Flow simulations in 3D Discrete Fracture Networks

J-R. de Dreuzy [1], J. Erhel [2], G. Pichot[3], B. Poirriez [2]

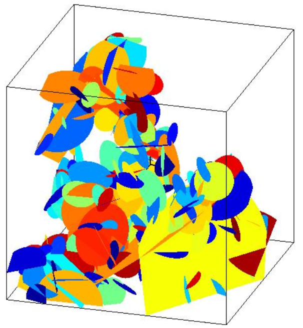
[1]Géosciences Rennes, [2]Inria Rennes, [3]LOMC, Université du Havre

June 21, 2010

Stochastic Generation of DFN

Following a Discrete Fracture Network approach, fractures are planes with the following statistical properties :

Parameter	Random distribution
length	power law
shape	disks / ellipses
position	uniform
orientation	uniform
Conductivity	homogeneous / correlated log-normal



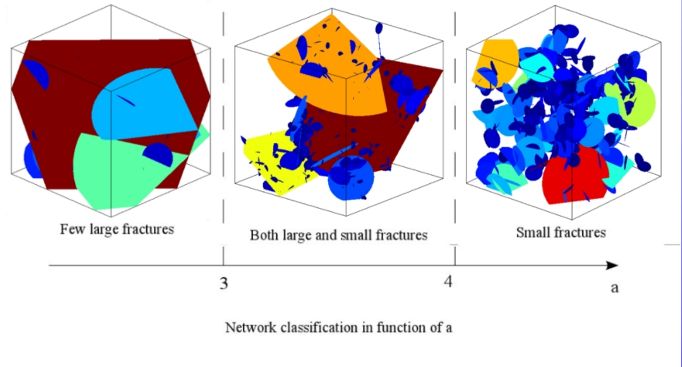
Example of DFN with 217 fractures

Stochastic Generation of DFN

The broad natural fracture length distribution is modeled by a power law distribution (Bour et al, 2002) :

$$p(l)dl = \frac{1}{a-1}\frac{l^{-a}}{l_{min}^{-a+1}}dl,$$

where $p(l)dl$ is the probability of observing a fracture with a length in the interval $[l, l+dl]$, l_{min} is the smallest fracture length, and a is a characteristic exponent.



Flow model

Current assumptions :

- The rock matrix is impervious : flow is only simulated in the fractures,
- Study of steady state flow,
- There is no longitudinal flux in the intersections of fractures.

Flow equations within each fracture Ω_f :

$$\begin{aligned} \nabla \cdot \mathbf{u}(\mathbf{x}) &= \mathbf{f}(\mathbf{x}), && \text{for } \mathbf{x} \in \Omega_f, \\ \mathbf{u}(\mathbf{x}) &= -\mathcal{T}(\mathbf{x}) \nabla p(\mathbf{x}), && \text{for } \mathbf{x} \in \Omega_f, \\ p(\mathbf{x}) &= p^D(\mathbf{x}), && \text{on } \Gamma_D \cap \Gamma_f, \\ \mathbf{u}(\mathbf{x}).\boldsymbol{\nu} &= q^N(\mathbf{x}), && \text{on } \Gamma_N \cap \Gamma_f, \\ \mathbf{u}(\mathbf{x}).\boldsymbol{\mu} &= \mathbf{0}, && \text{on } \Gamma_f \backslash \{(\Gamma_f \cap \Gamma_D) \cup (\Gamma_f \cap \Gamma_N)\}, \end{aligned}$$

- $\boldsymbol{\nu}$ (resp. $\boldsymbol{\mu}$) outward normal unit vectors
- $\mathcal{T}(\mathbf{x})$ a given transmissivity field (unit $[\mathrm{m}^2.\mathrm{s}^{-1}]$), $\mathbf{f}(\mathbf{x}) \in L^2(\Omega_f)$ sources/sinks.

Continuity conditions in each intersection :

$$\begin{aligned} p_{k,f} &= p_k, && \text{on } \Sigma_k, \forall f \in F_k, \\ \sum_{f \in F_k} \mathbf{u}_{k,f}.\mathbf{n}_{k,f} &= 0, && \text{on } \Sigma_k, \end{aligned}$$

with F_k the set of fractures with Σ_k (the k-th intersections) on the boundary,

Mixed-Hybrid Finite Element Method

Mixed-Hybrid Finite Element Method (MHFEM) for DFNs

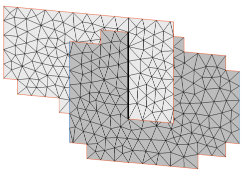
Réf. *J. Erhel et al., SIAM SISC, Vol. 31, No. 4, pp. 2688-2705, 2009*

- Makes it easy to deal with complex geometry ;
- Conforming mesh at the fracture intersections ;
- A linear system with only trace of pressure unknowns :

$$\mathbf{A}\boldsymbol{\Lambda} = \mathbf{b},$$

with **A** a symmetric positive definite matrix, the flux at the edges and the mean pressure are then easily derived locally on each triangle.

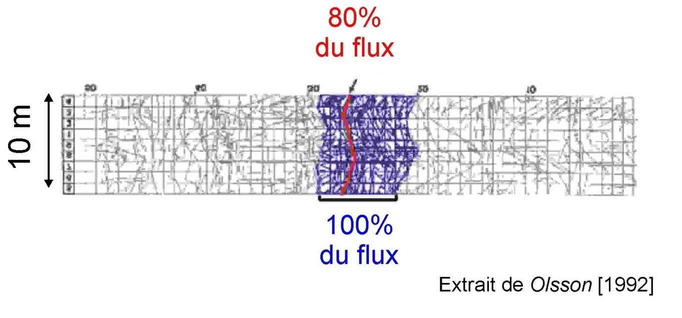
Specific mesh generation :



1. A first discretization of boundaries and intersections is done in 3D by using elementary cubes
2. The discretization of the boundaries and intersections within the fracture f is obtained by a projection of the previous voxel discretization within the fracture plane.
3. Some local corrections to ensure some topological properties.
4. Once the borders and intersections are discretized, a 2D mesh of each fracture, using triangular elements.

Mixed-Hybrid Mortar Finite Element Method

In DFN, flow is highly channelled = an opportunity to reduce the number of unknowns and the computational cost, by using a non conforming mesh at intersections.



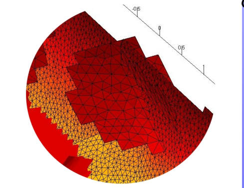
Mixed-Hybrid Mortar Finite Element Method

Mixed-Hybrid Mortar Finite Element Method (MHMFEM) for DFNs

Réf. *G. Pichot et al., Applicable Analysis, In print, 2010*

A method to mesh the fractures independently and to refine the chosen fractures using a posteriori estimators.

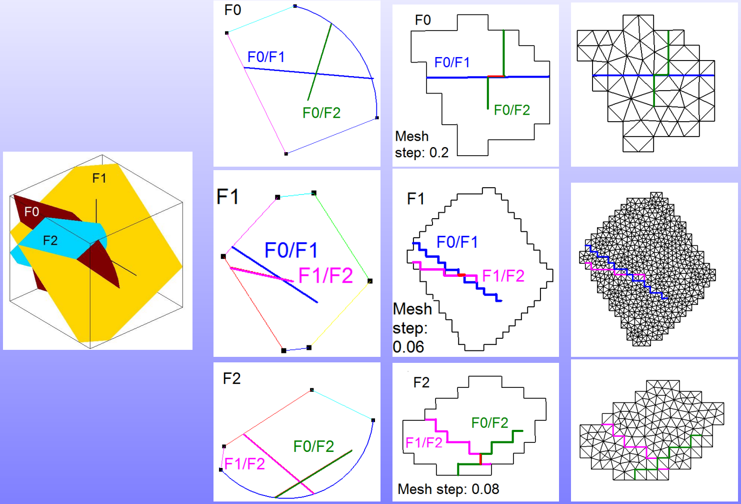
- Same advantages as MHFEM
- A simple mesh generation
- A reduced number of unknowns while keeping a solution of good quality
- a complex numerical method with Mortar conditions



A new specific mesh generation : For each fracture f, choose a mesh step and perform :

1. A first discretization of boundaries and intersections in 2D by using elementary squares, it leads to a stair-case like discretizations ;
2. Some local corrections to ensure some topological properties.
3. Once the borders and intersections are discretized, a 2D mesh of each fracture, using triangular elements.

Meshing procedure : Example



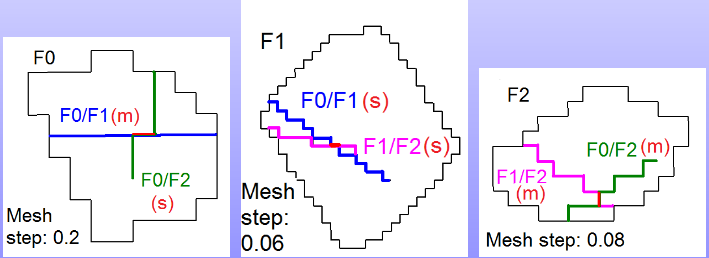
The discretization of intersections is non matching
$\Rightarrow$ Mortar conditions are required to ensure the continuity of heads and fluxes.

Mortar principle

J-R. de Dreuzy , J. Erhel , G. Pichot, B. Poirriez

Mortar method principle :

It consists in choosing arbitrarily for each intersection a master fracture (m) and a slave fracture (s).



- Particular case : each edge is either master or slave
- General case : some edges have several master or slave properties
  Réf. *G. Pichot et al., in preparation, 2010*

Definitions

Unknowns	Local (fracture f)	Global (network)
Cell mean hydraulic head	$\mathbf{P_f}$	$\mathbf{P} = (\mathbf{P_f})_\mathbf{f}$
Traces of hydraulic head	$\mathbf{\Lambda_f} = \begin{pmatrix} \mathbf{\Lambda_{f,in}} \\ \mathbf{\Lambda_{f,\Sigma}} \end{pmatrix}$ $\mathbf{\Lambda_{f,in}} = (\lambda_E)_{\mathrm{E\ inner\ edge}}$ $\mathbf{\Lambda_{f,\Sigma}} = (\lambda_E)_{\mathrm{E\ intersection\ edge}},$ $\mathbf{\Lambda_{f,m}} = (\lambda_E)_{\mathrm{E\ master\ edge}}$ $\mathbf{\Lambda_{f,s}} = (\lambda_E)_{\mathrm{E\ slave\ edge}}$	$\mathbf{\Lambda_{in}} = (\mathbf{\Lambda_{f,in}})_\mathbf{f}$ $\mathbf{\Lambda_\Sigma} = (\Lambda_{f,\Sigma})_f$ $\mathbf{\Lambda_m} = (\mathbf{\Lambda_{f,m}})_\mathbf{f}$ $\mathbf{\Lambda_s} = (\mathbf{\Lambda_{f,s}})_\mathbf{f}$
Jump of flux through the edges	$\mathbf{Q_{f,in}} = (\mathbf{Q_{E,f}})_{\mathrm{E\ inner\ edge}}$ $\mathbf{Q_{f,\Sigma}} = (\mathbf{Q_{E,f}})_{\mathrm{E\ intersection\ edge}}$ $\mathbf{Q_{f,m}} = (\mathbf{Q_{E,f}})_{\mathrm{E\ master\ edge}}$ $\mathbf{Q_{f,s}} = (\mathbf{Q_{E,f}})_{\mathrm{E\ slave\ edge}}$	$\mathbf{Q_\Sigma} = (\mathbf{Q_{f,\Sigma}})_f$ $\mathbf{Q_m} = (\mathbf{Q_{f,m}})_f$ $\mathbf{Q_s} = (\mathbf{Q_{f,s}})_f$

Notations : $N_{f,m}$ (resp. $N_{f,s}$) number of master (resp. slave) edges within the fracture f, and $N_{f,\Sigma}$, number of intersection edges. Global numbers are : $N_m = \sum_{f=1}^{N_f} N_{f,m}$, $N_s = \sum_{f=1}^{N_f} N_{f,s}$ and $N_\Sigma = \sum_{f=1}^{N_f} N_{f,\Sigma}$.

Mortar global conditions

Trace of hydraulic head	Jump of flux
$\mathbf{\Lambda_s} = \mathbf{C}\mathbf{\Lambda_m}$	$\mathbf{Q_m} + \mathbf{C}^T\mathbf{Q_s} = 0$
$\mathbf{\Lambda_\Sigma} = \mathbf{A_m}\mathbf{\Lambda_m} + \mathbf{A_s}\mathbf{\Lambda_s}$	$\mathbf{A_m}^T\mathbf{Q_\Sigma} = \mathbf{Q_m}$ $\mathbf{A_s}^T\mathbf{Q_\Sigma} = \mathbf{Q_s}$

with $\mathbf{C}$ a block matrix of dimension $N_s \times N_m$, with blocks $(\mathbf{C_k})$ of dimension $N_{k,s} \times N_{k,m}$ for which each block represents the L^2-projection from the master side to the slave side with coefficients C_{ln}, $l \in \{1, ..., N_{k,s}\}$, $n \in \{1, ..., N_{k,m}\}$:

$$C_{ln} = \left(\frac{|E_n^m \cap E_l^s|}{|E_l^s|} \right),$$

where the notation $|E|$ stands for the length of the edge E.

$\mathbf{A_s}$ and $\mathbf{A_m}$ are ponderation matrices that gives $\mathbf{\Lambda_\Sigma}$ as the mean of $\mathbf{\Lambda_m}$ and $\mathbf{\Lambda_s}$.

Linear system

$$\begin{cases} \mathbf{D}\,\mathbf{P} - \begin{pmatrix} \mathbf{R_{in}} & \mathbf{R_\Sigma}(\mathbf{A_m}+\mathbf{A_s}\mathbf{C}) \end{pmatrix} \begin{pmatrix} \mathbf{\Lambda_{in}} \\ \mathbf{\Lambda_m} \end{pmatrix} = \mathbf{f}, \\ \begin{pmatrix} \mathbf{M_{in}} & \mathbf{M_\Sigma}(\mathbf{A_m}+\mathbf{A_s}\mathbf{C}) \\ (\mathbf{A_m}^T+\mathbf{C}^T\mathbf{A_s}^T)\mathbf{M_\Sigma^T} & (\mathbf{A_m}^T+\mathbf{C}^T\mathbf{A_s}^T)\mathbf{B_\Sigma}(\mathbf{A_s}\mathbf{C}+\mathbf{A_m}) \end{pmatrix} \begin{pmatrix} \mathbf{\Lambda_{in}} \\ \mathbf{\Lambda_m} \end{pmatrix} \\ -\begin{pmatrix} \mathbf{R_{in}^T} \\ (\mathbf{A_m}^T+\mathbf{C}^T\mathbf{A_s}^T)\mathbf{R_\Sigma^T} \end{pmatrix} \mathbf{P} - \mathbf{v} = 0. \end{cases}$$

obtained by inverting locally Poiseuille's law on each triangle and by expressing the fluxes in term of traces of hydraulic head and mean heads in the following equations :

- First set of equations : mass conservation
- Second set of equations : continuity of flux through inner edges
- Third set of equations : continuity of flux through intersection edges

Linear system

We get a linear system of the form

$$\begin{pmatrix} \mathbf{D} & -\mathbf{R} \\ -\mathbf{R}^T & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{P} \\ \mathbf{\Lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{v} \end{pmatrix}$$

Then the system reduces to :

$$\mathbf{A}\mathbf{\Lambda} = \mathbf{b},$$

with $\mathbf{A} = \mathbf{M} - \mathbf{R}^T \mathbf{D}^{-1} \mathbf{R}$, $\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_{\mathbf{in}} \\ \mathbf{\Lambda}_{\mathbf{m}} \end{pmatrix}$ and $\mathbf{b} = \mathbf{v} + \mathbf{R}^T \mathbf{D}^{-1} \mathbf{f}$.

Assuming the transmissivity is locally symmetric positive definite, the matrix

$$\mathcal{J} = \begin{pmatrix} \mathbf{D} & -\mathbf{R} \\ -\mathbf{R}^T & \mathbf{M} \end{pmatrix}$$

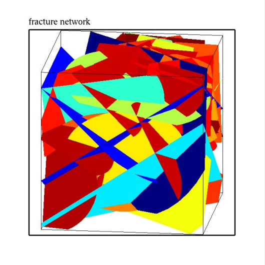
is symmetric and, with the presence of Dirichlet boundary conditions within at least one fracture, it is positive definite.

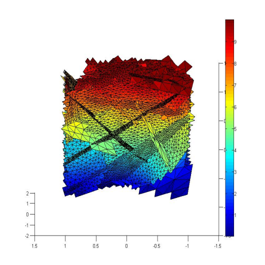
Then $\mathbf{A}$ is also symmetric positive definite.

Consistency of the results

Criteria checked for all simulations :

- Null sum of the fluxes over all the system
- Null sum of the fluxes over all intersections between fractures
- Boundary conditions satisfied
- Continuity of the flux on inner edges (that is egdes that are not intersection).





50 fractures, 315 intersections, 41967 edges, Mean head computation

Convergence criterium

Numerical convergence is estimated via a discrete relative L^2 error :

1. A computation is performed on a fine mesh $\mathcal{T}_\eta$ that gives a reference mean pressure p_η
2. Simulations are performed on coarsened grids $\mathcal{T}_h$ of mesh step $h > \eta$.

The mean head obtained on coarse meshes, p_h, are then compared with p_η [Martin et al. 2005] :

$$||p_h - p_\eta||^2_{L^2(\Omega)} = \frac{\sum_{T_\eta \in \mathcal{T}_\eta} (\Pi_\eta p_h - p_\eta)^2 |T_\eta|}{\sum_{T_\eta \in \mathcal{T}_\eta} (p_\eta)^2 |T_\eta|},$$

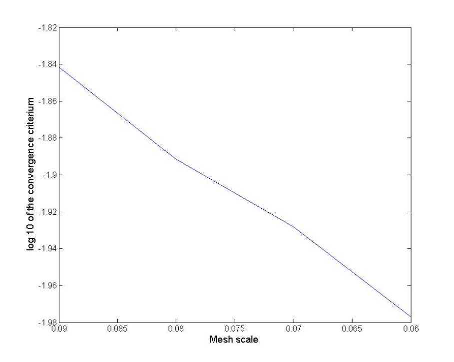
- $|T_\eta|$ the area of the triangle $T_\eta \in \mathcal{T}_\eta$
- $\Pi_\eta p_h$ the projection of p_h onto the fine mesh $\mathcal{T}_\eta$.

Convergence analysis

- On 25 Monte-Carlo simulations :

Parameter	Random distribution
length	power law
shape	disks
position	uniform
orientation	uniform

Parameter	Value
a	3.5
L/l_{min}	2
N_{MC}	25
Mesh step	from 0.05 to 0.09
Density	2



Log10 of the Convergence criterium vs mesh scale

Linear system

Linear system to solve :

In the previous section, we have seen that using MHFEM or MHFEM with Mortar leads to a linear system in term of trace of hydraulic head unknowns :

$$\mathbf{A}\mathbf{\Lambda} = \mathbf{b}.$$

with $\mathbf{A}$ a symmetric positive definite matrix.

Possible solvers :

- Direct solver (using Cholesky factorization) : ok but is memory and CPU expensive for large DFNs
- Iterative solver : multigrid method
- Iterative solver : Preconditioned Conjugate Gradient (PCG) method
- Semi-iterative solver : Domain Decomposition method

Remark :

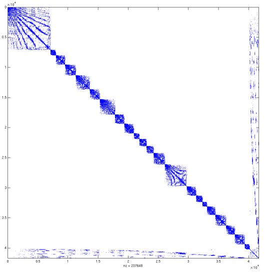
On the next slides, the PCG approach is applied to the matrix obtained via a conforming Mixed-Hybrid FEM. But there should be not difficulty to use it on the matrix obtained using a non-conforming mesh (with Mortar). It is part of our forthcoming work.

Schur complement matrix construction

A specific matrix shape :

The network : $\Omega_s = \cup_i \Omega_i$, $i = 1, \ldots, N_f$, N_f total number of fractures.

$$\begin{bmatrix} \mathbf{A}_{1,1} & \cdots & \cdots & \cdots & \mathbf{A}_{1,N_f+1} \\ \vdots & \ddots & & 0 & \vdots \\ \vdots & & \mathbf{A}_{i,i} & & \mathbf{A}_{i,N_f+1} \\ \vdots & 0 & & \ddots & \vdots \\ \mathbf{A}^T_{1,N_f+1} & \cdots & \mathbf{A}^T_{i,N_f+1} & \cdots & \mathbf{A}_{N_f+1,N_f+1} \end{bmatrix}$$



- One block for each fracture
- Same memory complexity as a 2D problem

Schur complement matrix :

$$\mathbf{S} = \mathbf{A}_{N_f+1,N_f+1} - \sum_{i=1}^{N_f} \mathbf{A}^T_{i,N_f+1} \mathbf{A}^{-1}_{i,i} \mathbf{A}_{i,N_f+1}$$

Equivalent system : $\mathbf{S}\boldsymbol{\Lambda}_{\boldsymbol{\Sigma}} = \tilde{\mathbf{b}}$, with $\tilde{\mathbf{b}} = \mathbf{b}_{N_f+1} - \sum_i \mathbf{A}^T_{i,N_f+1} \mathbf{A}^{-1}_{i,i} \mathbf{b}_i$

With $\boldsymbol{\Lambda}_{\boldsymbol{\Sigma}}$ the trace of hydraulic head unknowns at the intersection edges.

$\Rightarrow$ **Solving with PCG :** no need to compute $\mathbf{S}$: only matrix-vector products involving subdomain solutions

Schur complement and fractures

Local system construction : For a fracture numbered (i)

$$\mathbf{A}_i^{\#} = \begin{bmatrix} 0 & \dots & \dots & \dots \\ \vdots & \mathbf{A}_{i,i} & & \mathbf{A}_{i,N_f+1} \\ \vdots & & \ddots & \vdots \\ \vdots & \mathbf{A}_{i,N_f+1}^T & \dots & \mathbf{A}_{N_f+1,N_f+1}^{(i)} \end{bmatrix}, \quad \mathbf{A} = \sum_{i=1}^{N_f} \mathbf{A}_i^{\#}$$

- Network geometry is relevant for a Schur complement approach
- With too many fractures, one subdomain contains several fractures

Creation of connected fractures set

- Software Scotch : decomposes the network in N_k connected fractures sets F_k, $k = 1, \dots N_k$, $F_k = \cup_i \Omega_i$, $i \in I_k$, with I_k the set of indices generated by Scotch
- Constructs the k-th block-matrix $\mathbf{B}_k^{\#} = \mathbf{P}^T (\sum_{i \in I_k} \mathbf{A}_i^{\#}) \mathbf{P}$, $k = 1, \dots N_k$, and $\mathbf{P}$ a permutation matrix.

Schur complement and fractures

Local block matrix : For a block k, $k = 1, \dots N_k$:

$$\mathbf{B}_k^{\#} = \begin{bmatrix} 0 & \dots & \dots & \dots \\ \vdots & \mathbf{B}_{k,k} & & \mathbf{B}_{k,N_k+1} \\ \vdots & & \ddots & \vdots \\ \vdots & \mathbf{B}_{k,N_k+1}^T & \dots & \mathbf{B}_{N_k+1,N_k+1}^{(k)} \end{bmatrix}$$

Initial matrix :

$$\mathbf{B} = \sum_{k=1}^{N_k} \mathbf{B}_k^{\#} = \mathbf{P}^T \mathbf{A} \mathbf{P}$$

Schur Complement for the matrix B :

$$\mathbf{S_B} = \mathbf{B}_{N_k+1,N_k+1} - \sum_{k=1}^{N_k} \mathbf{B}_{k,N_k+1}^T \mathbf{B}_{k,k}^{-1} \mathbf{B}_{k,N_k+1}$$

Equivalent system : $\mathbf{S_B} \boldsymbol{\Lambda}_{\boldsymbol{\Sigma}} = \tilde{\mathbf{b}}_B$, with $\tilde{\mathbf{b}}_B = \mathbf{b}_{N_k+1} - \sum_k \mathbf{B}_{k,N_k+1}^T \mathbf{B}_{k,k}^{-1} \mathbf{b}_k$

Conjugate gradient

Initialisation :

- Choose $\Lambda_{\Sigma,0}$
- $\mathbf{r}_0 = \tilde{\mathbf{b}}_B - \mathbf{S}_\mathbf{B}\, \mathbf{\Lambda}_{\Sigma,0}$
- $\mathbf{p}_0 = r_0$

Iterations : Do

- $\mathbf{q}_j = \mathbf{S}_\mathbf{B} \mathbf{p}_j \Leftarrow$ Computation of matrix/vector product
- $\alpha_j = \dfrac{\mathbf{r}_j^T \mathbf{r}_j}{\mathbf{p}_j^T \mathbf{q}_j}$
- $\mathbf{\Lambda}_{\Sigma,j+1} = \mathbf{\Lambda}_{\Sigma,j} + \alpha_j \mathbf{p}_j$
- $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{q}_j$
- $\beta_j = \dfrac{\mathbf{r}_{j+1}^T \mathbf{r}_{j+1}}{\mathbf{r}_j^T \mathbf{r_j}}$
- $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
- $j = j + 1$

Until convergence.

Computation of matrix/vector product :

$$\begin{aligned}
\mathbf{S_B p}_j &= \sum_{k=1}^{N_k} \mathbf{S}_k \mathbf{p}_j \\
&= \sum_{k=1}^{N_k} \left(\mathbf{B}^{(k)}_{N_k+1,N_k+1} \mathbf{p}_j - \mathbf{B}^T_{k,N_k+1} \mathbf{B}^{-1}_{k,k} \mathbf{B}_{k,N_k+1} \mathbf{p}_j \right) \\
&= \sum_{k=1}^{N_k} \left(\mathbf{B}^{(k)}_{N_k+1,N_k+1} \mathbf{p}_j - \mathbf{B}^T_{k,N_k+1} \mathbf{B}^{-1}_{k,k} \mathbf{v}_{k,j} \right)
\end{aligned}$$

Cholesky factorization :

$$\mathbf{B}_{k,k} = \mathbf{L}_{k,k} \mathbf{L}^T_{k,k}$$

$$\mathbf{B}^{-1}_{k,k} \mathbf{v}_{k,j} = \mathbf{L}^{-T}_{k,k} \mathbf{L}^{-1}_{k,k} \mathbf{v}_{k,j}$$

PCG with Neumann-Neumann Preconditioning

Initialisation :

- Choose $\Lambda_{\Sigma,0}$
- $\mathbf{r}_0 = \tilde{\mathbf{b}}_B - \mathbf{S_B}\,\mathbf{\Lambda}_{\Sigma,0}$
- $\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$
- $\mathbf{p}_0 = z_0$

Iterations : Do

- $\mathbf{q}_j = \mathbf{S_B}\mathbf{p}_j$
- $\alpha_j = \dfrac{\mathbf{r}_j^T\mathbf{z}_j}{\mathbf{p}_j^T\mathbf{q}_j}$
- $\mathbf{\Lambda}_{\Sigma,j+1} = \mathbf{\Lambda}_{\Sigma,j} + \alpha_j\mathbf{p}_j$
- $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j\mathbf{q}_j$
- $\mathbf{z}_{j+1} = \mathbf{M}^{-1}\mathbf{r}_{j+1}$
- $\beta_j = \dfrac{\mathbf{r}_{j+1}^T\mathbf{z}_{j+1}}{\mathbf{r}_j^T\mathbf{z_j}}$
- $\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j\mathbf{p}_j$
- $j = j+1$

Until convergence.

Preconditioner : $\mathbf{M}^{-1} = \dfrac{1}{N_k}\displaystyle\sum_{k=1}^{N_k}\tilde{S}_k^{-1}$

- If floating subdomain = subdomain with no Dirichlet boundary condition : $\mathbf{S}_k$ singular $\Rightarrow$ Non singular approximation $\tilde{\mathbf{S}}_k$
- Otherwise $\tilde{\mathbf{S}}_k$=$\mathbf{S}_k$

Computation of $\mathbf{z}_{j+1} = \mathbf{M}^{-1}\mathbf{r}_{j+1}$

- $\mathbf{M}^{-1}$ not given explicitly
- Solving $\mathbf{z}_{k,j} = \tilde{\mathbf{S}}_k^{-1}\mathbf{r}_j$ can be done by solving

$$\mathbf{B}_k \begin{pmatrix} \mathbf{x}_k \\ \mathbf{z}_{k,j} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{r} \end{pmatrix} \text{ with } \mathbf{B}_k = \begin{pmatrix} \mathbf{B}_{k,k} & \mathbf{B}_{k,N_k+1} \\ \mathbf{B}^T_{k,N_k+1} & \mathbf{B}^{(k)}_{N_k+1,N_k+1} \end{pmatrix}$$

- $\mathbf{z_j} = \sum_{k=1}^{N_k} \mathbf{z}_{k,j}$

Cholesky factorization :

$$\mathbf{B}_k = \mathbf{L}_k\mathbf{L}_k^T \text{ with } \mathbf{L}_k = \begin{pmatrix} \mathbf{L}_{k,k} & 0 \\ \mathbf{L}_{k,N_k+1} & \mathbf{L}^{(k)}_{N_k+1,N_k+1} \end{pmatrix}$$

This factorization is used within CG to compute $\mathbf{B}_{k,k}^{-1}\mathbf{v}_{k,j}$.

Results

number of subdomains	Number of iterations and execution time	
	Without NN	With NN
2	153 it / 10.8 s	37 it / 4.4 s
4	164 it / 12.5 s	75 it / 9.3 s
8	166 it / 13.7 s	96 it / 12.4 s
16	165 it / 14.4 s	149 it / 20.2 s

TABLE: Iteration number and execution time for a network with 128 fractures for a varying number of subdomains

- PCG with Neumann-Neumann is efficient with a few subdomains
- Parallel computation can improve significantly the results (in terms of memory and CPU requirements)
- When the number of subdomains is too large, global preconditioning can improve convergence.

On-going work

Non conforming mesh and Mortar method

- Build *a posteriori* estimators to optimize mesh generation
- Perform more large scale Monte-Carlo simulations to check convergence
- Run large scale DFNs simulations to derive upscaling rules

Solving the linear system

- compare subdomain decompositions with other methods
- Apply PCG to the matrix obtained with a non conforming mesh
- Parallelize PCG with Neumann-Neumann to reduce memory requirements and improve execution time
- Use global preconditioning (Coarse Grid, Deflation, Balancing) to optimize PCG with Neumann-Neumann