GeMA – Dual Porosity 2D Example

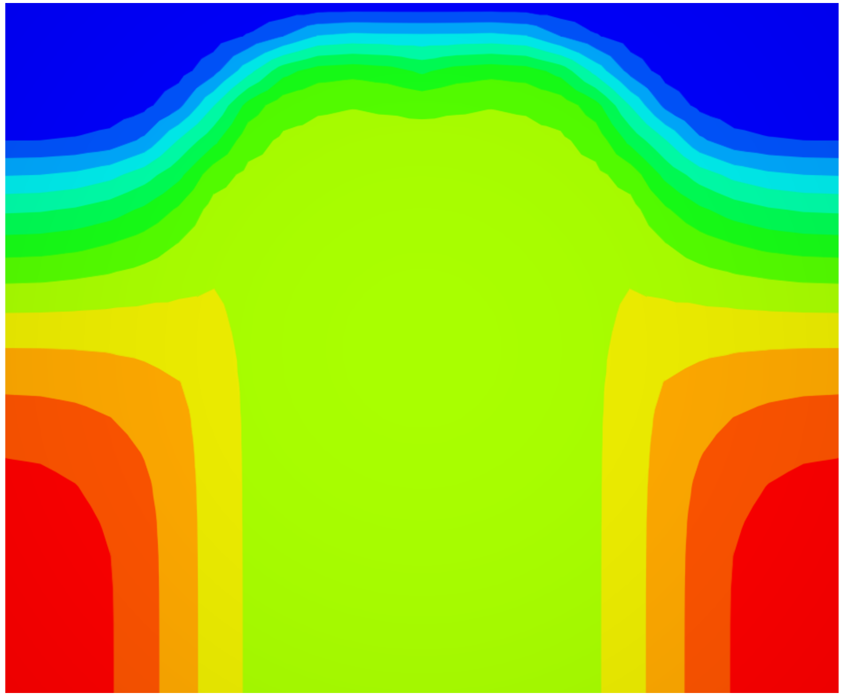
16/07/2018 – Version 1.0

Tecgraf
PUC-RIO

The example

- This example presents a coupled hydro-mechanical simulation of a naturally fractured reservoir using the dual porosity, dual permeability model.

- Only singular points are highlighted on this presentation. See the model files for additional details.

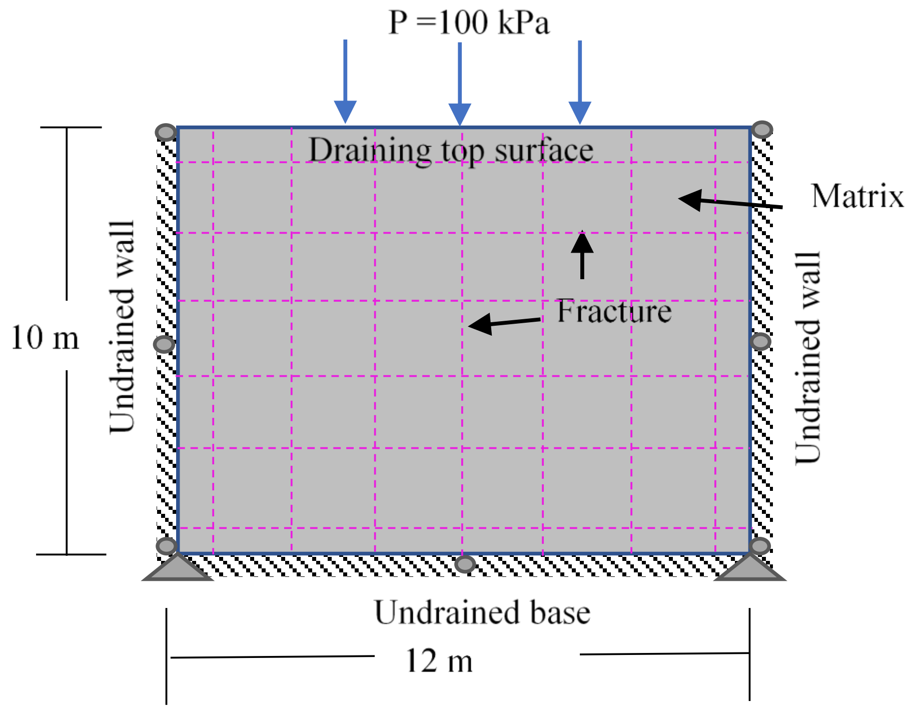


For validation, this model was compared with a discrete fracture model, presented in the Multiple Fracture Hydraulic Mechanical Coupling examples.

The model is also presented in: Rueda, C.J et al “Discrete and smeared fracture model to simulate fluid flow in naturally fractured reservoirs”, ARMA 18–0651

The Problem

- In this problem, a saturated poroelastic rectangle is compressed by applying uniform pressure at a 4 m region on the top of the model. The base is constrained in the y direction and the left and right sides are constrained in the x direction. Natural fractures are represented by the dual porosity / dual permeability model.



Parameters	Values
Young's modulus, E(kPa)	1.00E+05
Possion's ratio, υ	0.2
Normal stiffness of fracture, k_n (kPa/m)	2.00E+04
Tangencial stiffness of fracture, k_s(kPa/m)	1.00E+04
Hydraulic permeability of the matrix, k_m(m/s)	1.00E-11
Fracture aperture, $bx=by$(m)	4.9e-4
Relative compressibility, $\beta_{fr} = \beta_m$(1/kPa)	0
Fluid viscocity, μ (cp)	1
Fracture spacing, s(m)	1
Specific weight of water, γ_w(kN/m^3)	10
Analysis time, t(s)	1.00E+09

Dual porosity Dual Permeability

- The dual porosity, dual permeability concept consists in the superposition of two porous systems with different characteristics. In fractured reservoirs, the primary porosity of the matrix and the secondary porosity of the fracture network are physically overlapped in space and time so that the transfer of fluid occurs according to the potential of fluids between both, fracture and matrix media.

$$\nabla(D_{mfr}:\varepsilon - D_{mfr}:C_m:\alpha_m p_m - D_{mfr}:C_{fr}:\alpha_{fr} p_{fr}) + f = 0$$

$$\frac{k_m}{\mu}\nabla^2 p_m + D_{mfr}:C_m:\alpha_m:\frac{\partial \varepsilon}{\partial t} + \beta_m \frac{\partial p_m}{\partial t} + \omega\left(p_m - p_{fr}\right) + q_m = 0$$

$$\frac{k_{fr}}{\mu}\nabla^2 p_{fr} + D_{mfr}:C_{fr}:\alpha_{fr}:\frac{\partial \varepsilon}{\partial t} + \beta_{fr} \frac{\partial p_{fr}}{\partial t} - \omega\left(p_m - p_{fr}\right) + q_{fr} = 0$$

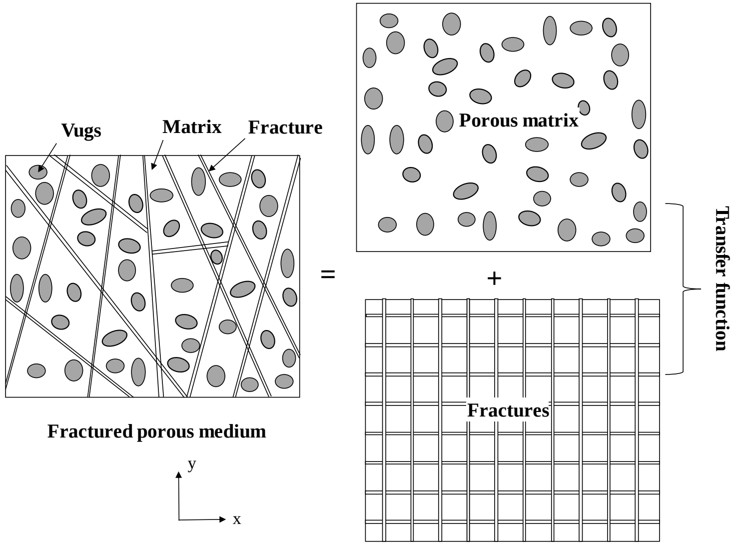


Fig. 1. Idealization of naturally fractured system with dual porosity model.

Model file: State variable

For a coupled Hydro-Mechanical, dual porosity simulation, the needed state variables are the displacement, the pore pressure and the fracture pore pressure:

```
-- State variables
StateVar{id = 'u', dim = 2, 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'}

StateVar{id = 'Pf', description = 'Fracture pore pressure', unit = 'kPa',
         format = '8.4f', groupName = 'hydraulic'}
```

The group name is used by the fem solver to group types of state variables and to define tolerances for each one

Model file: Material properties

```
PropertySet
{
  id          = 'MatProp',
  typeName    = 'GemaPropertySet',
  description = 'Material properties',
  properties  = {
    {id = 'Er',  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 = 'Bp',  description = 'Pore compressibility',               unit = 'kPa^-1'},
    {id = 'Knf', description = 'Normal elastic stiffness',           unit = 'kPa/m'},
    {id = 'Ksf', description = 'Shear elastic stiffness',            unit = 'kPa/m'},
    {id = 'Bpf', description = 'Pore compressibility of fracture',  unit = 'kPa^-1'},
    {id = 'Phf', description = 'Fracture porosity'},
    {id = 'uff', description = 'Fracture dynamic viscosity',         unit = 'kPa*s'},
    {id = 'Sx',  description = 'Fracture spacing along x',           unit = 'm'},
    {id = 'bx',  description = 'Fracture aperture along x',          unit = 'm'},
    {id = 'Sy',  description = 'Fracture spacing along y',           unit = 'm'},
    {id = 'by',  description = 'Fracture aperture along y',          unit = 'm'},
    {id = 'shpFacType', description = 'Shape factor type',           constMap = constants.HydroFemPhysics.shapeFactorModel},
    {id = 'material',   description = 'Mechanical material type',    constMap = constants.CoupledHMFemPhysics.materialModels},
    {id = 'h',          description = 'Element thickness',           unit = 'm'},
  },
```

The standard E property was renamed in order to allow user functions calculating fracture apertures to use as parameters both the Elasticity Modulus and the calculated strain vector (also named E).

HM Coupled Dual porosity properties

A map with the supported transfer function names

A map with the supported material names

… continues on the next slide

Model file: Material properties

```
values = {
   {Er = 1.00e+05, nu  = 0.2,        Knf = 2.00e+4,  Ksf = 1.00e+4,    K   = 1.00e-11, Kww = 2.20e+6,
    gw = 1.00e+01, Pht = 0.1,        Bp  = 1.44e-6,  Bpf = 0,          Phf =  0.0,     uff = 1.00e-06,
    Sx = 1.00e+00, bx  = 0.00004933, Sy  = 1.00e+00, by  = 0.00004933, shpFacType = 'kazemi',
    material = 'coupledDualPorosity', h = 1.000},
  }
}
```

A material type from `constants.CoupledHMFemPhysics.materialModels`
See the plugin docs for other available materials

A transfer function from `constants.HydroFemPhysics.shapeFactorModel`
See the plugin docs for other functions

Parameters	Values
Young's modulus, E(kPa)	1.00E+05
Possion's ratio, υ	0.2
Normal stiffness of fracture, k_n(kPa/m)	2.00E+04
Tangencial stiffness of fracture, k_s(kPa/m)	1.00E+04
Hydraulic permeability of the matrix, k_m(m/s)	1.00E-11
Fracture aperture, $bx=by$(m)	4.9e-4
Relative compressibility, $\beta_{fr} = \beta_m$(1/kPa)	0
Fluid viscocity, μ (cp)	1
Fracture spacing, s(m)	1
Specific weight of water, γ_w(kN/m^3)	10
Analysis time, t(s)	1.00E+09

Model file: Mesh

Cell user functions calculating the aperture from model parameters / results

```
Mesh
{
  ...
  -- Attributes for storing calculated fracture apertures
  gaussAttributes = {
    {id = 'Fopen1', description = 'aperture 1', functions = true, defVal = 'aperture1'},
    {id = 'Fopen2', description = 'aperture 2', functions = true, defVal = 'aperture2'}
  },
  ...
}

-- A user function to calculate the fracture aperture for vertical fractures
CellFunction { id = 'aperture1',
               parameters = { {src = 'E', dim=1},  -- x component from strain vector
                              {src = 'bx'},
                              {src = 'Er'},
                              {src = 'Sx'},
                              {src = 'Knf'},
                            },
               method = function(emf, b0, Em, s, kn)
                 local dbx = Em/(s*kn+Em)*emf
                 local bx  = b0 + dbx
                 if bx < 0 then
                   return 0
                 else
                   return bx/b0
                 end
               end
             }
```

Model file: Boundary conditions

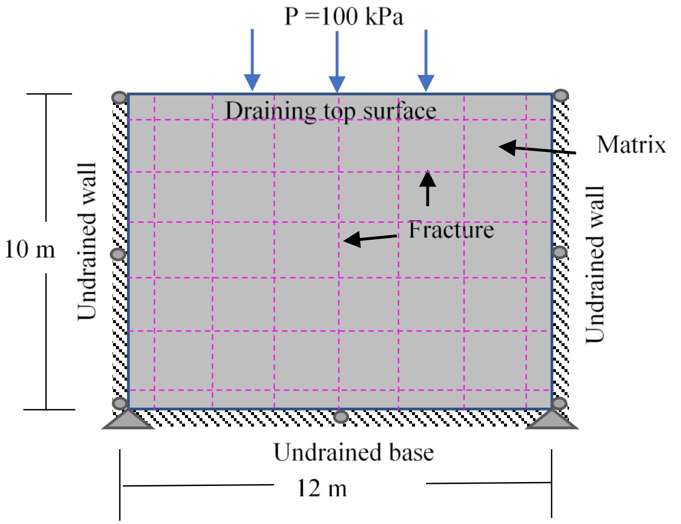
```
-- Pressure applied on the center nodes at the model top
BoundaryCondition {
  id   = 'bdp',
  type = 'node concentrated forces',
  mesh = 'mesh',
  properties  = {
    {id = 'f',  description = 'External force applied on the node', unit = 'kN', dim = 2},
  },
  nodeValues = {
    {58 ,     {     0.000,    -25.000}},
    {72 ,     {     0.000,    -50.000}},
    ...
  }
}
```

Boundary condition type for prescribed forces values

Force dimensions

Node set

Nodal concentrated forces {Fx, Fy}




```
-- Constrained displacements. Sides are constrained in the x direction while the base is constrained in y.
-- Bottom corner nodes are constrained in both directions.
BoundaryCondition {
  id   = 'bc',
  type = 'node displacement',
  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 = {
    {       1,  0.0000e+00,         nil},
    ...
  }
}
```

Boundary condition type for prescribed displacements

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

Prescribed displacement in x

Node free in y

… see model files for fixed pore pressure conditions

Solution file: Physics

The dual porosity model for a coupled HM problem is available through the DualPorosity object from the CoupledHMFemPhysics plugin.

```
PhysicalMethod {
  id       = 'DualPorosity',
  typeName = 'CoupledHMFemPhysics.DualPorosity',
  type     = 'fem',

  mesh                   = 'mesh',
  materials              = 'coupledDualPorosity',
  boundaryConditions = {'bpd', 'bpm', 'bpf', 'bc'},
  ruleSet                = 1,

  properties             = { E = 'Er' },

  permeabilityUpdate = true,
  penaltyStiffness   = false,
}
```

Since the elasticity modulus was renamed to Er in the property set, the physics must be warned about this change by a translation map that tells it that the expected property E will be given by property Er.

Solution file: Orchestration

```
local solverOptions = {
   type                 = 'transient automatic time step',
   timeMax              = 1.000E+09,
   timeInitIncrement    = 0.01,
   timeMinIncrement     = 0.01,
   timeMaxIncrement     = 1e8,
   iterationsMax        = 15,
   tolerance            = { mechanic = 1.0e-05, hydraulic = 1.0e-5 },
}
```

- type → Variable time step, transient, non-linear solver
- timeMax → Maximum time for the simulation
- timeInitIncrement → Size of the increment
- timeMinIncrement → Minimum increment
- timeMaxIncrement → Maximum increment
- iterationsMax → Maximum number of interactions
- tolerance → Associates a tolerance with each physics type

```
function ProcessScript()
   -- Create the solver model
   local solver = fem.init({'DualPorosity'}, 'solver', solverOptions)

   -- Prepare the file where results will be saved
   local file = io.prepareMeshFile('mesh', '$SIMULATIONDIR/out/DPP-kvar', 'nf', {'u','P','Pf'},
                                   {'S', 'E','Fopen1','Fopen2'}, {split = true, saveDisplacements = true})

   -- Saves initial state to the file
   io.addResultToMeshFile(file, 0.0)
```

{'S', 'E','Fopen1','Fopen2'}: Save stresses, strains and calculated apertures

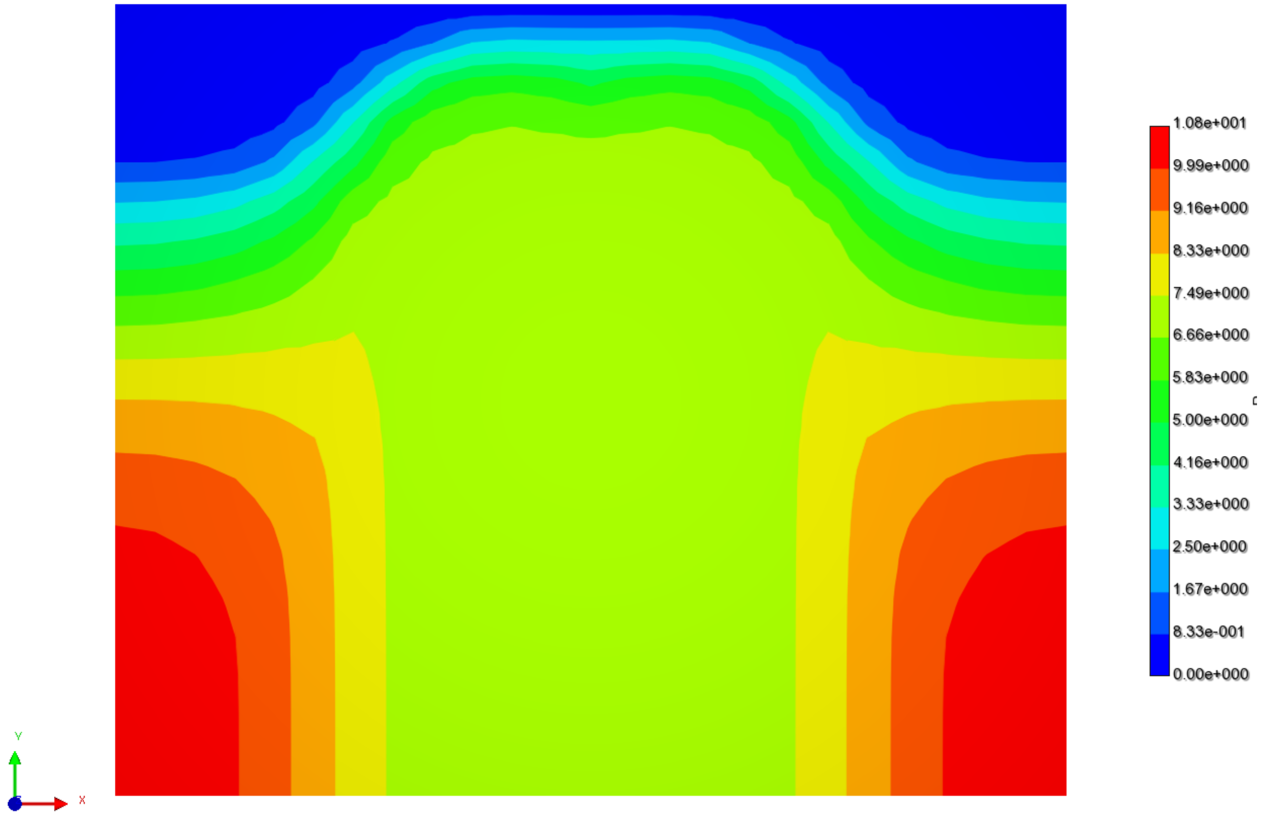
… continues on the next slide

Solution file: Orchestration

```
-- Flow loop
local dt        = solverOptions.timeInitIncrement
local FinalTime = solverOptions.timeMax
local Time      = dt
local OldTime   = 0.0
local LastStep  = false

while (Time <= FinalTime) do
  print('------------------')
  print(tr('Flow iteration - time = %1 s'):num(Time))
  print('------------------')

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

  -- If we are later on the simulation, lets ignore the
  -- suggested time step and use an increment that is 5%
  -- greater than the previous one
  if(Time >=1e6) then
    dt = (Time - OldTime) * 1.05
  end
    -- Save results
    io.addResultToMeshFile(file, Time)
    print(tr('Flow iteration - Dtime = %1 s'):num(dt))

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

    print('Time = ' .. Time)
  end

  -- Closes the result file
  io.closeMeshFile(file)
end
```

Results at t = 1.0e9 s

Pressure



Final Y Displacement