Solving flow in large scale discrete fracture networks with the hybrid high-order (HHO) method

Alexandre Ern, Florent Hédin, Géraldine Pichot, Nicolas Pignet

Solving flow in large scale discrete fracture networks with the hybrid high-order (HHO) method.

Alexandre Ern[2], Florent Hédin[1], Géraldine Pichot[1] *and* Nicolas Pignet[3]

[1]*Inria SERENA & ENPC*

[2] *ENPC & Inria SERENA*

[3]*EDF*

Work in collaboration with

Patrick Laug (Inria GAMMA3, Saclay)

Simon Legrand (Inria SED Paris)

Caroline Darcel and Romain Le Goc (ITASCA S.A.S)

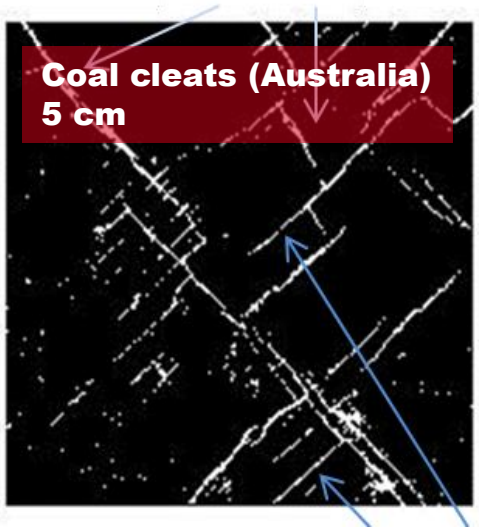
Philippe Davy (Geosciences Rennes, CNRS)

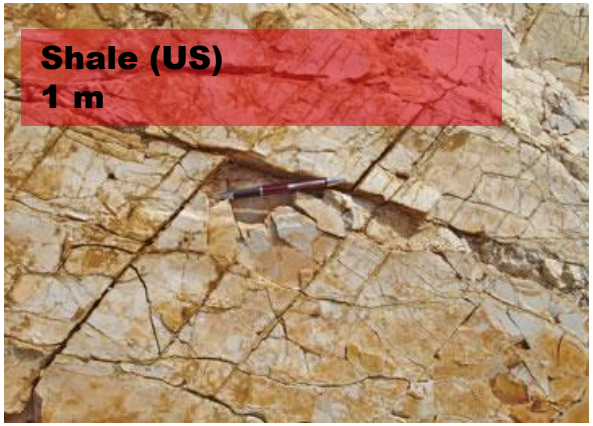
Context

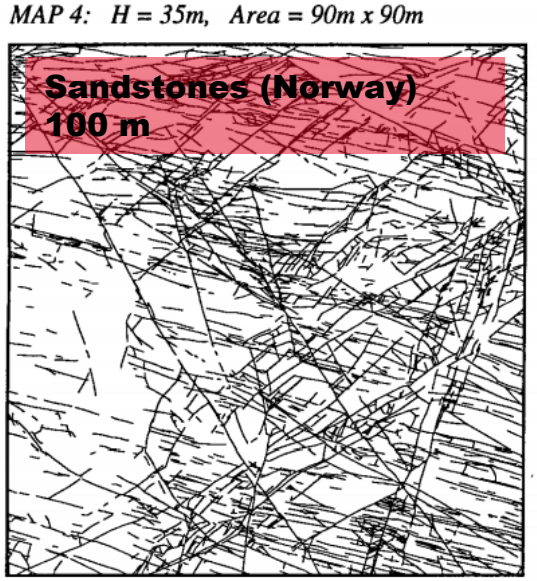
Introduction

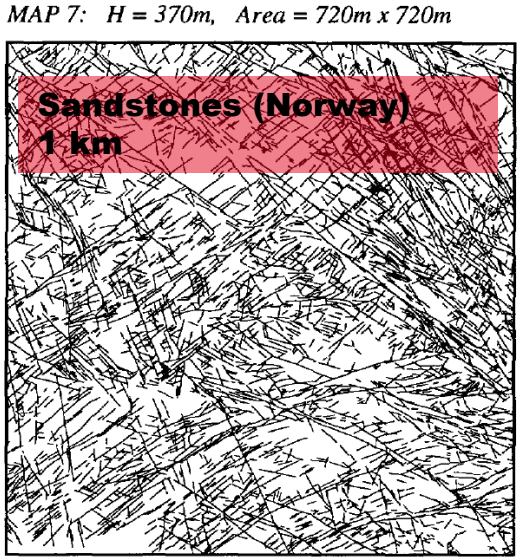
- **Fractures** play a key role in many physical and chemical phenomena
- **Characteristics** of fractures are totally different from surrounding rock matrix:
  - fractures: usually high **permeability**, yield a flow
  - rock: low permeability, will not yield a significant flow
- **Applications: many fields** covering **energy** and **environment**
  - **water extraction** (for drinking, irrigation, industrial processes, ...)
  - **geothermal energy production**
  - oil and gas extraction, nuclear waste storage, CO_2 sequestration, ...

Fracture structures: diversity in size and organization



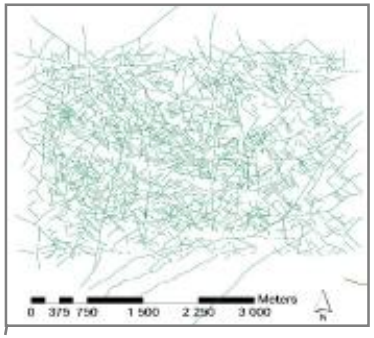


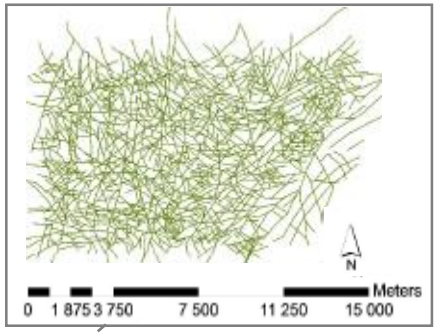


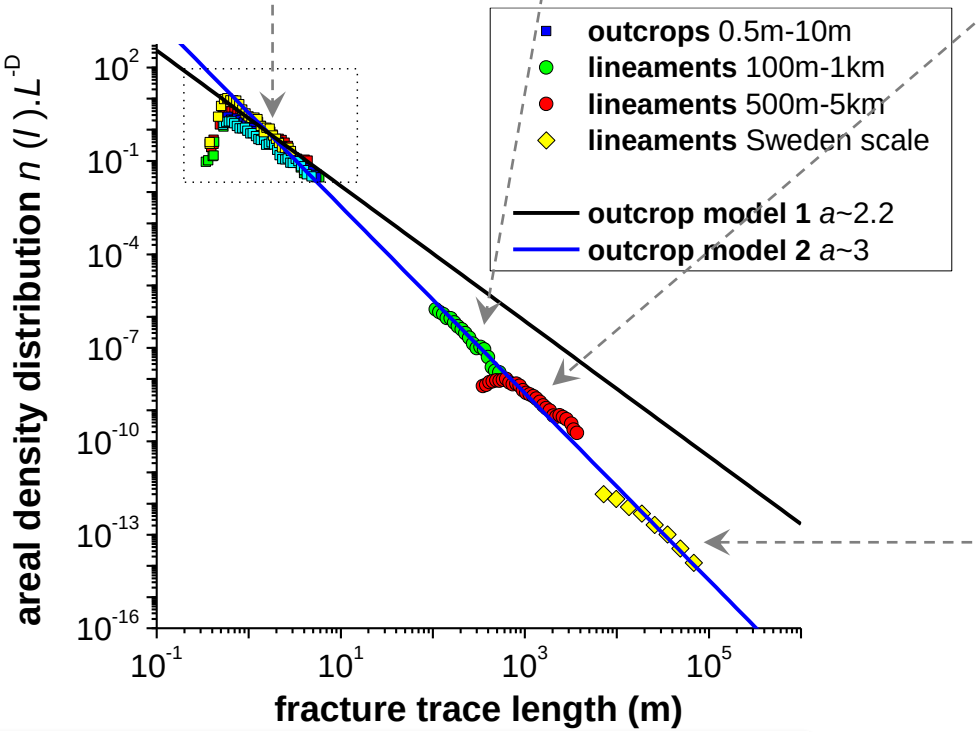


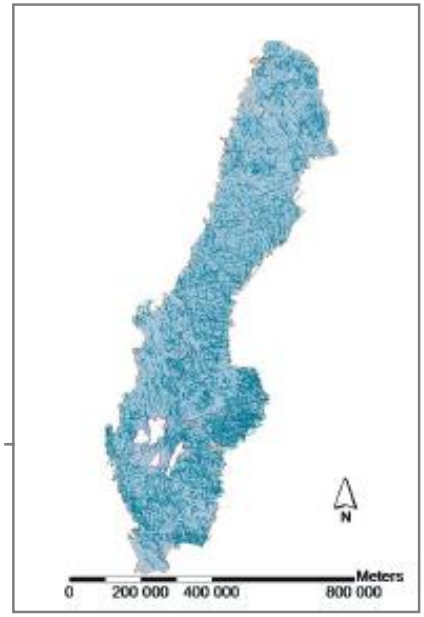
Observations and experiments (example in Sweden)

- **Itasca Consultants (France):** expertise and software in geomechanics
- **SKB (Sweden):** nuclear fuel and waste management



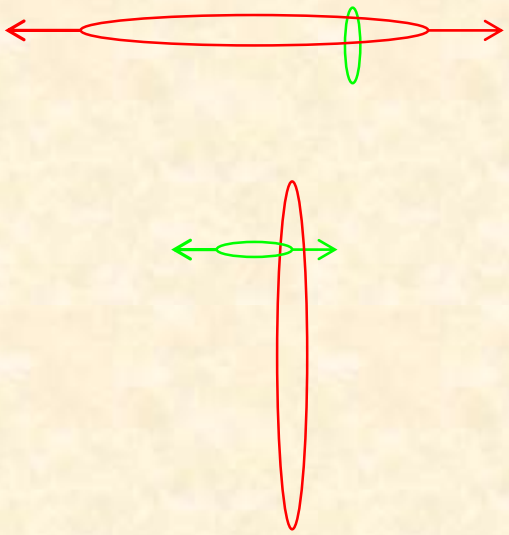






P. Davy, R. Le Goc, C. Darcel, O. Bour, J.R. de Dreuzy, R. Munier, A likely universal model of fracture scaling and its consequence for crustal hydromechanics, J. Geophys. Res., 2010

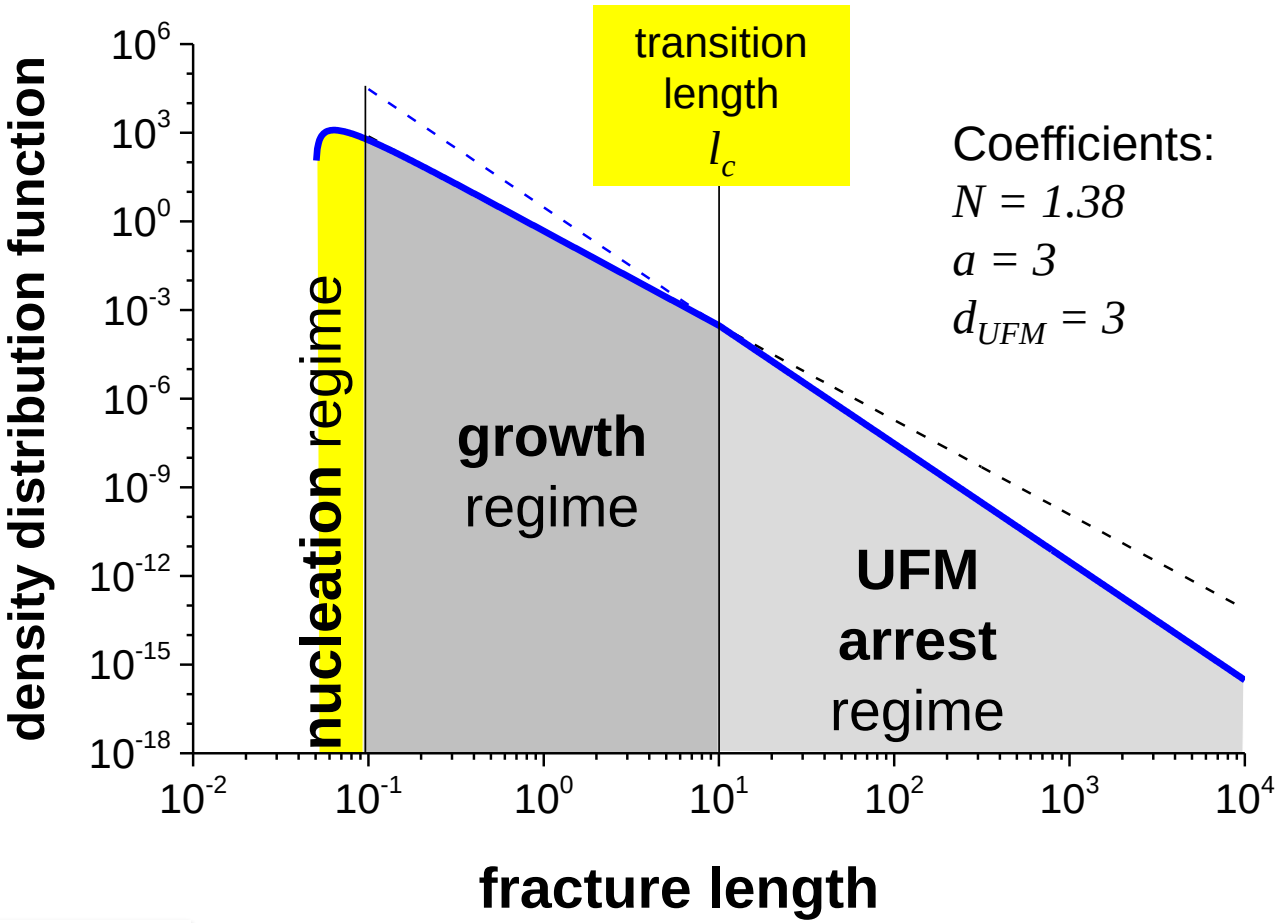
UFM (Unified Fracture Model)



The hierarchical rule: a large fracture can go across a smaller one, but the reverse is unlikely

$n(l) = N\, l^{-a}$

$n(l) = d_{UFM}\, l^{-(D+1)}$

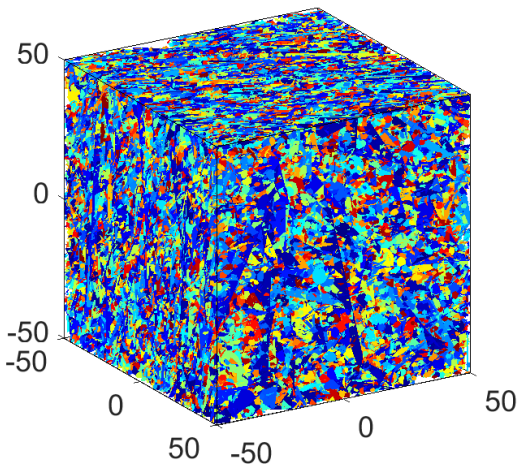


Davy, P., R. Le Goc, and C. Darcel (2013), A model of fracture nucleation, growth and arrest, and consequences for fracture density and scaling, *Journal of Geophysical Research: Solid Earth*, *118*(4), 1393-1407

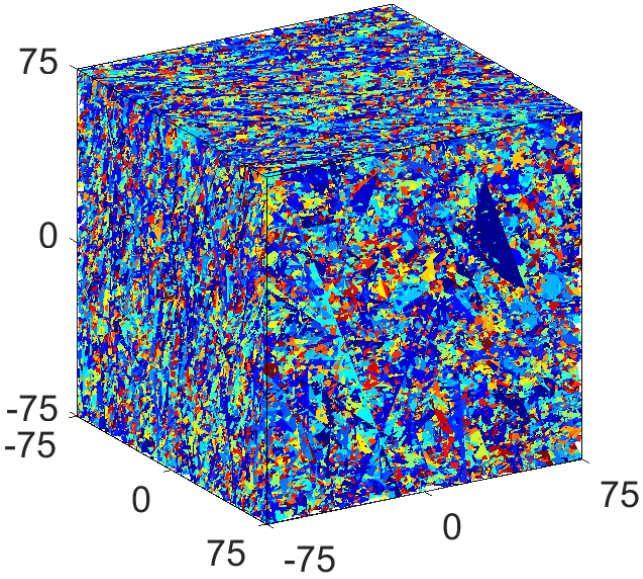
Two-power-law scaling distribution

DFN test cases provided by the LabCom fractory

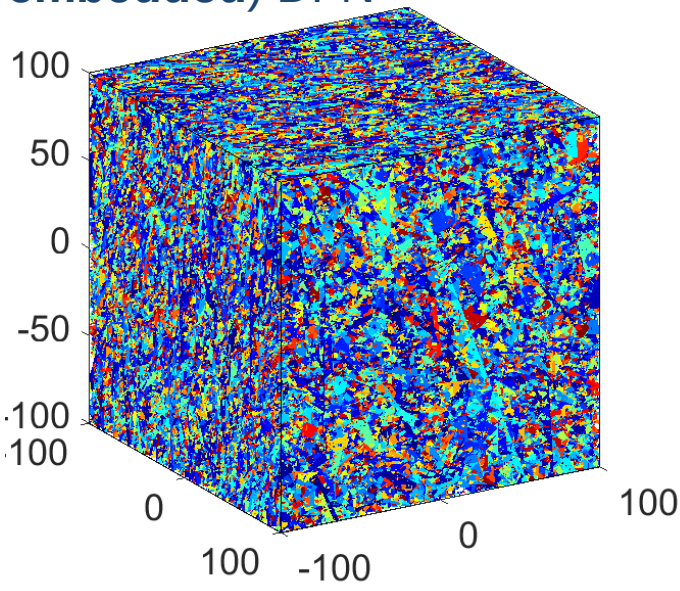
- We change the cube size L to work on different (but **embedded**) DFN



L=100 m



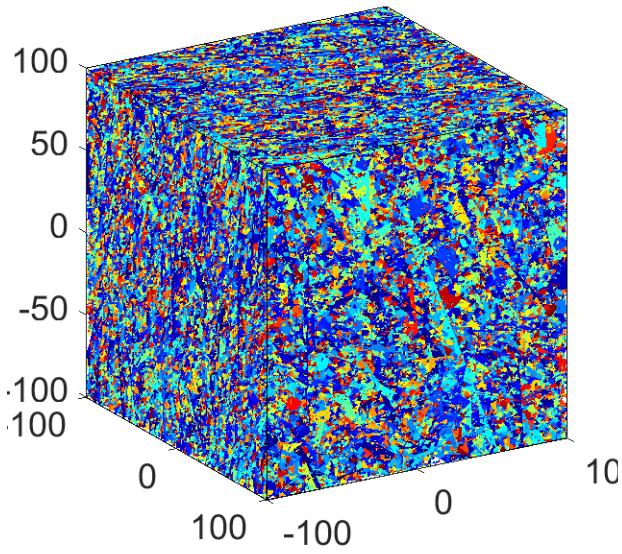
L=150 m



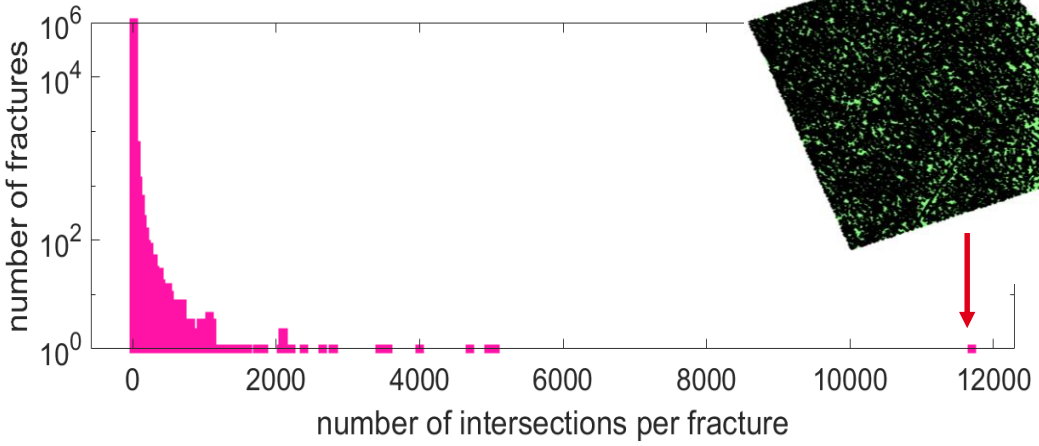
L=200 m

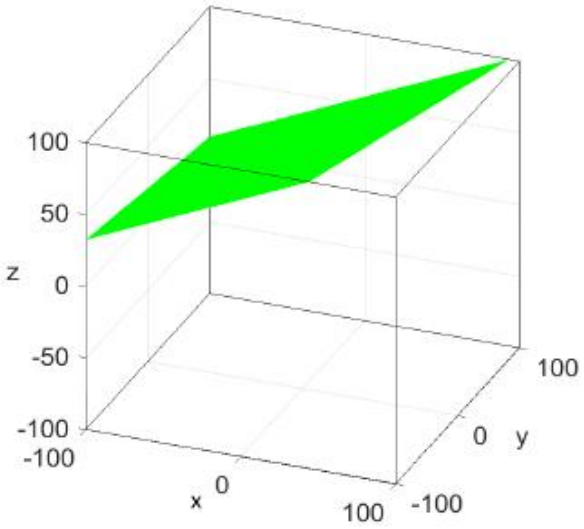
Cube size L	# fractures	# intersections	Max(#inter per fracture)	Transmissivity range [$m^2.s^{-1}$]
100	152,405	302,907	1,022	[3.5e-06; 20.33]
150	508,339	1,031,231	4,930	[3.38-06; 25.8]
200	**1,176,566**	2,410,539	11,710	[3.35-06; 25.8]

DFN test cases provided by the LabCom fractory

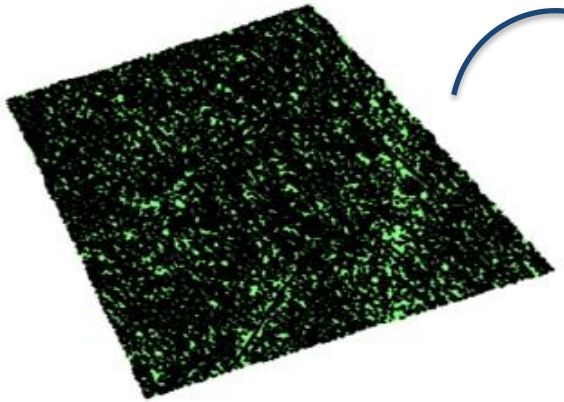


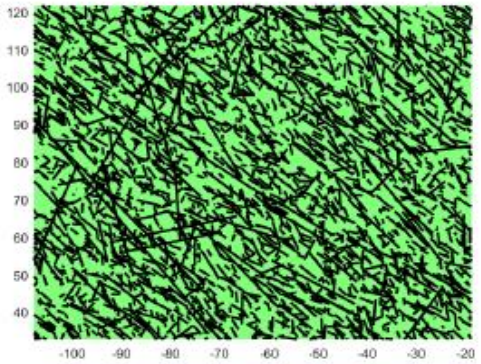
L=200 m
1,176,566 fractures





zoom





11,710 intersections

The flow problem

J. Erhel, J.-R. de Dreuzy, and B. Poirriez. SIAM J. Sci. Comput. (2009)

J. Maryska, O. Severyn, and M. Vohralik, Computational Geosciences (2004)

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 f_i with κ_i a positive definite tensor

$$\nabla \cdot \mathbf{u}_i(\mathbf{x}) = g_i(\mathbf{x}), \quad \text{for } \mathbf{x} \in f_i, \qquad \textit{(Continuity equation)}$$

$$\mathbf{u}_i(\mathbf{x}) = -\kappa_i \nabla p_i(\mathbf{x}), \quad \text{for } \mathbf{x} \in f_i. \qquad \textit{(Poiseuille's law)}$$

$$p_i(\mathbf{x}) = p_i^D(\mathbf{x}), \quad \text{for } \mathbf{x} \in \Gamma^D \cap \Gamma_i, \qquad \textit{(Dirichlet boundary conditions)}$$

$$\mathbf{u}_i(\mathbf{x}) \cdot \mathbf{n} = q_i^N(\mathbf{x}), \quad \text{for } \mathbf{x} \in \Gamma^N \cap \Gamma_i, \qquad \textit{(Neumann boundary conditions)}$$

$$\mathbf{u}_i(\mathbf{x}) \cdot \mathbf{n} = 0, \quad \text{for } \mathbf{x} \in \Gamma_i \backslash (\Gamma_i \cap (\Gamma_D \cup \Gamma_N)). \qquad \textit{(Impervious rock matrix)}$$

Continuity conditions at each intersection I_m with I_m the m^{th} intersection, $m = 1, \ldots, N_I$

N_I total number of intersections

$$p_k^{m,+} = p_l^{m,+} = p_k^{m,-} = p_l^{m,-}, \quad \text{on } I_m, \qquad \textit{(Continuity of the hydraulic heads)}$$

$$\sum_{j \in S_m} \mathbf{u}_j \cdot \mathbf{n}_{j,m}^{+} + \mathbf{u}_j \cdot \mathbf{n}_{j,m}^{-} = 0, \quad \text{on } I_m. \qquad \textit{(Conservation of fluxes)}$$

Mesh generation

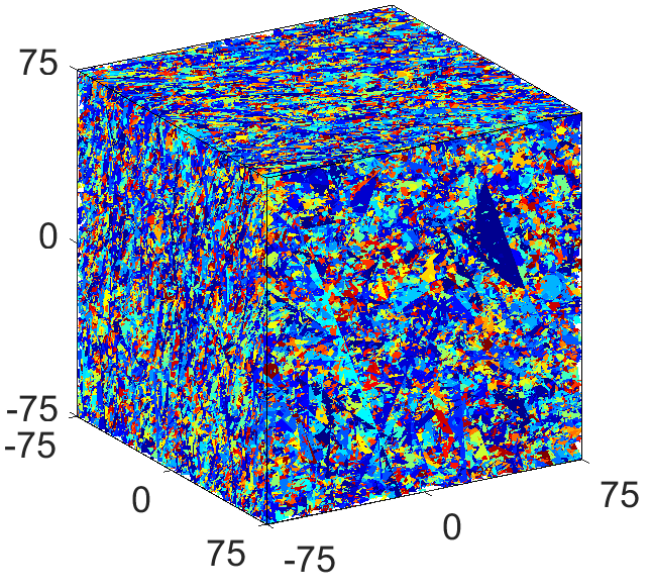
Mesh generation

❑ **MODFRAC software (Inria Gamma & UTT)**

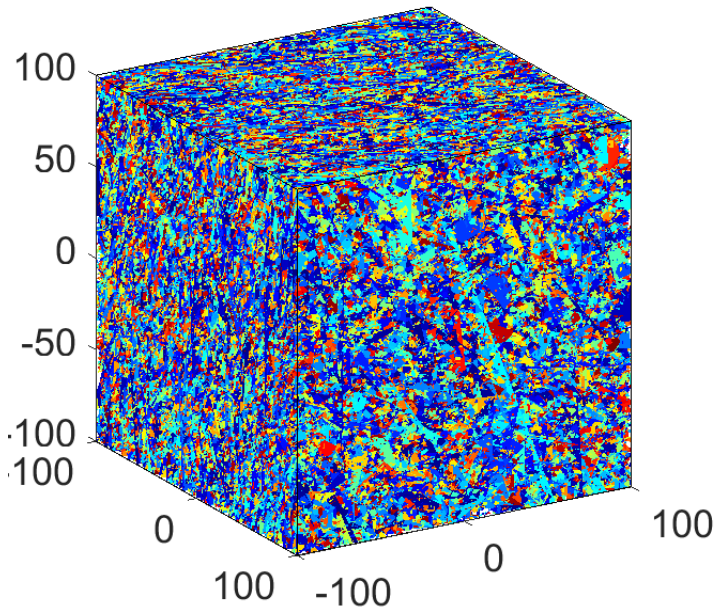
General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications



L=150 m
508,339 fractures



L=200 m
1,176,566 fractures

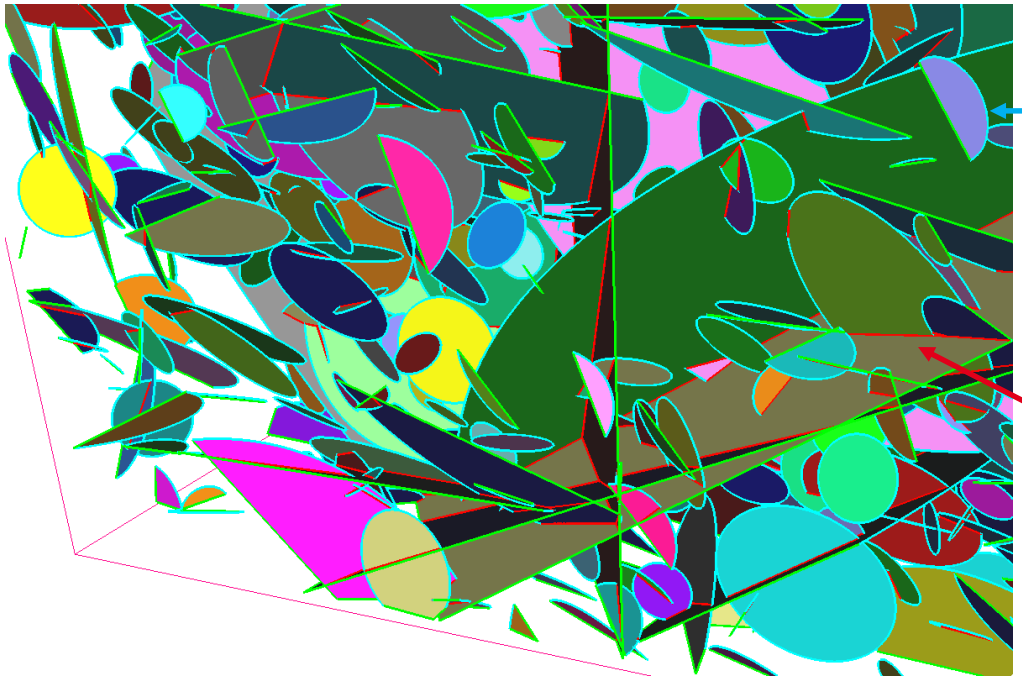
Mesh generation

❑ **MODFRAC software (Inria Gamma & UTT)**

General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications
2. Compute disk/disk and disk/cube **intersections**



Blue: disk contour

Green: disk/cube intersection,

Red: disk/disk intersection,

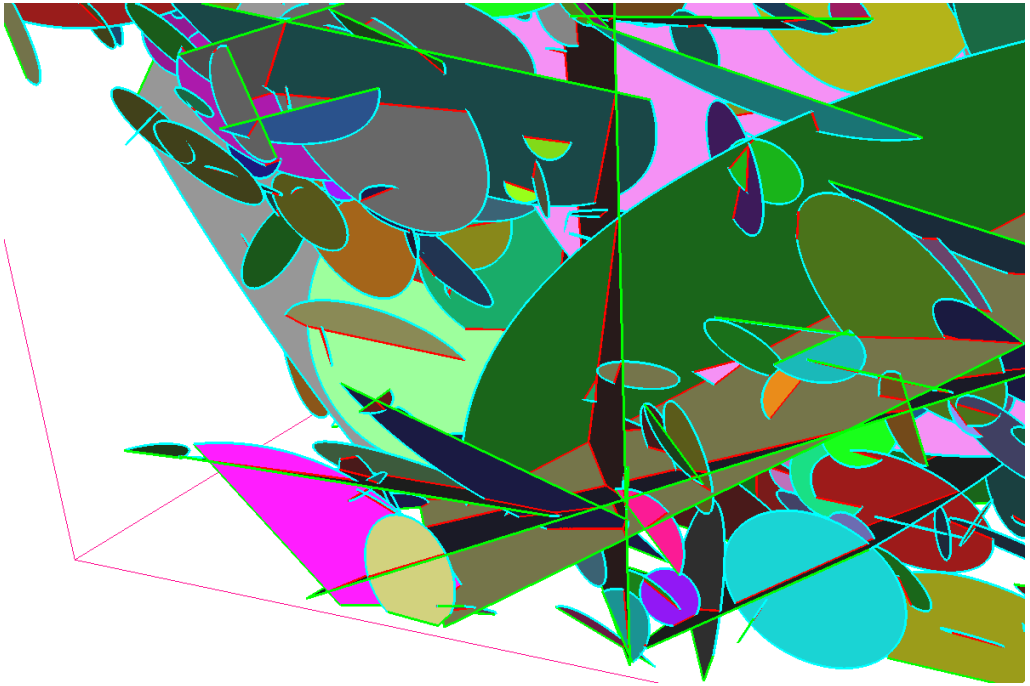
Mesh generation

❑ **MODFRAC software (Inria Gamma & UTT)**

General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications
2. Compute disk/disk and disk/cube **intersections**
3. Select disks that are **connected** to given cube faces



- Cube faces are selected by the user
- Here the **6 faces** of the cube are selected:
  any disk must have a path to all the faces of the cube (graph-based representation)

Mesh generation

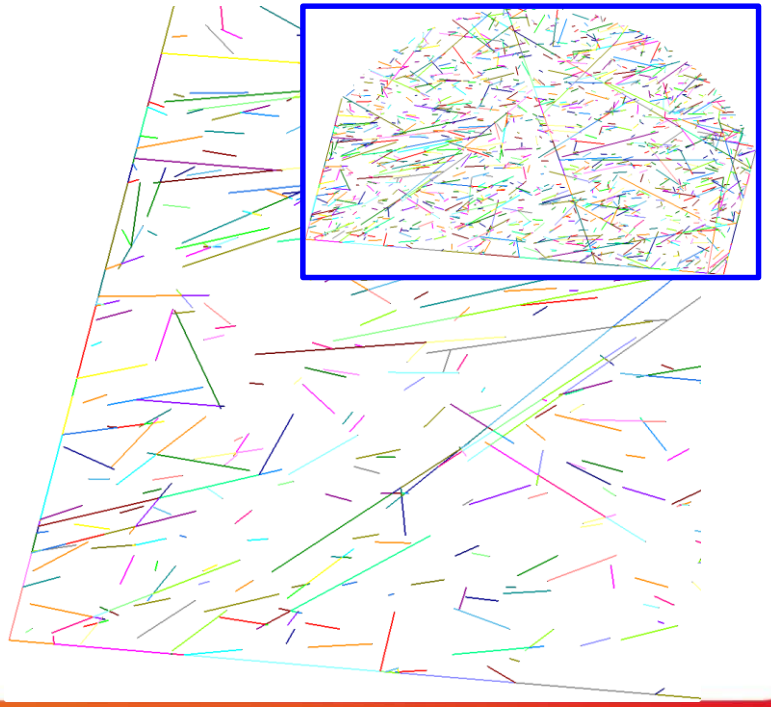
❑ **MODFRAC software (Inria Gamma & UTT)**

General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications
2. Compute disk/disk and disk/cube **intersections**
3. Select disks that are **connected** to given cube faces
4. For each reference domain, build a **conforming set** of the intersections

- Intersections of the line-segments (i.e., intersections of 3 fractures) are inserted making subsegments
- Topological consistency between patches is ensured
- Difficulties: small distances, small lengths, small angles, many intersections



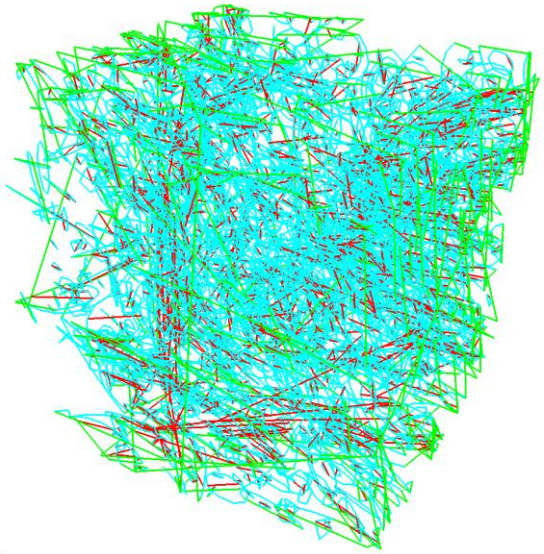
Mesh generation

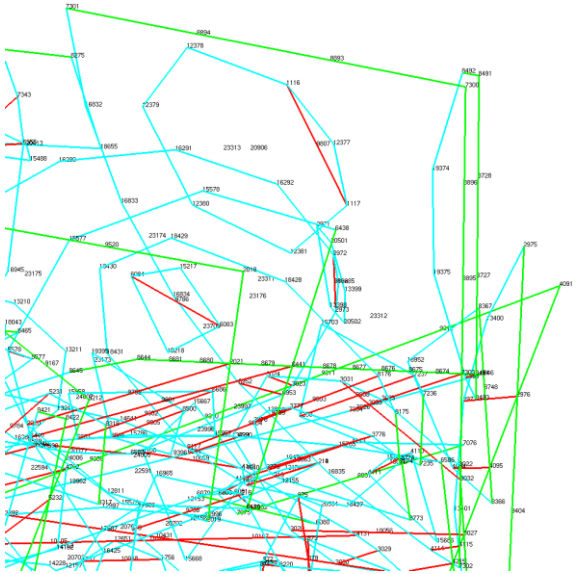
❑ **MODFRAC software (Inria Gamma & UTT)**

General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications
2. Compute disk/disk and disk/cube **intersections**
3. Select disks that are **connected** to given cube faces
4. For each reference domain, build a **conforming set** of the intersections
5. Build a **discretization of each 3D curve**





- Discretization depends on a size function: constant, curvature-dependent, adapted to a solution, etc.

Mesh generation

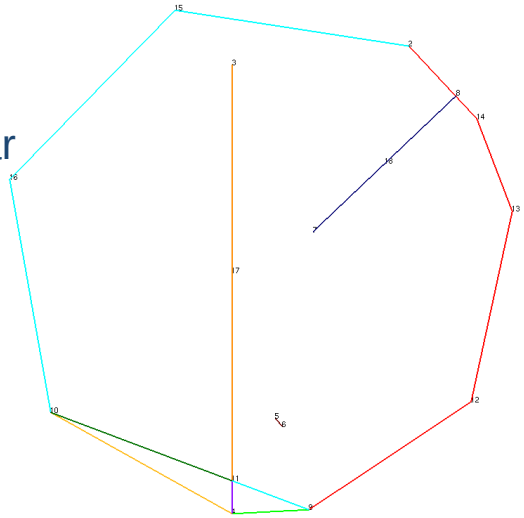
❑ **MODFRAC software (Inria Gamma & UTT)**

General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications
2. Compute disk/disk and disk/cube **intersections**
3. Select disks that are **connected** to given cube faces
4. For each reference domain, build a **conforming set** of the intersections
5. Build a **discretization of each 3D curve**
6. Project each 3D discretization to define the **2D discretized boundaries** of the corresponding parametric domains and check if the parametric domain is well defined, if not, subdivide and iterate

- Discretized 3D curves are projected to planar domains
- For each domain, these projections define geometric constraints (boundaries and internal curves)



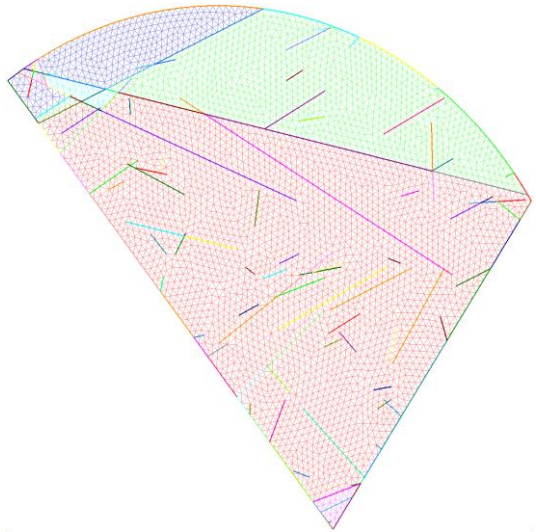
Mesh generation

❑ **MODFRAC software (Inria Gamma & UTT)**

General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications
2. Compute disk/disk and disk/cube **intersections**
3. Select disks that are **connected** to given cube faces
4. For each reference domain, build a **conforming set** of the intersections
5. Build a **discretization of each 3D curve**
6. Project each 3D discretization to define the **2D discretized boundaries** of the corresponding parametric domains and check if the parametric domain is well defined, if not, subdivide and iterate
7. Generate a **mesh of each parametric domain**



- Planar mesher:
  - ➢ advancing front, generalized Delaunay, anisotropic

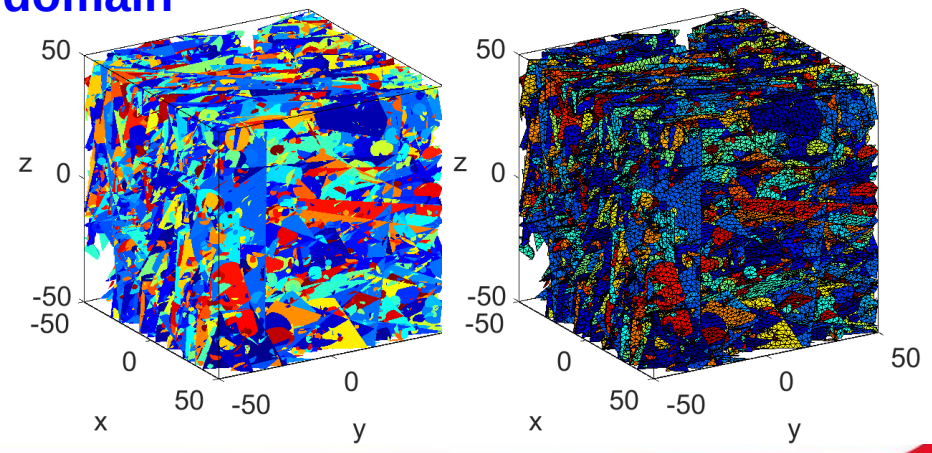
Mesh generation

❑ **MODFRAC software (Inria Gamma & UTT)**

General scheme

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

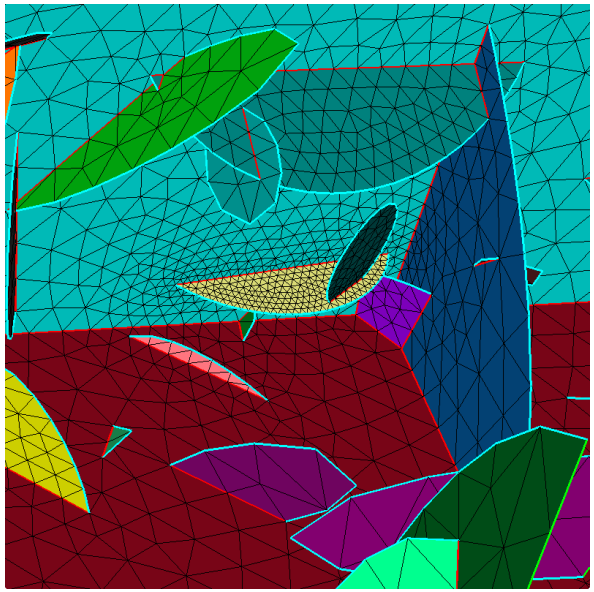
1. **Input:** DFN (up to millions of disks), delimiting cube, mesh size specifications
2. Compute disk/disk and disk/cube **intersections**
3. Select disks that are **connected** to given cube faces
4. For each reference domain, build a **conforming set** of the intersections
5. Build a **discretization of each 3D curve**
6. Project each 3D discretization to define the **2D discretized boundaries** of the corresponding parametric domains and check if the parametric domain is well defined, if not, subdivide and iterate
7. Generate a **mesh of each parametric domain**
8. Create the final **tridimensional mesh**



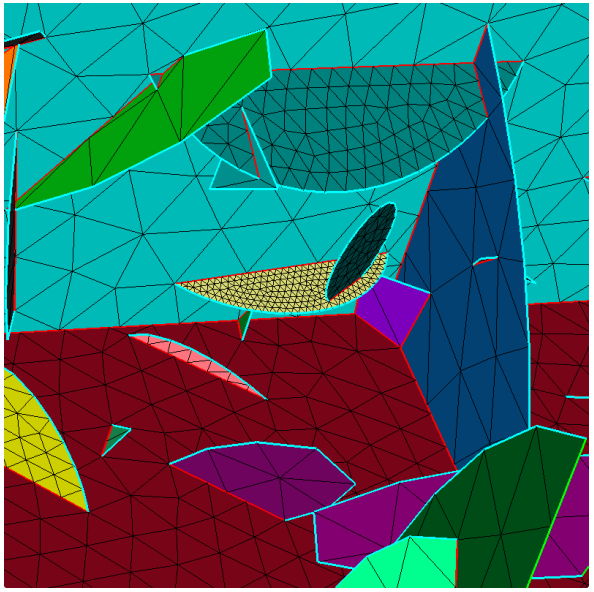
Mesh generation of the test cases

- MODFRAC software (Inria Gamma & UTT)

Cube size L	# fractures	#triangles	Min area	Mean(Q)	Min(Q)	Time hh:mm:ss
100	152,405	2,770,604	1.4e-09	0.66	3.1e-05	00:01:34
150	508,339	12,890,943	1.16e-11	0.77	1e-4	00:06:29
200	**1,176,566**	20,522,575	1.16e-11	0.63	9.25e-05	**00:23:25**



Matching surface mesh



Non-matching surface mesh

$$Q(T) := 4\sqrt{3}\frac{|T|}{l_1^2 + l_2^2 + l_3^2}$$

On a Laptop Intel Core i7
4 cores CPU
32GiB RAM

Parallelism using
4 POSIX-threads

H. Borouchaki, P. Laug, P.L. George, Parametric surface meshing using a combined advancing-front & generalized-Delaunay approach, Int. J. for Numerical Methods in Engineering, 2000

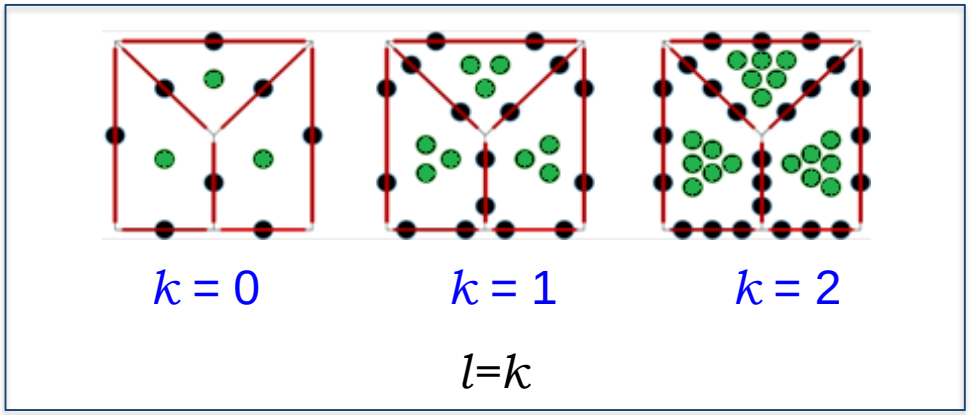
The HHO method

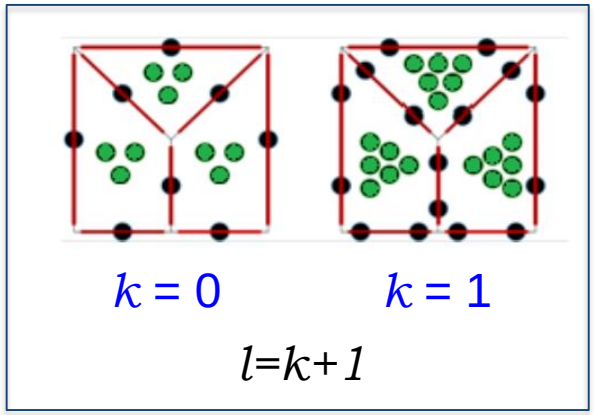
The Hybrid High Order (HHO) method

Di Pietro, D.A and Ern, A. and Lemaire, S. Computational Methods in Applied Mathematics (2014)

- Discrete unknowns on each finite element are:
  - On mesh faces
  - Inside the elements

- HHO unknowns are polynomial coefficients (no physical interpretation)
  - ✓ On mesh faces of order k
  - ✓ Inside the elements of order l





In our DFN applications, we choose $l=k+1$ to get a simpler stabilization operator than the equal-order choice $l = k$.

- Algebraically, we obtain a Schur complement system for only the faces unknowns (static condensation) and known to be symmetric positive definite.

The Hybrid High Order (HHO) method

Di Pietro, D.A and Ern, A. and Lemaire, S. Computational Methods in Applied Mathematics (2014)

- HHO required two main ingredients:
  - ✓ Local gradient reconstruction from cell and face unknowns
  - ✓ A stabilization term ensures that the traces of the cell unknows and the face unknowns match in a least-squares sense.

- Conservative fluxes are built from the sum of two terms
  - ✓ the normal component of the reconstructed gradient
  - ✓ A correction term coming from the stabilization

Cockburn, B. and Di Pietro, D.A. and Ern, A. ESAIM: M2AN (2016)

- Main advantages of HHO:
  - ✓ **Feature 1**: Globally and locally **mass conservative method** (whatever $k \geq 0$)
  - ✓ **Feature 2**: Face polynomials of **order $k \geq 0$**
  - ✓ **Feature 3**: Support **general meshes** (polygonal/polyhedral cells)

- Implementation:

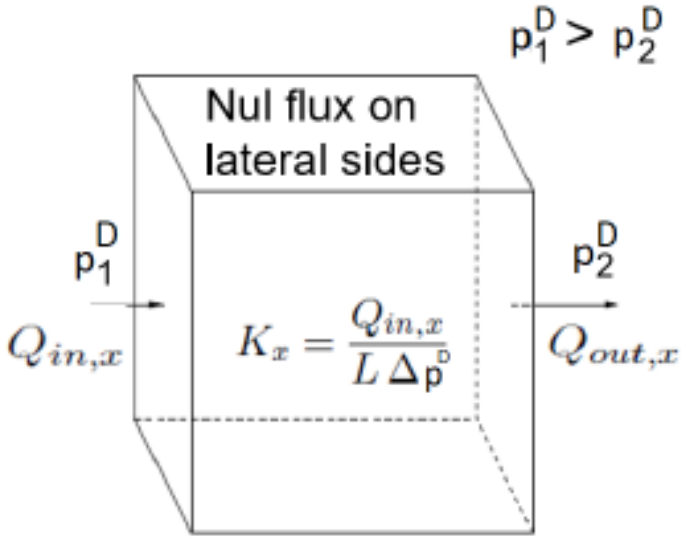
❑ **Library Disk++ for HHO (open-source)**

Cicuttin, M. and Di Pietro, D.A. and Ern, A. Journal of Computational and Applied Mathematics (2018)

Computational performance

Computational performance of HHO method in the context of extremely large DFNs.

- Evaluation of the equivalent permeability of a DFN by running a permeameter test case



- ✓ Software:
  - ❑ **Library Disk++ for HHO (open-source)**
  - ❑ **NEF++ (Inria) for an application to DFNs**
  - ❑ **Prune (Inria) for automatic parameter studies**

Inria Cluster CLEPS: 4x Intel Xeon E7-4860 v2; 12 cores, 2.6-3.2GHz; **3T RAM**

- Objectives:

✓ **Choice of the basis functions**

✓ **Trade-off between increasing the polynomial order and refining the mesh**

✓ **Can resources be saved by the means of polygonal cells ?**

Choice of the basis functions

- HHO unknowns are polynomial coefficients (no physical interpretation)
  - ✓ No reference element
  - ✓ Defined on physical element
  - ✓ No continuity is required between face and cell

A lot of possibilities
BUT
the choice
may affect the condition number of local matrices and performances

- **On mesh faces:** scaled monomials x_F^α with $\alpha = 0, \ldots, k$ and $x_F = 2\dfrac{(x - \bar{x}_F)}{|F|}$

- **On every mesh cell T:** we compare two choices of monomials with

$$\alpha, \beta = 0, \ldots, l = k+1, \alpha + \beta \leq l$$

[Cartesian] $x_T^\alpha y_T^\beta$

$$x_T = 2\frac{(x - \bar{x}_T)}{h_x^T}, \qquad y_T = 2\frac{(y - \bar{y}_T)}{h_y^T}$$

[Rotated] $\xi_T^\alpha \zeta_T^\beta$

$$\xi_T = 2\frac{(x - \bar{x}_T, y - \bar{y}_T)\cdot a_1^T}{h_1^T} \qquad \zeta_T = 2\frac{(x - \bar{x}_T, y - \bar{y}_T)\cdot a_2^T}{h_2^T}$$

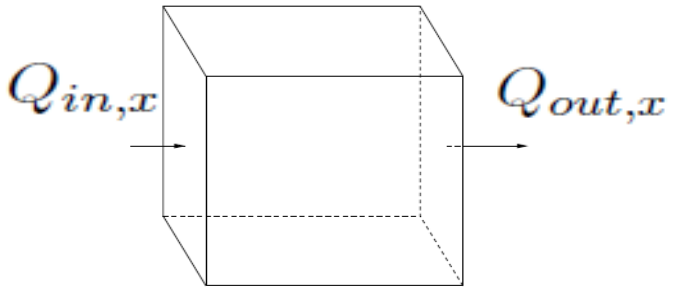
- **Hierarchical** : easy to change order *k* (only few monomials to compute)
- **Centered** : independent of translation (use barycenter)
- **Scaled** : independent of homothety (use length of the bounding box)

Cartesian & Rotated

- **Rotated** : independent of rotation (use inertia axes) → **Rotated**

Choice of the basis functions

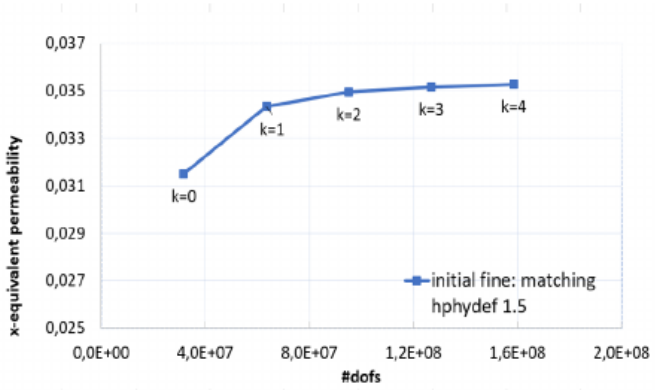
		$\lvert Q_{in,x} + Q_{out,x} \rvert$	
	k	[Cartesian]	[Rotated]
L100	0	1.44e-07	9.82e-11
	1	1.66e-4	5.26e-10
	2	2.31e-2	4.63e-09
	3	15.21	4.20e-08
L150	0	3.12E-07	7.32e-10
	1	1.58e-3	1.31e-08
	2	38.22	1.85e-08
	3	45.73	4.85e-08



- Mass conservation
  - ✗ [Cartesian]
  - ✓ [Rotated]
    - Improve the condition number of the local matrices
    - Global and local mass conservation ensured whatever k

Trade-off between polynomial order and mesh refinement

- Evaluation of the reference equivalent permeability on two fine meshes



L150: 508,339 fractures
12,890,943 triangles

k	#dofs	Total Time (hh:mm:ss)	RAM (GiB)	Rel. Error Kx
0	19,999,230	0:49:29	61	8.4%
1	39,998,460	1:52:24	144	2.0%
2	59,997,690	2:55:54	271	0.7%
3	79,996,920	4:46:51	457	0.2%
4	99,996,150	6:16:19	681	0.0%

Reference values for $k = 4$

$$K_x = 3.51\text{e}{-2}\ \text{m}^2.\text{s}^{-1}$$

L200: 1,176,566 fractures
20,522,575 triangles

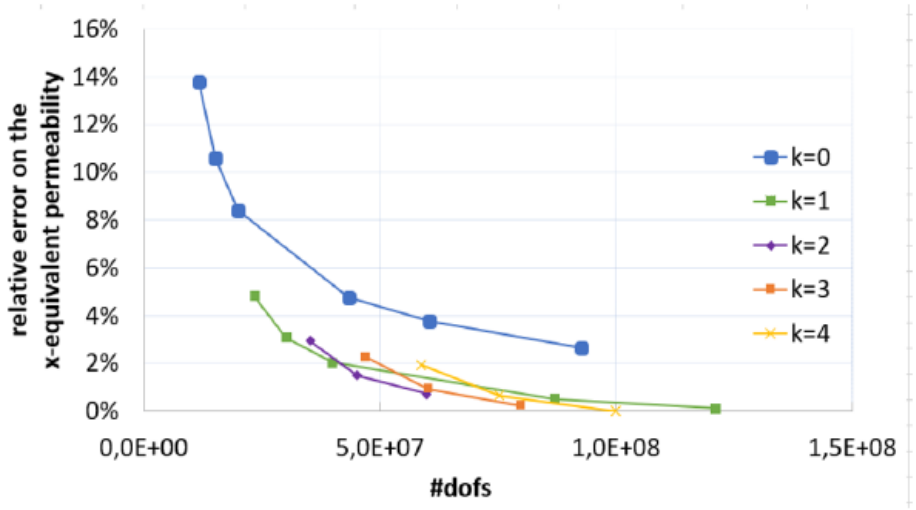
k	#dofs	Total Time (hh:mm:ss)	RAM (GiB)	Rel. Error Kx
0	31,711,430	1:28:45	100	10.7%
1	63,422,860	3:16:14	246	2.6%
2	95,134,290	6:19:46	481	0.9%
3	126,845,720	9:44:50	784	0.3%
4	158,557,150	14:39:47	1186	0.0%

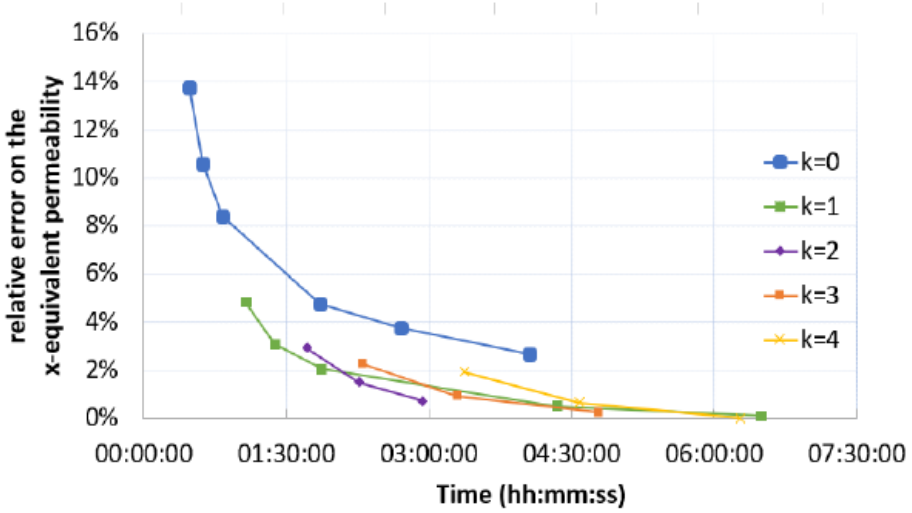
Reference values for $k = 4$

$$K_x = 3.53\text{e}{-2}\ \text{m}^2.\text{s}^{-1}$$

Trade-off between polynomial order and mesh refinement

- Evaluation of the equivalent permeability on coarser meshes

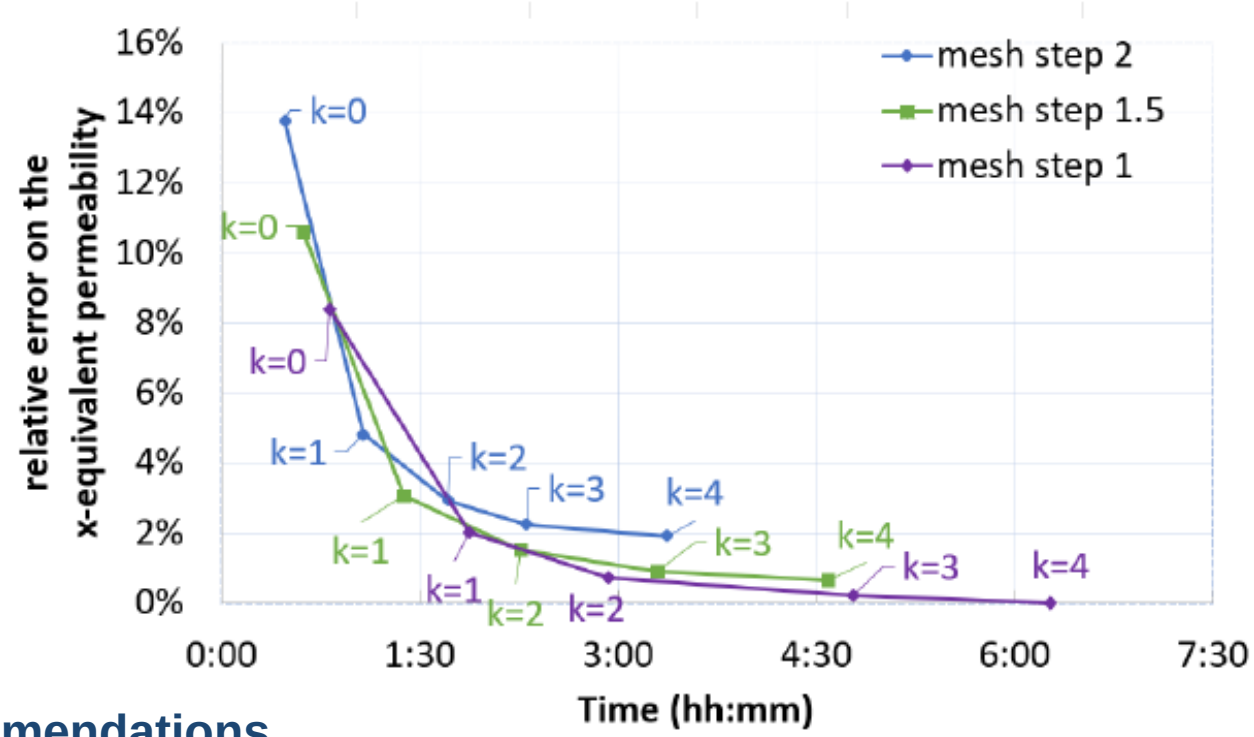




- **Recommendations**
  - ✓ **use a face polynomial order k at least equal to 1, instead of refining a mesh while keeping k = 0**

Trade-off between polynomial order and mesh refinement

- Evaluation of the equivalent permeability on coarser meshes



Recommendations

- ✓ use a face polynomial order k at least equal to 1, instead of refining a mesh while keeping $k = 0$
- ✓ choose $k = 1$ on a mesh fine rather than $k > 1$ on a coarser mesh

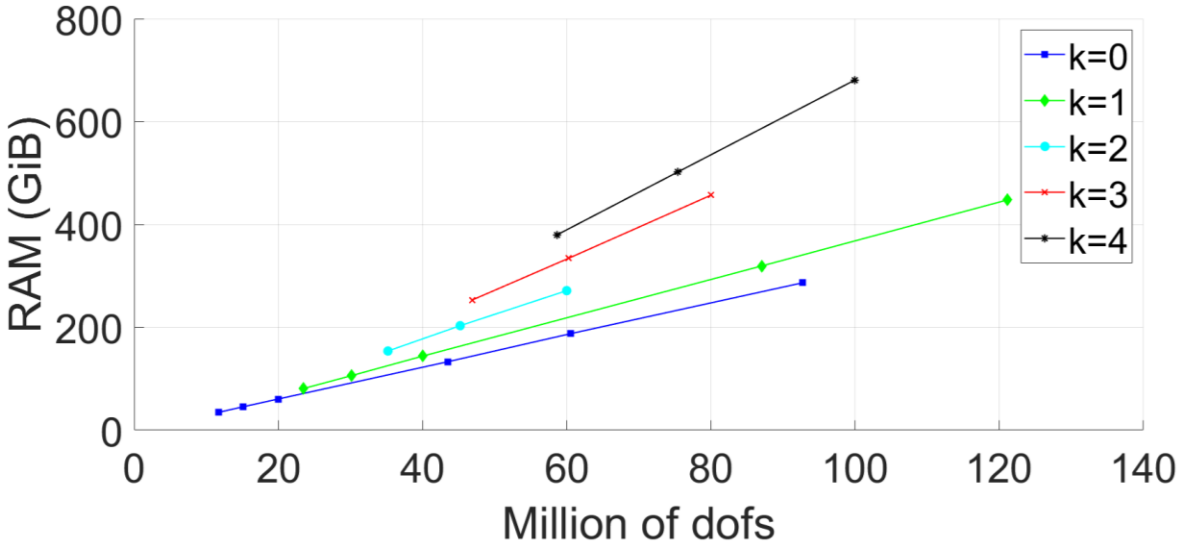
Trade-off between polynomial order and mesh refinement

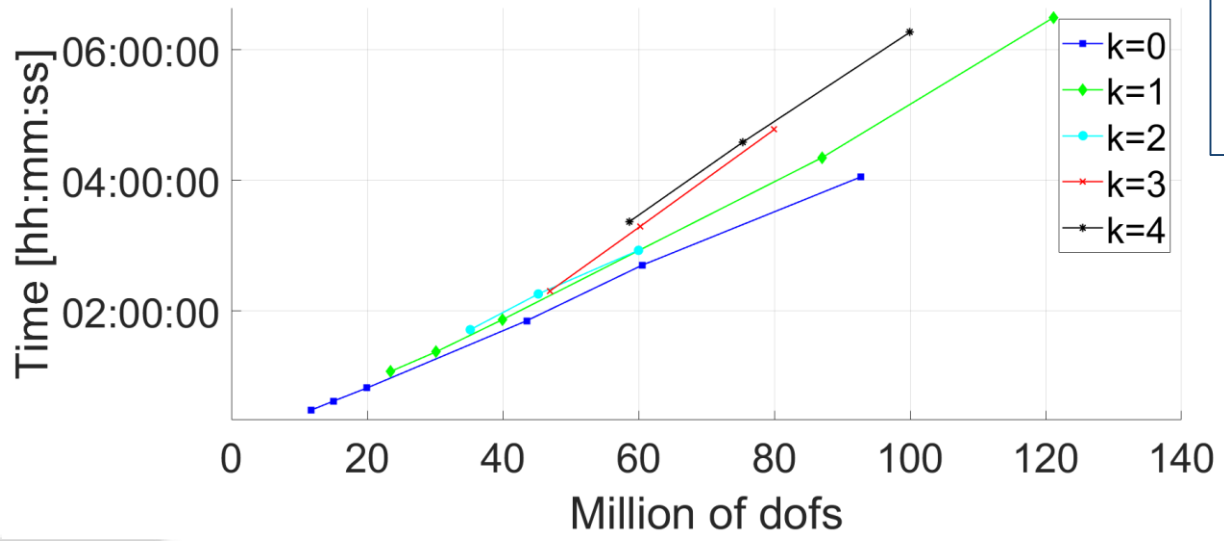
- Computational resources for approximately the same number of dofs

L150: 508,339 fractures					
k	**#dofs**	**Total Time (hh:mm:ss)**	**RAM (GiB)**	**Rel. Error *Kx***	***DFN Mesh step***
0	**43,531,537**	**01:51:14**	**133**	**4.8%**	**0.6**
1	**39,998,460**	**01:52:24**	**144**	**2.0%**	**1**
2	**45,222,504**	**02:15:48**	**203**	**1.5%**	**1.5**
3	**46,930,868**	**02:18:27**	**253**	**2.3%**	**2**

- **Comments**
  - ✓ **As expected, time and RAM increase with *k***

Trade-off between polynomial order and mesh refinement

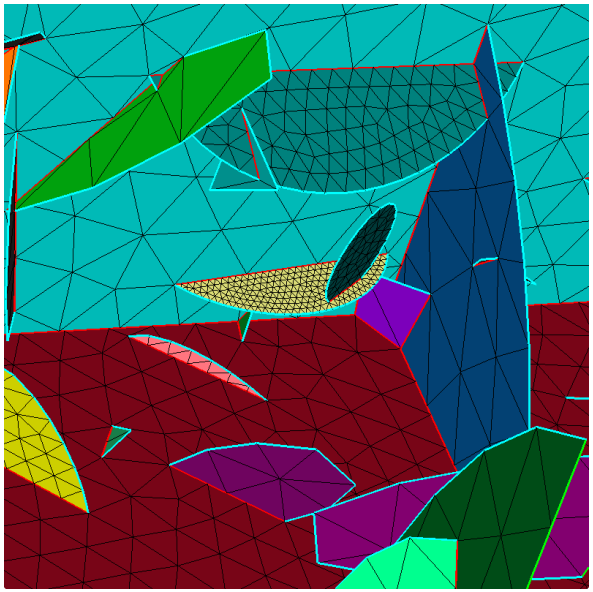




- ❑ **Comments**
- ✓ **The extra time and peak of RAM for $k \leq 2$ seems negligible by comparison with the case $k = 0$**
- ✓ **For $k > 2$, the slopes becomes higher**

How to take advantage of polygonal cells?

- **How ?**
  - **With non matching mesh:** the mesh generator **MODFRAC** (Inria&UTT) allows to choose an independent mesh step from one fracture to another
  - **With the channelling effect of DFN flow:** the fractures that carry most of the flow could be refined, the other could be coarsened.
  - **With the creation of polygons** from the non-matching discretization



- **Limitation:**
  - Coarsening is constrained by the geometry and intersections of the fractures!

How to take advantage of polygonal cells?

Algorithm 1 Standard procedure

1: (F1) Generate a mesh of the DFN with matching cells (in our case triangles) at the intersections
2: (F2) Perform the corresponding flow simulation

VERSUS

Algorithm 2 A mesh refinement/coarsening procedure

1: (I1) Generate a coarse mesh of the DFN
2: (I2) Perform the coarse flow simulation
3: (I3) Decide which fractures to refine according to their flow contribution,
4: (I4) Remesh the DFN according to step (I3): only the selected fractures are refined and the mesh is therefore nonmatching at the intersections between fractures
5: (I5) Run a node insertion algorithm to build a matching polygonal mesh at intersections: the mesh is therefore composed of triangles and polygons
6: (I6) Perform the flow computation on the new polygonal mesh obtained at step (I5)

- **Questions:**
  - Step (I3): How to select the set of fractures that needs to be refined ?
  - Step (I5): How to create the polygons from the non-matching meshes?

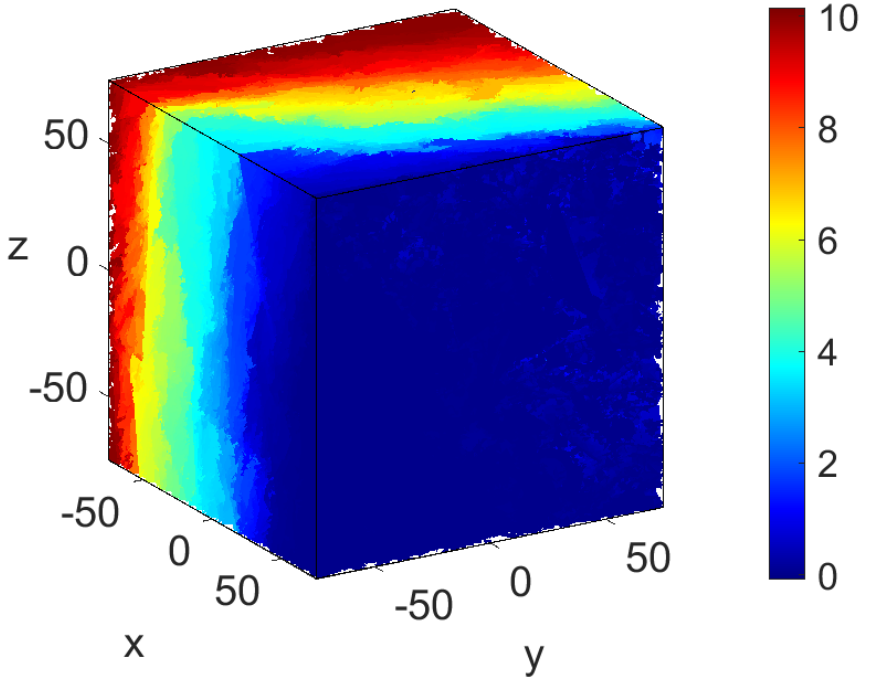
How to take advantage of polygonal cells?

- **Steps (I1):** generate a coarse mesh

Cube size L	# fractures	#triangles	Min area	Mean(Q)	Min(Q)	Time hh:mm:ss
150	508,339	**7,029,255**	1.16e-11	0.55	5.5e-5	00:05:30

- **Steps (I2):** perform a coarse flow simulation

L150
508,339 fractures
x direction: h1:10, h2 :0
Mesh load: 6 min,
Computation: 16 minutes (Laptop)
Coarse mesh: **7,029,255** triangles



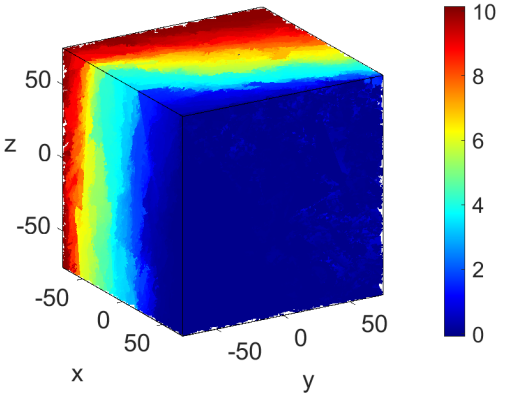
PC Intel Corei7 4 cores CPU @ 2.90 GHz
32GB RAM

How to take advantage of polygonal cells?

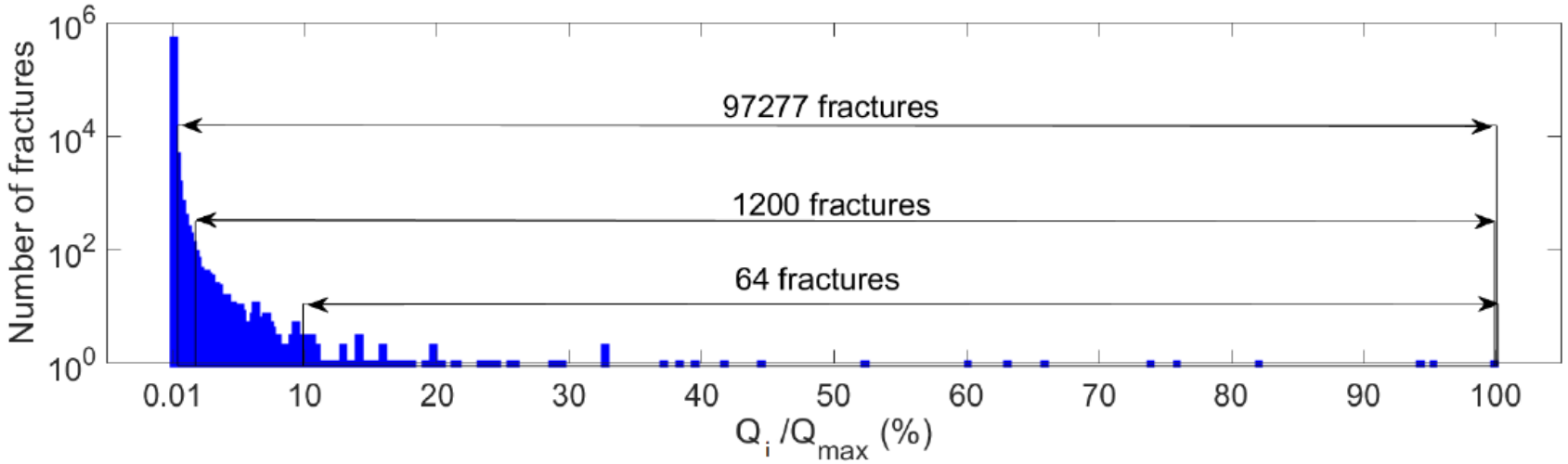
Step (I3): select the set of fractures that needs to be refined.

- **A first basic idea: from the former coarse flow simulation,** for each fracture f_i compute the total flow value Q_i exchanged between f_i and its intersecting fractures:

L150
508,339 fractures
x direction: h1:10, h2 :0
Mesh load: 6 min,
Computation: 16 minutes (Laptop)
Coarse mesh: 7,029,255 triangles



$$\begin{cases} Q_i := \frac{1}{2}\sum_{m=1}^{N_I^i} |Q_{i,m}| \\ Q_{max} := max_i(Q_i) \end{cases}$$

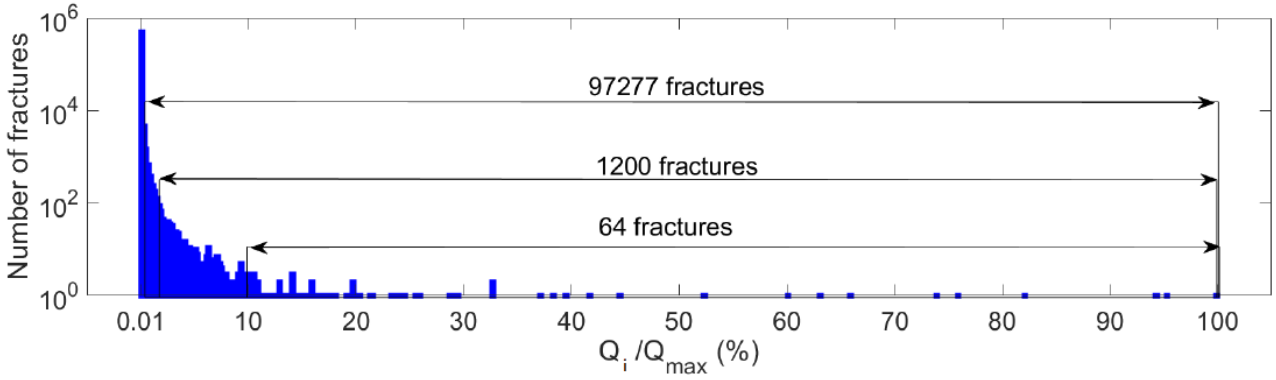


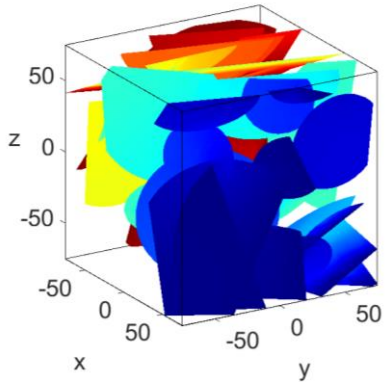
How to take advantage of polygonal cells?

Step (I3): select the set of fractures that needs to be refined

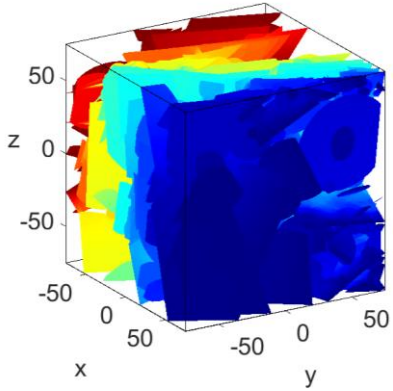
L150
508,339 fractures

Threshold: % of the maximum output flux	#fractures above the threshold
10%	64
1%	1200
0.01%	97,277

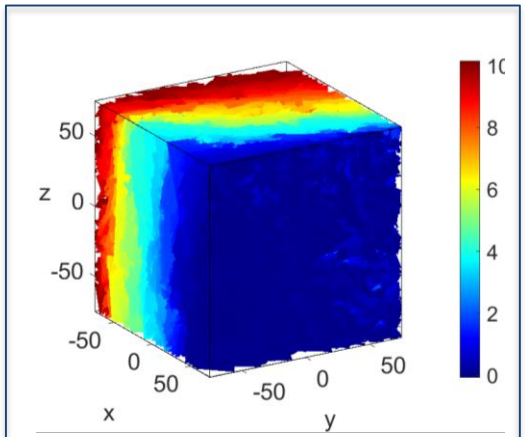




64 fractures
(0.01% of the DFN)



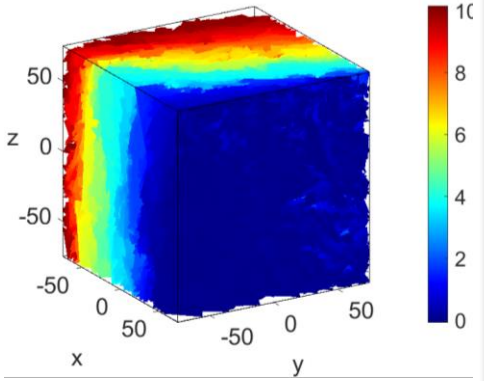
1200 fractures
(0.23% of the DFN)



97,277 fractures
(19% of the DFN)

How to take advantage of polygonal cells?

Step (I4): only the the set of fractures selected at step (I3) are refined



97,277 fractures
(19% of the DFN L150)
(8% of the DFN L200)

L150:

- We keep a mesh **step of 1** for 97,277 selected fractures (19% of the DFN L150) and coarsen the other fractures (**mesh step of 2**)

L200:

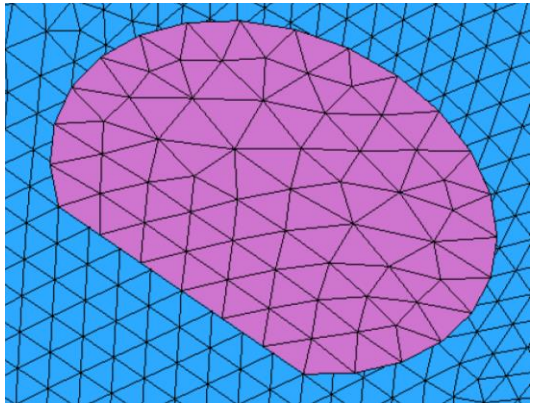
- To **save a coarse simulation** and thanks to the **embedded feature of the** DFN, we keep the 97,277 selected fractures (8% of the DFN L200) of test case L150 and mesh them with a mesh **step of 1.5**, the others with a mesh **step of 2**.

Cube size L	# fractures	#triangles	Min area	Mean(Q)	Min(Q)	Time mm:ss	#tri wrt fine matching mesh
150	508,339	**10,461,407**	1.16e-11	0.69	1e-4	07:01	**-19%**
200	1,176,566	**18,648,084**	1.16e-11	0.59	7.8e-05	27:32	**-9%**

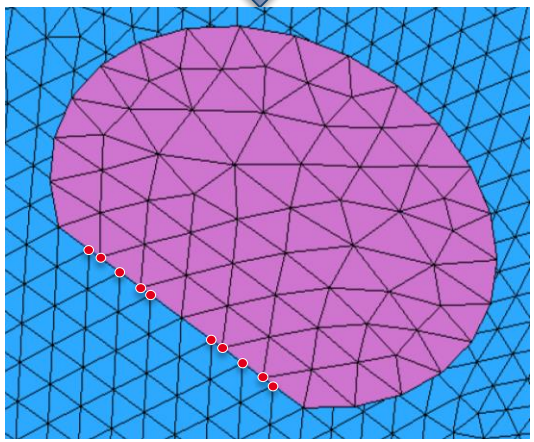
How to take advantage of polygonal cells?

Step (I5): create the polygons from the non-matching meshes

- **Development of a node insertion algorithm**

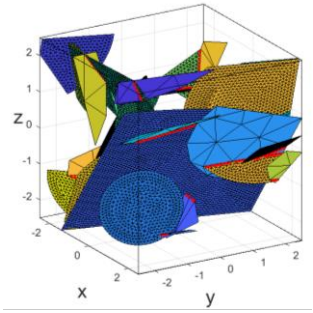


non-matching triangular mesh

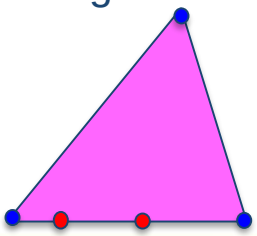


polygonal mesh

- **Algorithm to create the polygons:**
  1. Load the non-matching triangular mesh, especially the edges of each intersection on both side
  2. Define the linear coordinates of the vertices on the intersection to **find where to insert the vertices**
  3. Define a **new global numbering** of the intersection vertices (no duplicate indices!)
  4. Add the extra-vertices to **create the polygons** (red dots) and check the insertion in the connectivity table that defines the polygons in terms of vertices
  5. Compute the **2D coordinates** of extra vertices in the fractures they now belong to



- **Challenges**
  - Large number of intersections
  - Efficient algorithms

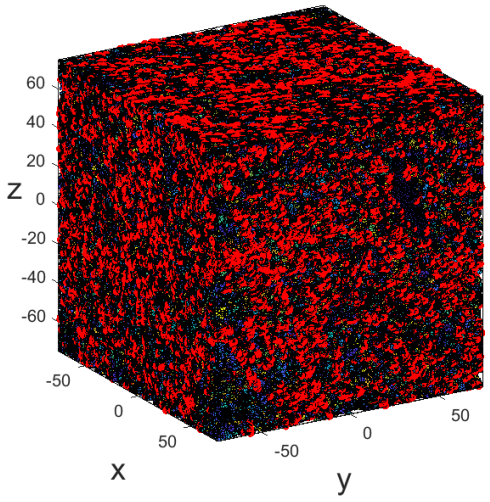


- Vertices of the triangles
- Extra vertices to create a polygon

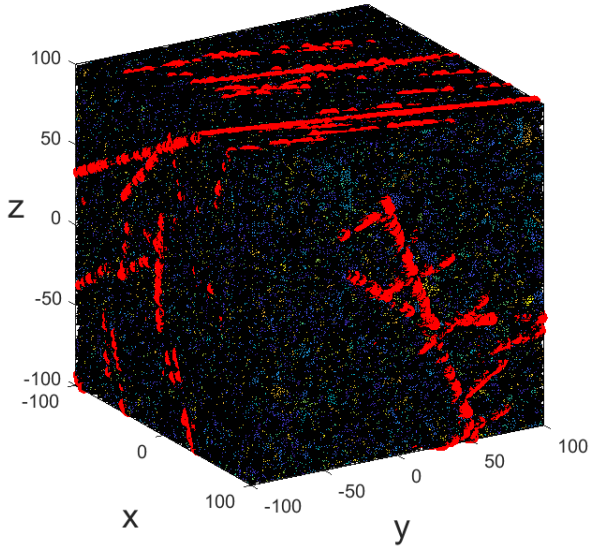
How to take advantage of polygonal cells?

Step (I5): create the polygons from the non-matching meshes

L	# fractures	#tri	#extra-vertices (red dots)	%polygons	#tri wrt fine matching mesh
150	508,339	10,461,407	201,583	3.5%	**-19%**
200	1,176,566	18,648,084	96,143	1.0%	**-9%**



L150
508,339 fractures



L200:
1,176,566 fractures

How to take advantage of polygonal cells?

Step (I6): perform the flow computation on the new polygonal meshes

L150: 508,339 fractures
10,461,407 triangles

k	#dofs	Total Time (hh:mm:ss)	RAM (GiB)	Rel. Error Kx
0	16,094,932	0:42:29	51	8.7%
1	32,189,864	1:34:38	121	2.3%
2	48,284,796	2:51:21	238	0.9%
3	64,379,728	4:01:37	392	0.4%
4	80,474,660	5:54:06	593	0.2%

Comparison with the fine mesh flow simulation
-15% time
-12% to -15% RAM

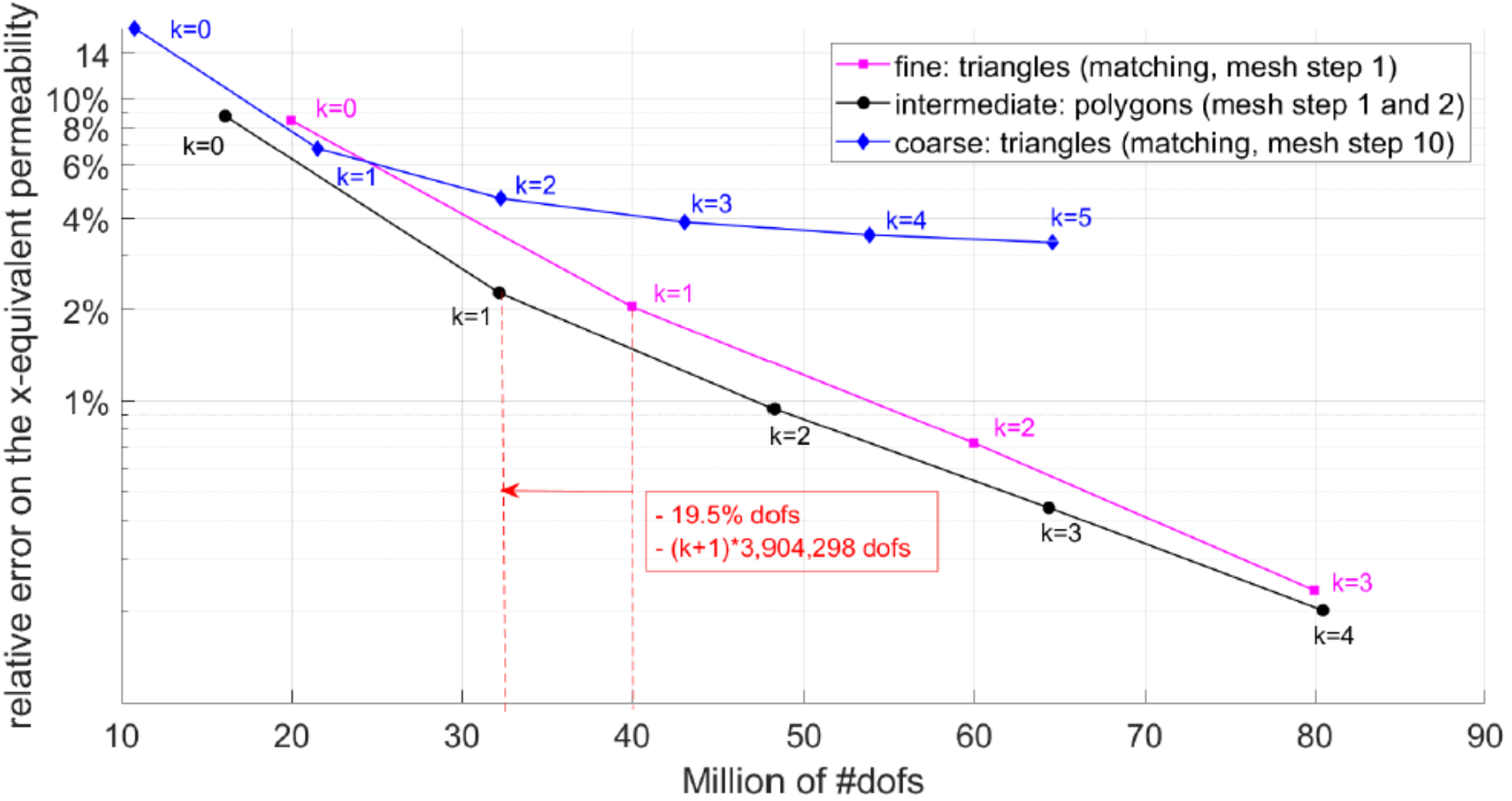
L200: 1,176,566 fractures
18,648,084 triangles

k	#dofs	Total Time (hh:mm:ss)	RAM (GiB)	Rel. Error Kx
0	28,685,198	01:18:01	93	11.7%
1	57,370,396	03:12:49	223	3.3%
2	86,055,594	05:24:48	435	1.5%
3	114,740,792	08:16:42	732	0.8%
4	143,425,990	12:10:58	1087	0.5%

Comparison with the fine mesh flow simulation
-12% to -16% time
-7% to -10% RAM

How to take advantage of polygonal cells?

Step (I6): perform the flow computation on the new polygonal meshes



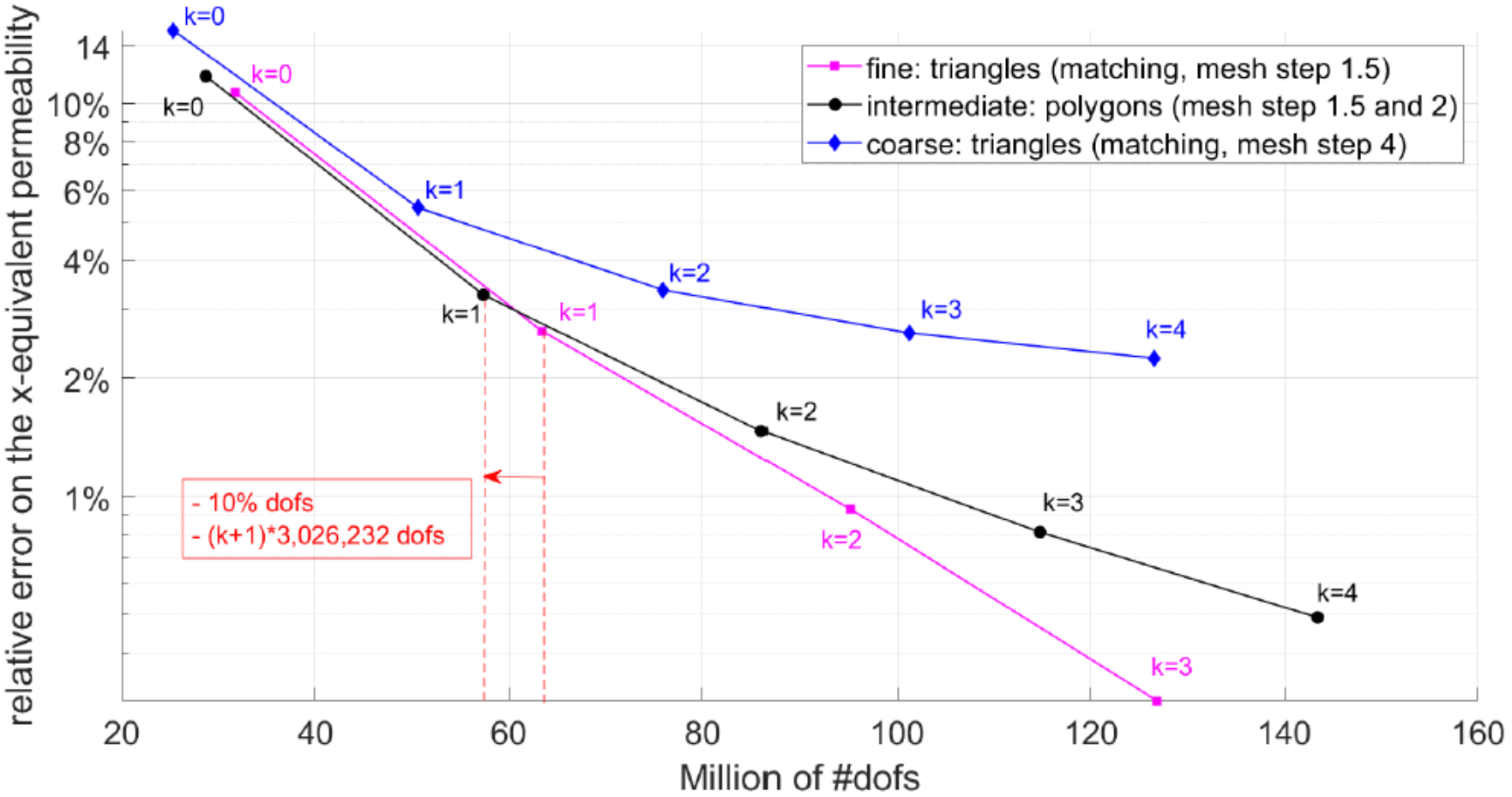
Initial coarse mesh:
7,029,255 triangles

Intermediate mesh:
10,461,407 triangles

Initial fine mesh:
12,890,943 triangles

How to take advantage of polygonal cells?

Step (I6): perform the flow computation on the new polygonal meshes



Initial coarse mesh:
16,554,413 triangles

Intermediate mesh:
18,648,084 triangles

Initial fine mesh:
20,522,575 triangles

Conlusion

- We have **successfully** tested the **HHO method on large scale DFN** (more than **1 million of fractures**)

- The implementation of the method is **locally and globally conservative** (whatever k ≥0) thanks to the use of basis functions based on **hierarchical scaled centered and rotated monomials**

- The use of high order is an advantage to compute a **more accurate** flow in DFN **with the following recommendations:**
  - ✓ use a face polynomial order k at least equal to 1, instead of refining a mesh while keeping $k = 0$
  - ✓ choose $k = 1$ on a mesh fine rather than $k > 1$ on a coarser mesh

- We have shown that the use of non-matching meshes, the channelling effect of DFN flow and the polygonal feature of the **HHO method allows to reduce the number of #dofs and therefore the RAM requirements** in DFN flow simulations. In terms of **computational time however**, **so far**, the polygonal strategy is not the most relevant according to the extra time spent in the steps to know which fractures to refine/coarsen and to build the polygonal mesh from the non-matching triangular one.

Perpectives

- Further reduction of the #dofs by using the **additional recent features of HHO**
  - ✓ **Unfitted HHO**
  - ✓ Possibility to **merge the triangles** in (possibly non convex) **polygons**

- A **C-version** of the **node insertion algorithm**

- Studying **an iterative solver to reduce the RAM requirements**

- Studying other **algorithms to know which fractures to refine :**
  - ✓ From **a posteriori error** estimates
  - ✓ From **graph connectivity of the fracture network** and BCs. (no need any flow computation)

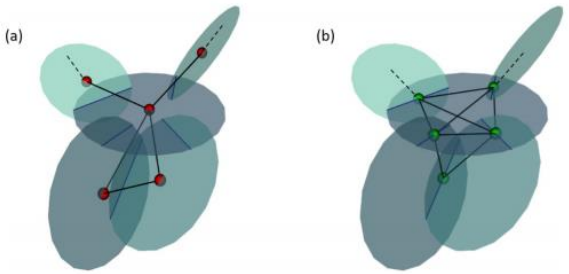


FIG. 1. DFNs and equivalent graphs: (a) fracture graph, (b) intersection graph.

©From Doolaeghe et al. 2020

D. Doolaeghe & P. Davy & J. Hyman & C. Darcel. Graph-based flow modeling approach adapted to multiscale discrete fracture network models (2020), Physical review E, 102(5), 053312.

E. Burman and A. Ern, An unfitted hybrid high-order method for elliptic interface problems, SIAM J. Numer. Anal., 56(3), 1525-1546 (2018)

A. Miraçi, J. Papež, and M. Vohralík (2020)
A Multilevel Algebraic Error Estimator and the Corresponding Iterative Solver with *p*-Robust Behavior. SIAM J. Numer. Anal., 58(5), 2856–2884.

M.Vohralík (2007). A Posteriori Error Estimates for Lowest-Order Mixed Finite Element Discretizations of Convection-Diffusion-Reaction Equations. SIAM Journal on Numerical Analysis. 45.

S. Karra, D. O'Malley, J. Hyman, H. Viswanathan and G. Srinivasan (2017). Modeling flow and transport in fracture networks using graphs. Physical Review E. 97.

Thanks a lot for your attention !