18/06/2018

GeMA - Modelling the dissolution process of rocks

Version 1.0

Key Example Points

This example shows:

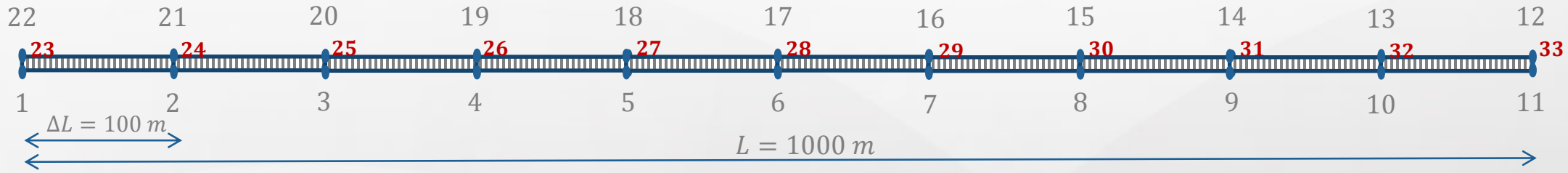
- How to setup models for transient flow analysis
- How to use an interface element as a single fracture model to represent the dissolution process in the evolution of the early karst
- How to create a Multiphysics simulation where one of the physics is implemented wholly in Lua and integrated with another GeMA physics through the global simulation loop
- How to setup and to update Gauss point attributes with values calculated by Lua functions during the transient flow analysis

1 – DISSOLUTION PROCESS OF A SIMPLE FRACTURE USING INTERFACE ELEMENTS

Example I

- The example presents the early karst aquifer evolution where the fracture is modelled as a conduit element. The fissure enlargement is governed by the dissolution process based on the algorithm proposed by Kaufmann and Braun (1999).

- The interface element int2dl6 is used to discretize the fracture as can be seen below. The conduit has a constant pore pressure difference of $\Delta p = 98.1\ kPa$ and the fissure enlargement will be analyzed at 10000 years.



Note: Each interface element of length ΔL is subdivided in 1000 increments of 0.1 m for the conduit evolution calculation.

Reference: Kaufmann, G.; Braun, J.. "**Karst aquifer evolution in fractured rocks**". Water Resources Research 35, pp. 3223-3238, 1999

A short explanation about the Kaufmann and Braun algorithm – fissure enlargement algorithm

The conduit element is subdivided into n elements.

Step 1. Calculate the flow rate Q from conduit geometry l and d, hydrostatic pressure difference Δh, and flow regime f.

The flux rate is, a priori, calculated as laminar

$$Q = \frac{g d^4 \Delta h}{128 \nu L}$$

So, it is verified the Reynolds number. If $Re \leq 2200$, returns the laminar flow. Otherwise, the friction factor for turbulence is determined from the maximum value of:

$$f_s = 0.3164 Re^{-0.25}$$

$$f_t^{-0.50} = 1.14 - 2\log\left(\frac{w}{d} + \frac{9.35}{Re} f_t^{-0.50}\right)$$

$$f_r^{-0.50} = 1.14 - 2\log\left(\frac{w}{d}\right)$$

Then,

$$K_g = f\frac{8L}{\pi^2 g d^5} \Rightarrow Q = \sqrt{\frac{\Delta h}{K_g}}$$

Step 2. Assign the $[Ca^{2+}]_j$ at the upstream end of element j and calculate the Ca^{2+} flux rate $[F_{Ca^{2+}}]_j$ in the element. The dissolution rate is modeled by a potential law and depends of the actual limit thickness δ and calcium concentration $[Ca^{2+}]$.

$$k_1 = k_0\left(1 + \frac{k_0 d(t)}{6D[Ca^{2+}]_{eq}}\right)^{-1}$$

If $[Ca^{2+}] < [Ca^{2+}]_s$

$$F_{Ca^{2+}} = k_1\left(1 - \frac{[Ca^{2+}]}{[Ca^{2+}]_{eq}}\right)^{n_1}$$

Otherwise,

$$k_2 = k_1\left(1 - \frac{[Ca^{2+}]_s}{[Ca^{2+}]_{eq}}\right)^{-1}$$

$$F_{Ca^{2+}} = k_2\left(1 - \frac{[Ca^{2+}]}{[Ca^{2+}]_{eq}}\right)^{n_2}$$

A short explanation about the Kaufmann and Braun algorithm – fissure enlargement algorithm

Step 3. Calculate additionally dissolved calcium in length element j, using

$$d[Ca^{2+}] = \frac{F_{Ca^{2+}} dA}{Q}$$

where $dA = 2\pi \Delta r_j \Delta L_j$.

Step 4. Calculate calcium concentration at the downstream end of length element j,

$$[Ca^{2+}]_{j+1} = [Ca^{2+}]_j + d[Ca^{2+}]$$

Step 5. Calculate the wall retreat within length element j:

$$\frac{dr_j}{dt} = \frac{F_{Ca^{2+}} m_r}{\rho_c}$$

Step 6. Update the radius of length element j for time step Δt

$$\Delta r_j = \Delta r_j + \frac{1}{2}\frac{dr_j}{dt}$$

Step 7. Repeat steps 2-6 for all length elements, then calculate a new effective diameter:

$$d_e^4 = L\left[\sum_{j=1}^{n} \frac{\Delta L_j}{\left(2\Delta r_j\right)^4}\right]^{-1}$$

Simulation file: simpleKarstEvolution.lua

Solution file: Friction factor

The fissure enlargement algorithm is implemented basically by a main function that calculates the conduit evolution, and other two auxiliary functions that return the friction factor and flux rate, respectively.

```
function frictionFactor(Re,      d,      w)
```

Reynolds number

Conduit diameter [cm]

Wall roughness [cm]

```
  -- Smooth friction factor
  local fs = 0.3164*Re^(-0.25)

  -- Transition friction factor (nonlinear calculations)
  -- Initial friction factor was adopted is the value of smooth friction factor, fs
  local ft = fs

  -- tolerance
  local tol = 1e-5
  local F   = ft^(-0.5)-1.14+2*math.log(w/d+9.35/Re*ft^(-0.50), 10)
  -- index of no. of iteration
  local it       = 1
  local iter_max = 10
```

… continues on the next slide

Solution file: Friction factor

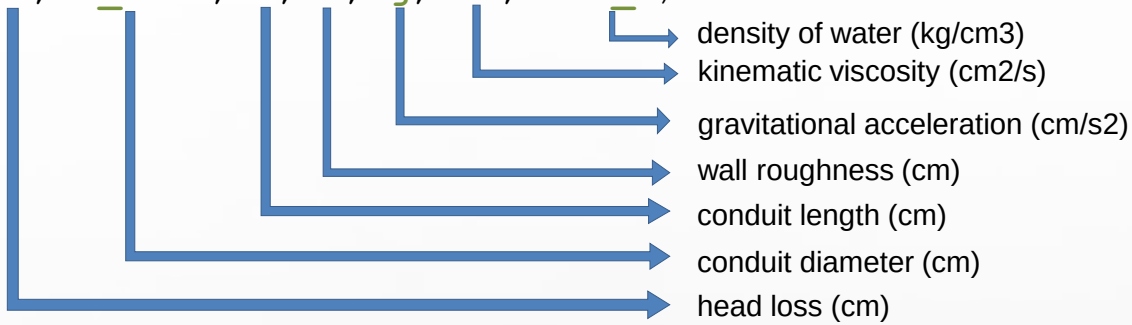
```
  while (F >= tol) do
    -- Derivative of function with respect to dgamma
    local dF = -.5/ft^1.5-9.350/(Re*ft^1.5*(w/d+9.35/(Re*ft^.5))*math.log(10))
    -- Increment of dgamma
    local dft = -F/dF
    -- Update dgamma
    ft = ft+dft
    -- Stop criteria
    F = ft^(-0.5)-1.14+2*math.log(w/d+9.35/Re*ft^(-0.5), 10)

    if (it==iter_max) then
      error("Max. number of iterations!!!")
    end
    it = it+1
  end
  -- Rough friction factor
  local Bcont = 1.14-2*math.log(w/d, 10)
  local fr    = (1/Bcont)^2

  return math.max(fs,ft,fr)
end
```

Nonlinear method to calculate the transition friction factor

Returns the maximum among the friction factors

Solution file: Flux rate

```
function fluxrateQ(dH, d_init, L, w, g, nu, rho_w)
```




```
  -- Flux rate for laminar flow
  local Q = dH*math.pi*g*(d_nit)^4/(128*nu*L)

  -- Reynolds number
  local Re = 4*Q/(math.pi*nu*d_init)

  if (Re <= 2200) then
    --is laminar?
    return Q
  else
    local f = frictionFactor(Re, d_init, w)
    return (math.pi^2*g*d_init^5*dH/8/f/L)^0.50
  end
end
```

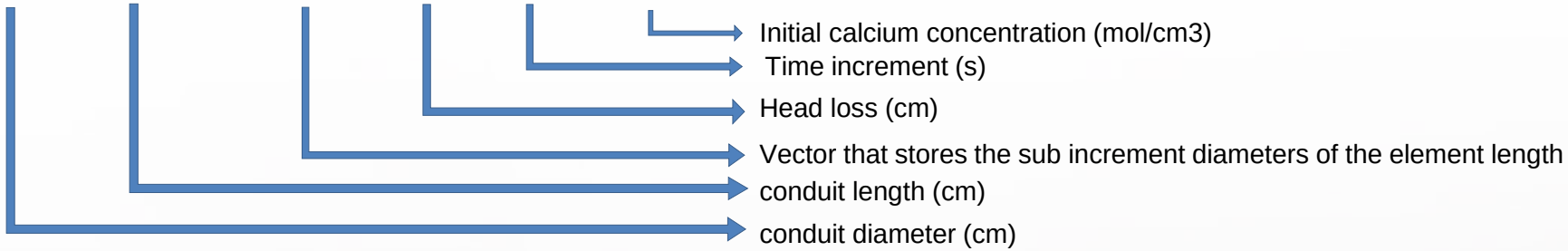
If the flux is laminar, it returns the flux rate. Otherwise, the flux rate is calculated as turbulent.

Calculates the friction factor

Returns the flux rate for a turbulent flow.

Solution file: Conduit evolution

```
function conduitEvolution(d_init, L, vector_DELm, dH, dTime, Ca_0)
```




```
  local sumDen = 0
  local dL     = L/(#vector_DELm)        -- sub increment length
  ------------------------------------------------------------------------------
  -- Parameters
  ------------------------------------------------------------------------------
  local Ca_eq = 2e-6                     -- calcium equilibrium concentration [mol/cm3]
  local Ca_s  = 0.90*Ca_eq    -- calcium threshold concentration [mol/cm3]
  local k0    = 4e-11                    -- low-order rate coefficient [mol/cm3]
  local D     = 1e-5                     -- diffusivity [cm2/s]
  local n1    = 1                        -- low-order exponent
  local n2    = 4                        -- high-order exponent
  local rho_c = 2700e-6                  -- density of calcite [kg/cm3]
  local mr    = 100.1e-3                 -- molar mass [kg/mol]
  local w     = 2e-3                     -- wall roughness [cm]
  local g     = 9.81e2                   -- gravitational acceleration [cm2/s]
  local nu    = 0.0114                   -- kinematic viscosity [cm2/s]
  local rho_w = 1000e-6                  -- density of water [kg/cm3]
  ------------------------------------------------------------------------------
  -- Calculate the flux rate in the conduit
  local Q = fluxrateQ(dH, d_init, L, w, g, nu, rho_w) -- cm3/s
  local concentrationCa = Ca_0
```

Parameters for the limestone dissolution process

Calls the flux rate function

… continues on the next slide

Solution file: Conduit evolution

```
  -- loop in each sub increment of the conduit
  for ie=1,(#vector_DELM) do
    local k1 = k0/(1+k0*vector_DELM[ie]/(6*D*Ca_eq)) -- low order coefficient
    local k2 = k1*(1-Ca_s/Ca_eq)^(n1-n2)              -- high-order coefficient

    -- Calculate the dissolution rate FCa in mol/(cm2.s)
    if (concentrationCa < Ca_s) then
      Fca = k1*(1-concentrationCa/Ca_eq)^n1
    else
      Fca = k2*(1-concentrationCa/Ca_eq)^n2
    end
    -- Calculate the calcium additionally dissolved
    local dA  = math.pi*vector_DELM[ie]*dL
    local dCa = FCa*dA/Q
    -- Calcium concentration at the downstream end of the conduit subdivision
    concentrationCa=concentrationCa+dCa
    -- is saturated?
    if(concentrationCa > Ca_eq) then
      concentrationCa=Ca_eq
    end
    -- wall retreat
    dr_dt=FCa*mr/rho_c
```

Update the diameter of sub increment length

```
    vector_DELM[ie]= vector_DELM[ie]+2*dr_dt*dTime
    sumDen=sumDen+dL/(vector_DELM[ie]^4)
  end

  return (L/sumDen)^(0.25), vector_DELM, concentrationCa
end
```

Returns the new effective diameter, the updated vector of the sub increment diameter and calcium concentration at the downstream of the conduit

Solution file: Orchestration script

```
function ProcessScript()
  -- Initialize the accessors
  local em = modelData:mesh('mesh')
  assert(em)
  -- nodal coordinate accessor
  local acNode = em:nodeCoordAccessor()
  -- pore pressure accessor
  local PpAc = assert(em:nodeValueAccessor('P'))
  -- interface element gap (diameter) accessor
  local dAc  = assert(em:gaussAttributeAccessor('An'))
  -- integration rule for interface elements
  local ir = em:elementIntegrationRule('int2dl6', 1)

  -- Initialize the matrix that stores the diameter of element lengths
  local matDiameterLength = {}
  -- Get the attribute diameter for each interface element
  local ndiv = 1000
  local d    = 0.10  -- cm
  for ie = 1, em:numCells() do
    matDiameterLength[ie] = {}
    for isub = 1,ndiv do
      matDiameterLength[ie][isub] = d
    end
  end

  -- Setting the solver
  local solver = fem.init({'interface'}, 'NumSolver', solverOptions)
  io.printMeshNodeData('mesh', {'P'}, {header_title = true})
```

Create the needed accessors to retrieve and set mesh data: nodal values and coordinates and Gauss attributes.

… continues on the next slide

Solution file: Orchestration script

```
-- Prepare the output file
local file = io.prepareMeshFile('mesh', '$SIMULATIONDIR/out/simple_KARST', 'nf', {'P'}, {'An'})
io.addResultToMeshFile(file, 0.0)   -- Save initial result

-- Transient flow analysis
local dt        = solverOptions.timeInitIncrement
local FinalTime = solverOptions.timeMax
local Time      = dt
local LastStep  = false
while (Time < FinalTime) do
```

The global simulation loop

```
  print('------------------')
  print(tr('Flow iteration - time = %1 yr'):num(Time/31536000))
  print('------------------')

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

Executes the Hydraulic step

```
  io.printMeshNodeData('mesh', {'P'}, {header_title = true})
  io.addResultToMeshFile(file, Time)

  -- loop over all interface elements for dissolution calculation
  local Ca = 0.0    -- set the initial calcium concentration
  for i = 1, em:numCells() do
    local ec = em:cell(i)
    assert(ec:type() == 'int2dl6')

    local id1 = ec:nodeIndex(1)
    local id2 = ec:nodeIndex(2)
```

… continues on the next slide

Solution file: Orchestration script

```
  -- Get the coordinates vector
  local x1 = acNode:value(id1)
  local x2 = acNode:value(id2)
  -- Get the pore pressure
  local p1 = PpAc:value(id1)
  local p2 = PpAc:value(id2)

  -- Get the current effective diameter (Accessor - Gap)
  local d0 = dAc:value(ec, 1)*100

  -- Calculate the conduit length
  local L_el = math.sqrt((x2(1)-x1(1))^2+(x2(2)-x1(2))^2)*100  -- length [convert in cm]

  -- Calculate the head loss [cm]
  local gamma_w = 9.81                  -- specific weight of water [kN/m3]
  local dh      = (p1-p2)/gamma_w*100
  local dnew                            -- new diameter

  -- Update the diameter in the interface element
  dnew, matDiameterLength[i], Ca = conduitEvolution(d0, L_el, matDiameterLength[i], dh, dt, Ca)

  -- Set the new diameter([m]) in the gauss points of the interface element
  for j = 1, ir:numPoints() do
    dAc:setValue(ec, j, dnew*0.01)
  end
end
```

Get the global nodal coordinates (X, Y) at the ends of the interface element

Get the pore pressure values at the ends of interface elements

Get the current diameter (gap) of the interface element

Calls the conduit evolution function

For each gauss point of interface element, set the new effective diameter in m

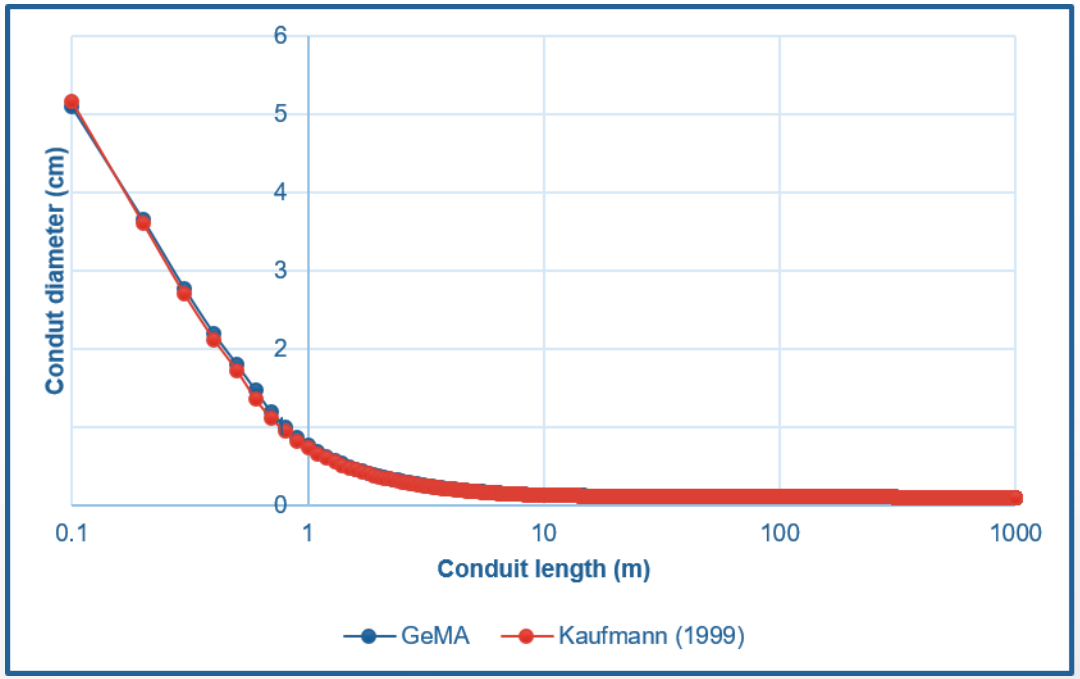
… continues on the next slide

Solution file: Orchestration script

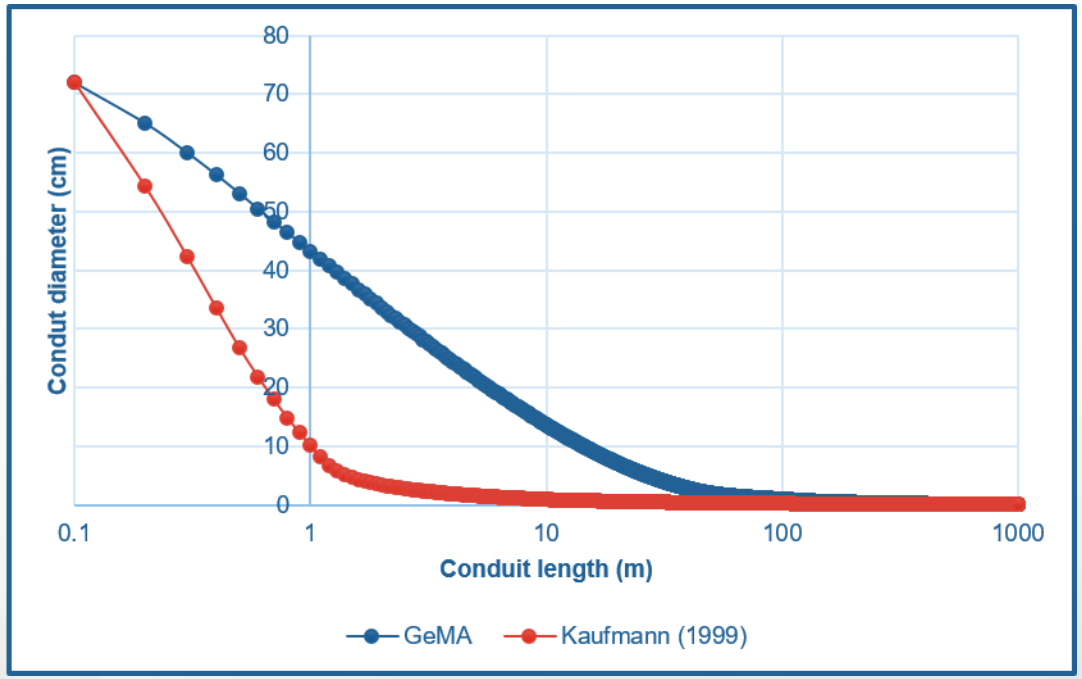
```
    -- Adjust time to guarantee that the last iteration will be on the
    -- requested final simulation time
    dt = newt
    if (Time + dt >= FinalTime and not LastStep) then
      dt   = FinalTime - Time
      Time = FinalTime
      LastStep = true
      if equal(dt, 0.0) then break end -- Test special case when Time == FinalTime
    else
      Time = Time + dt
    end
    print(string.format("Time = %6.4f years",Time/31536000))
  end

  -- write the dissolution
  writeDissolutionResults(matDiameterLength, Time)
  io.closeMeshFile(file)
end
```

Results

$\Delta t = 1000\ yr$



$\Delta t = 10000\ yr$



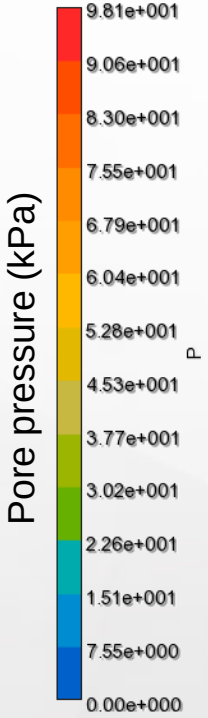
In the early phase of the dissolution process only the entrance section of the fissure is enlarged, so there is no disparity between the fissure enlargement between the model discretized in GeMA (with ten interface elements) and the Kaufmann's results (using only one conduit). However, with the conduit evolution and higher-order kinetics, the model in GeMA has no more a constant head loss along the interface elements, unlike that found in Kaufmann's model with constant head loss.

Reference: Kaufmann, G.; Braun, J.. "**Karst aquifer evolution in fractured rocks**". Water Resources Research 35, pp. 3223-3238, 1999

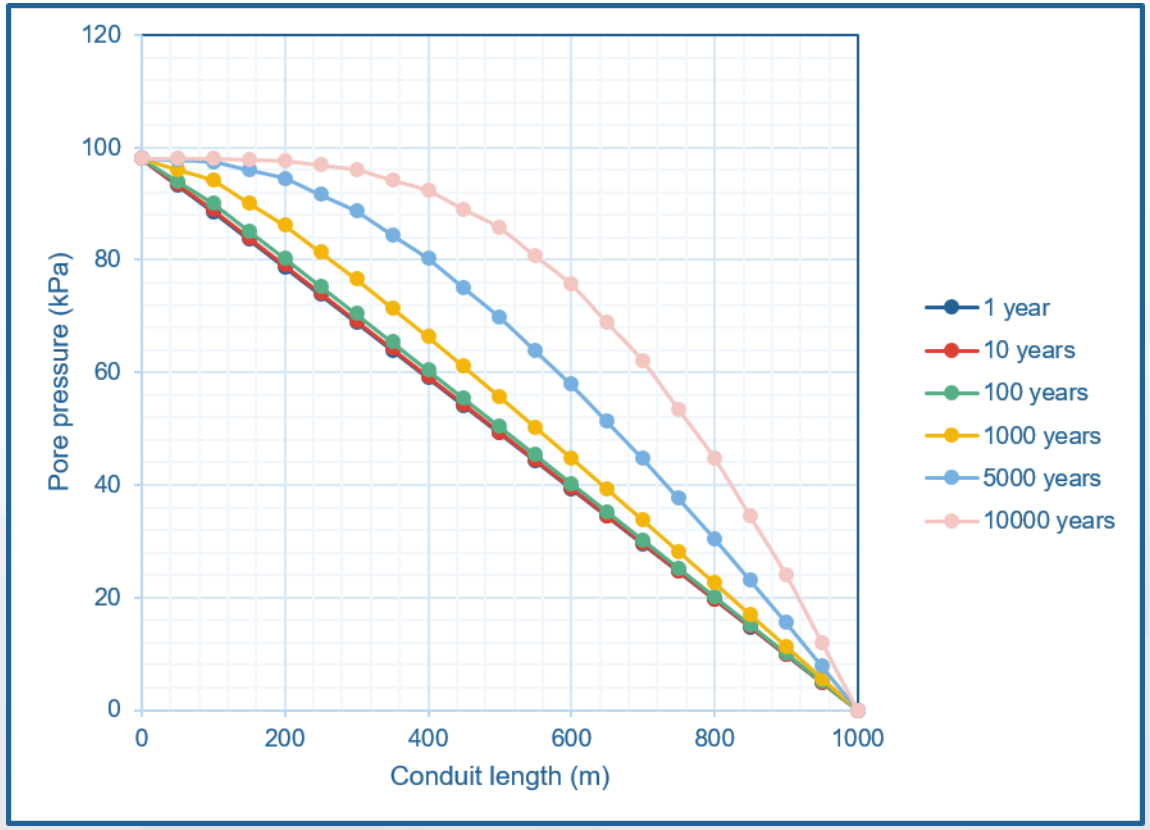
Results

Time: 1 year

Time: 10000 years



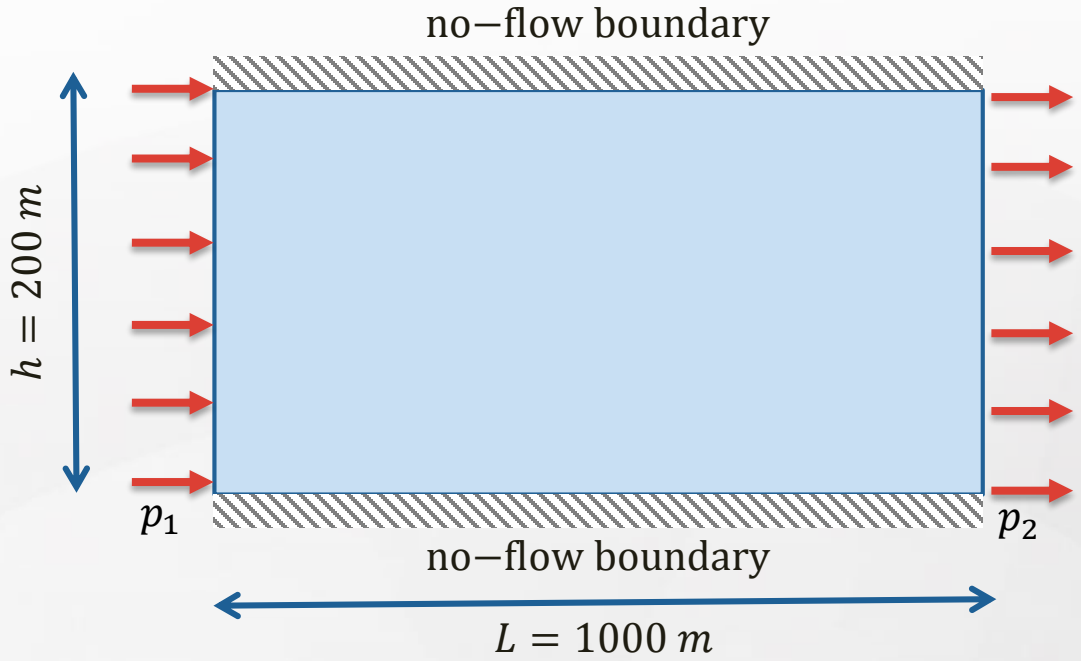
Pore pressure along of the transient flow analysis



2 – EVOLUTION OF A 2D KARST AQUIFER BASED ON DISCRETE FRACTURE APPROACH

Example II

In this simple example, the interface elements and fissure enlargement algorithm are incorporated to a porous medium in order to represent the fracture in the aquifer and to assess the early evolution of vertical karst aquifer due the dissolution process along the transient analysis. The cross-section model has length of 1000 m and height of 200m. The top and bottom borders are no-flow boundaries, and the left and right borders are prescribed pore pressure boundaries with $p_1 = 0\ kPa$ and $p_2 = 1982\ kPa$, respectively.



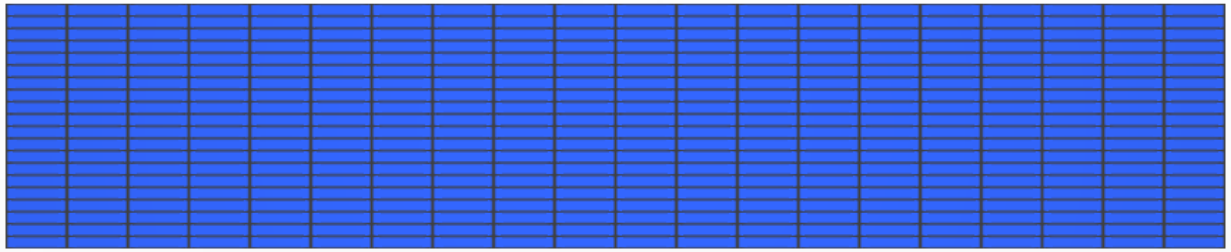
Reference:
Kaufmann, G..“**Modelling unsaturated flow in a evolving karst aquifer**”. Journal of Hydrology 276, pp. 53-70, 2003

Kaufmann, G.. “**Structure and evolution of karst aquifers: a finite-element numerical modelling approach**”. In W. Dreybrodt, F. Gabrovsek and D. Romanov (editors), Processes of Speleogenesis: A modeling approach, pp. 323–371. Karst Research Institute, ZRZ SAZU, Postojna, Slovenia, 2005.

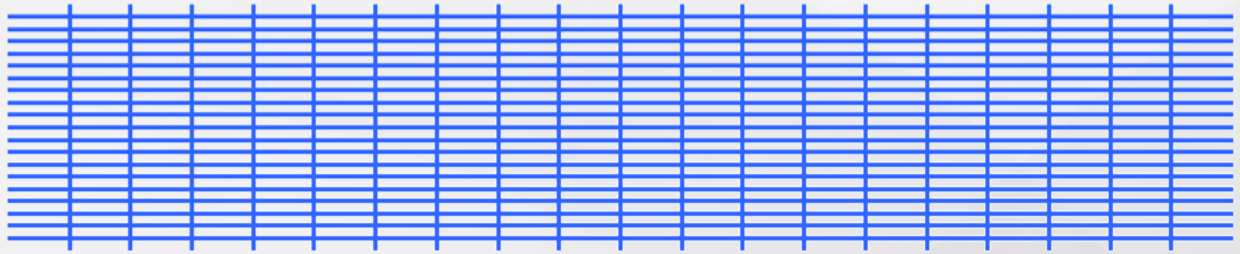
Simulation file: conceptualKarst2D.lua

Model file: Mesh

Karst aquifer mesh



The incorporated interface elements (conduits)
into the porous medium

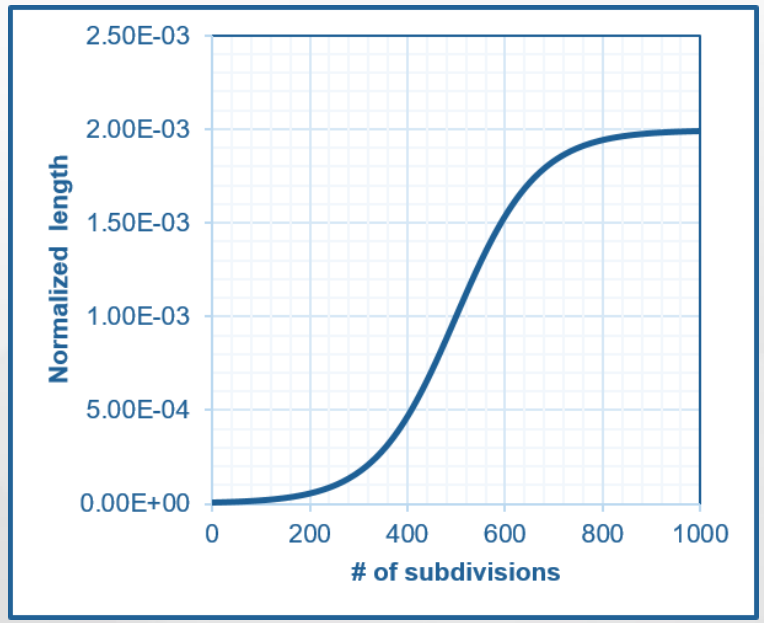


Solution file: Conduit evolution and Orchestration

The same fissure enlargement algorithm is used to evaluate the evolution of early karst aquifer. The only difference is in the interface element length subdivisions using a logarithmic distribution as proposed by Groves and Howard (1994) in their study about the development of early karst. This logarithmic division ensures that the steep gradient in calcium concentration in the entrance section of the interface element (or conduit) is properly resolved (Kaufmann, 1999).

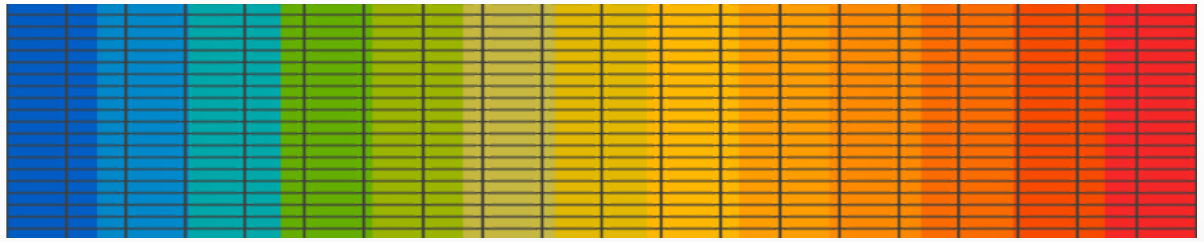
Reference:
Kaufmann, G.; Braun, J.. "**Karst aquifer evolution in fractured rocks**". Water Resources Research 35, pp. 3223-3238, 1999
Groves, C.; Howard, A.. "**Early development of karst systems: 1. Preferential flow path enlargement under laminar flow**". Water Resources Research 30, pp. 2837-2845, 1994

The orchestration strategy closely follows the same algorithm seen on the previous example. Refer to the solution file for details.

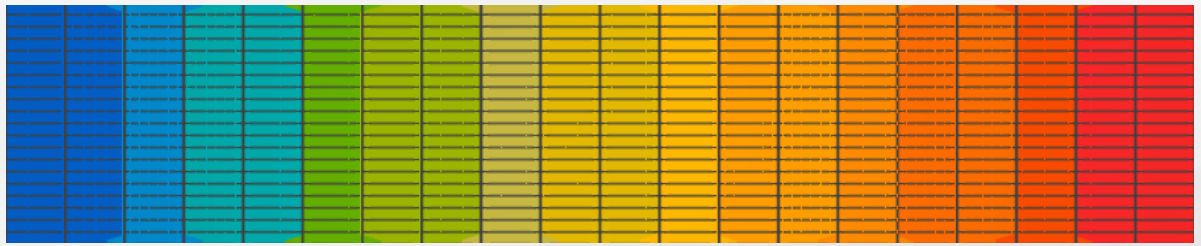


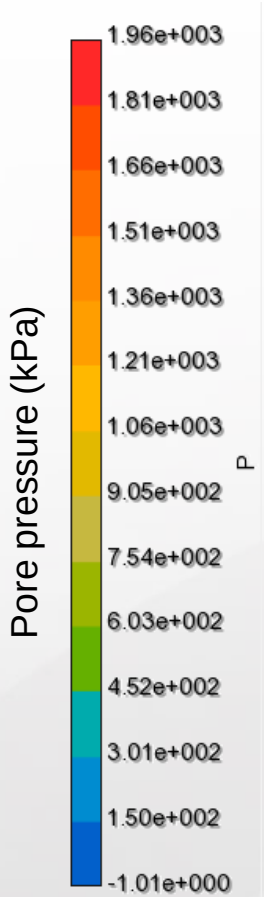
Results

Time: 1 year



Time: 10000 years

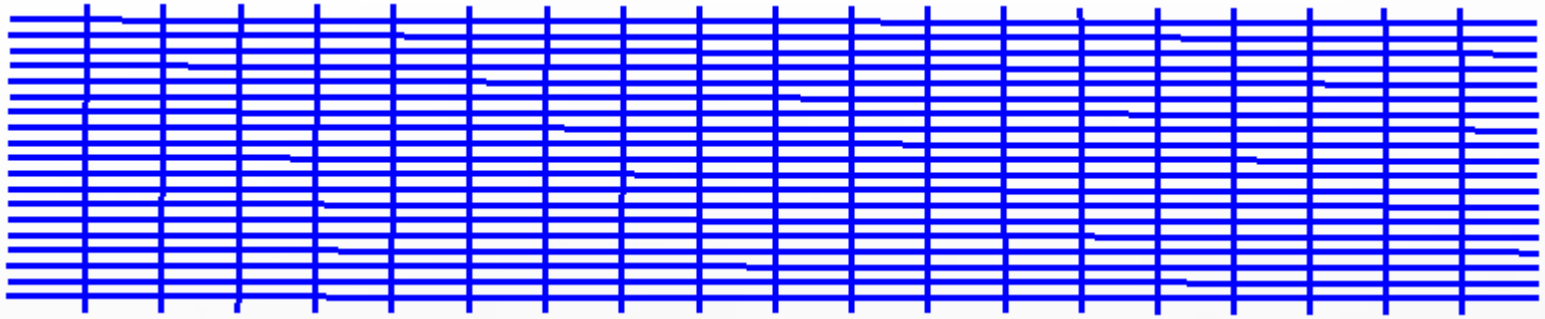




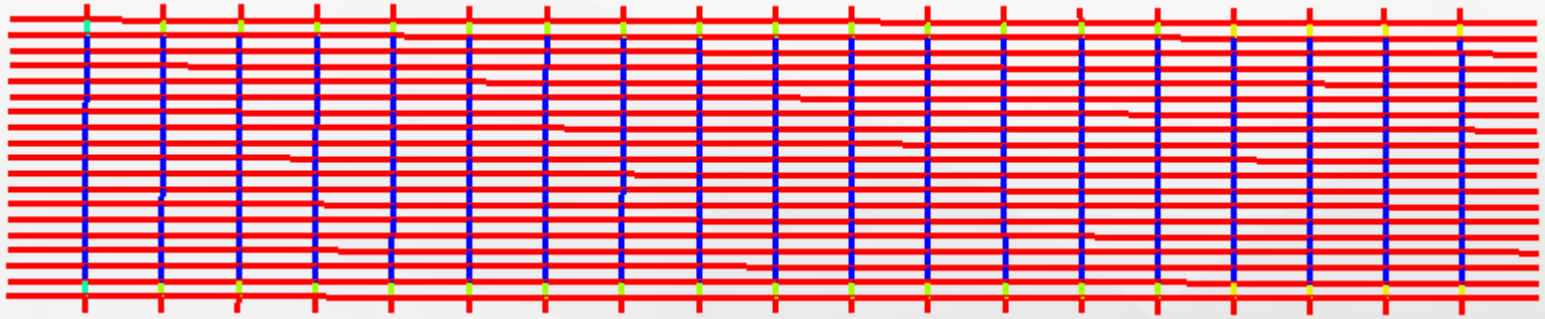
Evolution of the pore pressure along of the transient flow analysis

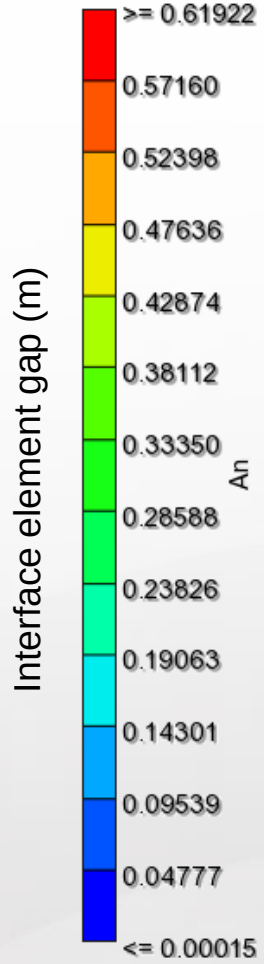
Results

Time: 1 year



Time: 10000 years





As the pore pressure gradient is the same for all horizontal interface elements (conduits) in the early years (1~3000 years), a similar evolution of the fissure enlargement is expected for these elements. In vertical interface elements with the change of the pore pressure gradient, the fracture opening begins from the borders to middle region of the karst aquifer.