Solution file: solverOptions and xfemOptions

solverOptions and xfemOptions are the objects that provide specific options used to solve the simulation.

```
local solverOptions = {
  type                  = 'transient nonlinear',
  timeMax               = 10,                    -- Total time of analysis
  timeInitIncrement     = 0.05,                  -- Initial time increment
  timeMinIncrement      = 1,                     -- Minimum time increment
  timeMaxIncrement      = 5.0,                   -- Maximum time increment
  iterationsMax         = 15,                    -- Maximum number of iterations
  tolerance             = {mechanic = 1e-4, hydraulic = 1e-4}  -- tolerances for convergence criteria
}
```

Group names from state variable definitions

```
local xfemOptions = {
  propagationCriteria = 'MaxPS',         -- Maximum principal stress criterium for crack propagation
  evaluationZone      = 'nonLocalQuad', -- Create a quadrilateral region at the crack tip to compute the average
                                         -- maximum principal stress of the gauss points which are inside this region
  weightFunction      = 'uniform',       -- Every gauss point inside the quadrilateral region used for MaxPS has the
                                         -- same weight value even though a point is nearer than other to the crack tip
  geometricTol        = 1e-6,            -- Geometric tolerance to determine which gauss points are considered for
                                         -- Maximum principal stress calculation
  geoStatic           = false,           -- False when no geoStatic step is performed
}
```

Solution file: Orchestration script

Finally, the orchestration script, provided by the ProcessScript() Lua function, drives the simulation by calling the XFEM process to execute the simulation.

```
function ProcessScript()
  -- Just a definition of a parameter which represents the mesh object in the orchestration
 local m  = modelData:mesh('mesh')

  -- Just a definition of a parameter which represents the empty mesh object in the orchestration
 local em = modelData:mesh('emptyMesh')

  -- Just a definition of a parameter to access the node value of the fracture pressure in the mesh "m"
 local pfAcc = m:nodeValueAccessor('Pf')

  -- Create the solver model and execute the initial step in which preprocessing (crack nodes
  -- are added to the mesh) is performed. In the xfem.init, the hydromechanical physic, the solver id,
  -- solver options and xfemOptions are passed for the initial step
  local solver = xfem.init({'HMCoupledXfem'}, 'solver', solverOptions, xfemOptions)

  -- Changing node 1 in bc5 by the second ghost node to inject fluid flow rate
  local bc = modelData:boundaryCondition('bc5')
  bc:setNode(1, setMeshGhostFlag(2)) -- Second ghost node of the first crack
```

… continues on the next slide

Solution file: Orchestration script

… continued from previous slide

```
--The initial state of the calculation is saved in a neutral file for postprocessing purposes
io.saveMeshFile(m, '$SIMULATIONDIR/out/$SIMULATIONNAME.nf', 'nf', {'u','a','P','Pa'},{'S', 'E'},
                {split = true, saveDisplacements = true})
```

The mesh can be given either as an object (m) or through its id ('mesh')

Save options

Saved nodal values are the state variable

Save stresses and strains

```
--This file is created to save the results of each step in one archive for postprocessing
--purposes such as analysis of variables along time
local file = io.prepareMeshFile(m, '$SIMULATIONDIR/out/KGDtotal', 'nf',
                                {'u','a','P','Pa','Pf'}, {'S'}, {saveDisplacements = true})

-- The initial state of the analysis is saved
io.addResultToMeshFile(file, 0.0)

--Definition of the file location which would contain the crack pressure along time
local loadPf = io.open(translatePath('$SIMULATIONDIR/out/KGDloadPf.txt'), "w+")

--Write the initial state of the three cracks pressure
loadPf:write("Inc\t Pf\n")
loadPf:write(0, "       ", 0.0, "\n")
```

… continues on the next slide

Solution file: Orchestration script

… continued from previous slide

```
--Definition of the parameters used in the orchestration
local dt     = solverOptions.timeInitIncrement
local endt   = solverOptions.timeMax
local nsteps = endt / dt
local dstep  = 1        --FREQUENCE OF SAVING RESULTS ON EMPTY MESH
local cont   = 1        --a counting parameter
local cont2  = dt*dstep --a counting parameter

for i=1, nsteps do
  --In the xfem.step, the solver parameters and time step are passed
  --Run transient analysis
  xfem.step(solver, dt)

  --After the analysis step finished, results are saved in the output file
  io.addResultToMeshFile(file, i*dt)

  --Get fracture pressure
  local Pfdata = pfAcc:value(setMeshGhostFlag(2))
  --write time, and fracture pressure
  loadPf:write(i*dt, "  ", Pfdata, "\n")
```

… continues on the next slide

Solution file: Orchestration script

… continued from previous slide

```
    if (i*dt == cont2) then  --If a different frequency of saving results is used then save results
      --Save results in an empty mesh passing original mesh, empty mesh, solver parameters, displacements,
      --pressures and fracture pressures. On the new mesh, Xfem sub-elements are transformed into regular
      --mesh elements. Split fractures=true activates a postprocessing tool to visualize
      --fracture aperture in the neutral file
      xfem.copyToElementMesh(m, em, solver, {'u', 'a','P','Pa','Pf'}, {'S', 'E'},
                             {createMissing = true, splitFractures = true})

      --save results of each step  in the neutral file
      io.saveMeshFile(em,  '$SIMULATIONDIR/out/$SIMULATIONNAME.nf'..'_'..i, 'nf',
                     {'u','a','P','Pa','Pf'}, {'S'}, {split = true, saveDisplacements = true})
      em:clear()
      cont  = cont+1
      cont2 = cont*dstep*dt
    end  --end if
  end --end for

  --close output files
  io.closeMeshFile(file)
  loadPf:close()
end
```

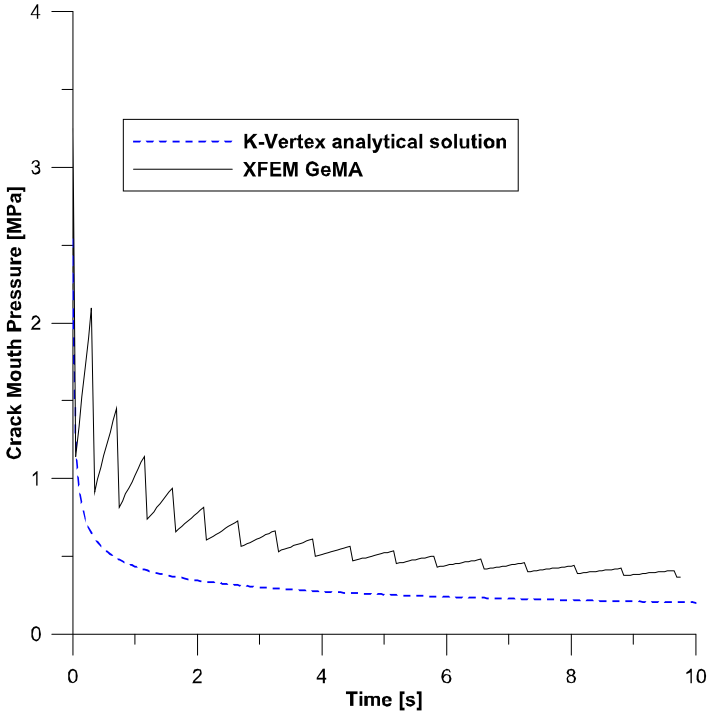
Results: KGD Analytical Model in K-vertex Propagation Regime (Detournay, 2004)

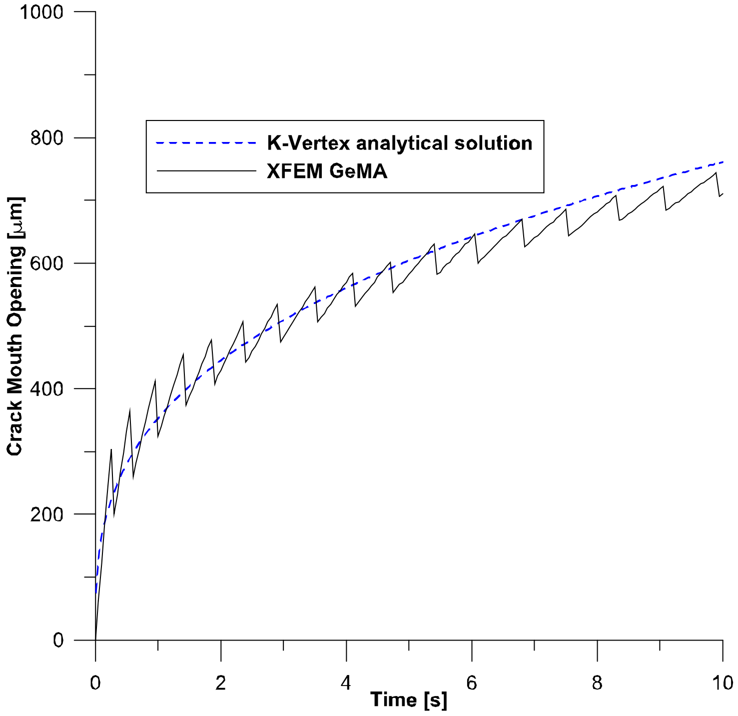
$$\omega(x,t) = \epsilon_k * L_k * \Omega(\xi)$$

$$P(x,t) = \epsilon_k * E' * \Pi$$

$$\epsilon_k = \left(\frac{K'}{E'^4 Q_0 t}\right)^{\frac{1}{3}}, \qquad \Omega = \frac{\pi^{1/3}}{2}\sqrt{1-\xi^2}, \qquad E' = \frac{E}{1-v^2}$$

$$L_k = \left(\frac{E' Q_0 t}{K'}\right)^{\frac{2}{3}}, \qquad K' = 4K_{Ic}\left(\frac{2}{\pi}\right)^{1/2}, \qquad \Pi = \frac{\pi^{1/3}}{8}$$



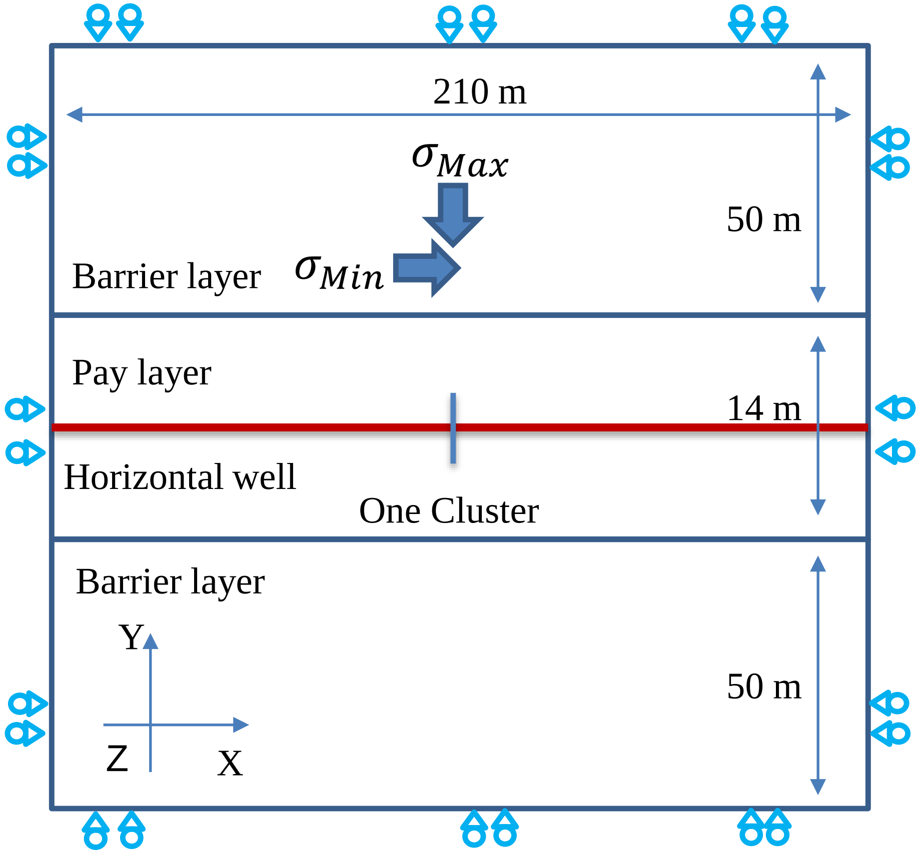


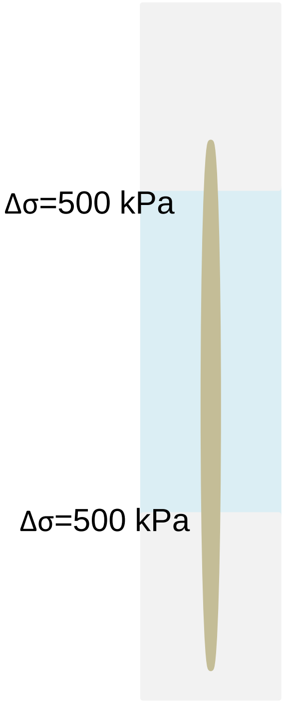
Good agreement is obtained between the profile of the crack mouth opening and the analytical solution.
The pressure obtained from numerical simulations is always greater than the pressure obtained analytically because it neglects the hydro-mechanical coupled behavior of the surrounding porous medium, our model takes this coupling into account (Carrier & Granet, 2012; Mohammadnejad & Khoei, 2013).

2 – A SINGLE HYDRAULIC FRACTURE ENTERING ANOTHER LAYER WITH SYMMETRIC STRESS CONTRAST

The Problem

- This example presents a transient analysis of a single hydraulic fracture entering another layer with symmetric stress contrast to be compared with Simonson analytical solution





Reference: "Containment of Massive Hydraulic Fractures", Simonson et al.

Key Example Points

- This example shows how to create a simple orchestration of a geostatic step which is intended to get equilibrium in the model for the initial state of a hydraulic fracturing simulation.

- This example shows the orchestration of the SIGINI subroutine which distributes the initial stress states in the model

- This example builds heavily on the previous one and only shows key difference points. Please refer to the simulation files for the complete source.

Simulation file: Simonson.lua

Solution File: SIGINI

This orchestration script distributes the initial stress state in the model.

```
--This subroutine mimics the subroutine of the same name in Abaqus to define the initial
--state of stress in situ
-- Stress accessor
local accS = m:gaussAttributeAccessor('S',1, true)
-- node coordinate accessor
local coordAc = m:nodeCoordAccessor()

-- For each element
for i=1, m:numCells() do

  local e   = m:cell(i)                                 -- element
  local ir  = m:elementIntegrationRule(e:type(), 1)    -- integration rule
  local shp = assert(e:shape())                         -- element shape

  -- Get elemental node coordinates
  local Xnode = e:nodeMatrix(coordAc, true)
  assert(ir:numPoints() == 4) -- verify that elements contain 4 integration points
```

… continues on the next slide

Solution File: SIGINI

… continued from previous slide

```
  -- 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
    --These conditionals are set to define different stress state for three layers depending on depth
    if (coord(2) >= 14) then
      Sv = -2000 --Vertical stress component
      Sh = -500  --Horizontal stress component
    end
    if (coord(2) < 14 and coord(2) > 7) then
      Sv = -2000 --Vertical stress component
      Sh = -500  --Horizontal stress component
    end
    if (coord(2) <= 7) then
      Sv = -2000 --Vertical stress component
      Sh = 0     --Horizontal stress component
    end
    --fill stress vector to pass it to the XFEM code
    local v = {Sh, Sv, Sh, 0} -- stress vector
    accS:setValue(e, j, v)    -- Passing element and gauss point indexes and stress vector to the XFEM code
  end
end
```

Solution File: Geostatic Step

```
-- Create the solver model and execute the Geostatic step in which preprocessing (crack nodes are
-- added to the mesh) is NOT performed. In the xfem.init, the hydromechanical physic, the solver id,
-- solver options and xfemOptions are passed for the Geostatic step
local solver2 = xfem.init({'HMCoupledXfem'}, 'solver', solverOptions, xfemOptions)

--this file is created to save the results of each step in one archive for postprocessing purposes
--such as analysis of variables along time
local file = io.prepareMeshFile(m, '$SIMULATIONDIR/out/Simonsontotal', 'nf',
                                {'u','a','P','Pa','Pf'}, {'S'}, {saveDisplacements=true})

--Definition of the parameters used in the orchestration
local dt      = solverOptions.timeInitIncrement
local TimeFin = solverOptions.timeMax
local Time    = dt
local nsteps  = TimeFin / dt
for i=1, nsteps do
  --Run transient analysis
  --In the xfem.step, the solver parameters and time step are passed
  xfem.step(solver2,dt)
  --After the Geostatic step finished, results are saved in the output file
  io.addResultToMeshFile(file, i*dt)
end

--set current time to zero for the subsequent analysis (Injection Step)
setCurrentTime(0.0)
```

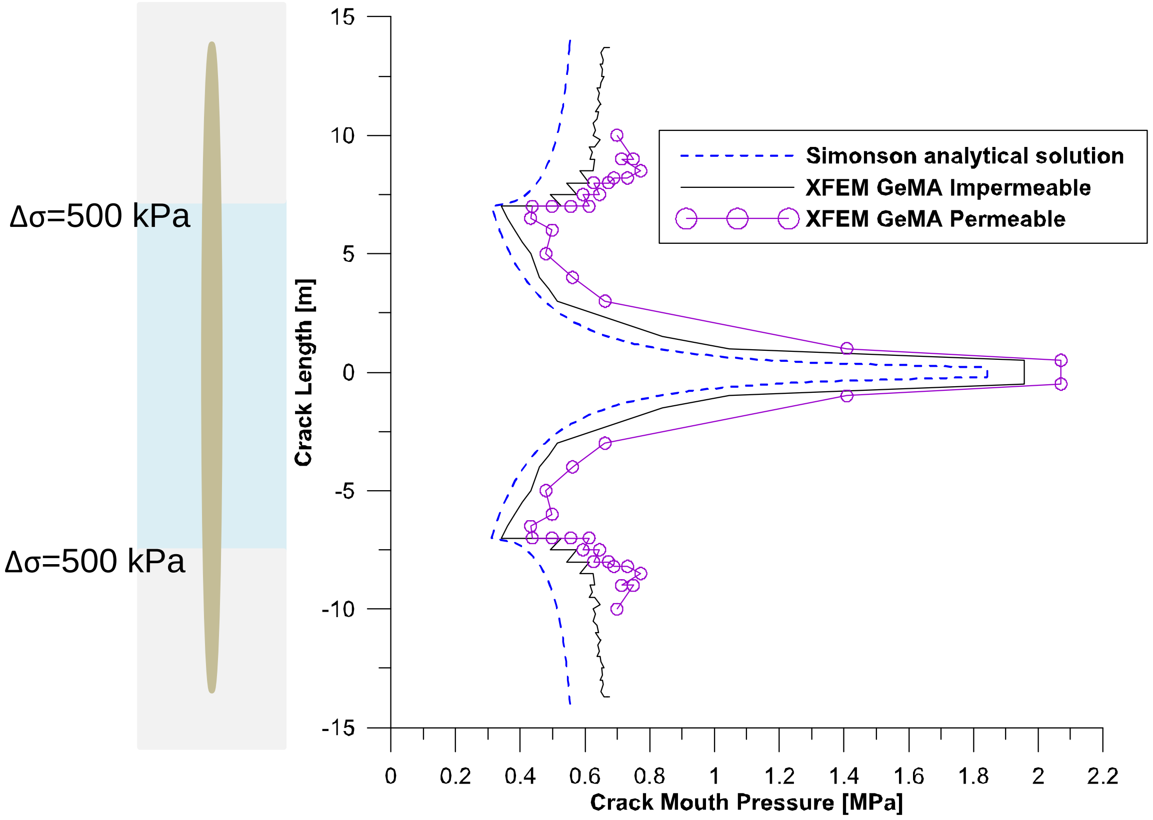
RESULTS: Simonson's Analytical Solution

According to Simonson's solution (1978), the injection pressure required for fracture penetration into the adjacent layers with symmetrical in-situ stress contrast is

$$P = \sigma_{res} + \frac{K_{1c}}{\sqrt{\pi L}} + \frac{2(\sigma_{barrier} - \sigma_{res})}{\pi} \cos^{-1}\left(\frac{h_{res}}{2L}\right)$$

$$h_f = 2L$$

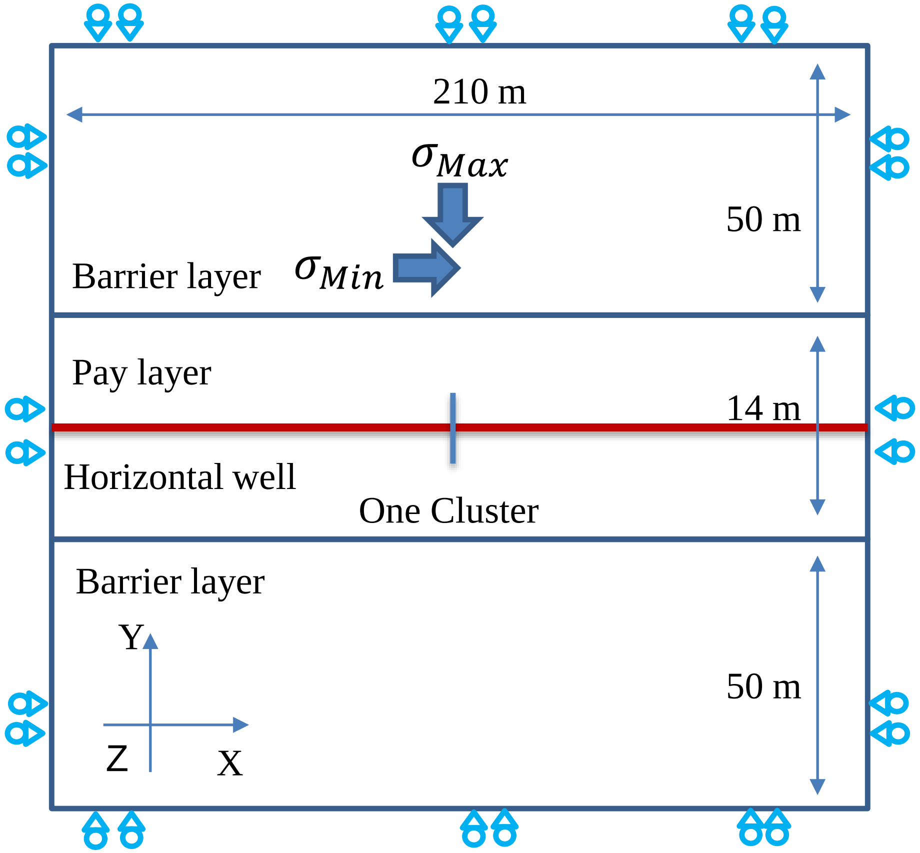
When the fracture penetrates into the adjacent layers where the in-situ horizontal stress is greater than that of the reservoir, the pressure required for fracture propagation increases (h > 7 m and h< -7 m).

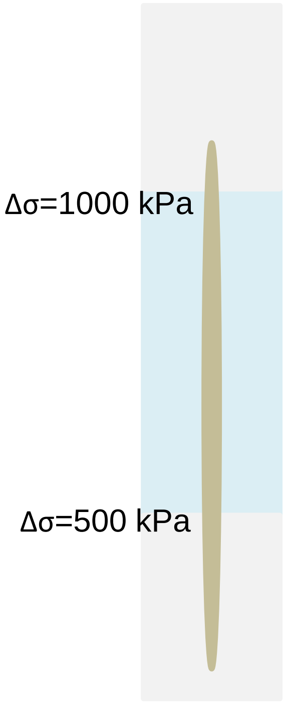


3 – A SINGLE HYDRAULIC FRACTURE ENTERING ANOTHER LAYER WITH ASYMMETRIC STRESS CONTRAST

The Problem

- This example presents a transient analysis of a single hydraulic fracture entering another layer with Asymmetric stress contrast to be compared with Fung analytical solution





Reference: **“**Calculation of Vertical Fracture Containment in Layered Formations.”, Fung et al.

Simulation file: Fung.lua

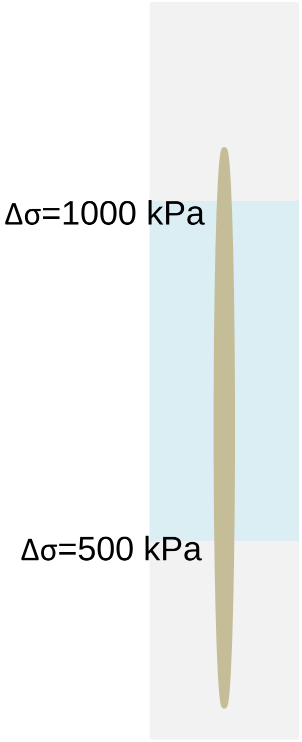
RESULTS: Fung’s Semianalytical Solution

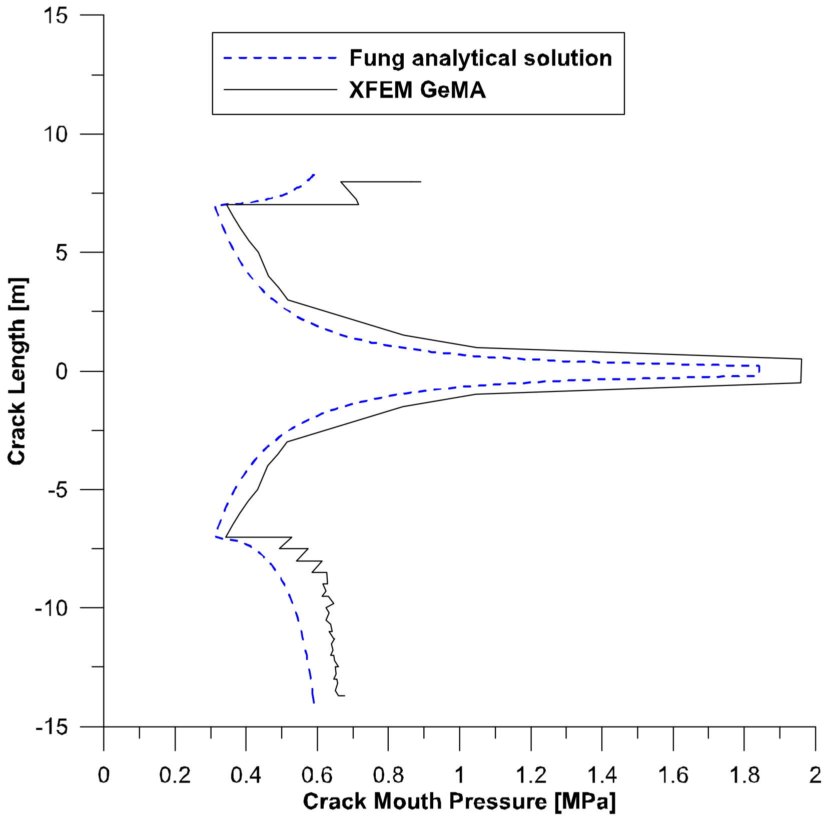
Fung et al. (1987) developed a semi-analytical procedure for computing the injection pressure required for fracture penetration into the adjacent layers with an arbitrary in-situ horizontal stress distribution.

$$K_{Ic1} - K_{Ic2} = \sqrt{\frac{h}{2\pi}} \sum_{i=1}^{n} \left\{ (\sigma_{i+1} - \sigma_i) . \sqrt{1 - \left(\frac{2h_i - h}{h} \right)^2} \right\}$$

$$K_{Im} = \sqrt{\frac{h}{2\pi}} . \left\{ (P - \sigma_n)\pi + \sum_{i=1}^{n} (\sigma_{i+1} - \sigma_i) . \left[2\sin^{-1} \sqrt{\frac{h_i}{h}} - (-1^m) \sqrt{1 - \left(\frac{2h_i - h}{h} \right)^2} \right] \right\}$$

The fracture penetrates farther into the lower layer due to lower in-situ stress in comparison with the upper layer.

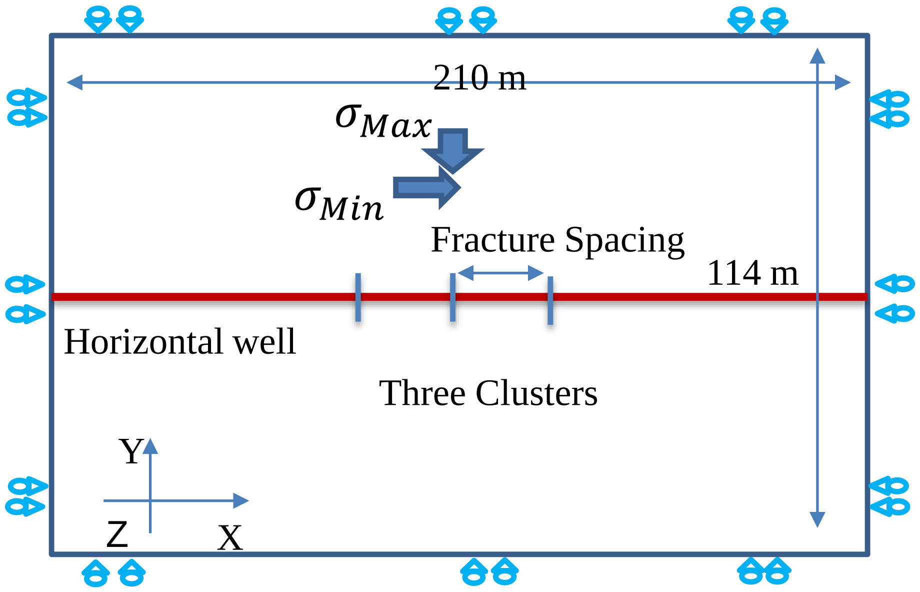




- 4 –TRANSIENT ANALYSIS OF THREE SIMULTANEOUS HYDRAULIC FRACTURES WITH A HOMOGENEOUS STRESS STATE IN A SINGLE LAYER AND FRACTURE SPACINGS OF 7 AND 20 METERS.

The Problem

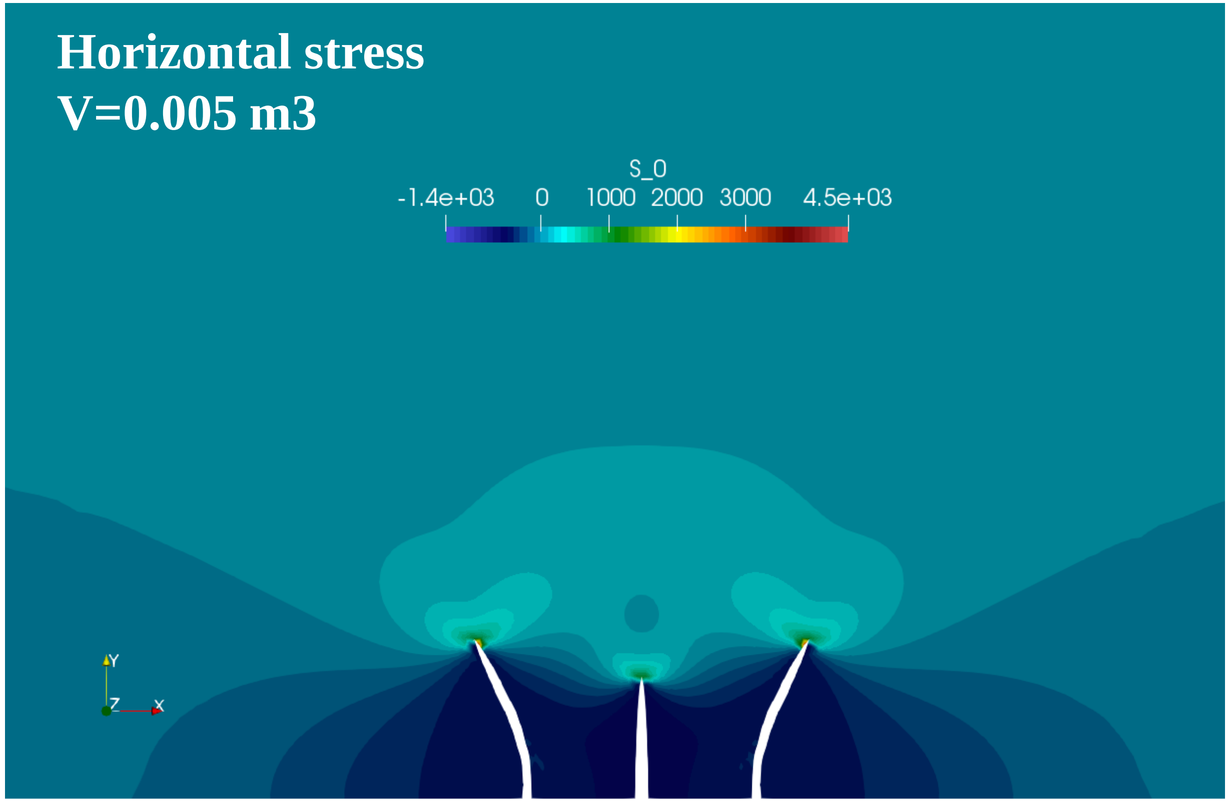
- This example presents a transient analysis of three simultaneous hydraulic fractures with a homogeneous stress state in a single layer and fracture spacings of 7 and 20 meters.

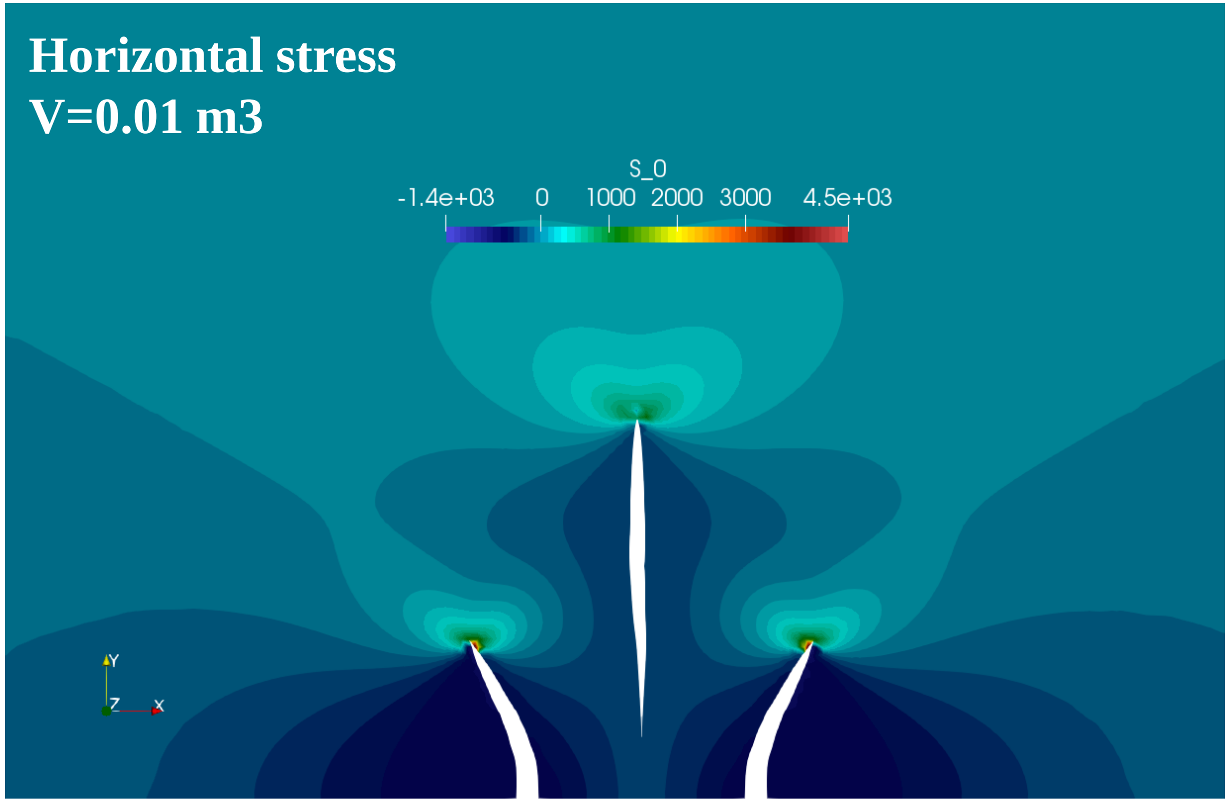


Simulation files:
SimHF7.lua
SimHF20.lua

Multiple Simultaneous Hydraulic Fracturing

Homogeneous stress state, Fracture Spacing =7 m

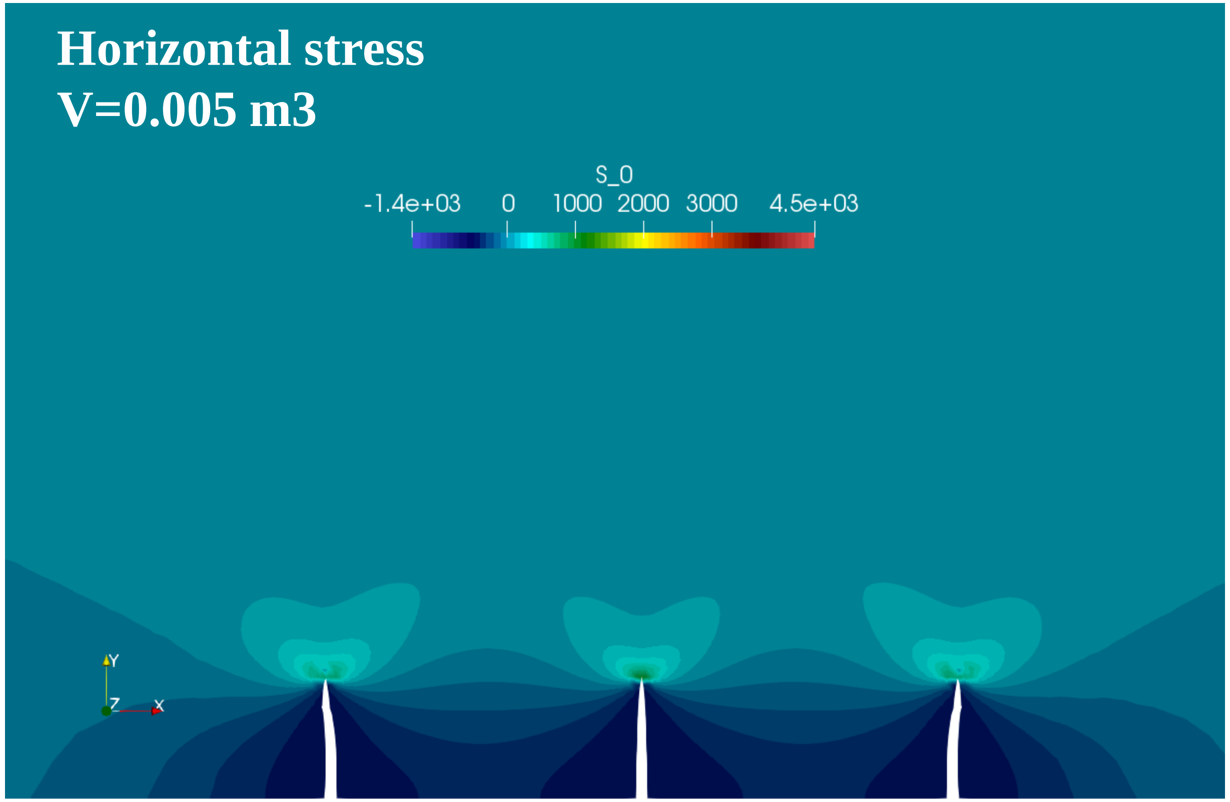


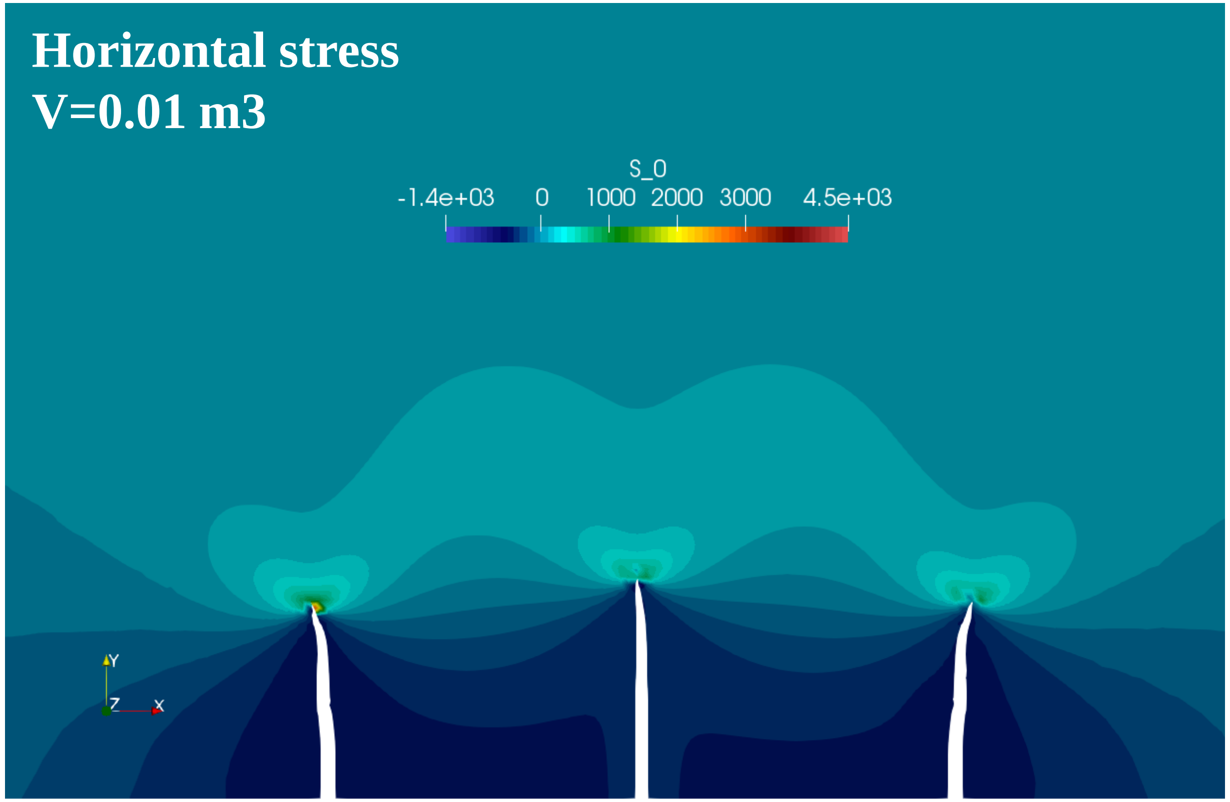


For a simultaneous scheme, stress shadowing effects of closely spaced clusters generates shorter and outward deviated side fractures and a straight longer fracture in the middle

Multiple Simultaneous Hydraulic Fracturing

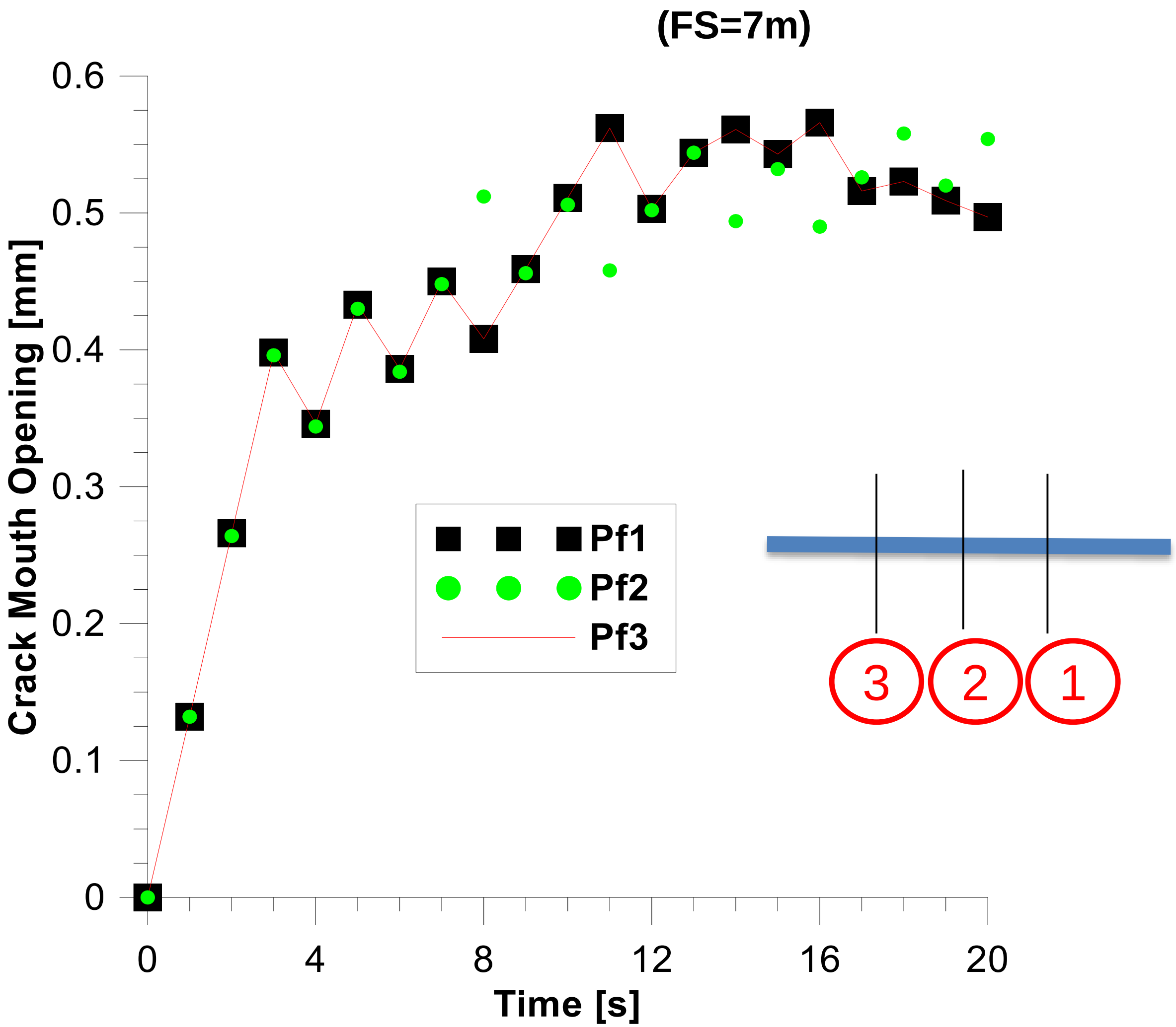
Homogeneous stress state, Fracture Spacing =20 m

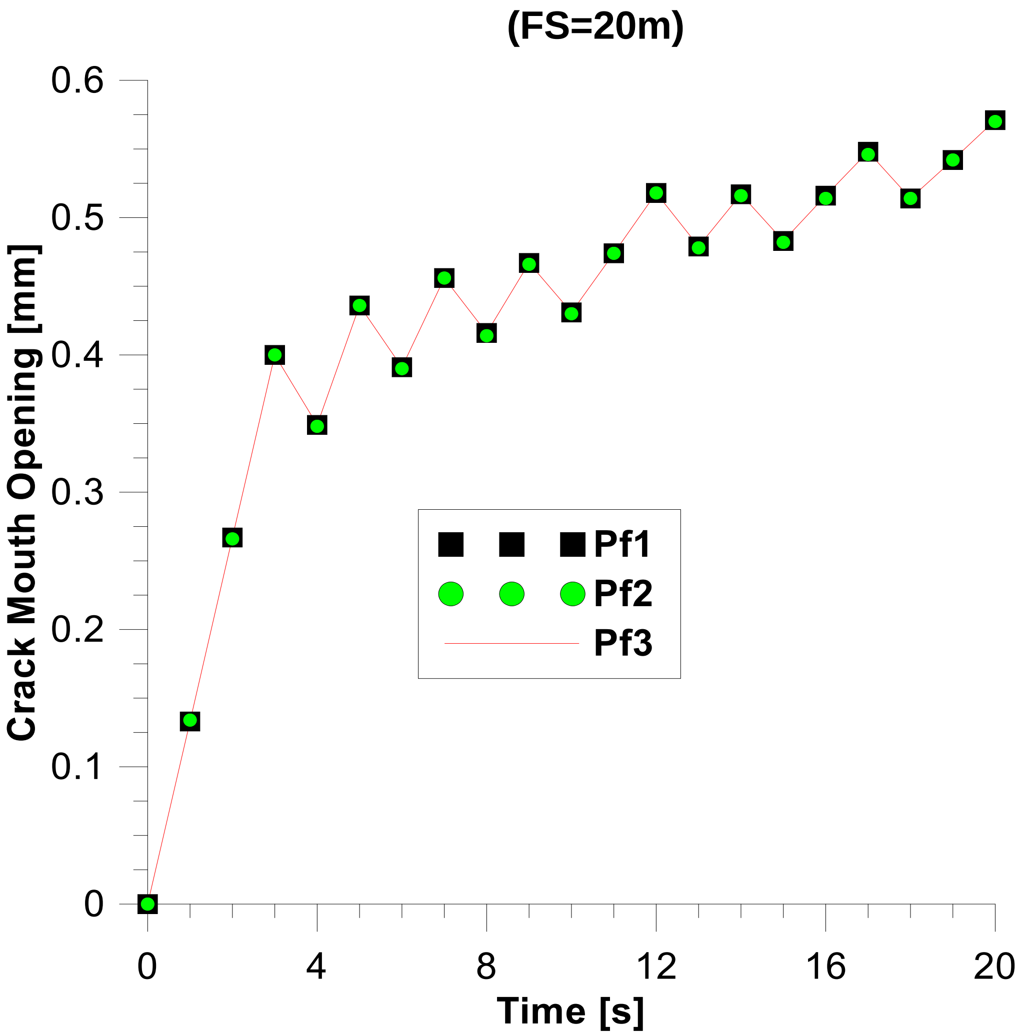




As fracture spacing is increased (right), the stress shadowing effect decreases allowing all three fractures to propagate straight.

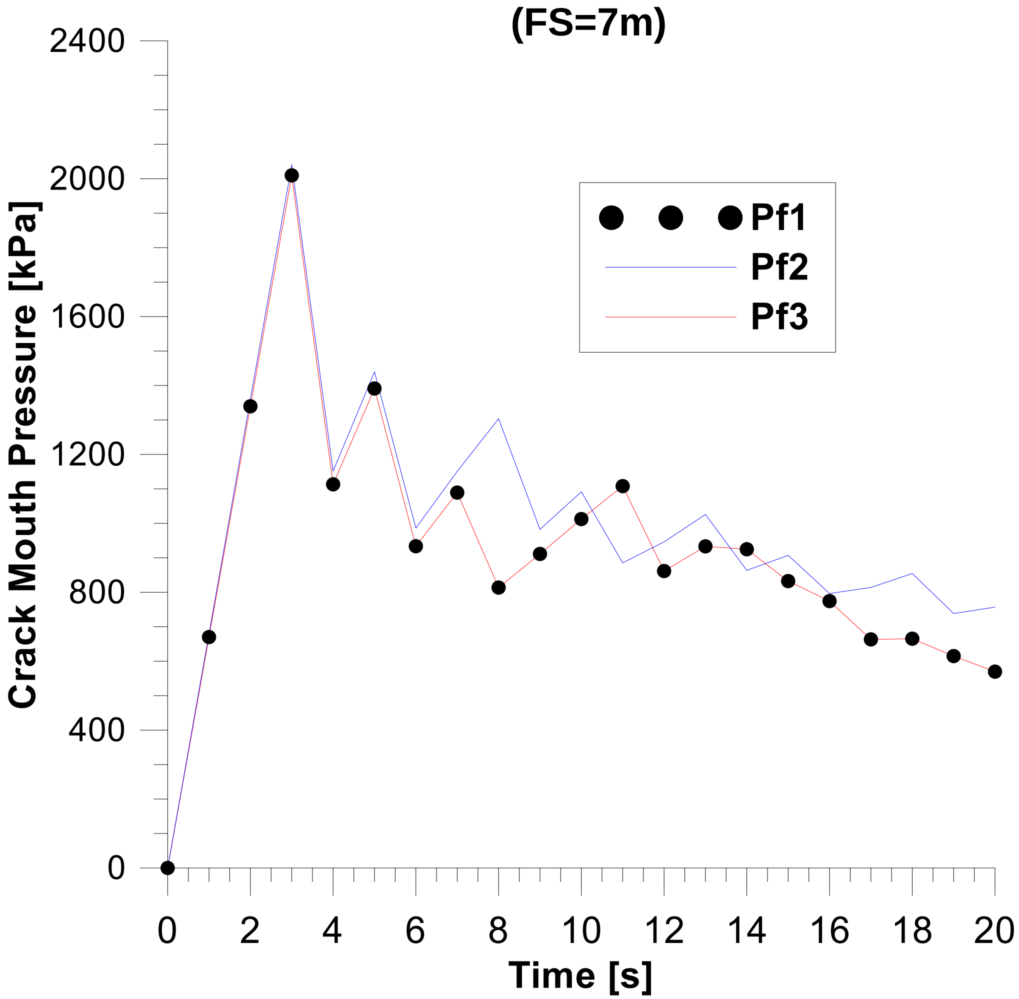
Multiple Simultaneous Hydraulic Fracturing

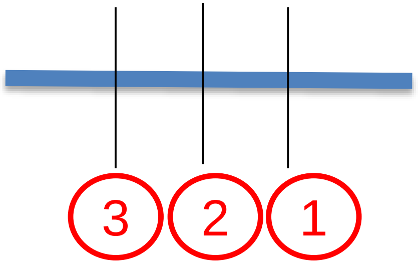


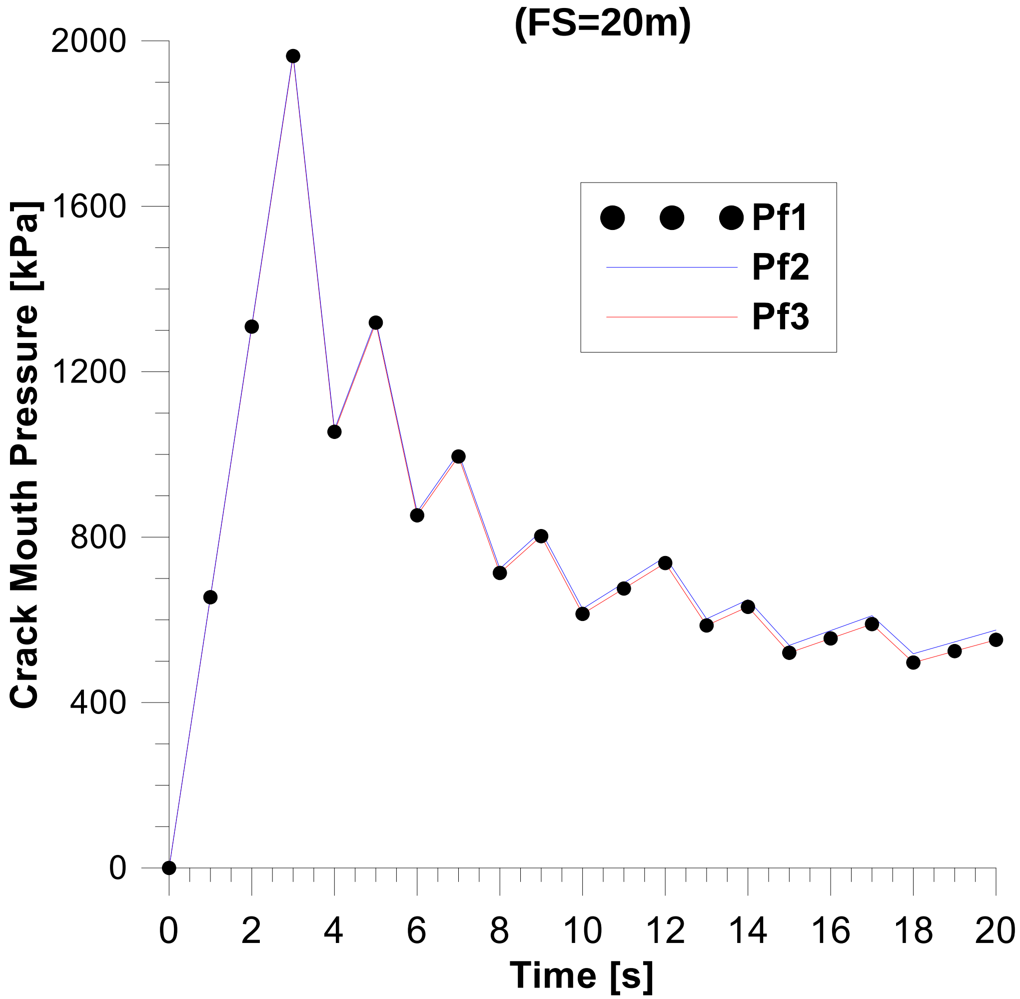


Due to the stress shadowing effect, fracture propagation occurs with a time lag between middle and side clusters. In contrast, a synchronized fracture propagation is observed when the spacing is enough to avoid stress interference between fractures.

Multiple Simultaneous Hydraulic Fracturing



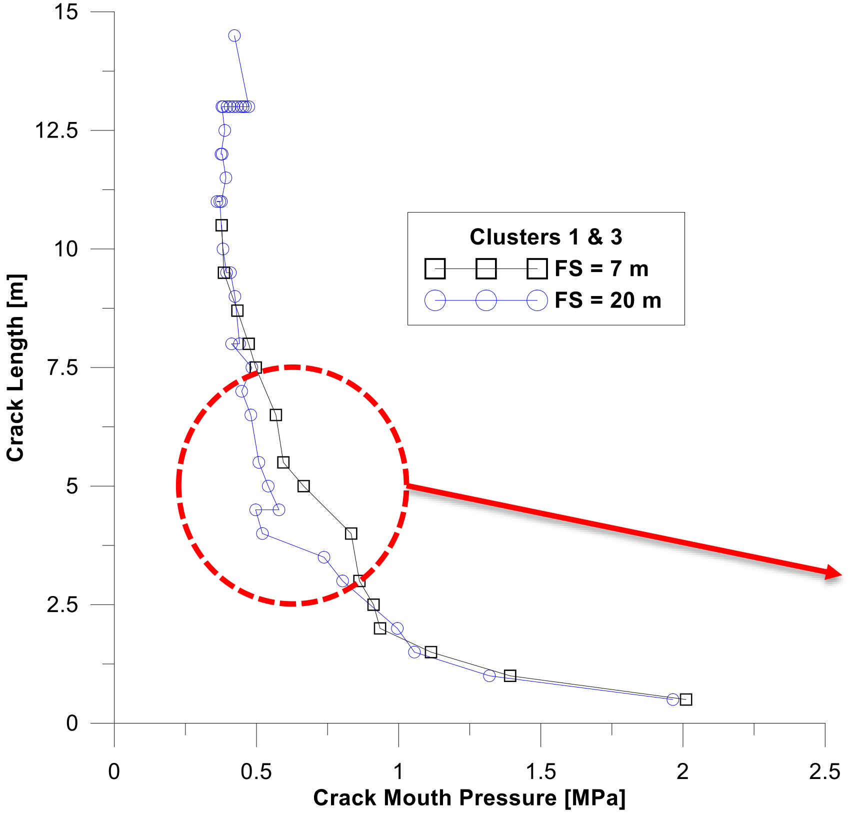


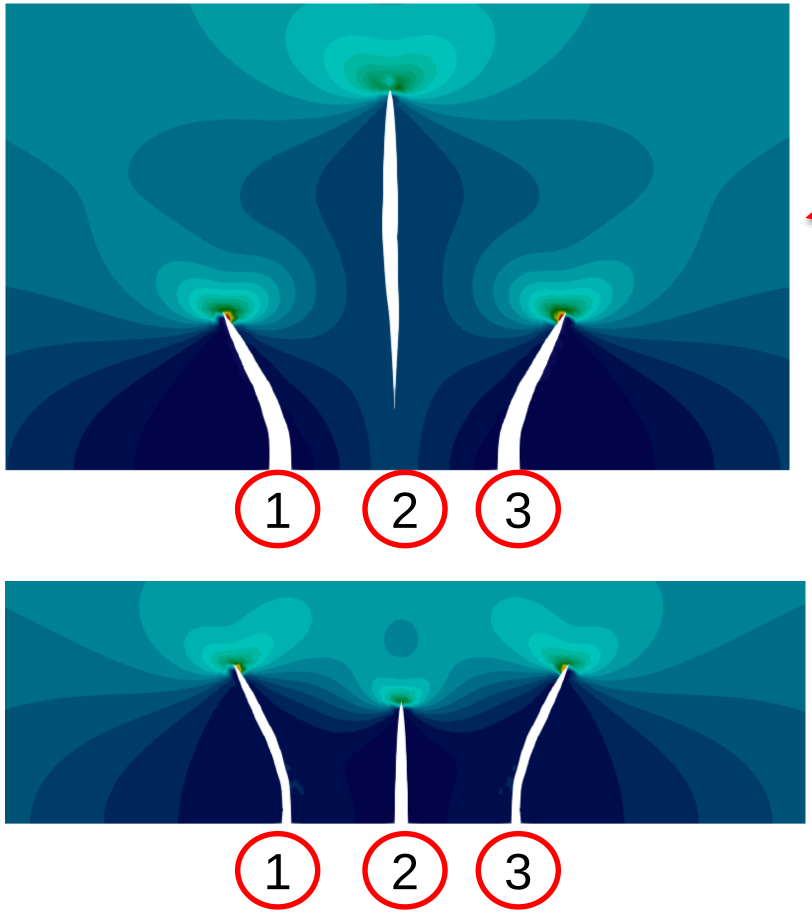


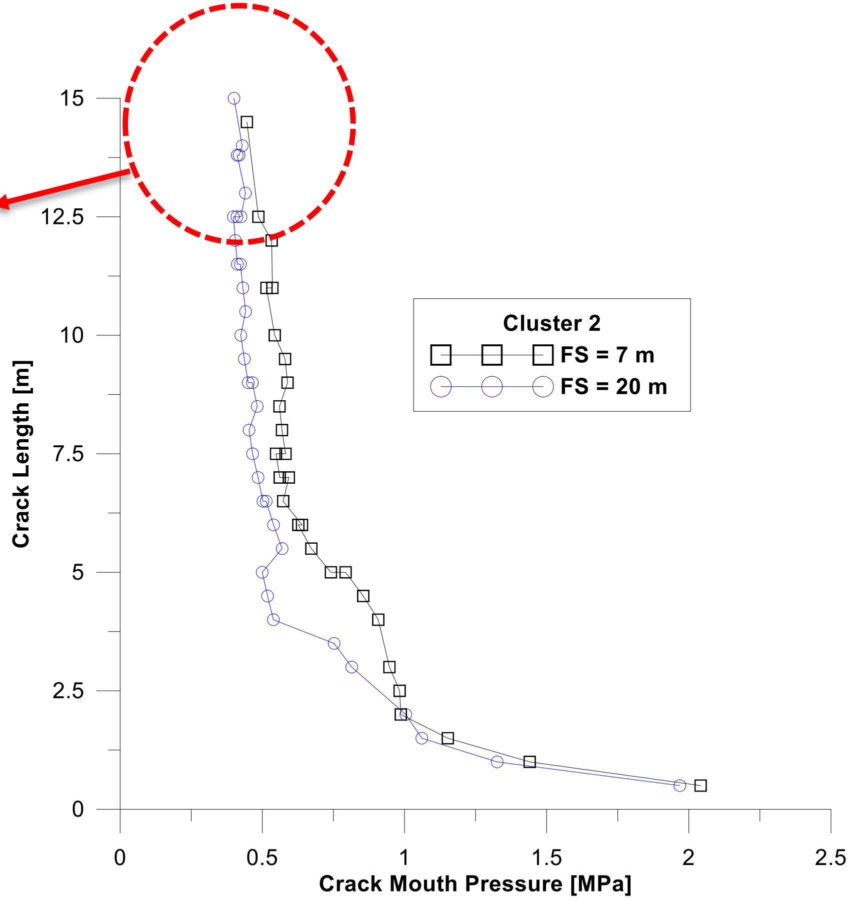
As stress interference on fracture propagation increases, the breakdown pressure required to propagate increases as well. Consequently, a pressure difference can be observed between the side and middle fractures.

Multiple Simultaneous Hydraulic Fracturing

Homogeneous stress state





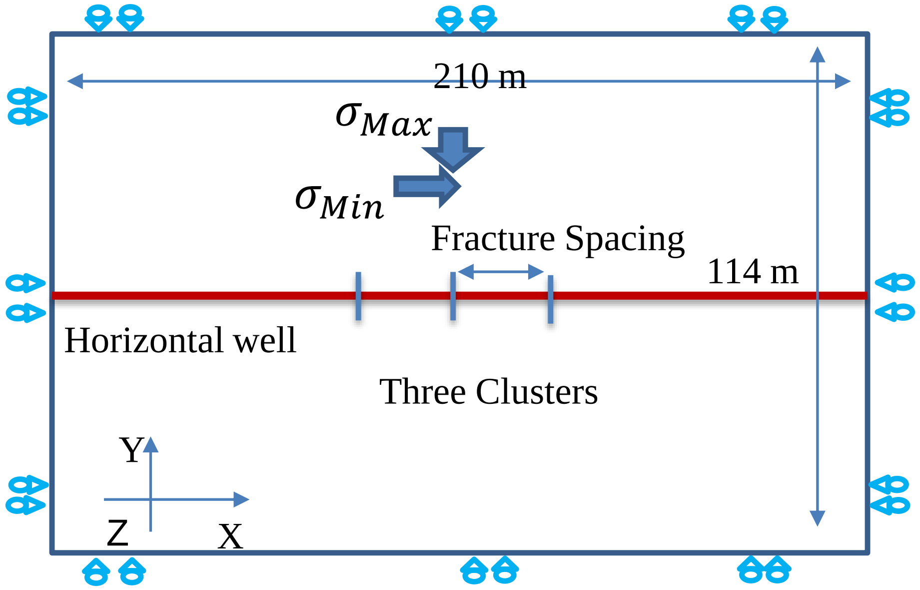


The stress shadowing effect increases the fracture pressure required to propagate (left figure). As middle fracture propagates further than the side ones, less pressure needs to be accumulated due to the lower stress interference.

- 5 –TRANSIENT ANALYSIS OF THREE SEQUENTIAL HYDRAULIC FRACTURES WITH A HOMOGENEOUS STRESS STATE IN A SINGLE LAYER AND A FRACTURE SPACING OF 7 AND 20 METERS.

The Problem

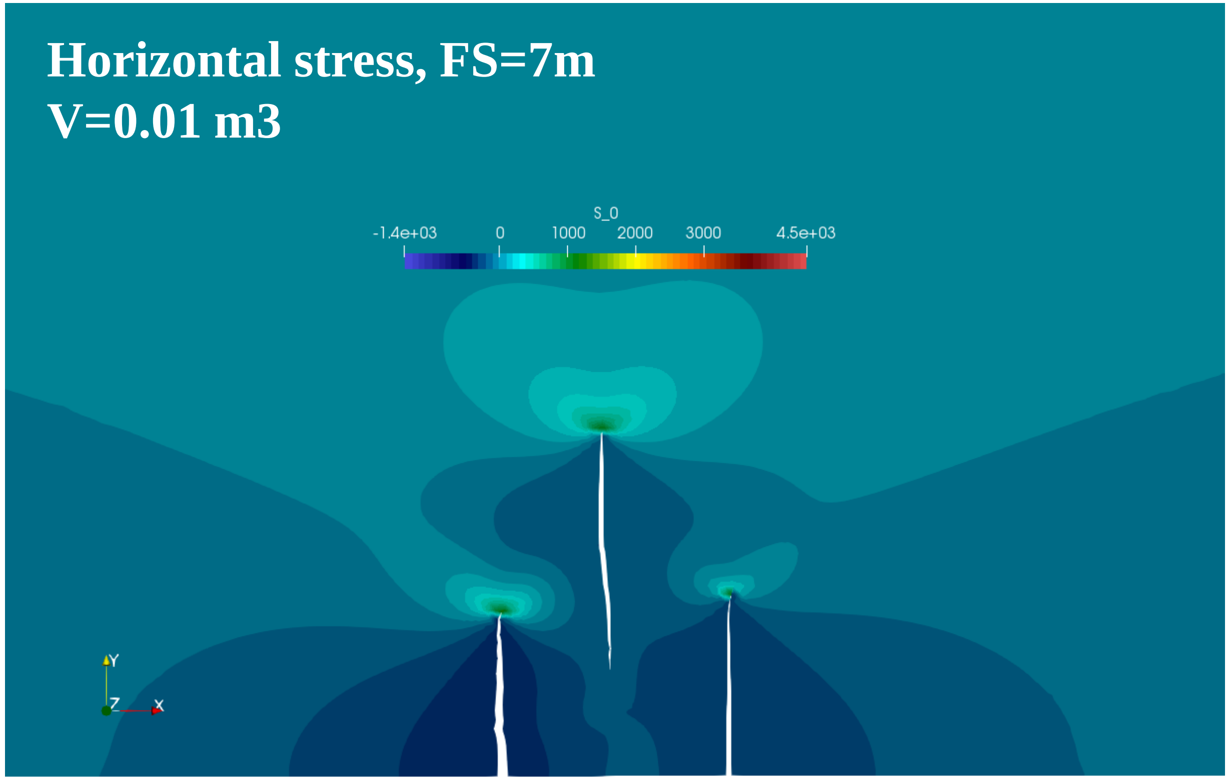
- This example presents a transient analysis of three sequential hydraulic fractures with a homogeneous stress state in a single layer and fracture spacings of 7 and 20 meters.

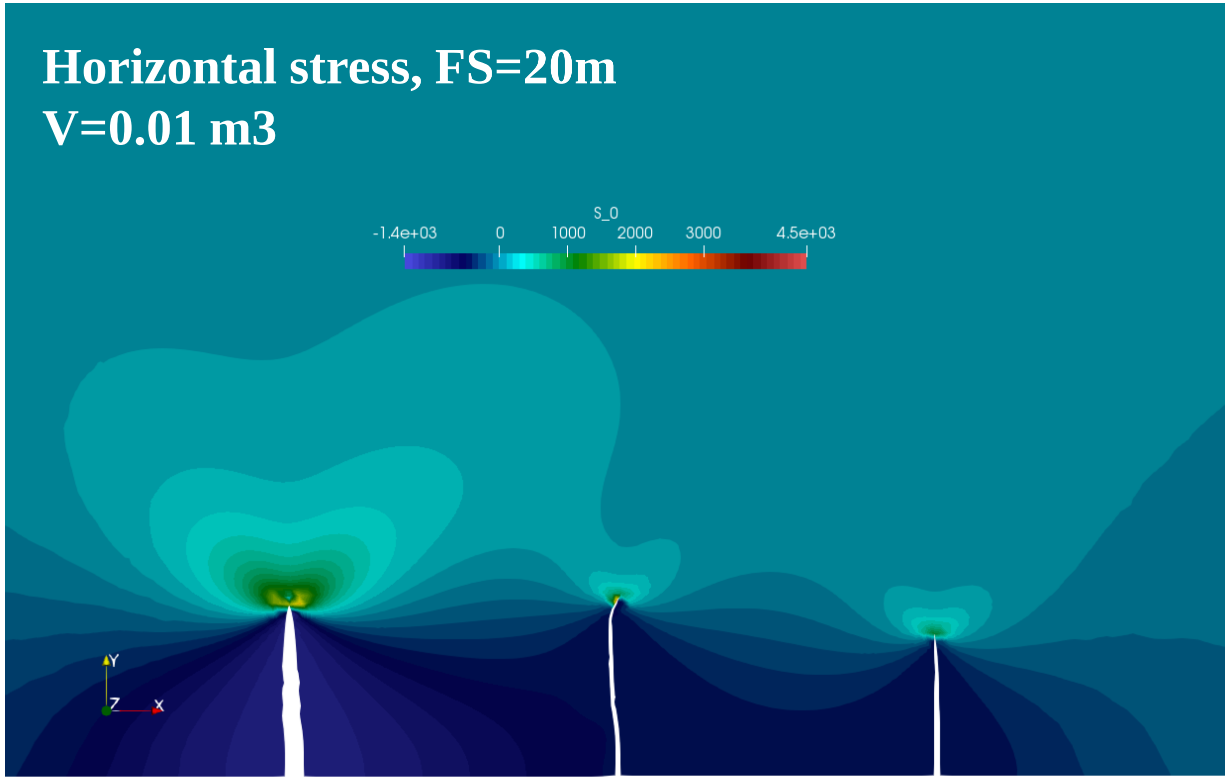


Simulation files:
SeqHF7.lua
SeqHF20.lua

Multiple Sequential Hydraulic Fracturing

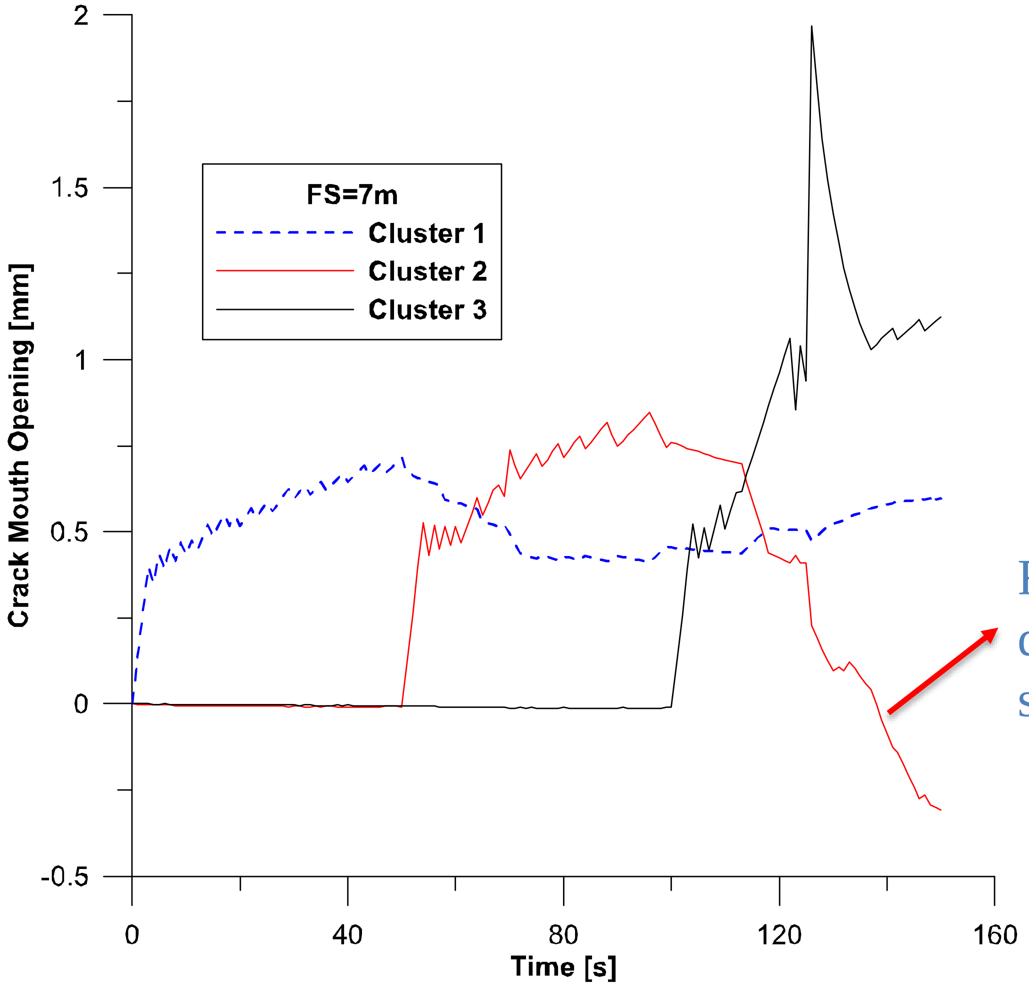
Homogeneous stress state

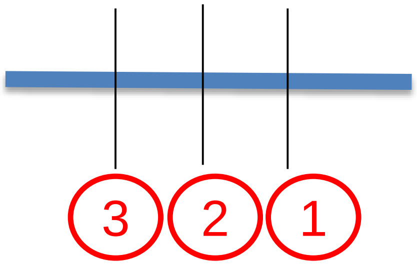




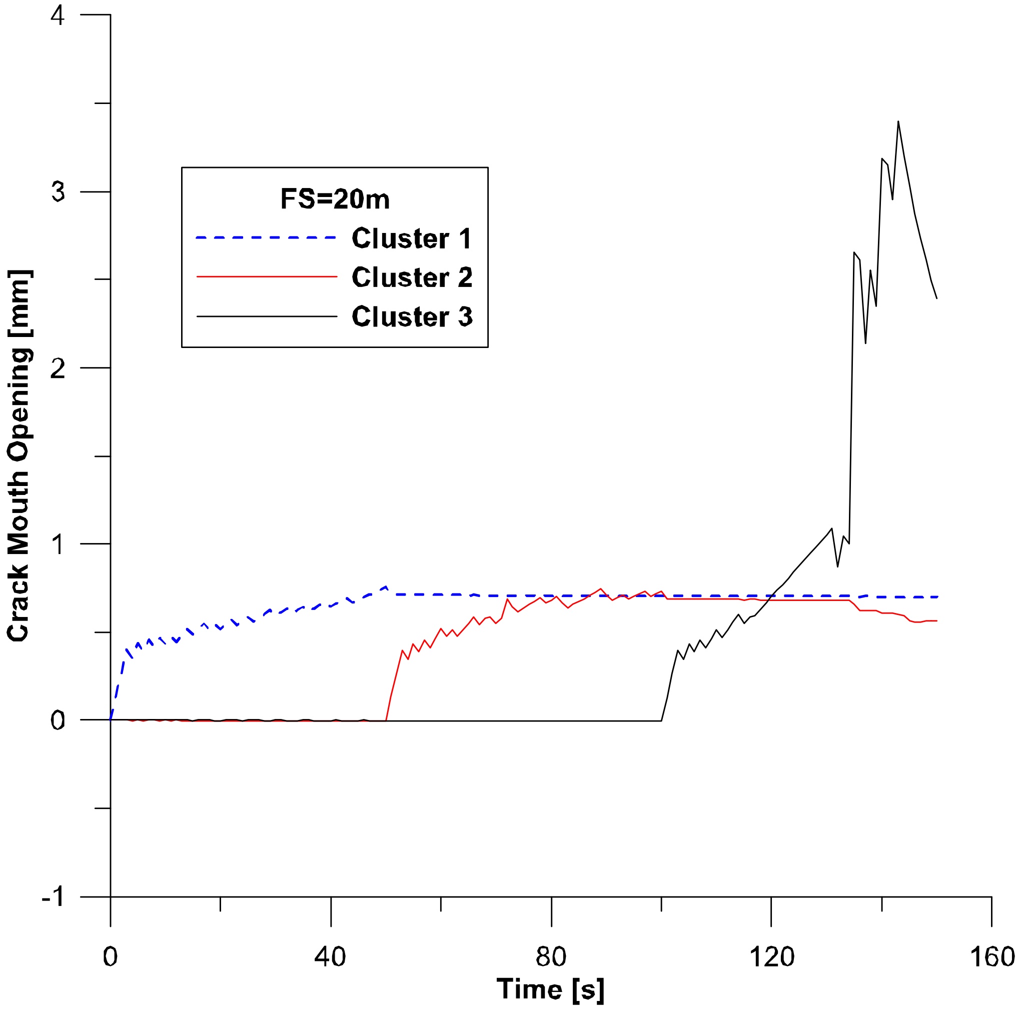
As fracture spacing is increased (from left figure to right), the stress shadowing effect decreases allowing all three fractures to propagate mostly straight.

Multiple Sequential Hydraulic Fracturing



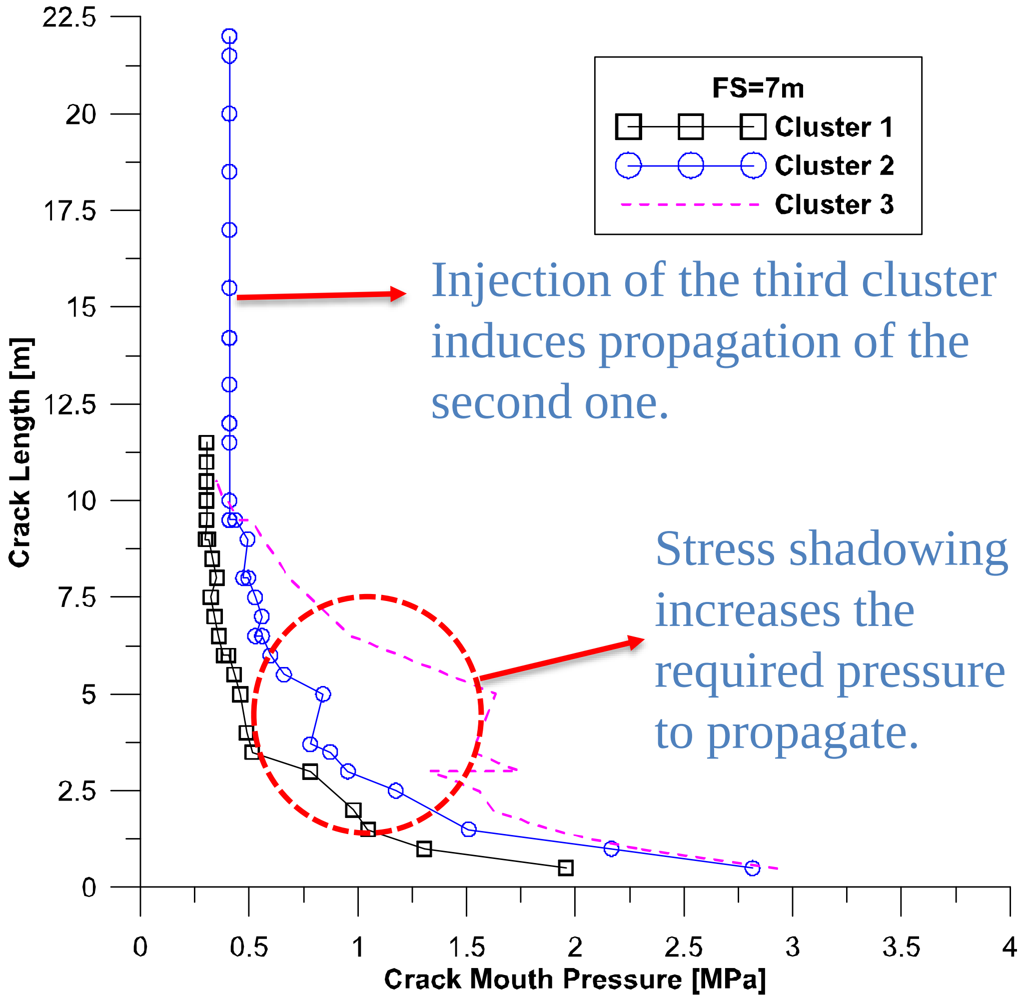


Fracture closure due to stress shadowing effect.



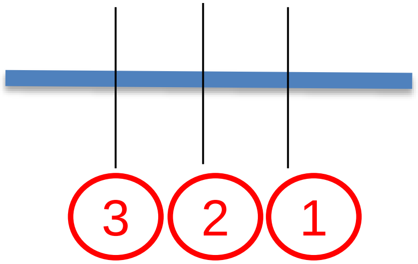
As fracture spacing is increased (right), no fracture closing is observed due to less stress shadowing effect and wider aperture is obtained for the last injected cluster

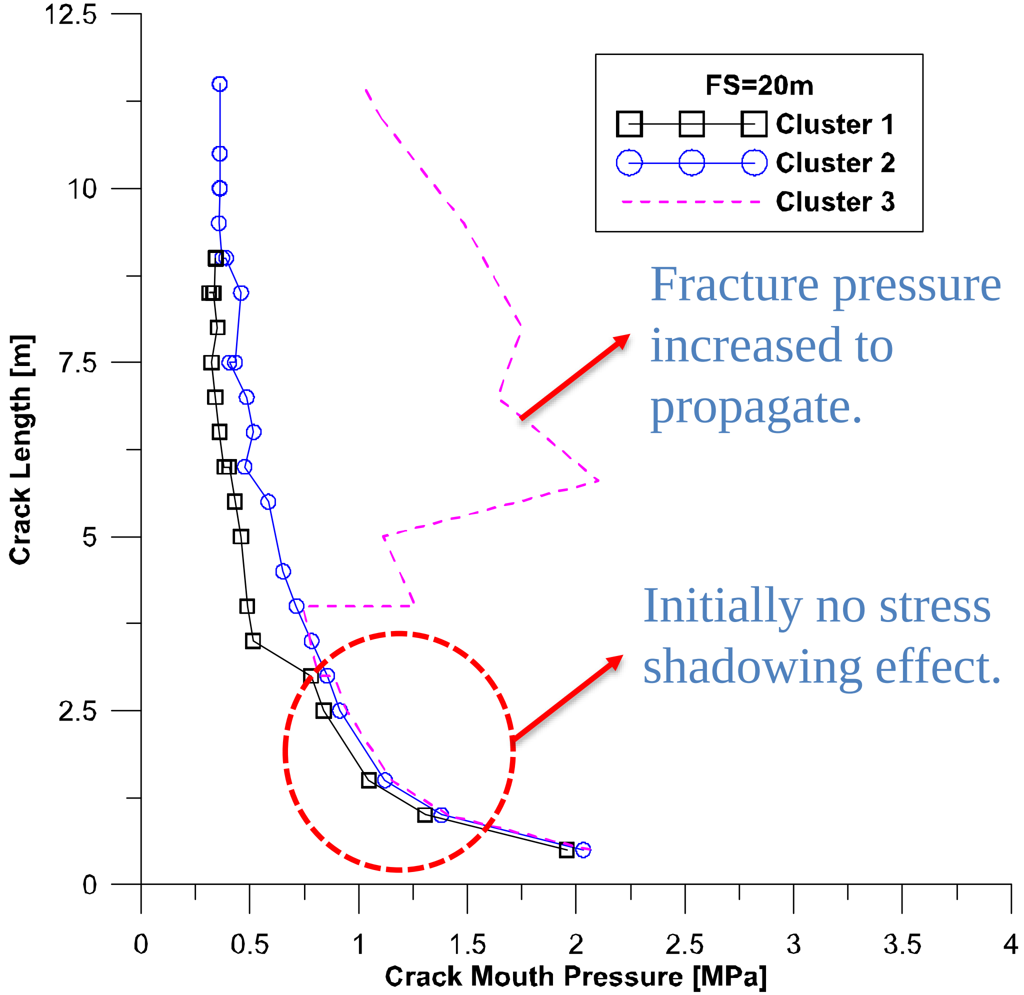
Multiple Sequential Hydraulic Fracturing



Injection of the third cluster induces propagation of the second one.

Stress shadowing increases the required pressure to propagate.





Fracture pressure increased to propagate.

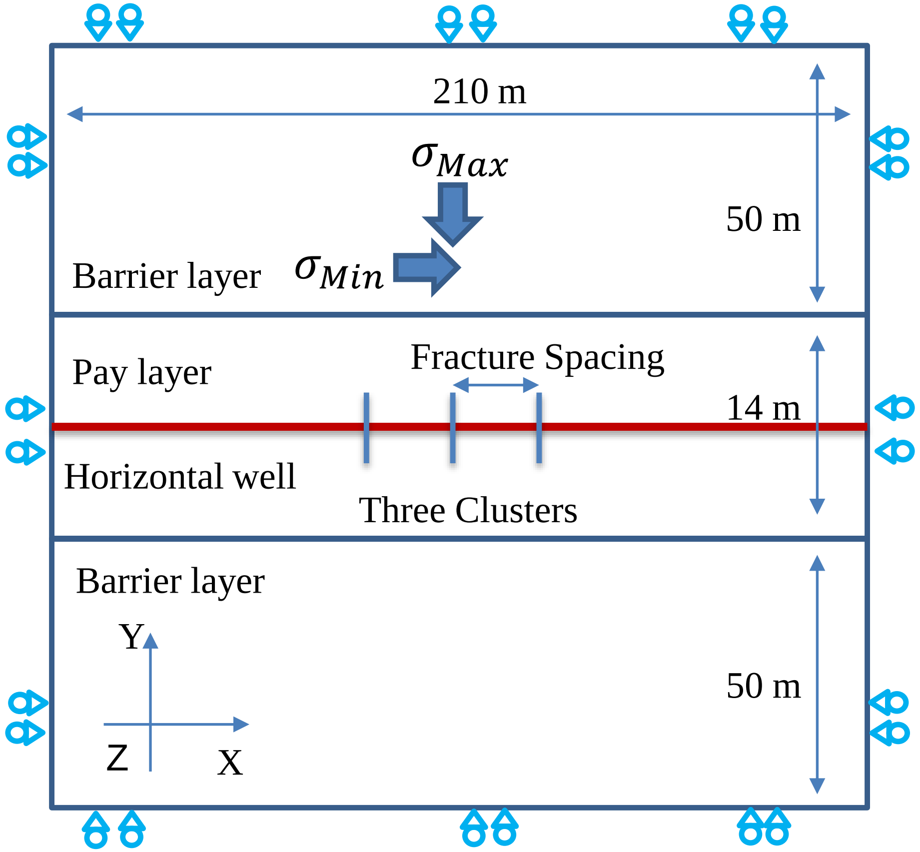
Initially no stress shadowing effect.

As clusters are injected sequentially, the required pressure to propagate is increased due to the stress interference of the previous propagated fractures.

6 –TRANSIENT ANALYSIS OF THREE SIMULTANEOUS HYDRAULIC FRACTURES ENTERING ANOTHER LAYER WITH SYMMETRIC STRESS CONTRAST AND FRACTURE SPACINGS OF 7 AND 20 METERS.

The Problem

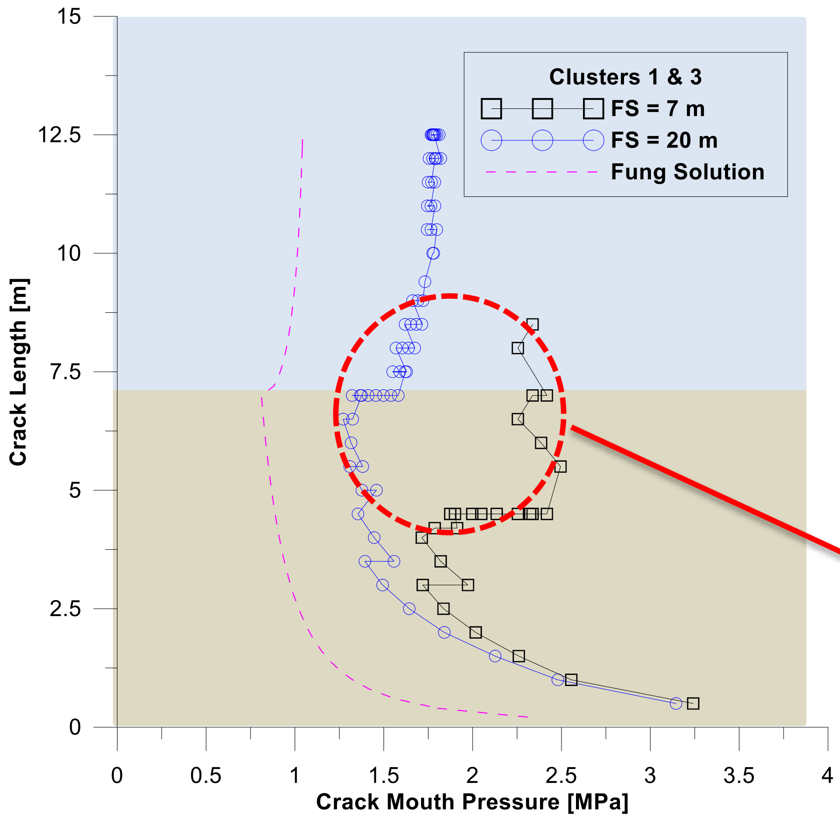
- This example presents a transient analysis of three simultaneous hydraulic fractures with a homogeneous stress state in a single layer and fracture spacings of 7 and 20 meters.

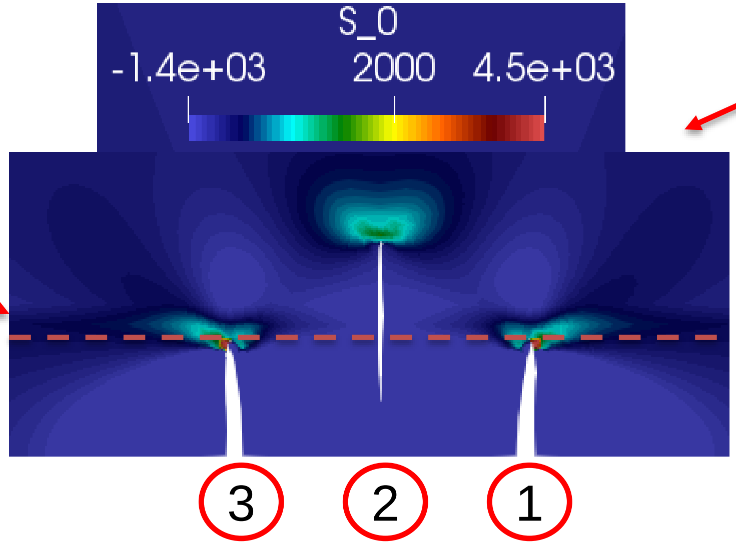


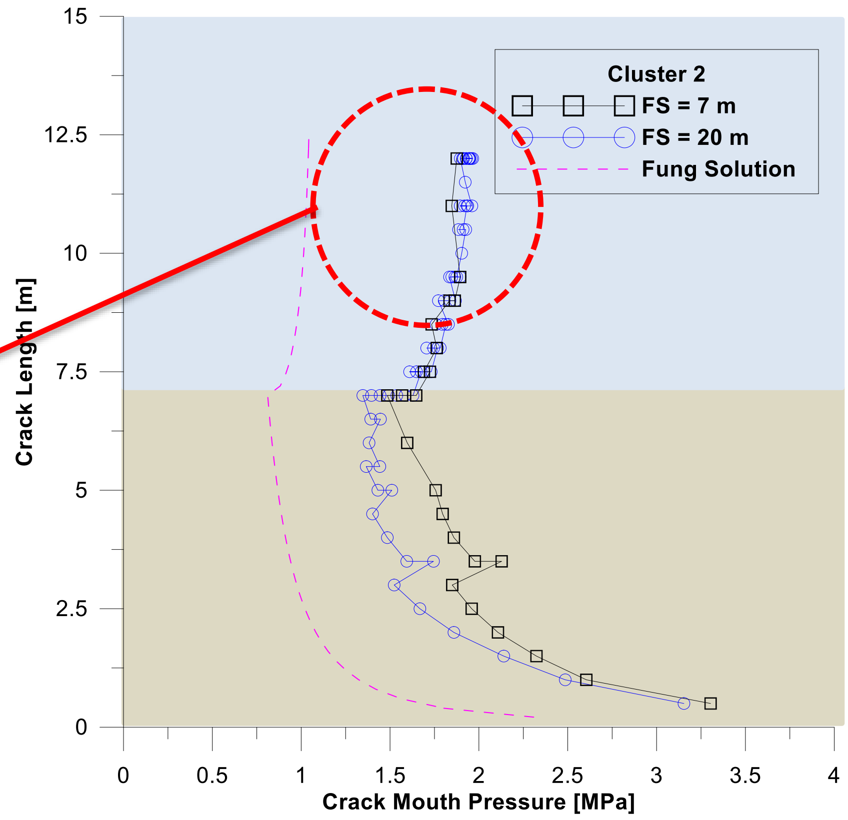
Simulation files:
SimHF7SC.lua
SimHF20SC.lua

Multiple Simultaneous Hydraulic Fracturing

Stress contrast between layers Δσ=500 kPa







The stress shadowing effect and the stress contrast between adjacent layers increase the fracture pressure required to propagate