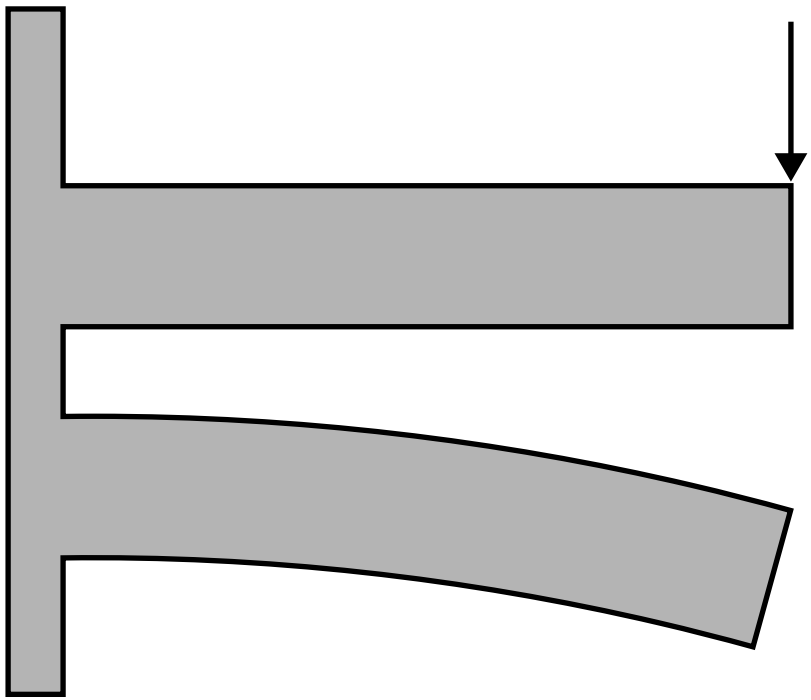
LEPL1110 - Elements finis

A la découverte de la géométrie

Michel Henry, Thomas Leyssens,
Vincent Legat

24 Mars 2023

Une nouvelle année, un nouveau projet...



Elasticité linéaire,

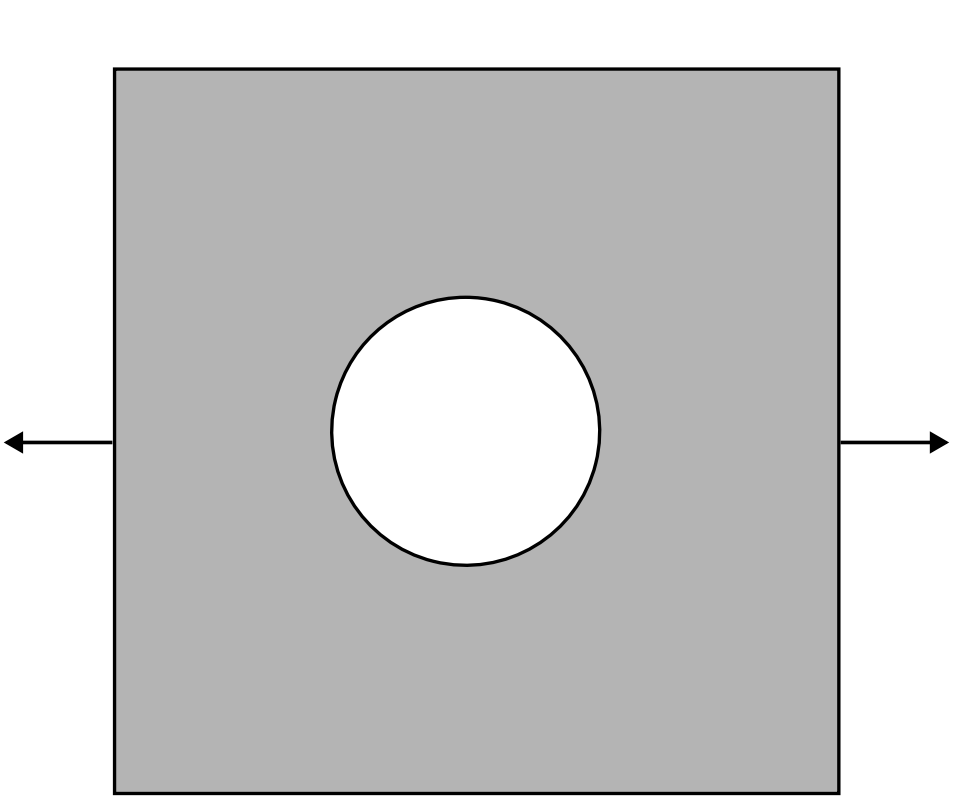
$$\frac{\mathrm{d}\sigma}{\mathrm{d}x} + \rho g = 0$$

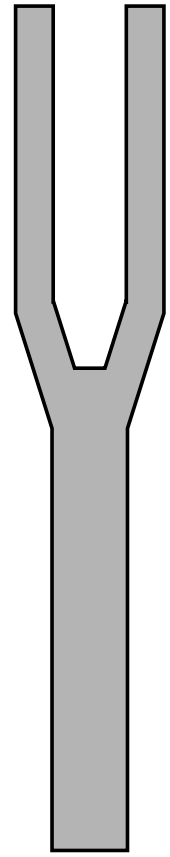
Faibles déformations,

$$\sigma = E\frac{\mathrm{d}u}{\mathrm{d}x}$$

... avec la pièce mécanique de votre choix

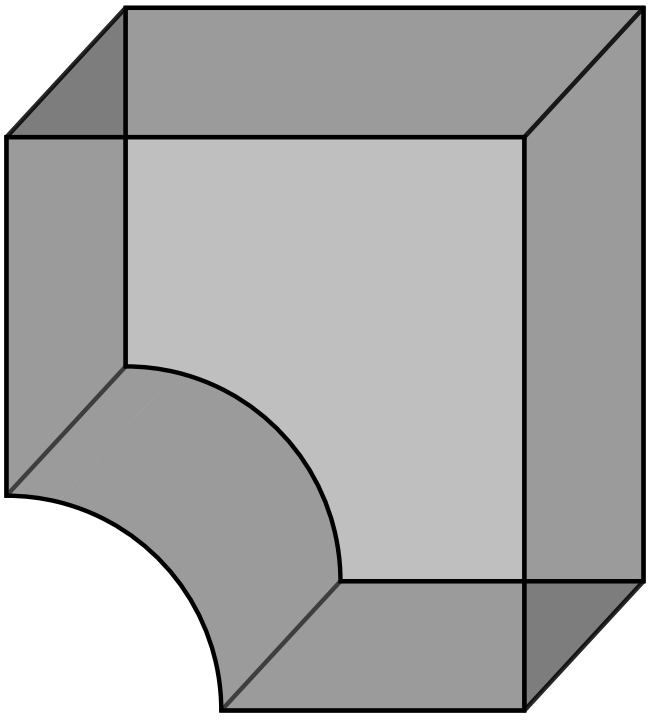
Un code FEM classique...





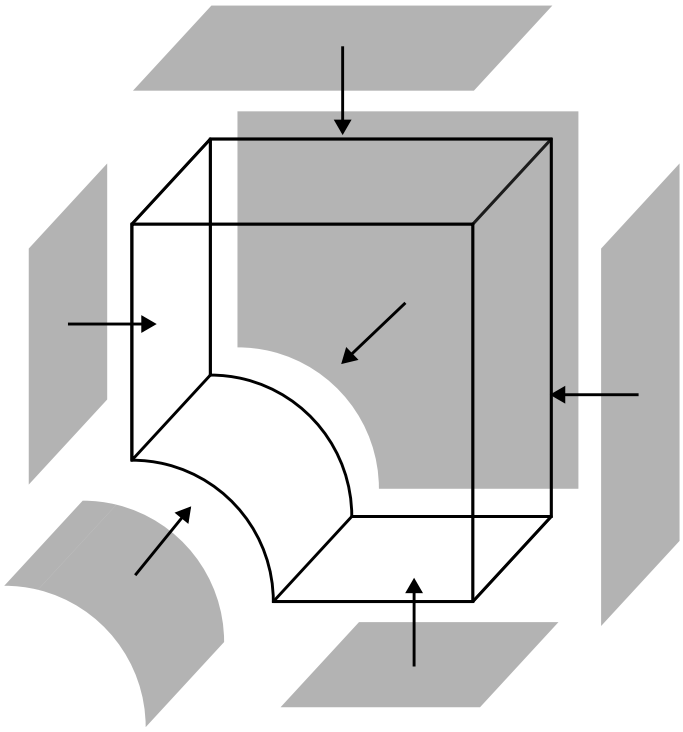
... épatez nous par votre créativité

La géométrie est une structure hiérachique...



Construction abstraite :

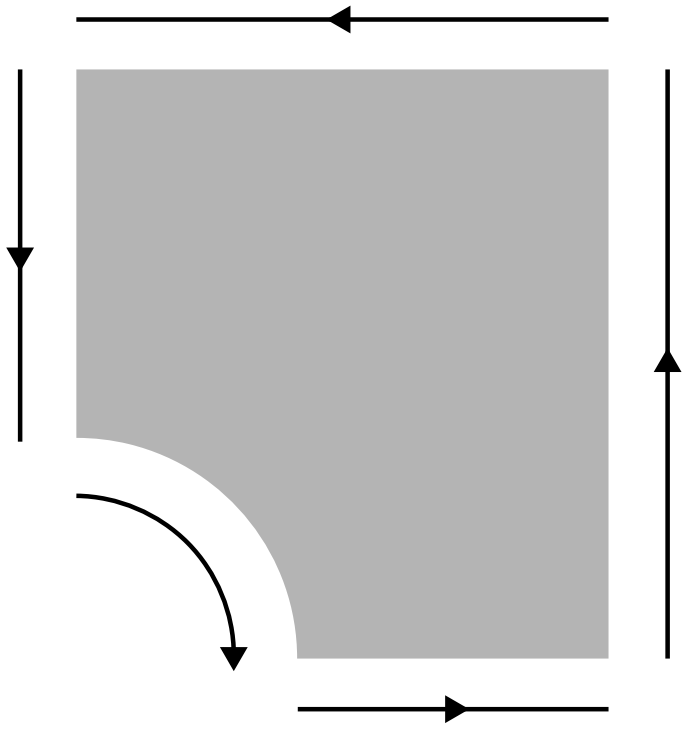
La géométrie est une structure hiérachique...



Construction abstraite :

$\mathcal{V}$: volume décrit par ses surfaces

La géométrie est une structure hiérachique...

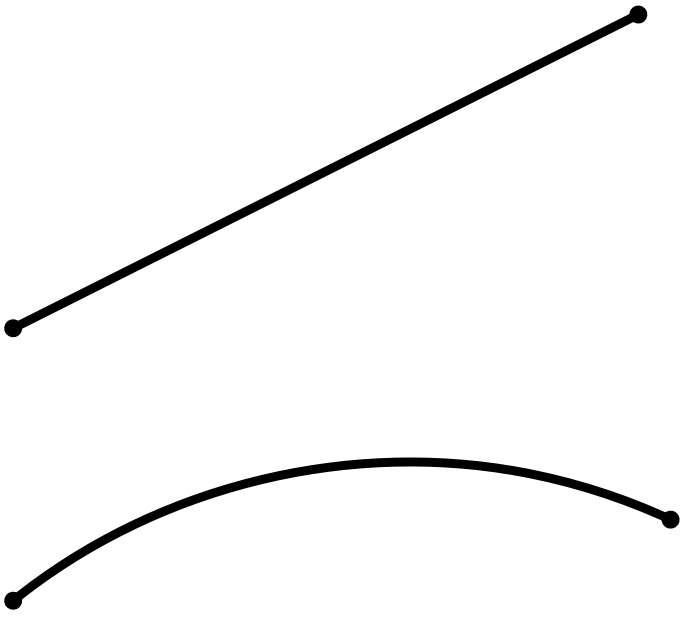


Construction abstraite :

$\mathcal{V}$: volume décrit par ses surfaces

$\mathcal{S}$: surface décrit par ses arêtes

La géométrie est une structure hiérachique...



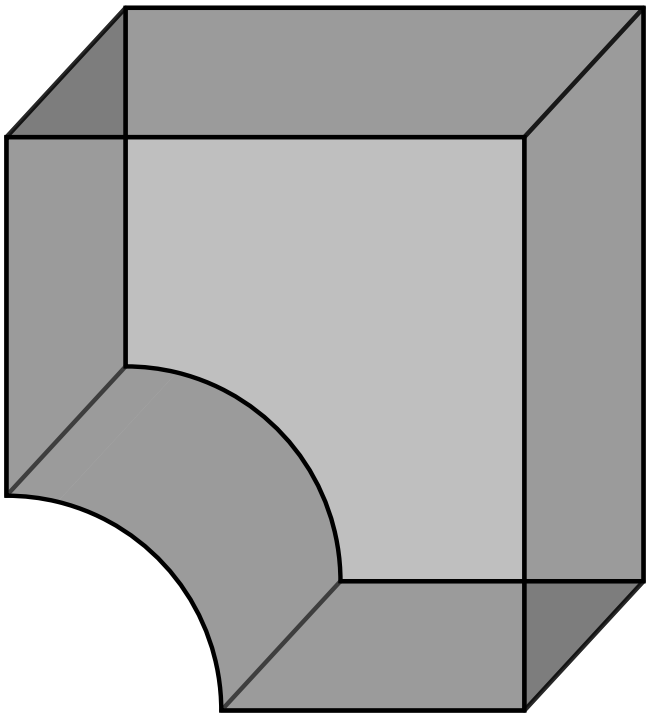
Construction abstraite :

$\mathcal{V}$: volume décrit par ses surfaces

$\mathcal{S}$: surface décrit par ses arêtes

$\mathcal{A}$: arête décrit par ses noeuds

La géométrie est une structure hiérachique...



Construction abstraite :

$\mathcal{V}$: volume décrit par ses surfaces

$\mathcal{S}$: surface décrit par ses arêtes

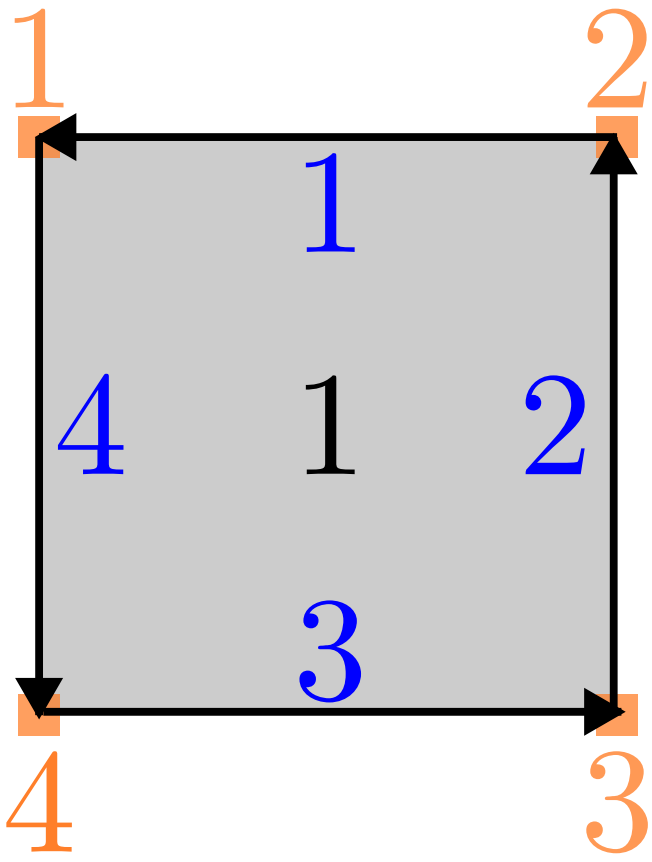
$\mathcal{A}$: arête décrit par ses noeuds

$\mathcal{N}$: noeud décrit par un indice

... décrite par une connectivité

Construisons une géométrie simple

Chaque entité géométrique est représentée par une paire (dimension, tag)



Dim	Tag	Dim	Tag	Dim	Tag

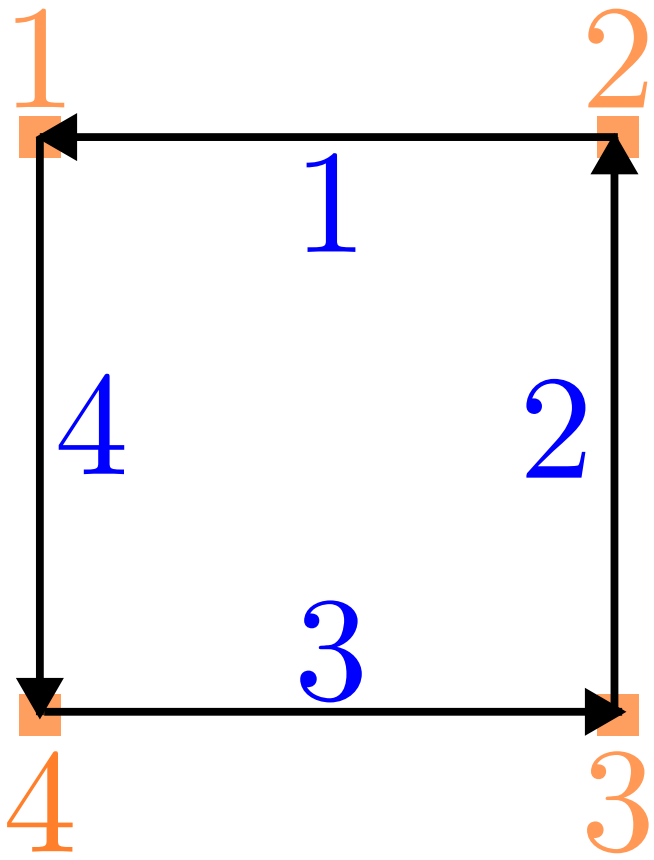
Construisons une géométrie simple

Chaque entité géométrique est représentée par une paire (dimension, tag)

Dim	Tag	Dim	Tag	Dim	Tag
0	1				
0	2				
0	3				
0	4				

Construisons une géométrie simple

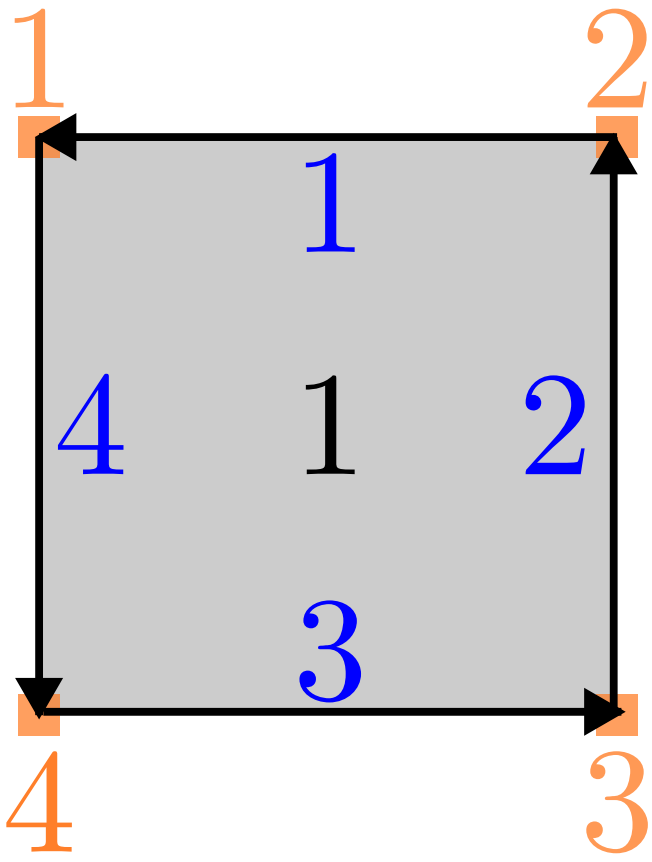
Chaque entité géométrique est représentée par une paire (dimension, tag)



Dim	Tag	Dim	Tag	Dim	Tag
0	1	1	1		
0	2	1	2		
0	3	1	3		
0	4	1	4		

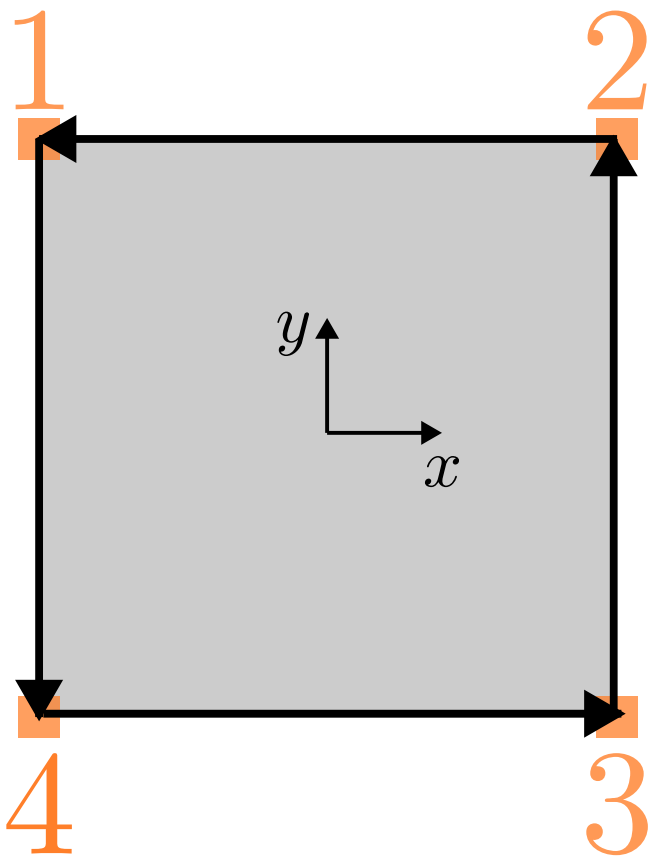
Construisons une géométrie simple

Chaque entité géométrique est représentée par une paire (dimension, tag)



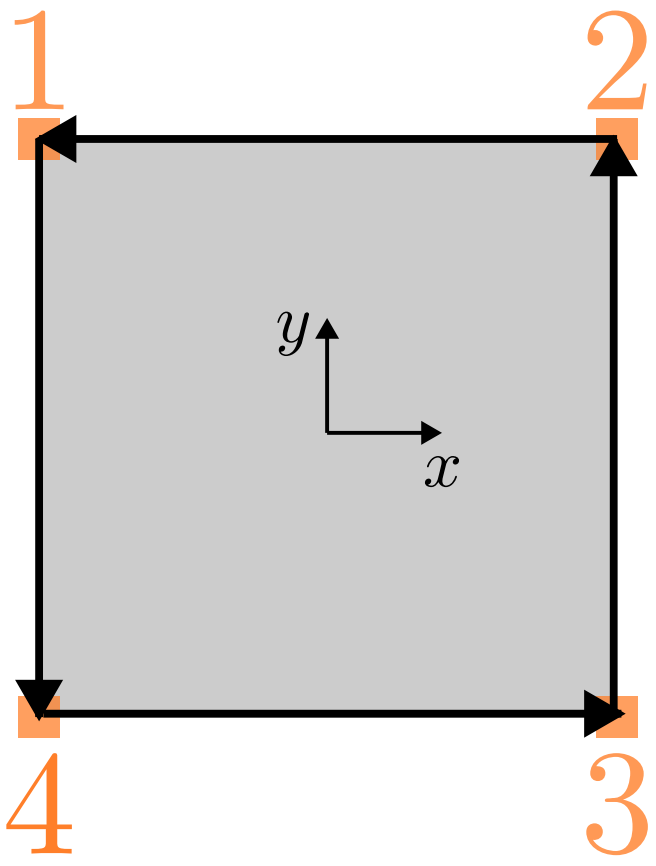
Dim	Tag	Dim	Tag	Dim	Tag
0	1	1	1	2	1
0	2	1	2		
0	3	1	3		
0	4	1	4		

Lions la representation à un système d'axes, ...



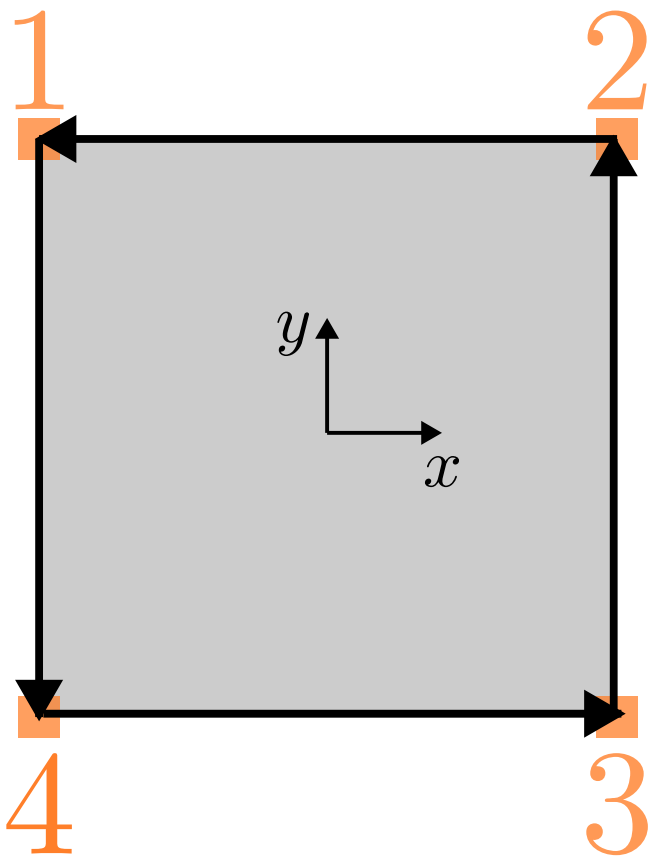
Dim	Tag	x	y
0	1	−1.0	1.0

Lions la representation à un système d'axes, ...



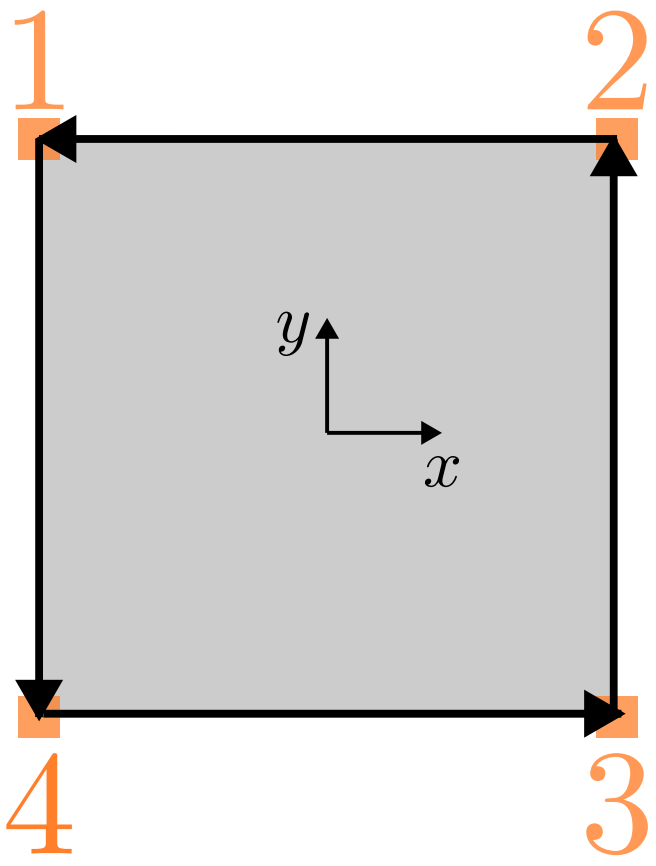
Dim	Tag	x	y
0	1	−1.0	1.0
0	2	1.0	1.0

Lions la representation à un système d'axes, ...



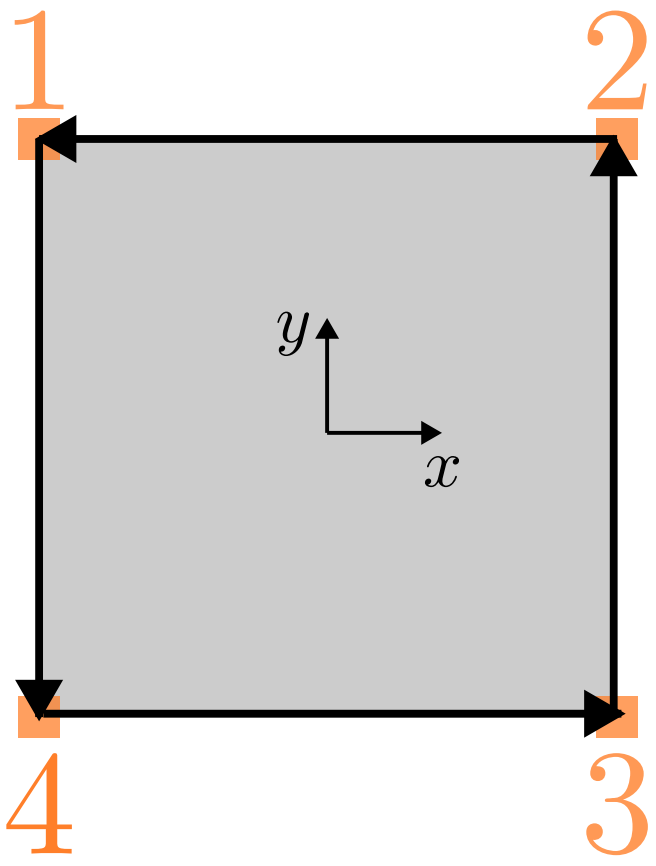
Dim	Tag	x	y
0	1	−1.0	1.0
0	2	1.0	1.0
0	3	1.0	−1.0

Lions la representation à un système d'axes, ...



Dim	Tag	x	y
0	1	−1.0	1.0
0	2	1.0	1.0
0	3	1.0	−1.0
0	4	−1.0	−1.0

Lions la representation à un système d'axes, ...

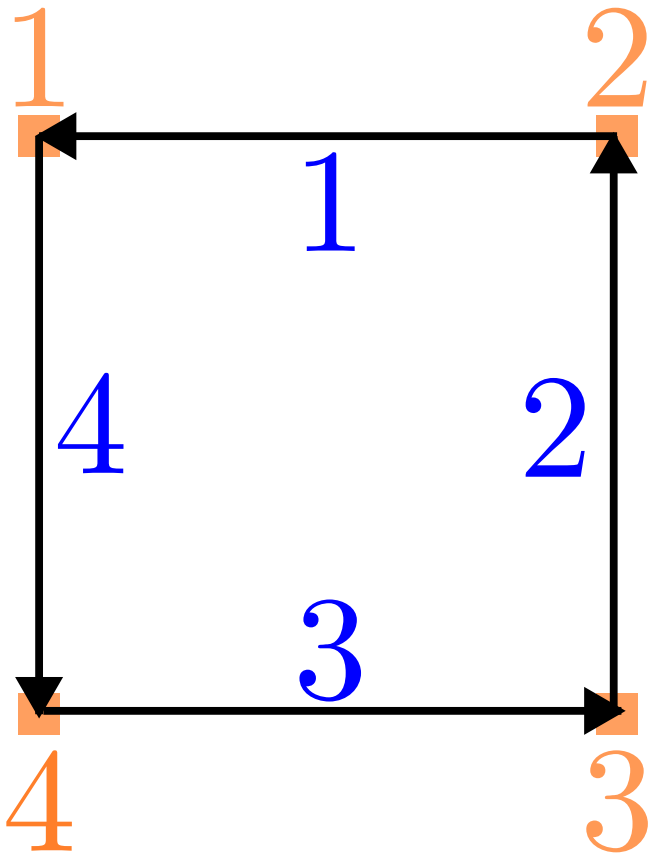


Dim	Tag	x	y
0	1	−1.0	1.0
0	2	1.0	1.0
0	3	1.0	−1.0
0	4	−1.0	−1.0

Seuls les noeuds ont des coordonnées

Une collection d'entités forme un groupe physique

Il est utile d'associer un ensemble d'entité géométrique à un groupe physique

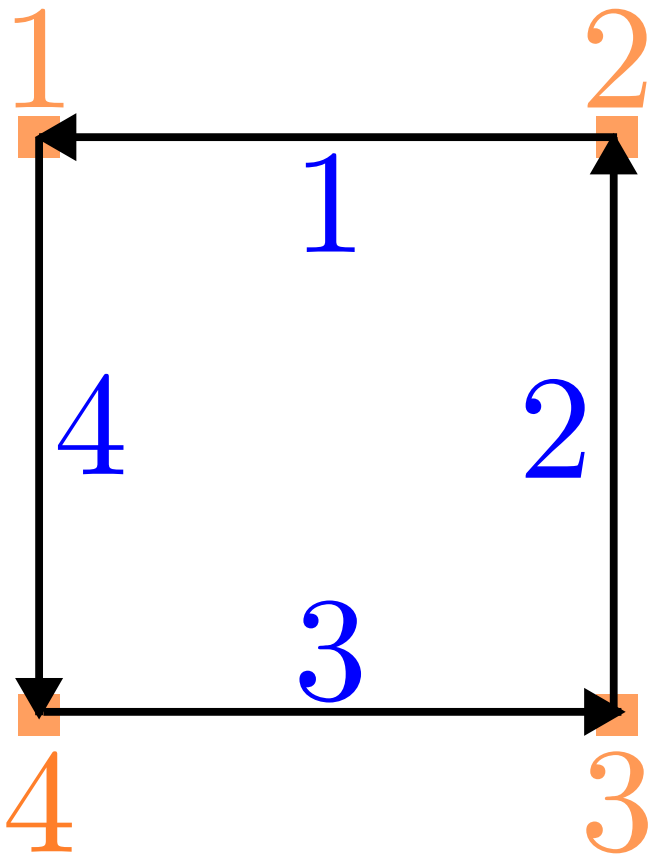


Dim	Tag	Nom
1	1	Top

Nom	Dim	Tag
Top	1	1

Une collection d'entités forme un groupe physique

Il est utile d'associer un ensemble d'entité géométrique à un groupe physique

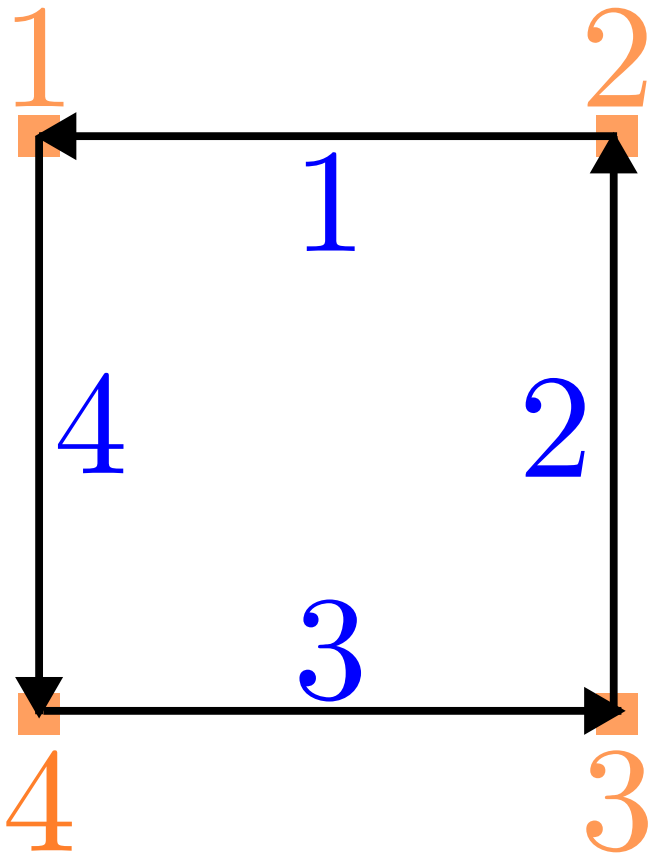


Dim	Tag	Nom
1	1	Top
1	2	Lateral

Nom	Dim	Tag
Top	1	1
Lateral	1	2

Une collection d'entités forme un groupe physique

Il est utile d'associer un ensemble d'entité géométrique à un groupe physique

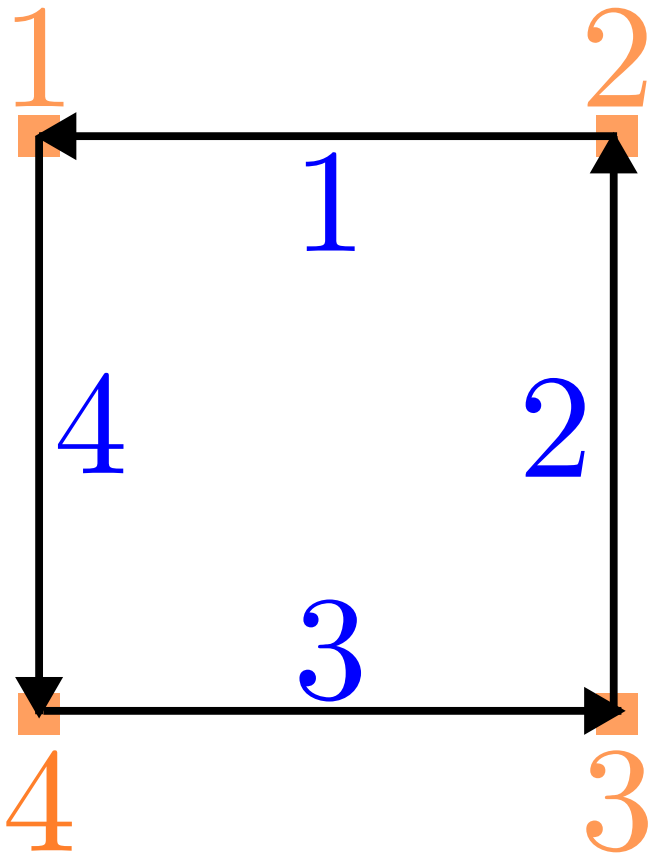


Dim	Tag	Nom
1	1	Top
1	2	Lateral
1	3	Bottom

Nom	Dim	Tag
Top	1	1
Lateral	1	2
Bottom	1	3

Une collection d'entités forme un groupe physique

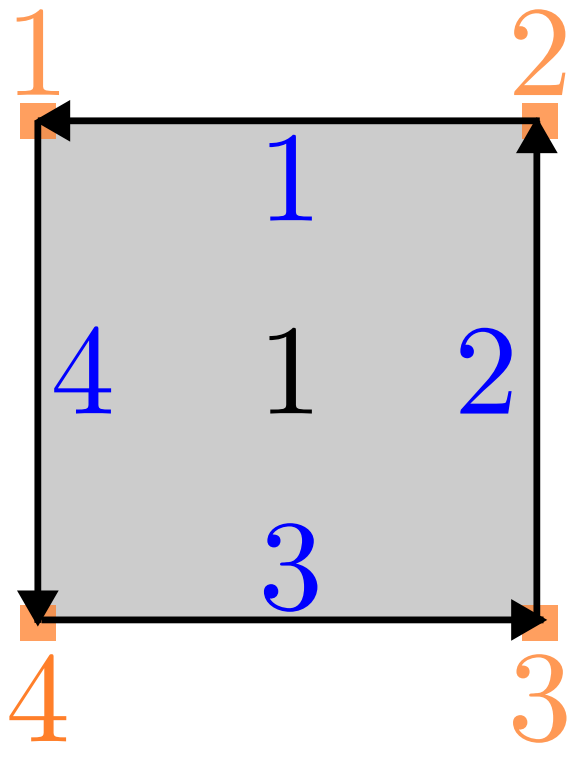
Il est utile d'associer un ensemble d'entité géométrique à un groupe physique



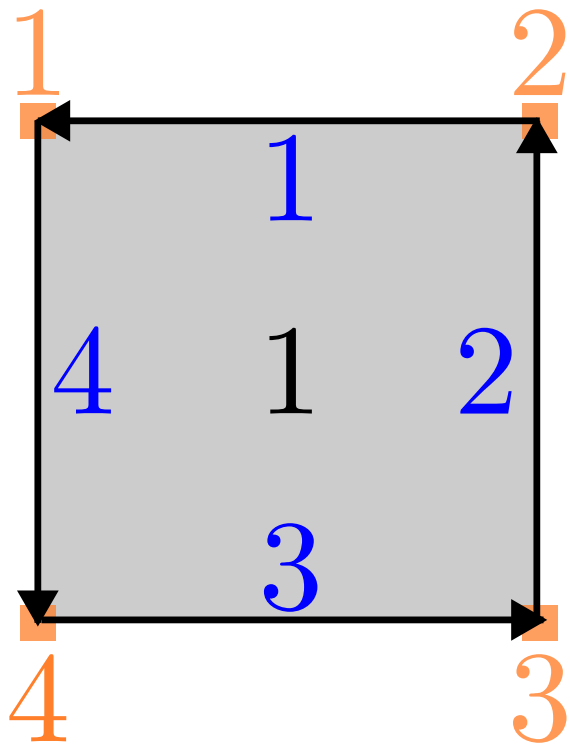
Dim	Tag	Nom
1	1	Top
1	2	Lateral
1	3	Bottom
1	4	Lateral

Nom	Dim	Tag
Top	1	1
Lateral	1	2, 4
Bottom	1	3

Trois tableaux décrivent notre géométrie

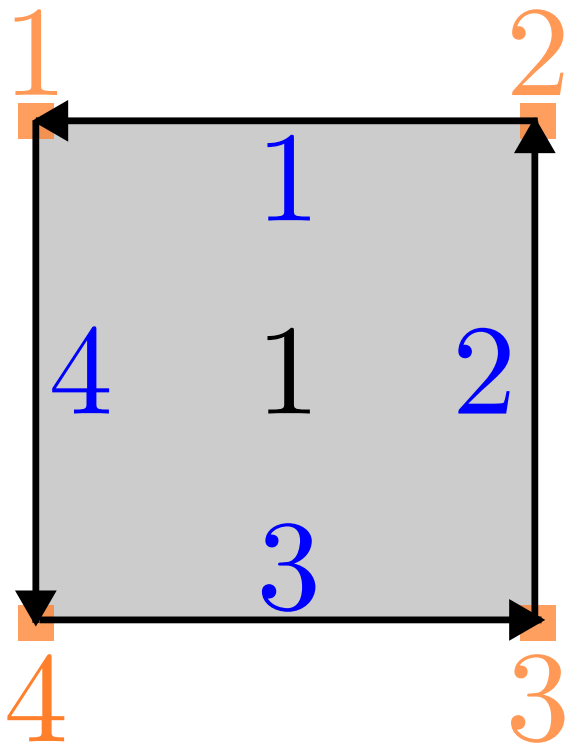


Trois tableaux décrivent notre géométrie



Dim	Tag	Dim	Tag	Dim	Tag
0	1	1	1	2	1
0	2	1	2		
0	3	1	3		
0	4	1	4		

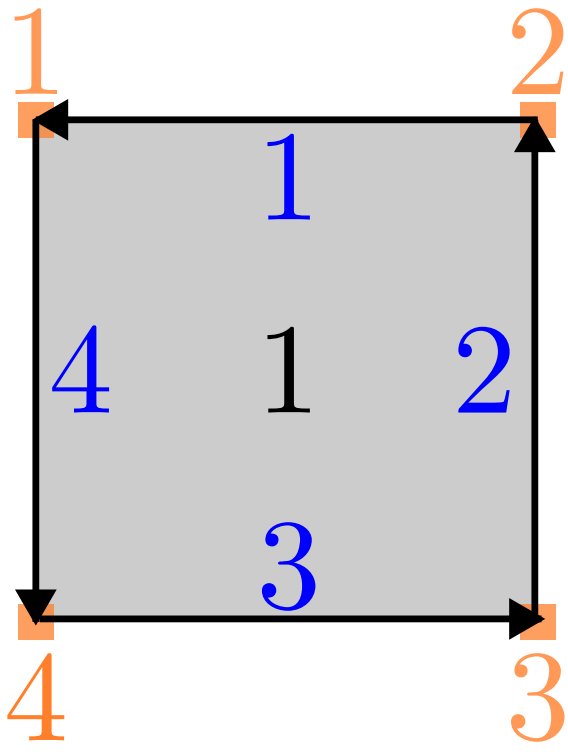
Trois tableaux décrivent notre géométrie



Dim	Tag	x	y
0	1	−1.0	1.0
0	2	1.0	1.0
0	3	1.0	−1.0
0	4	−1.0	−1.0

Dim	Tag	Dim	Tag	Dim	Tag
0	1	1	1	2	1
0	2	1	2		
0	3	1	3		
0	4	1	4		

Trois tableaux décrivent notre géométrie

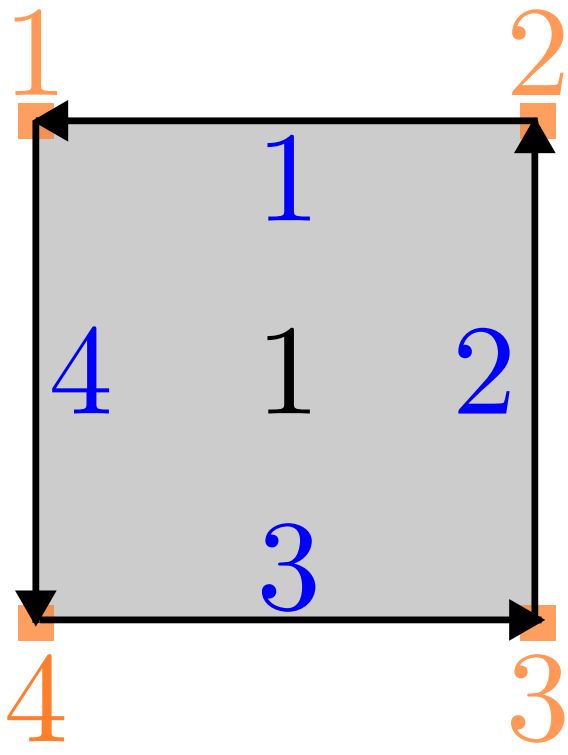


Dim	Tag	x	y
0	1	−1.0	1.0
0	2	1.0	1.0
0	3	1.0	−1.0
0	4	−1.0	−1.0

Dim	Tag	Dim	Tag	Dim	Tag
0	1	1	1	2	1
0	2	1	2		
0	3	1	3		
0	4	1	4		

Nom	Dim	Tag
Top	1	1
Lateral	1	2, 4
Bottom	1	3

Trois tableaux décrivent notre géométrie

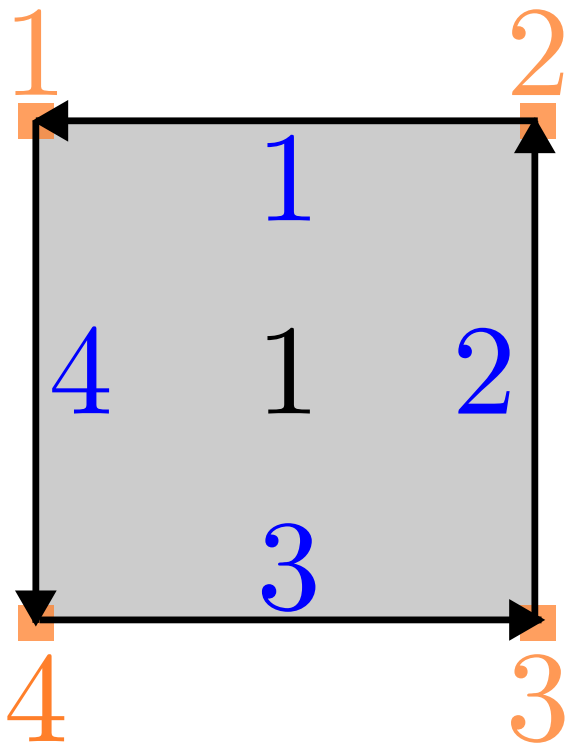


Dim	Tag	x	y
0	1	−1.0	1.0
0	2	1.0	1.0
0	3	1.0	−1.0
0	4	−1.0	−1.0

```c
typedef struct {
    int nLocalNode;
    int nElem;
    int *elem;
    femNodes *nodes;
    char name[256];
} femMesh;
```

Nom	Dim	Tag
Top	1	1
Lateral	1	2, 4
Bottom	1	3

Trois tableaux décrivent notre géométrie

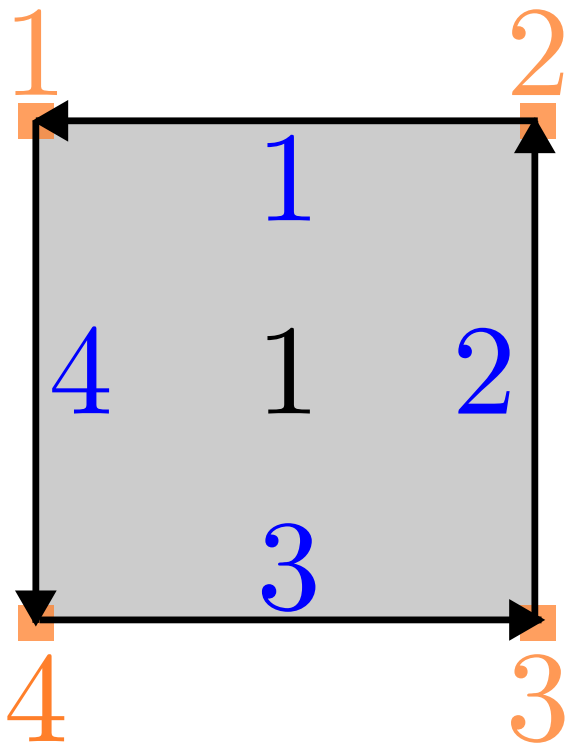



```c
typedef struct {
    int nNodes;
    double *X;
    double *Y;
} femNodes;
```

```c
typedef struct {
    int nLocalNode;
    int nElem;
    int *elem;
    femNodes *nodes;
    char name[256];
} femMesh;
```

Nom	Dim	Tag
Top	1	1
Lateral	1	2, 4
Bottom	1	3

Trois tableaux décrivent notre géométrie

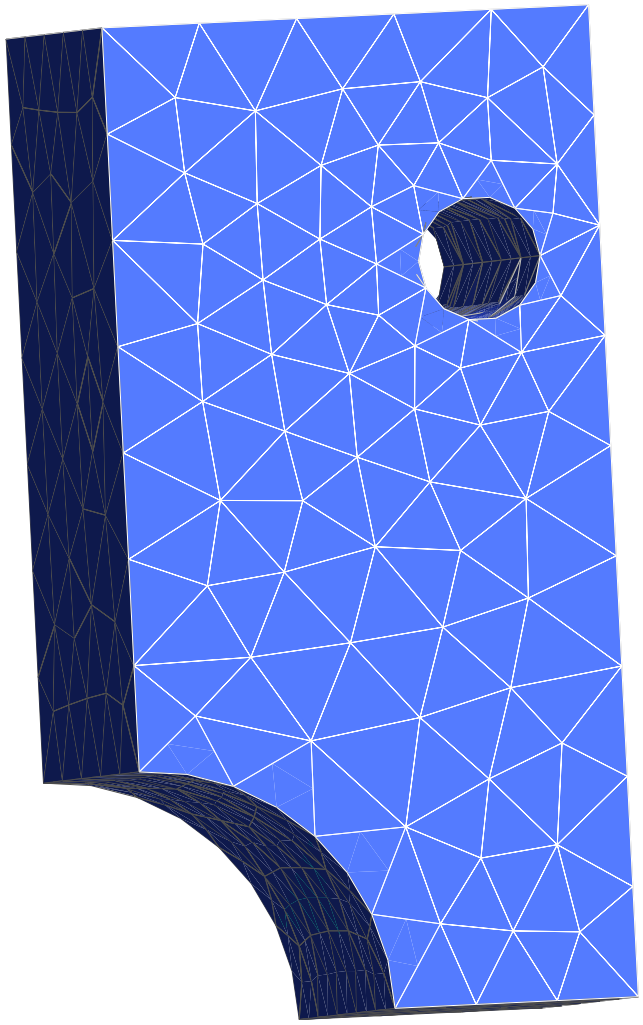



```c
typedef struct {
    int nNodes;
    double *X;
    double *Y;
} femNodes;
```

```c
typedef struct {
    int nLocalNode;
    int nElem;
    int *elem;
    femNodes *nodes;
    char name[256];
} femMesh;
```

```c
typedef struct {
    femMesh *mesh;
    int nElem;
    int *elem;
    char name[256];
} femDomain;
```

Obtenons un maillage sur notre géométrie

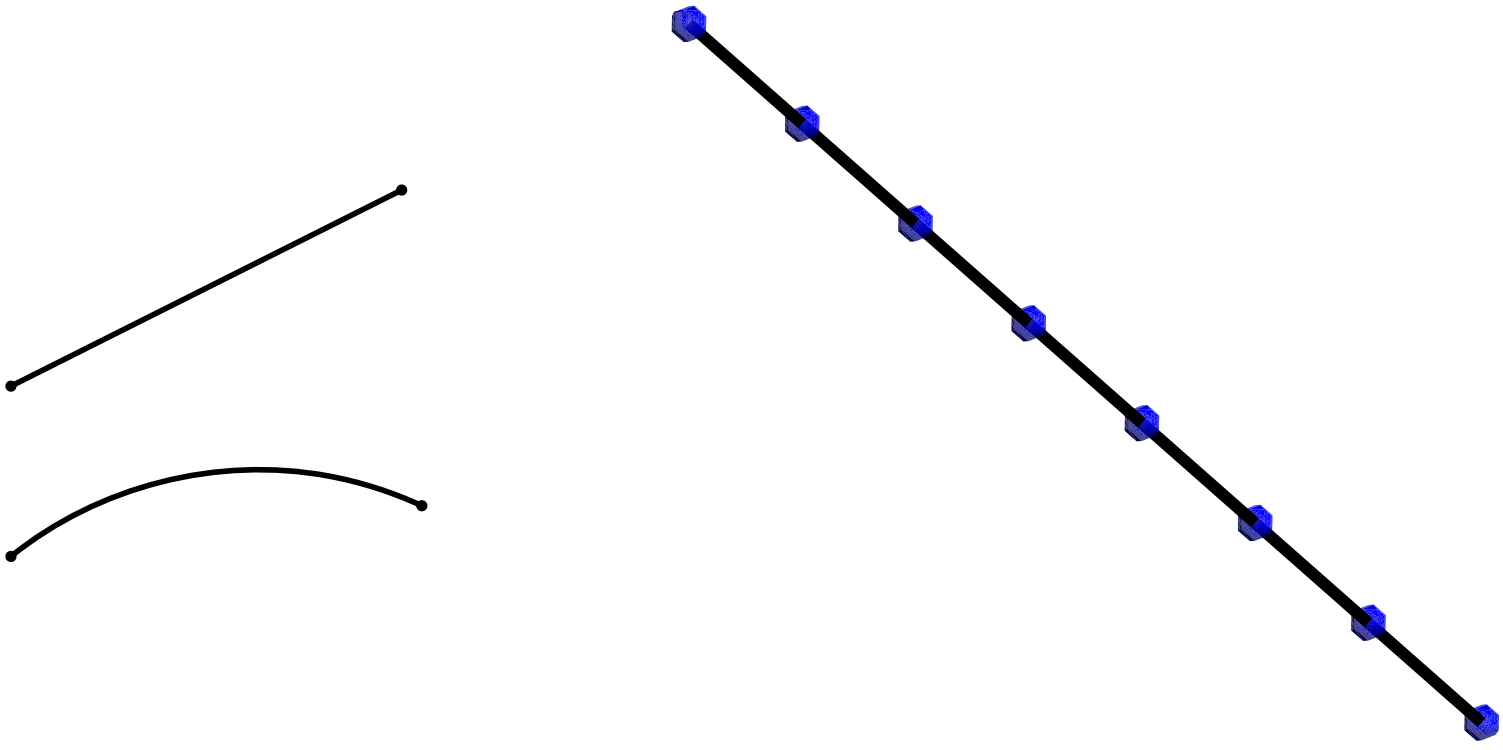


Un maillage doit

- être conforme à la géométrie
- avoir des éléments de qualité
- respecter les contraintes de taille

La construction est hiérachique.

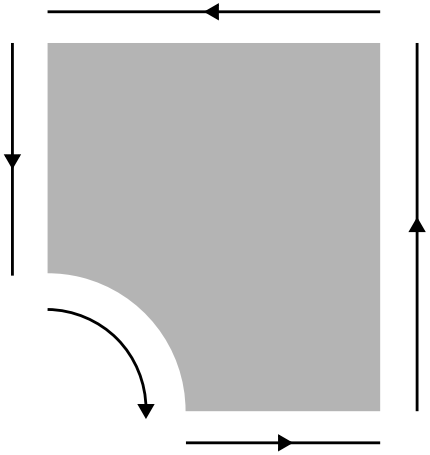
Types d'éléments : 1D

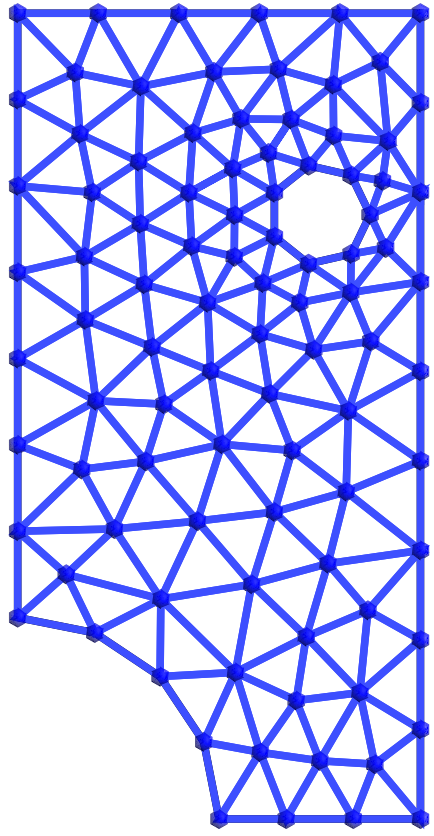


Les éléments 1D sont des segments, délimités par 2 noeuds

Types d'éléments : 2D

Triangles

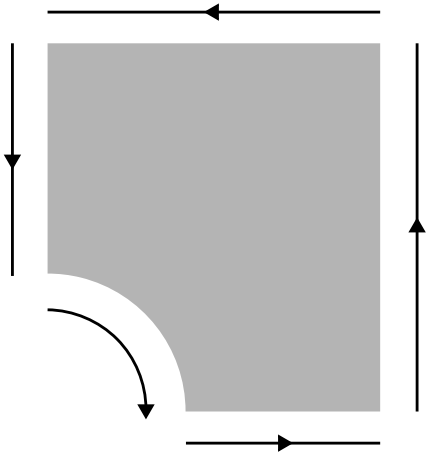


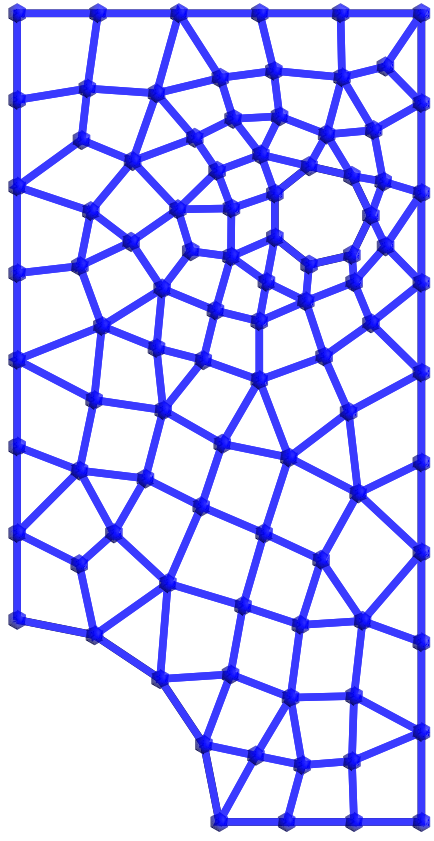


Les éléments 2D sont des triangles ou des quadrilatères, délimités par 3 ou 4 noeuds

Types d'éléments : 2D

Triangles & Quadrilatères

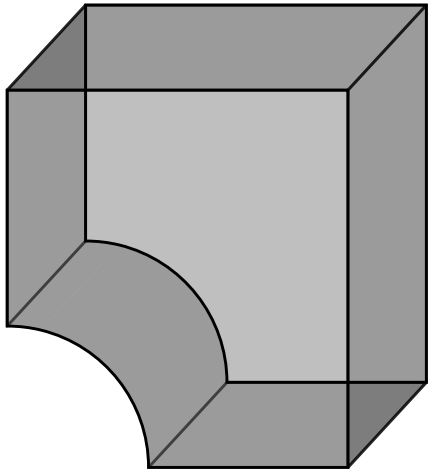


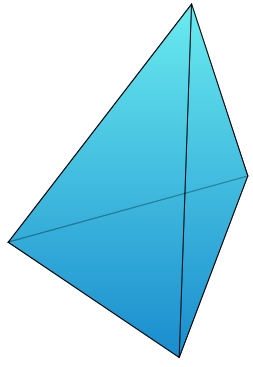


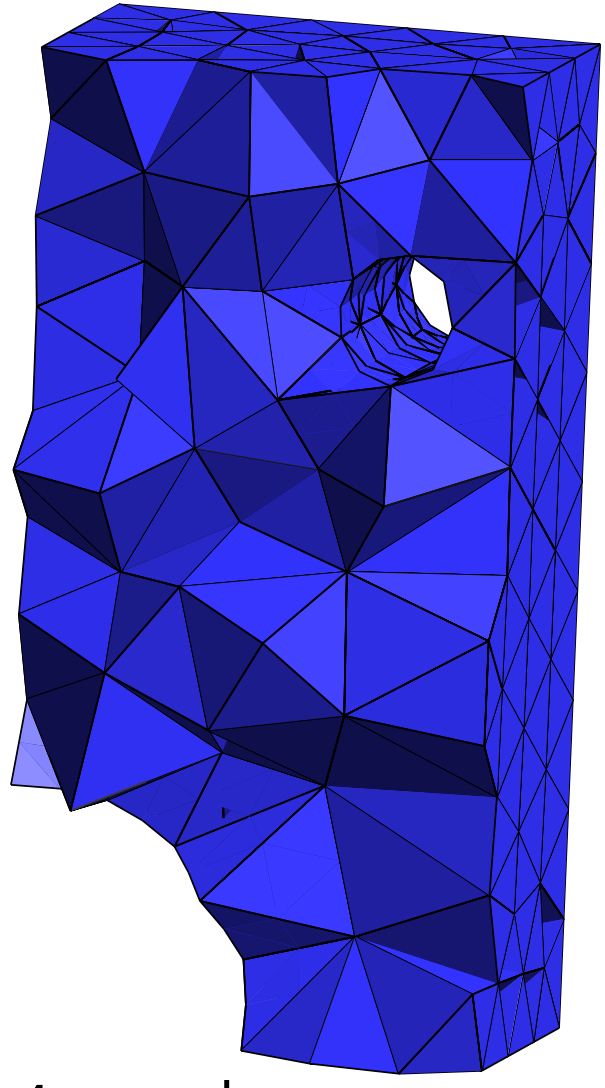
Les éléments 2D sont des triangles ou des quadrilatères, délimités par 3 ou 4 noeuds

Types d'éléments : 3D

Tétrahèdres







Les tétrahèdres sont délimités par 4 noeuds

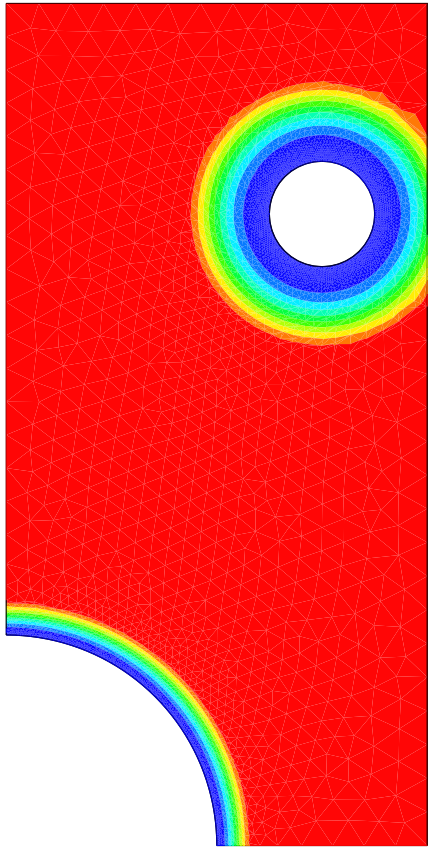
Carte de taille

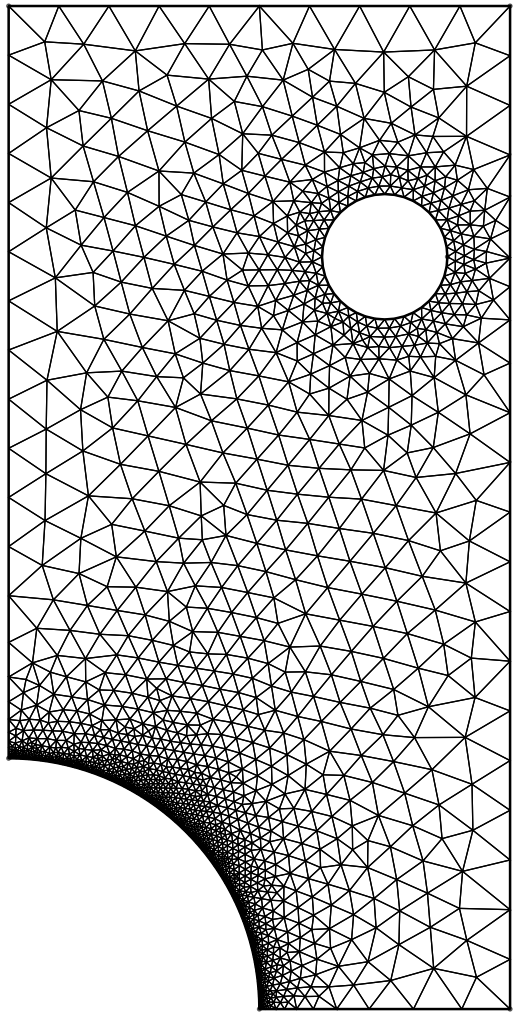
Comment définir la taille des éléments dans le maillage ?

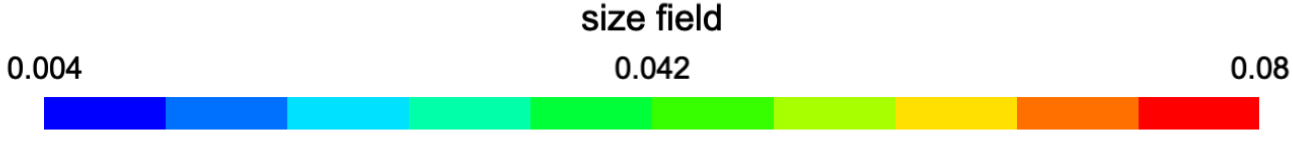
En un point (x, y) *de mon domaine, je veux des éléments de taille* $f(x, y)$.
Plus de raffinement dans les régions où l'on souhaite plus de précision.

$$f : \mathbb{R}^d \to \mathbb{R}$$

Carte de taille



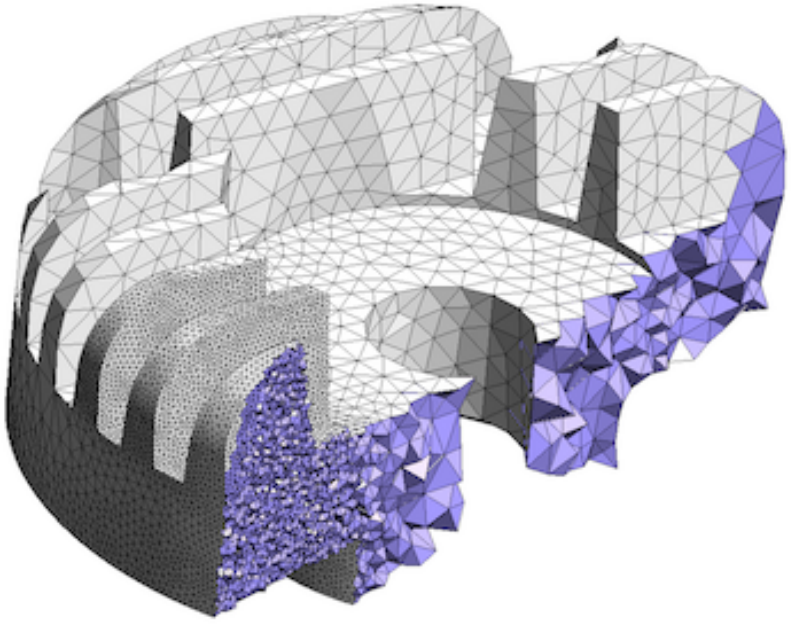




Découvrons GMSH...

Qu'est ce que GMSH ?

- **Générateur de maillage open-source**
- Un solveur éléments finis
- Un logiciel de post-traitement



API disponibles en :

C, C++, Python, Julia,

...pour mailler la géométrie de vos rêves

Comment construire votre géométrie ?

```c
typedef struct {
    double width;
    double height;
    double meshSize;
} femGeo;

extern femGeo *theGeometry;
```

- Initialiser la géométrie

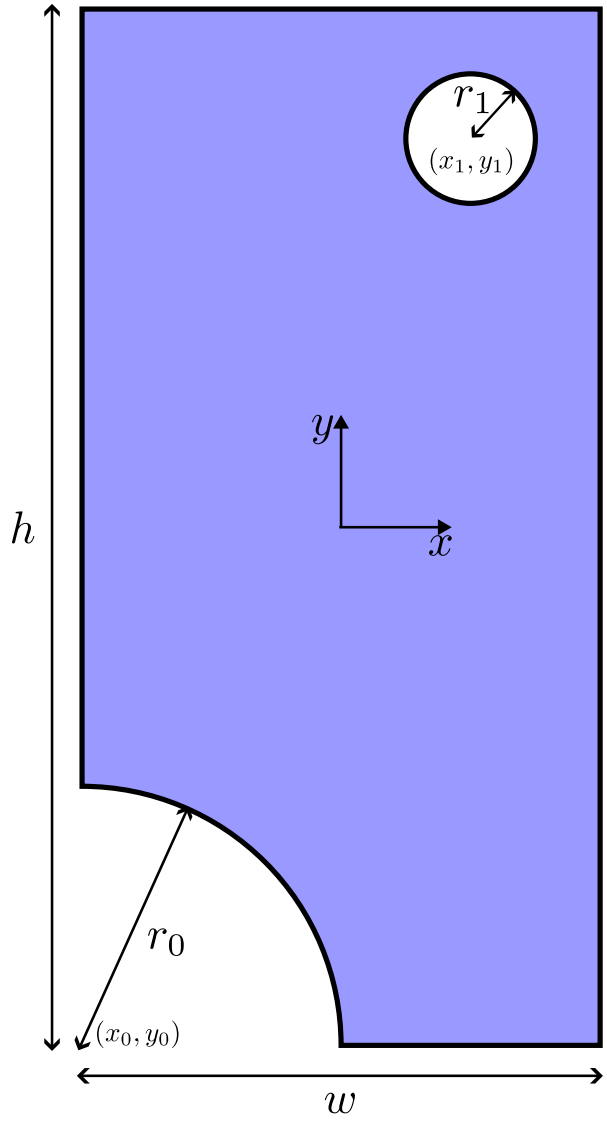
```c
double w = 1.0;
double h = 2.0;
double meshSize = 0.1;
double x0 = -w/2, y0 = -h/2, r0 = w/2, lc0 = 0.05;
double x1 =  w/4, y1 =  h/4, r1 = w/8, lc1 = 0.20;
theGeometry = geoCreate(w, h, meshSize);
```

- Créer la géométrie

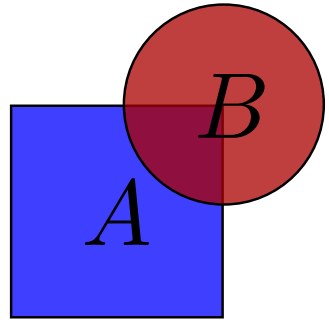
```c
geoGenerate(theGeometry, x0, y0, r0, lc0, x1, y1, r1, lc1);
```

- Extraire les connectivités

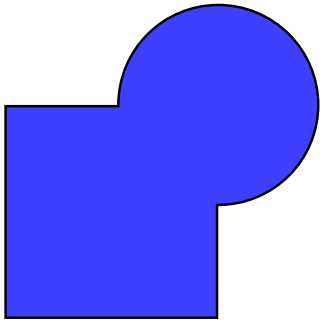
```c
femNodes *theNodes = femNodesCreate(theGeometry);
femMeshes* theMeshes = femMeshesCreate(theGeometry, theNodes);
```



A l'attaque des géométries complexes

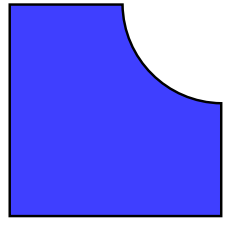


$A \cup B$



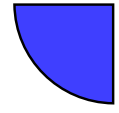
Union

$A - B$



Différence

$A \cap B$



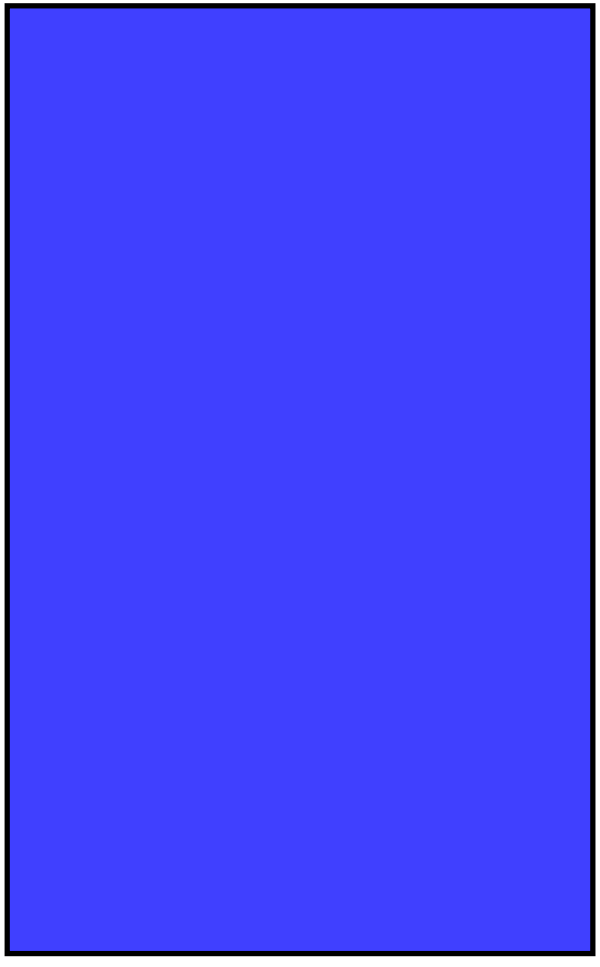
Intersection

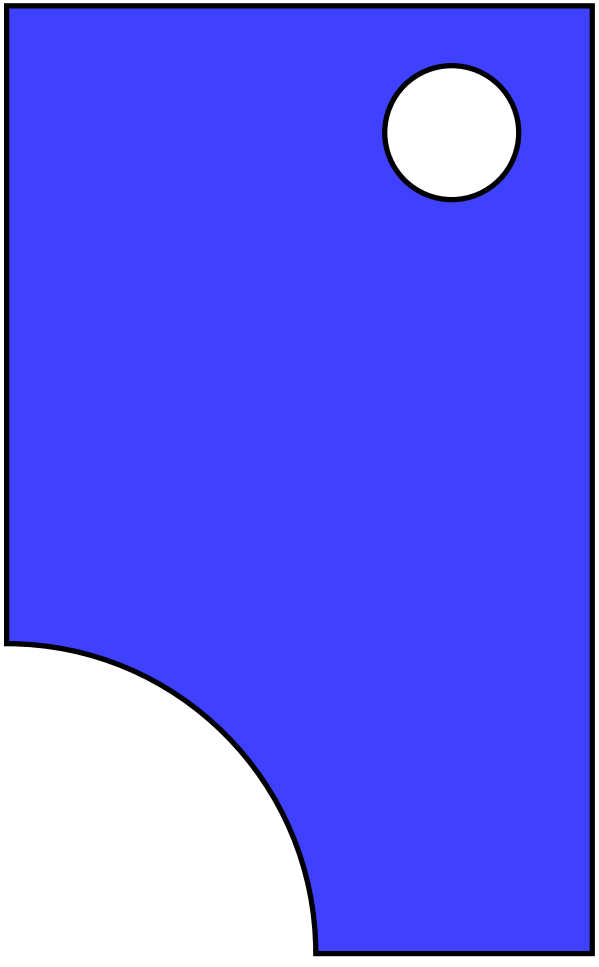
... grâce aux opérations booléennes

Construisons la géométrie du devoir

```
int rect_tag = gmshModelOccAddRectangle(___, ___, ___, ___, ___, ___, ___, &ierr); chk(ierr);
```

1 : Ajout Rectangle $\Rightarrow \{(2,1)\}$

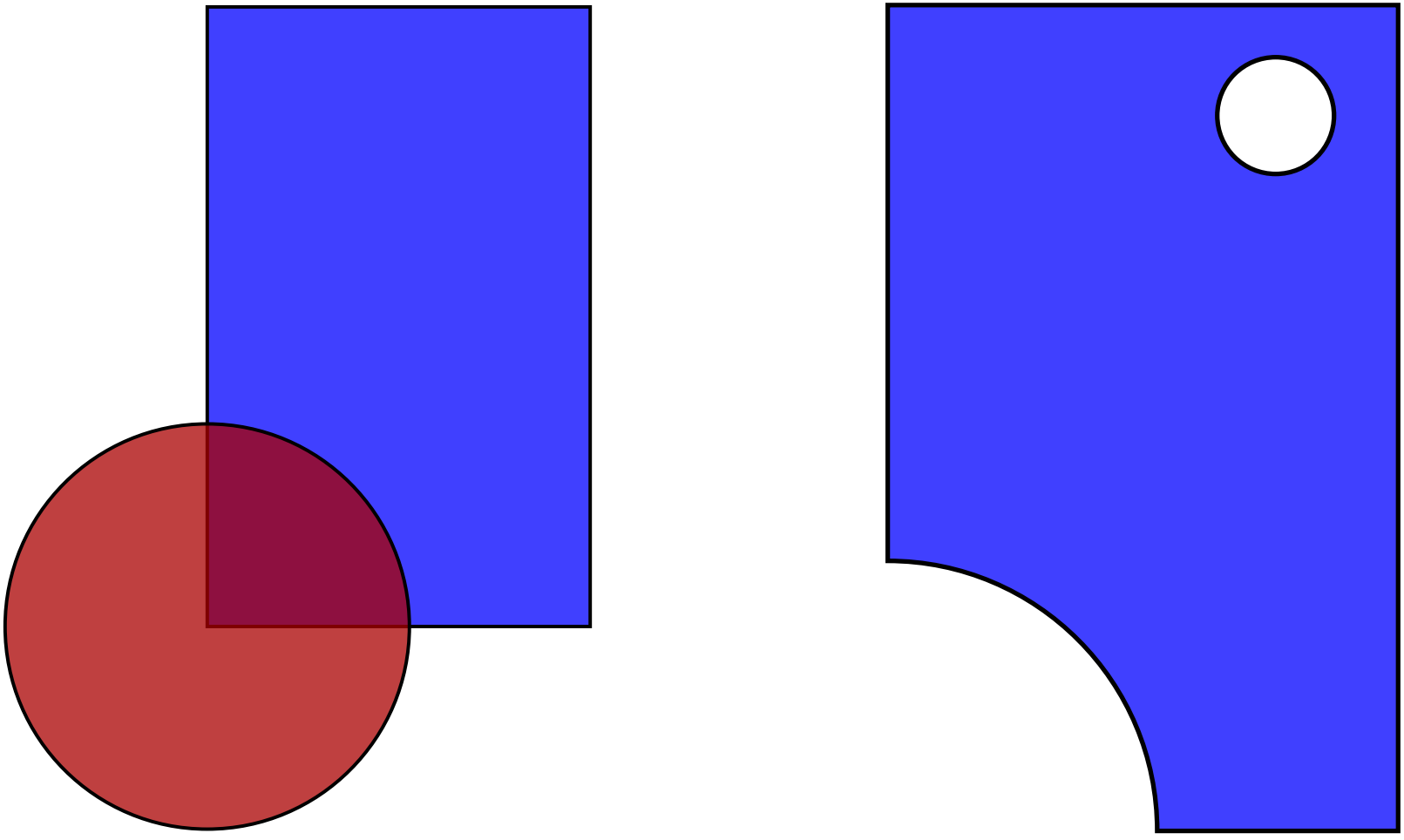




Construisons la géométrie du devoir

```
disk_tag[0] = gmshModelOccAddDisk(___, ___, ___, ___, ___, ___, NULL, 0, NULL, 0, &ierr);
```

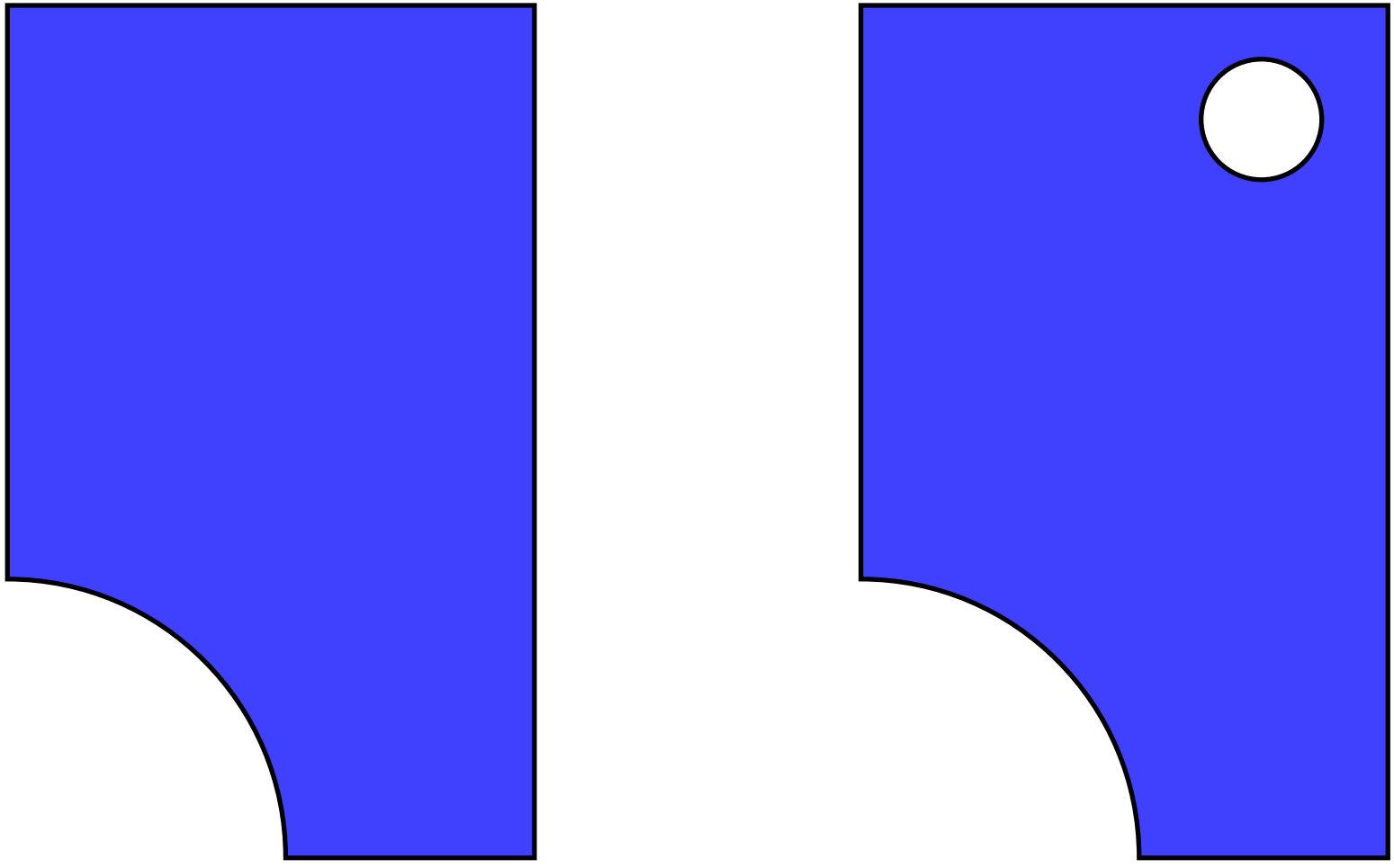
2 : Ajout disque $\Rightarrow \{(2,1),(2,2)\}$



Construisons la géométrie du devoir

```
gmshModelOccCut(___,___,___,___,
                NULL, NULL, NULL, NULL, NULL, -1, 1, 1, &ierr); chk(ierr);
```

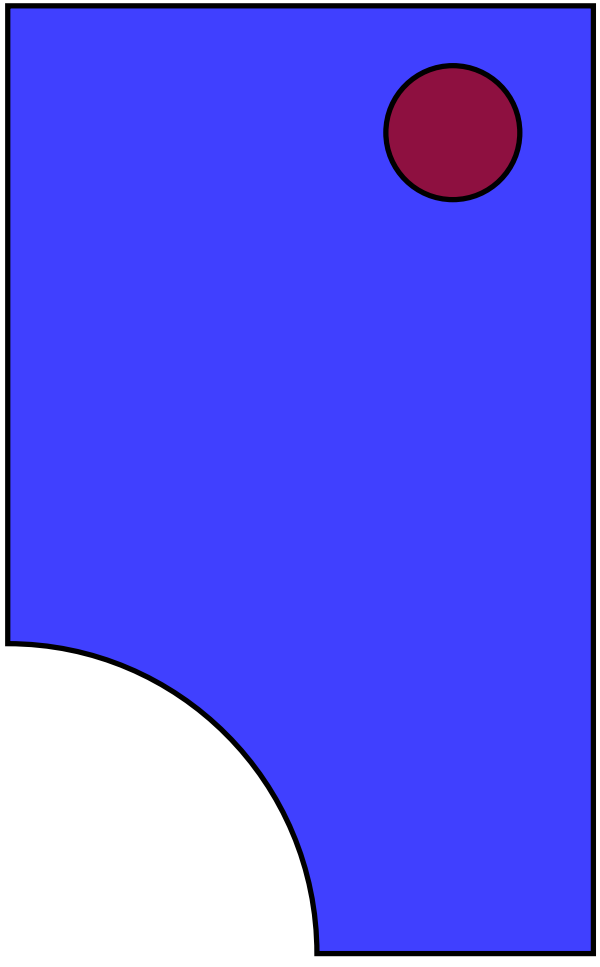
3 : Différence $\Rightarrow \{(2,1)\}$

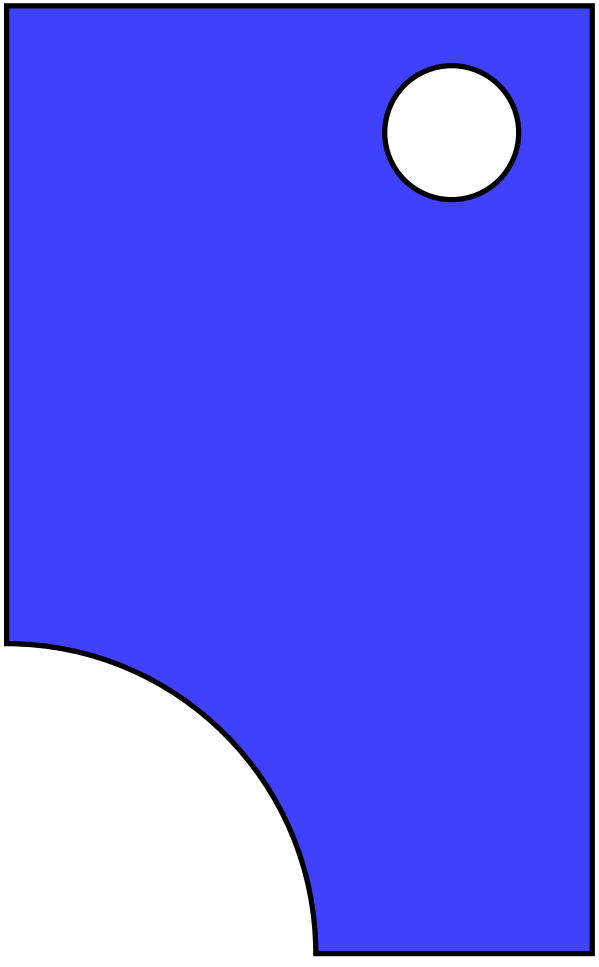


Construisons la géométrie du devoir

```
disk_tag[0] = gmshModelOccAddDisk(___, ___, ___, ___, ___, ___, NULL, 0, NULL, 0, &ierr);
```

4 : Ajout disque $\Rightarrow \{(2,1),(2,3)\}$

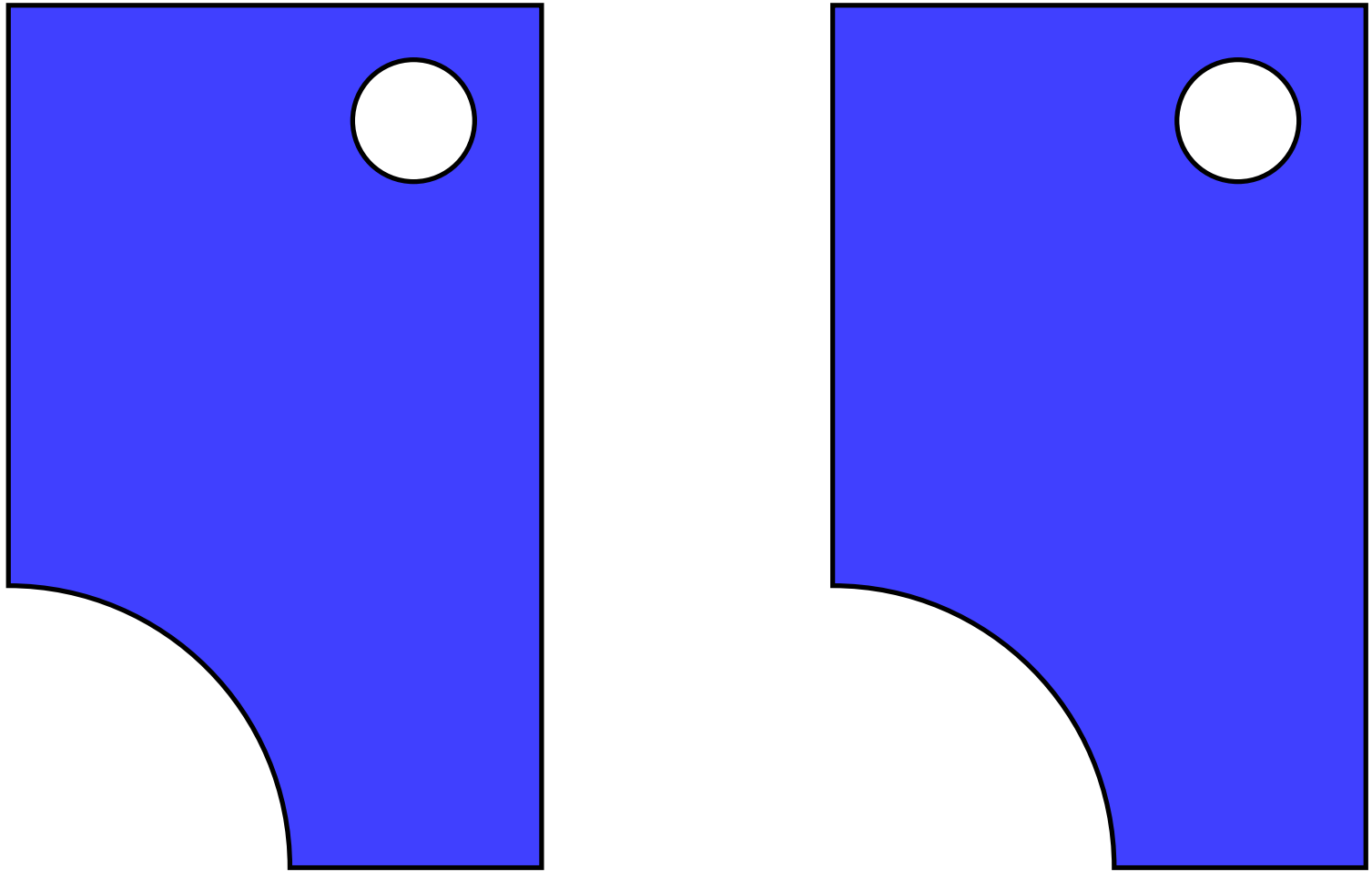




Construisons la géométrie du devoir

```
gmshModelOccCut(___,___,___,___,
                NULL, NULL, NULL, NULL, NULL, -1, 1, 1, &ierr); chk(ierr);
```

5 : Différence $\Rightarrow \{(2,1)\}$

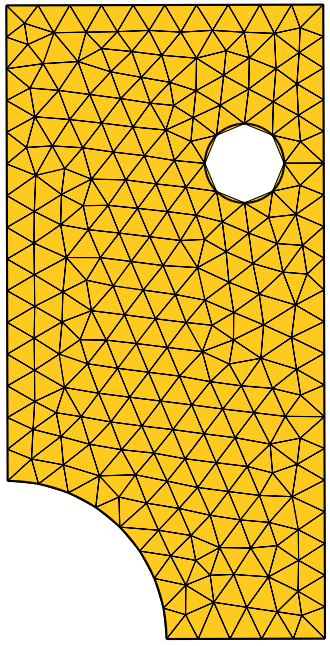


... grâce aux opérations booléennes

Le maillage peut être généré,...

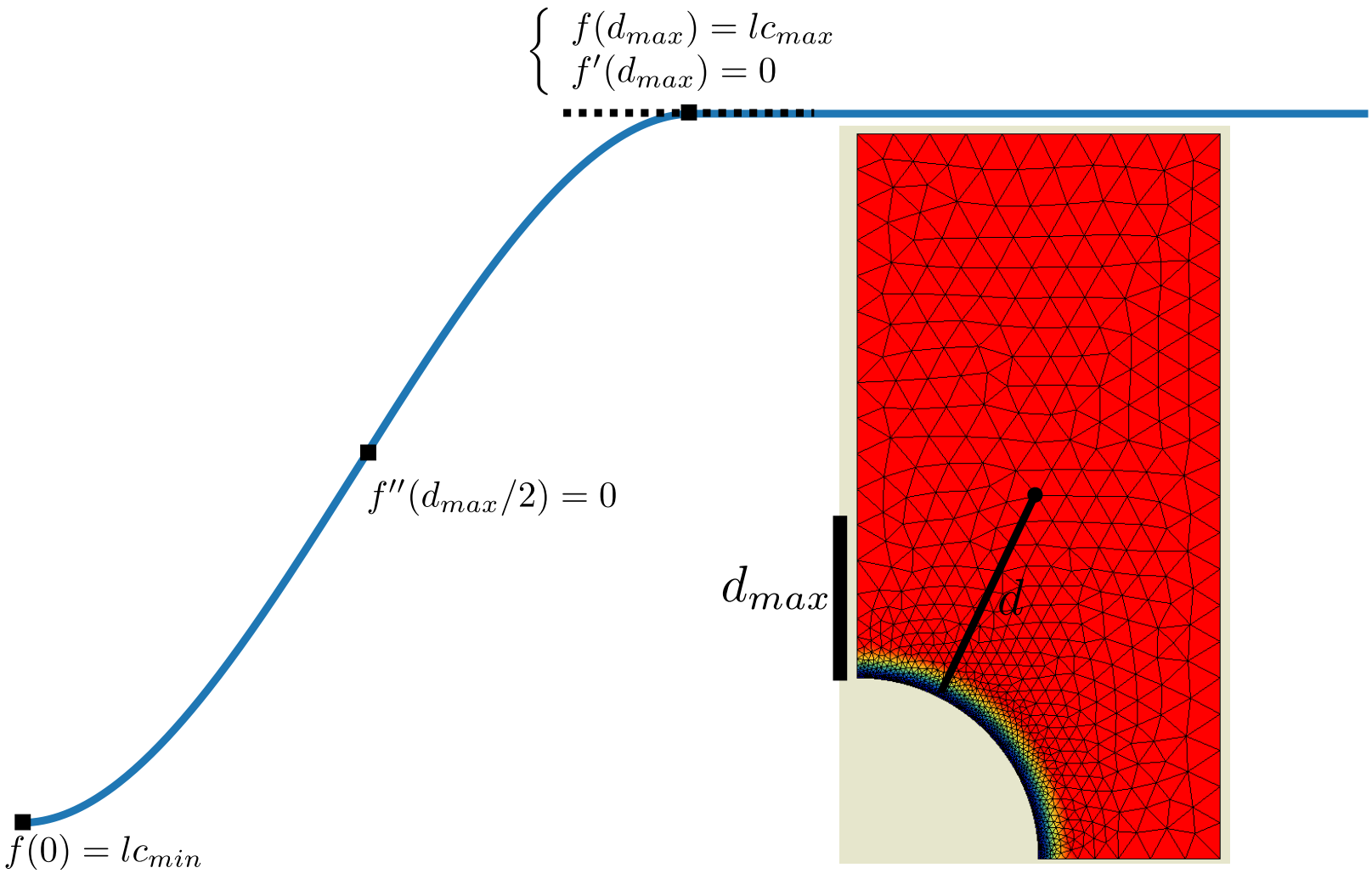
```
double data[] = {x0, y0, r0, lc0, x1, y1, r1, lc1, meshSize};
geoSetSizeCallback(data);
gmshModelMeshGenerate(2, &ierr);
```

```
double geoSize(double x, double y, double *data){
    double x0 = data[0], y0 = data[1], r0 = data[2], lc0=data[3]; // data from disk0
    double x1 = data[4], y1 = data[5], r1 = data[6], lc1=data[7]; // data from disk1
    double lc_global = data[8];
    // Your contribution starts here




    return lc_global;
    // Your contribution ends here
}
```



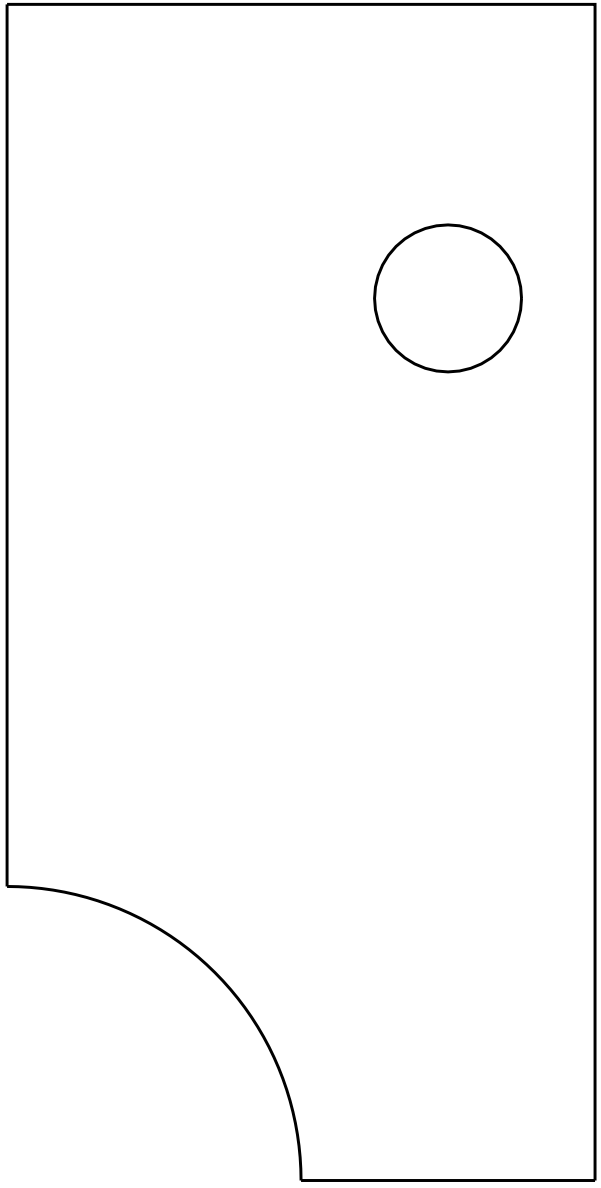
... mais comment améliorer sa qualité ?

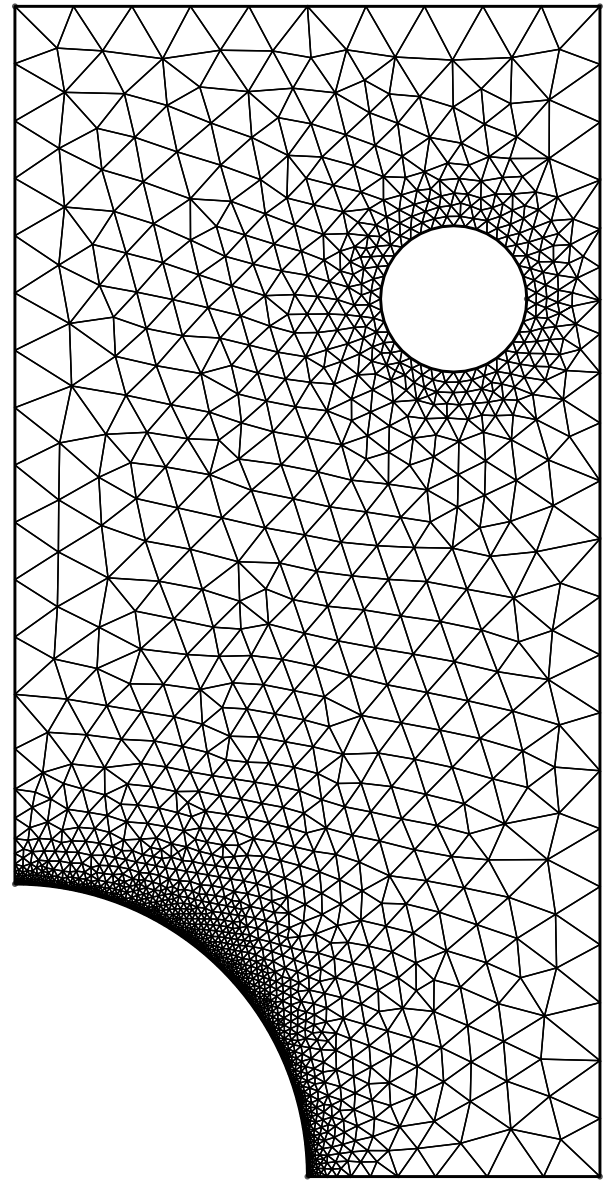
La carte de taille à imposer...



... pour chaque disque

Un maillage est associé à un type d'éléments ...





La connectivité est extraite...

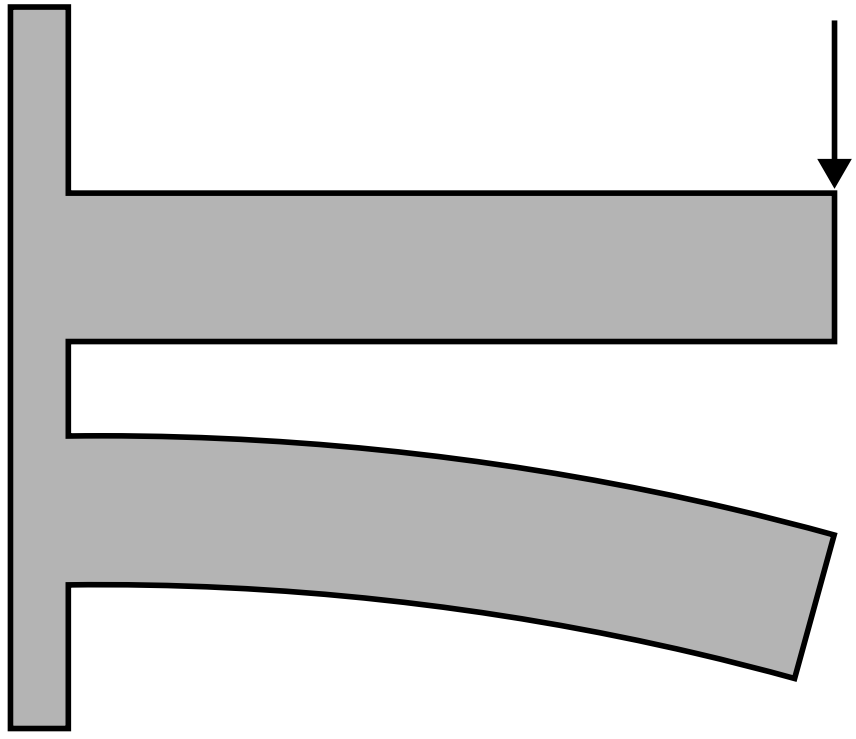
```c
typedef struct {
    int nLocalNode;
    int nElem;
    int *elem;
    femNodes *nodes;
    char name[256];
} femMesh;
```

```c
typedef struct {
    int nMesh;
    femMesh **mesh;
} femMeshes;
```

```c
{theMesh->nLocalNode = 2; strncpy(theMesh->name, "mesh_edge",     256);}
{theMesh->nLocalNode = 3; strncpy(theMesh->name, "mesh_triangle", 256);}
{theMesh->nLocalNode = 4; strncpy(theMesh->name, "mesh_quad",     256);}
{theMesh->nLocalNode = 1; strncpy(theMesh->name, "mesh_node",     256);}
```

... par type d'éléments

La fin...



...,il ne reste plus qu'à résoudre le problème d'élasticité linéaire !