Génération et adaptation de maillage en géométries complexes

Frédéric Alauzet

INRIA Saclay Ile-de-France - Projet Gamma - Palaiseau, France
Frederic.Alauzet@inria.fr

http://pages.saclay.inria.fr/frederic.alauzet/download/Cours1_AMS314.pdf
http://pages.saclay.inria.fr/frederic.alauzet/download/projet-AMS314_Partie1.pdf
http://pages.saclay.inria.fr/frederic.alauzet/download/projet-AMS314.zip

Partie 1: Structure de données et algorithmes élémentaires

1. Définition des éléments élémentaires
2. Notions de triangulation et de maillage
3. Algorithme de localisation
4. Vers les algorithmes informatiques...

Les simplexes : cas du triangle

Triangle = 2-simplexe :

- Enveloppe convexe de 3 points (P_1, P_2, P_3) dans $\mathbb{R}^2$
- Un triangle est définit par ses 3 sommets : $P_1P_2P_3$
- 3 arêtes : e_1, e_2, e_3 de longueur ℓ_1, ℓ_2, ℓ_3
- Aire signée

$$\text{Aire}(K) = \frac{1}{2}\begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix} = \frac{1}{2}\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}$$

- Éléments de surface non signé

$$\text{Aire}(K) = |K| = \frac{1}{2}\|P_1P_2 \times P_1P3\| = \frac{1}{2}\|e_3 \times e_2\|$$

- Rayon cercle circonscrit $\quad r(K) = \dfrac{\ell_1\ell_2\ell_3}{4\,|K|}$
- Rayon cercle inscrit $\rho(K) = \dfrac{2\,|K|}{p(K)} = \dfrac{2\,|K|}{\ell_1 + \ell_2 + \ell_3}$

Tétraèdre = 3-simplexe :

- Enveloppe convexe de 4 points (P_1, P_2, P_3, P_4) dans $\mathbb{R}^3$
- Un tétraèdre est définit par ses 4 sommets : $P_1P_2P_3P_4$
- 6 arêtes $\{e_i\}$ de longueur $\{\ell_i\}$, 4 faces triangulaires $\{f_i\}$ de surface $\{|F_i|\}$
- Volume signé

$$\mathrm{Vol}(K) = |K| = \frac{1}{6}\begin{vmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{vmatrix} = \frac{1}{6}\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$

- Rayon sphère circonscrite $r(K) = \dfrac{\|\ell_1^2(e_2 \times e_3) + \ell_2^2(e_3 \times e_1) + \ell_3^2(e_1 \times e_2)\|}{12\,|K|}$
- Rayon sphère inscrite $\rho(K) = \dfrac{3\,|K|}{|F_1| + |F_2| + |F_3| + |F_4|}$

Validité d'un élément

- Le volume (aire) signé(e) permet de déclarer un élément valide
  On verra par la suite que cela permet de nous repérer dans un maillage

- Problème lié à son évaluation numérique

  Exercice 1 :

  • Calculer le volume du tétraèdre régulier
  $P_1 = (0, 0, 0)$, $P_2 = (1, 0, 0)$
  $P_3 = (0.5, \frac{\sqrt{3}}{2}, 0)$, $P_4 = (0.5, \frac{\sqrt{3}}{6}, \frac{\sqrt{2}}{\sqrt{3}})$

Validité d'un élément

- Le volume (aire) signé(e) permet de déclarer un élément valide

  On verra par la suite que cela permet de nous repérer dans un maillage

- Problème lié à son évaluation numérique

  Exercice 2 :

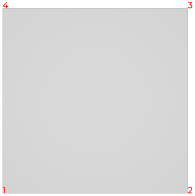
  • Calculer le volume du tétraèdre dégénéré
  $P_1 = (0, 0, 0)$, $P_2 = (1, 0, 0)$
  $P_3 = (0.5, \frac{\sqrt{3}}{2}, 0)$, $P_4 = \frac{P_1+P_2+P_3}{3} + (0, 0, \varepsilon)$

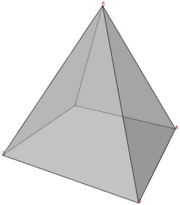
  • Appliquer une rotation de 45° suivant l'axe x et 30° suivant l'axe y à P_1, P_2, P_3 puis $P_4 = \frac{P_1+P_2+P_3}{3}$ et recalculer le volume

  • Translater tous les points de 1000 suivant x et recalculer le volume

  • Qu'en concluez vous ?

Éléments hybrides

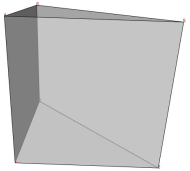


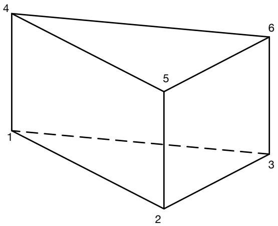


- Quadrilatère $P_1P_2P_3P_4$
  - Décomposition en 2 triangles
- Pyramide $P_1P_2P_3P_4P_5$
  - Décomposition en 2 tétraèdres

Prisme

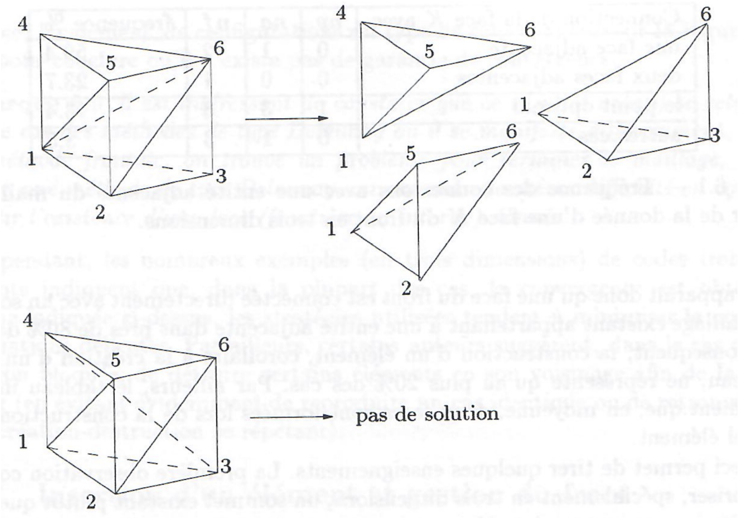
- On considère un prisme régulier $P_1P_2P_3P_4P_5P_6$





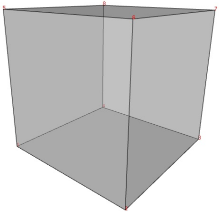
- Combien de tétraèdres sont nécessaires pour décomposer ce prisme ?
- Combien de décompositions en tétraèdres valides ?

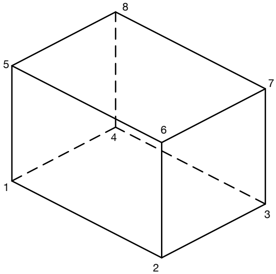
Prisme



Hexaèdre

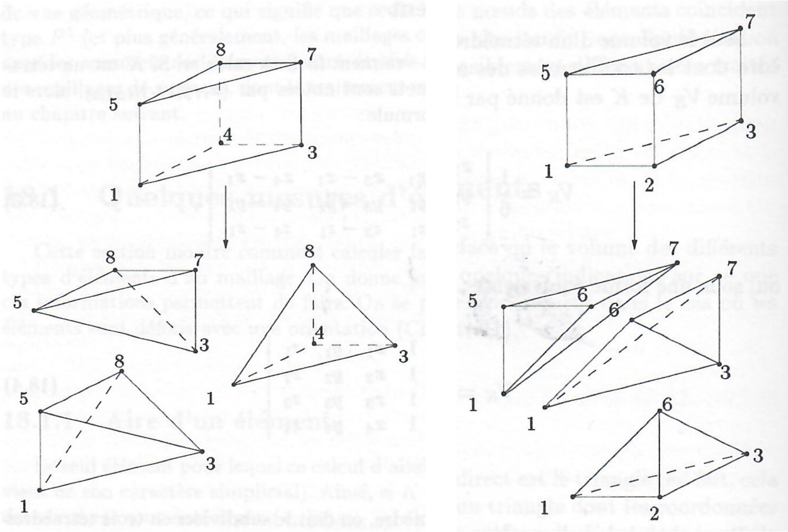
- On considère un hexaèdre régulier $P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8$.



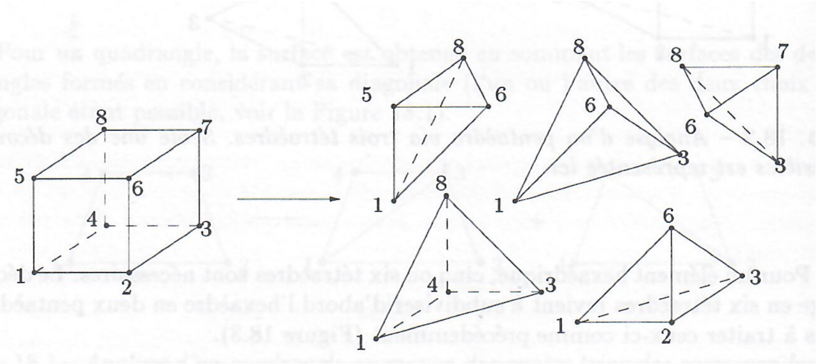


- Combien de tétraèdres sont nécessaires pour décomposer cet hexaèdre ?

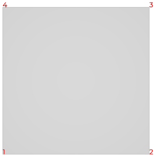
Hexaèdre

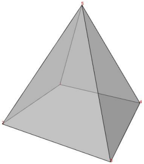


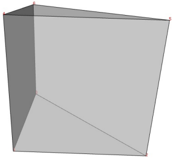
Hexaèdre

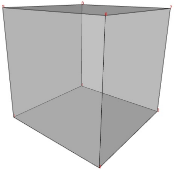


Éléments hybrides



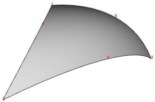


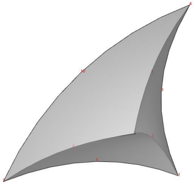




- Quadrilatère
  - Décomposition en 2 triangles
- Pyramide
  - Décomposition en 2 tétraèdres
- Prisme
  - Décomposition en 3 tétraèdres
  - Il existe des décompositions invalides !!
- Hexaèdre
  - Décomposition en 5/6 tétraèdres
  - Il existe des décompositions invalides !!

Éléments courbes P2





Triangle P2

- 6 nœuds

Tetraèdre P2

- 10 nœuds

Dans ce cours, on s'intéresse seulement aux maillages non-structurés formés de simplexes.

Ils sont les plus efficaces pour mailler des **géométries complexes**

Partie 1: Structure de données et algorithmes élémentaires

1. Définition des éléments élémentaires
2. Notions de triangulation et de maillage
3. Algorithme de localisation
4. Vers les algorithmes informatiques...

Triangulation/Tesselation

Soit S un ensemble de N points de $\mathbb{R}^2/\mathbb{R}^3$

Un ensemble de simplexes, $\mathcal{T}_h = \{K_i\}_i$, s'appuyant sur les points de S tel que :

H_1 L'ensemble des sommets de $\mathcal{T}_h$ est exactement S

H_2 $\overset{\circ}{K} \neq \emptyset \quad \Leftrightarrow \quad |K| \neq 0 \qquad \forall K \in T_h$ $\Longleftarrow$ Validité

H_3 $\overset{\circ}{K}_i \cap \overset{\circ}{K}_j = \emptyset \qquad \forall K_i, K_j \in T_h \quad (i \neq j)$ $\Longleftarrow$ Pas d'intersection

est un **recouvrement simplicial**.

Soit S un ensemble de N points de $\mathbb{R}^2/\mathbb{R}^3$

Un ensemble de simplexes, $\mathcal{T}_h = \{K_i\}_i$, s'appuyant sur les points de S tel que :

H_1 L'ensemble des sommets de $\mathcal{T}_h$ est exactement S

H_2 $\overset{\circ}{K} \neq \emptyset \quad \Leftrightarrow \quad |K| \neq 0 \qquad \forall K \in T_h$ $\Longleftarrow$ Validité

H_3 $\overset{\circ}{K}_i \cap \overset{\circ}{K}_j = \emptyset \qquad \forall K_i, K_j \in T_h \;\; (i \neq j)$ $\Longleftarrow$ Pas d'intersection

H_4 L'intersection de deux éléments de $\mathcal{T}_h$ est $\Longleftarrow$ Conformité

- soit $\emptyset$
- soit un simplexe de dim $<$ d

(en 2D un point ou une arête, en 3D un point ou une arête ou une face)

est une **triangulation / tesselation**.

Remarque :

Parmi les différents types de triangulations, la triangulation de Delaunay est d'un grand intérêt. Dans ce cas particulier, l'union de tous les éléments est *l'enveloppe convexe* de S : $\mathrm{conv}(S) = \bigcup_i K_i$. On y reviendra plus loin dans ce cours.

Soit Ω un domaine de $\mathbb{R}^2/\mathbb{R}^3$ de frontière Γ.

Un ensemble de simplexes $\mathcal{H} = \{K_i\}_i$ vérifiant :

H_1 $\forall P \in \partial\mathcal{H}$ alors $P \in \Gamma$ $\Longleftarrow$ Contrainte la géom est imposée

H_2 $\Omega = \bigcup_i K_i$

H_2 $\mathring{K} \neq \emptyset \quad \Leftrightarrow \quad |K| \neq 0 \qquad \forall K \in T_h$ $\Longleftarrow$ Validité

H_3 $\mathring{K}_i \cap \mathring{K}_j = \emptyset \qquad \forall K_i, K_j \in T_h \quad (i \neq j)$ $\Longleftarrow$ Pas d'intersection

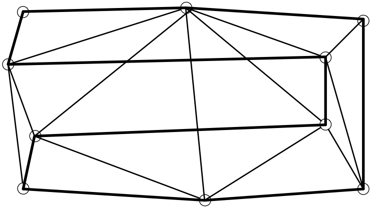
H_4 L'intersection de deux éléments de $\mathcal{T}_h$ est $\Longleftarrow$ Conformité

- soit $\emptyset$
- soit un simplexe de dim $<$ d

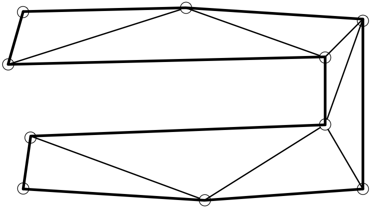
(en 2D un point ou une arête, en 3D un point ou une arête ou une face)

est **un maillage** (ou triangulation contrainte) de Ω.

Différence entre une triangulation et un maillage



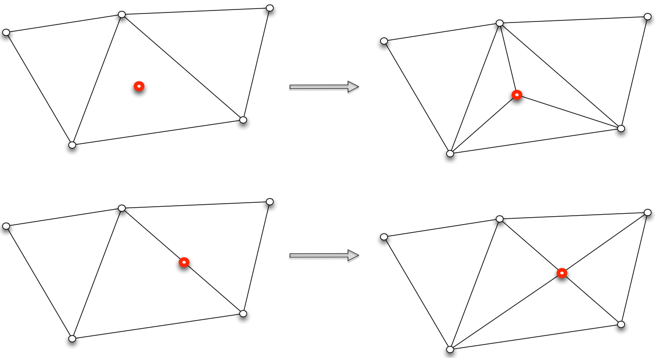
Triangulation de S



Maillage de Ω

Méthode de Triangulation Facile

Opérateurs d'insertion d'un point :

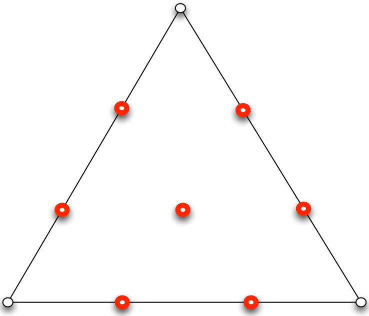


Il faut savoir **localiser** un point dans un élément / sur une arête

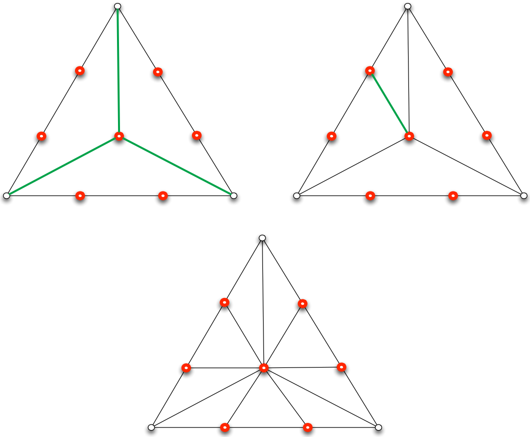
Méthode de Triangulation Facile

On se donne un domaine triangulaire

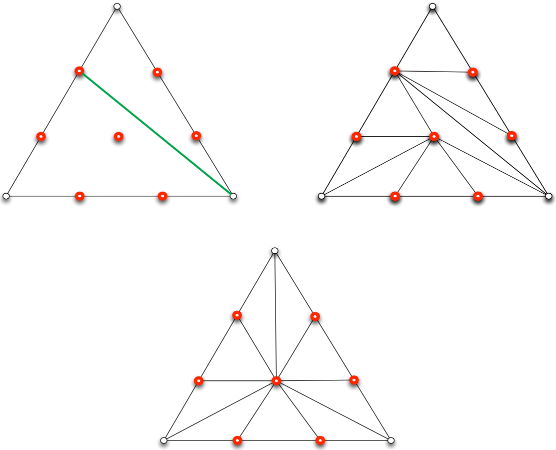
Objectif: insérer les points rouges dans le domaine
en utilisant les deux opérateurs d'insertion



Méthode de Triangulation Facile

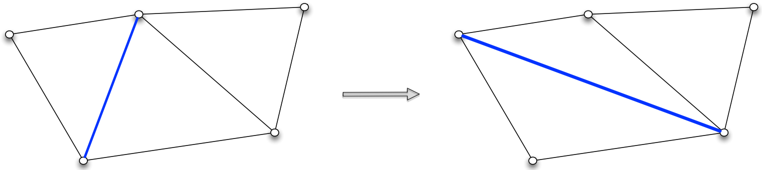


Méthode de Triangulation Facile



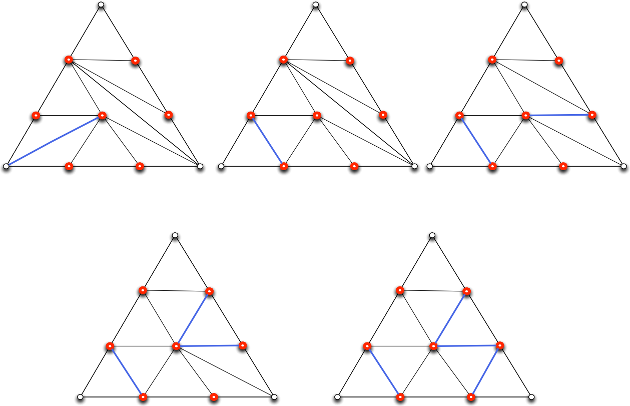
Est ce que l'on peut faire un maillage plus esthétique ?

Opérateur de bascule d'arête :



Il faut savoir trouver les deux **triangles voisins** par une arête.

Méthode de Triangulation Facile

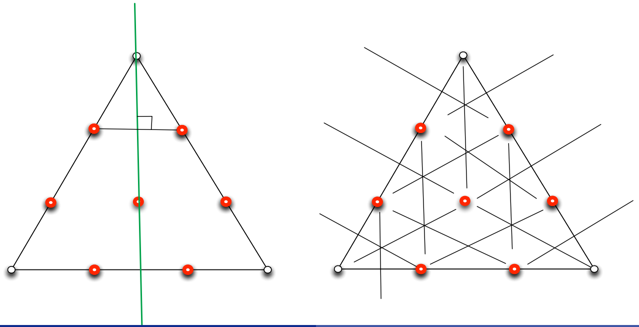


Méthode géométrique d'existence : le diagramme de Voronoï

Soit $S = \{P_i\}_i$ un nuage de points. On appelle cellule de Voronoï V_i associée au point P_i la région du plan où tout point est plus proche de P_i que n'importe quel autre $P_j \in S$ avec $j \neq i$:

$$V_i = \left\{ Q \;\middle|\; P_iQ \leq P_jQ \;,\; \forall j \neq i \right\}$$

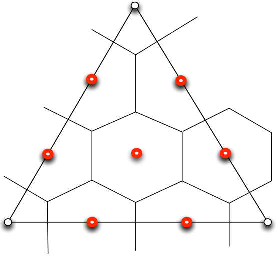
On appelle diagramme de Voronoï le recouvrement du plan définit par l'ensemble des V_i : $\{V_i\}_i$. Ce diagramme **existe** et est **unique**.

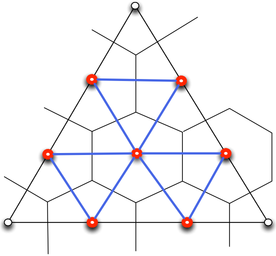


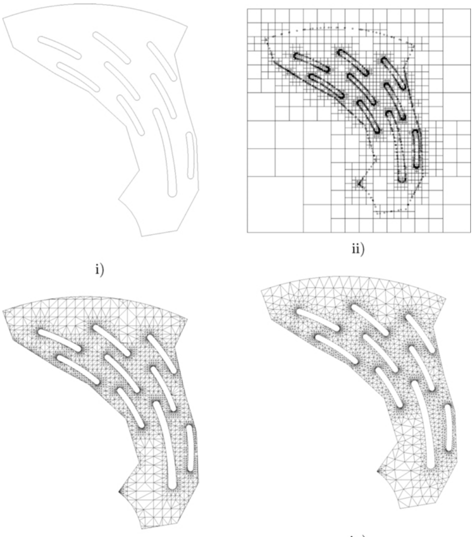
Méthode géométrique d'existence : le diagramme de Voronoï

On appelle diagramme de Voronoï le recouvrement du plan définit par l'ensemble des V_i : $\{V_i\}_i$. Ce diagramme **existe** et est **unique**.

Son dual est obtenu en reliant les points qui partage une face de cellule de Voronoï. C'est la triangulation de Delaunay.
Cette triangulation **existe** et est **unique** (si il y a au plus 3 points cocyclique).

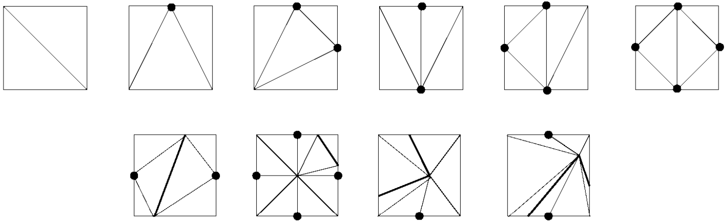




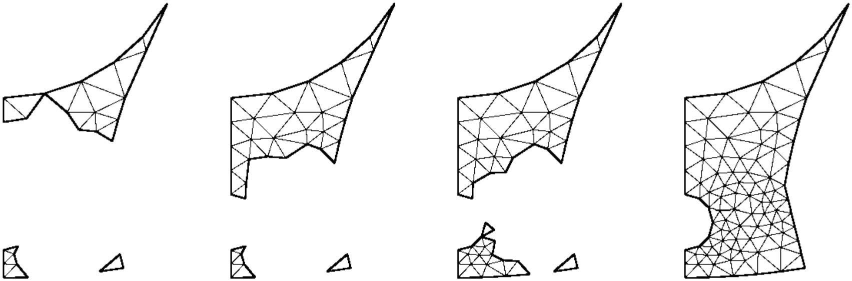


i) ii) iii) iv)

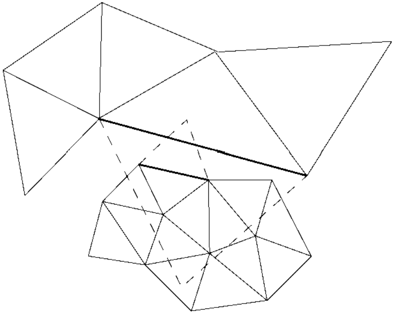
Problématique principale : Respect de la géométrie



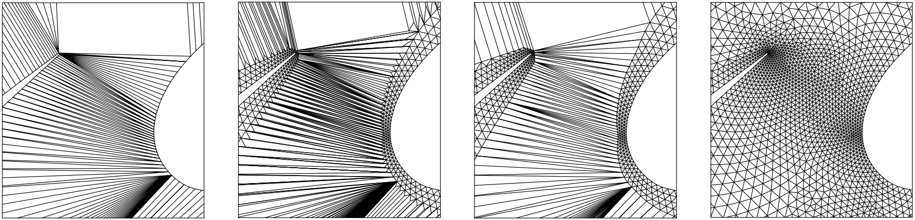
Méthodes frontales



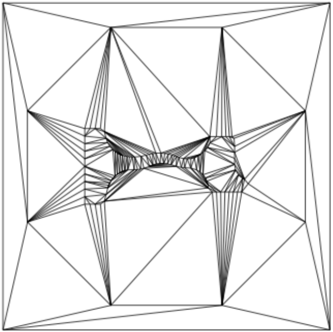
Problématique principale : fermeture du front

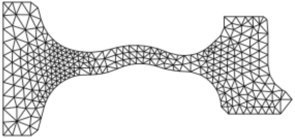


Problématique principale : forçage de la frontière

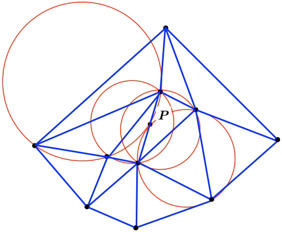


Problématique principale : forçage de la frontière

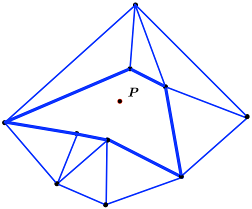




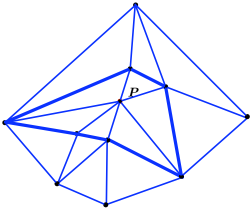
Noyau de Delaunay



$\mathcal{T}^n$

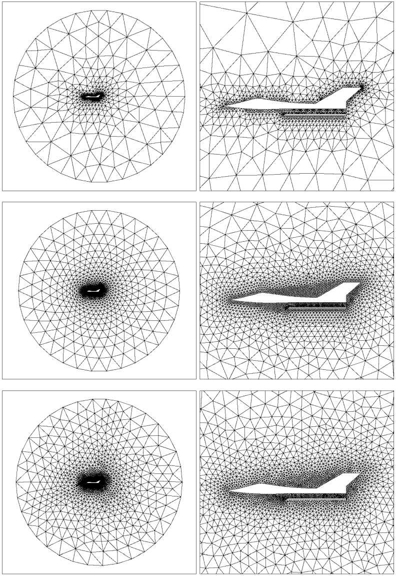


$\mathcal{T}^n - \mathcal{C}_P$



$\mathcal{T}^n - \mathcal{C}_P + \mathcal{B}_P = \mathcal{T}^{n+1}$

Quel est le type de méthode utilisé dans les exemple ci-dessous ?



Partie 1: Structure de données et algorithmes élémentaires

1. Définition des éléments élémentaires
2. Notions de triangulation et de maillage
3. Algorithme de localisation
4. Vers les algorithmes informatiques...

Localisation

Savoir **localiser** si un point est dans un élément ou pas.

Coordonnées barycentriques (similaire en 3D)

- On se place dans $\mathbb{R}^2$

  Soit $K = P_1P_2P_3$ alors $\forall P \in \mathbb{R}^2$, $\exists \beta_i \in \mathbb{R}$ pour $i = 1, 2, 3$ tel que

  $$P(x, y) = \sum_{i=1}^{3} \beta_i(x, y)\, P_i$$

  Preuve : vrai car P_1, P_2, P_3 forme une base de $\mathbb{R}^2$

- Interprétation géométrique:

$$\beta_1(x, y) = \frac{|PP_2P_3|}{|K|}, \qquad \beta_2(x, y) = \frac{|P_1PP_3|}{|K|}, \qquad \beta_3(x, y) = \frac{|P_1P_2P|}{|K|}$$

Un point P est dans un élément $K = P_1P_2P_3$ ssi $0 \leq \beta_i \leq 1 \;\; \forall i = 1, 2, 3$.

Savoir **localiser** si un point est dans un élément ou pas.

Méthode de localisation naïve : algorithme quadratique

```
For iPts=1,NbrPts
  For iElt=1,NbrElt
    [b1,b2,b3] = ComputeBarycentrics(P,K);
    If ( b1 > 0 && b2 >0 && b3 > 0 )
      Grm[iPts] = iElt;
      break;
    EndIf
  EndFor
EndFor
```

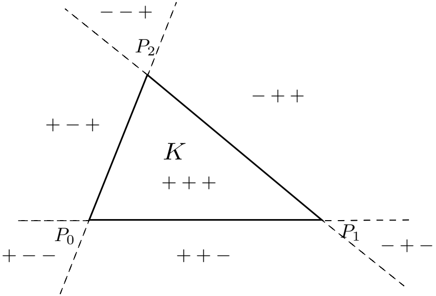
Il est de complexité $N_{pts} * N_{elt}$

On verra que ce type d'algorithme n'est pas acceptable

Savoir **localiser** si un point est dans un élément ou pas.

- Le maillage est orienté $\Longrightarrow$ on a des aires ou volumes signés (positif)

$\Longrightarrow$ Les coordonnées barycentriques séparent le plan en 7 espaces convexes

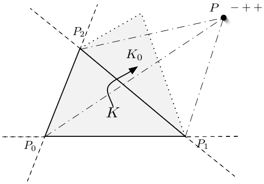


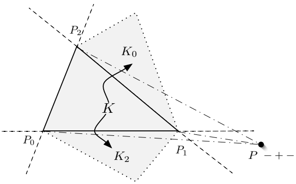
On peut utiliser cette propriété pour écrire un algorithme linéaire

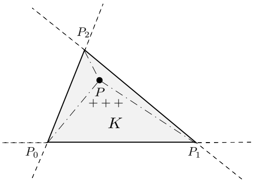
Savoir **localiser** si un point est dans un élément ou pas.

- Le maillage est orienté $\Longrightarrow$ on a des aires ou volumes signés (positif)

$\Longrightarrow$ L'idée est de se déplacer dans le maillage en utilisant le signe des coordonnées barycentriques



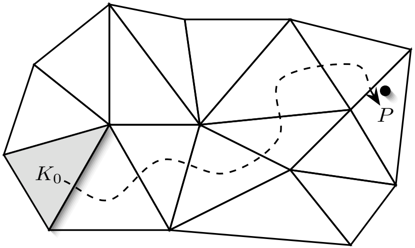




Savoir **localiser** si un point est dans un élément ou pas.

- Le maillage est orienté $\Longrightarrow$ on a des aires ou volumes signés (positif)

$\Longrightarrow$ L'idée est de se déplacer dans le maillage en utilisant le signe des coordonnées barycentriques



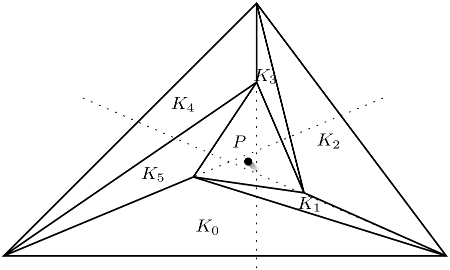
$\Longrightarrow$ Cet algorithme a une complexité de $N_{pts} * n_{move}$

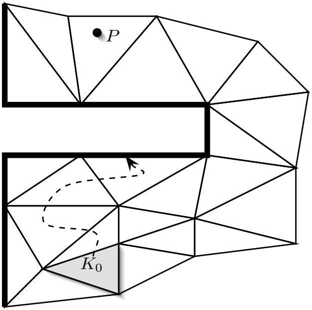
Il faut connaître les **triangles voisins** d'un triangle.

Savoir **localiser** si un point est dans un élément ou pas.

Il existe quand même des problématiques :

- Cycle limite $\Longrightarrow$ faire de l'aléatoire ou du coloriage
- Bloqué $\Longrightarrow$ recherche exhaustive ou se déplacer sur la frontière
- En dehors du domaine

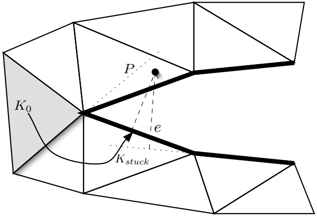


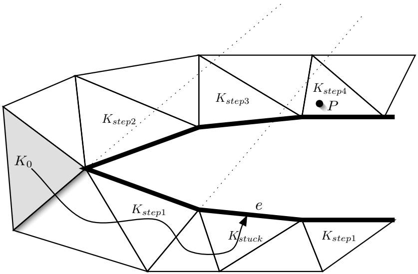


Savoir **localiser** si un point est dans un élément ou pas.

Il existe quand même des problématiques :

- Cycle limite $\Longrightarrow$ faire de l'aléatoire ou du coloriage
- Bloqué $\Longrightarrow$ recherche exhaustive ou se déplacer sur la frontière
- En dehors du domaine

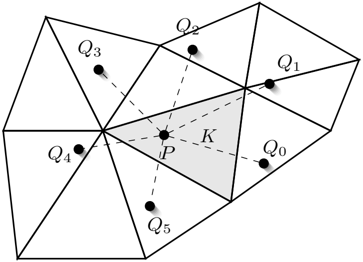




Savoir **localiser** si un point est dans un élément ou pas.

Mais on a des remèdes et on peut même réduire n_{move}:

- On peut utiliser une structure de grille ou un quadtree/octree
- On peut utiliser la connectivité des deux maillages



Exercices

On considère le triangle K ayant pour sommets $A = (0, 0)$, $B = (1, 0)$ et $C = (0.5, 0.5)$:

Q1. Calculer les cordonnées barycentriques du point $P = (1, \frac{1}{3})$.

Q2. En déduire par quelle(s) arête(s), on peut passer pour localiser P à partir de K. Illustrer graphiquement votre réponse.

Partie 1: Structure de données et algorithmes élémentaires

1. Définition des éléments élémentaires
2. Notions de triangulation et de maillage
3. Algorithme de localisation
4. Vers les algorithmes informatiques...

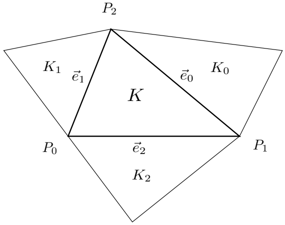
Structure de données

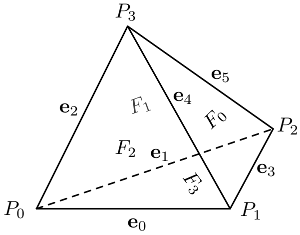
Comment représenter un maillage sur ordinateur ?

- Structures de données **locales**
  - Définissent les éléments
  - Des tables de connectivités de petite taille basées sur des **conventions**. Cela définit notamment **l'orientation**
  - Les indices démarrent à 0 en C/C++ (pour les modulos) et à 1 en Fortran

- Structures de données **globales**
  - Donnent la liste des entités (sommets, éléments, ...)
  - Des tableaux (matrices) de grande taille (en général)
  - Les indices démarrent à 1 (pour ne pas toujours décaler)

Structures de données locales

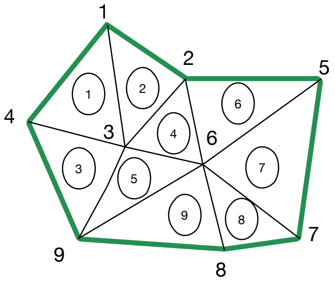




$$tri2edg = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}$$

$$tet2fac = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 3 & 0 & 1 & 2 \end{bmatrix}$$

$$tet2edg = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 2 \\ 1 & 2 & 3 & 2 & 3 & 3 \end{bmatrix}$$

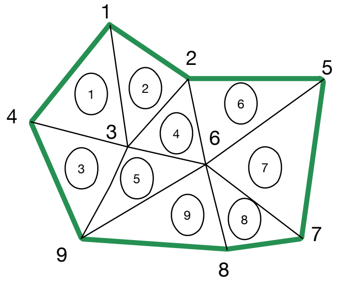


- Données géométriques : **coordonnées**

$$Crd = \begin{bmatrix} x_1 & \dots & x_9 \\ y_1 & \dots & y_9 \end{bmatrix}$$

- Connectivités entre les points : **éléments**

$$Tri = \begin{bmatrix} 1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix}$$

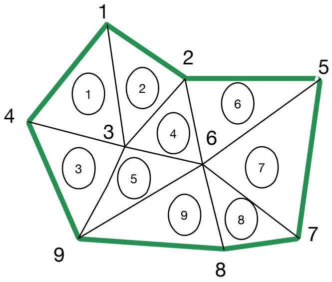


- Arêtes internes/frontières

$$\begin{bmatrix} 1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix}$$

Interprétation



- Relation de voisinage

$$
\begin{array}{c}
\begin{bmatrix}
1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\
3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\
4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8
\end{bmatrix} \\
\begin{array}{ccccccccc}
1 & 2 & & & & & 7 & 8 &
\end{array}
\end{array}
$$

voisin par l'arête 3-4

$$\begin{bmatrix} \bullet & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix} \quad \begin{bmatrix} 3 & 1 & 3 \\ -1 & -1 & 5 \\ 2 & 4 & -1 \end{bmatrix}$$

voisin par l'arête 1-4

$$\begin{bmatrix} 1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ \bullet & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix} \quad \begin{bmatrix} 3 & 1 & 3 \\ -1 & -1 & 5 \\ 2 & 4 & -1 \end{bmatrix}$$

voisin par l'arête1-3

$$\begin{bmatrix} 1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ \bullet & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix} \quad \begin{bmatrix} 3 & 1 & 3 \\ -1 & -1 & 5 \\ 2 & 4 & -1 \end{bmatrix}$$

voisin par l'arête 3-1

$$\begin{bmatrix} 1 & \bullet & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix} \qquad \begin{bmatrix} 3 & 1 & 3 \\ -1 & -1 & 5 \\ 2 & 4 & -1 \end{bmatrix}$$

voisin par l'arête 2-1

$$\begin{bmatrix} 1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & \bullet & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & 1 & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix} \qquad \begin{bmatrix} 3 & 1 & 3 \\ -1 & -1 & 5 \\ 2 & 4 & -1 \end{bmatrix}$$

voisin par l'arête 2-3

$$\begin{bmatrix} 1 & 2 & 9 & 2 & 6 & 5 & 7 & 6 & 9 \\ 3 & 3 & 4 & 6 & 9 & 6 & 6 & 7 & 6 \\ 4 & \bullet & 3 & 3 & 3 & 2 & 5 & 8 & 8 \end{bmatrix} \qquad \begin{bmatrix} 3 & 1 & 3 \\ -1 & -1 & 5 \\ 2 & 4 & -1 \end{bmatrix}$$

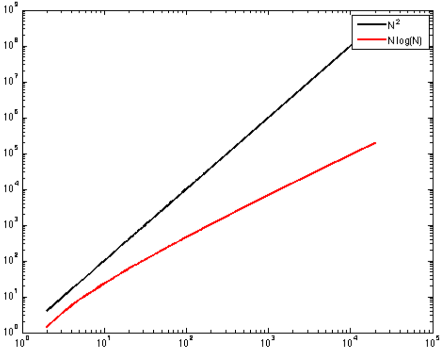
Algorithme de construction des voisins

```
% Function SetNeighbors

tri2edg = [1 2 0; 2 0 1];
memset(TriVoi, 0, sizeof(int3)*(NbrTri+1));

for (iTri=1; iTri<=NbrTri; ++iTri) {
  for (ied=0; ied<3; ++ied) {
     % sommets de l'arete ied du triangle iTri
     iv1 = Tri[iTri][tri2edg[ied][0]];
     iv2 = Tri[iTri][tri2edg[ied][1]];

     for (jTri=1; jTri<=NbrTri; ++jTri) {
       if ( iTri == jTri ) continue;
       for (jed=0; jed<3; ++jed) {
          % sommets de l'arete jed du triangle jTri
          jv1 = Tri[jTri][tri2edg[jed][0]];
          jv2 = Tri[jTri][tri2edg[jed][1]];

          if ( ((iv1==jv1) && (iv2==jv2)) || ((iv1==jv2) && (iv2==jv1)) ) {
            % Mise a jour des voisins
            TriVoi[iTri][ied] = jTri;
            TriVoi[jTri][jed] = iTri;
            break;
          }
       }
     }
  }
}
```

Ce type d'algorithme n'est pas acceptable

Complexité

- Estimer la mémoire/le temps CPU pour un algorithme
- 2 boucles sur le nombre de triangles, 2 boucles sur le nombres d'arêtes
- Chaque triangle est visité $O(9\,n^2)$ fois avec $n = NbrTri$



# de triangles	temps CPU
38	0.0006s
5084	8.78s
20336	140.8s

Objectif :

Casser la complexité, *i.e.*, retrouver un algorithme de complexité globalement linéaire

- Trier par clé des objets. Exemple de clé pour une arête : $\texttt{cle}(is_1, is_2) = is_1 + is_2$
- Les objets en collision sont ceux ayant la même clé
- Algorithme quadratique sur les objets en collision seulement

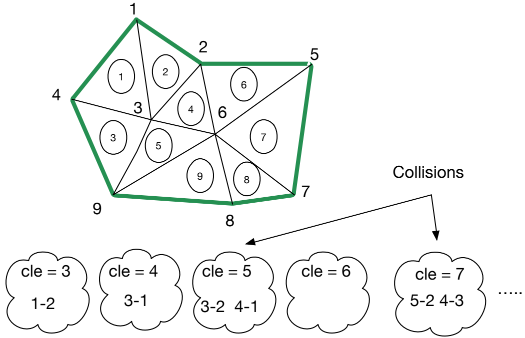


Table de hachage

Une table de hachage est représentée par un ensemble de listes chaînées

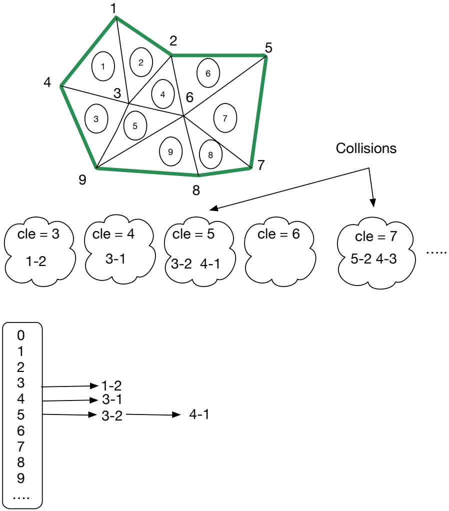


Table de hachage

Une table de hachage est représentée par un ensemble de listes chaînées

Pour cela, il faut les données:

- Taille du tableau de tête (de clés) : `SizHead`
- Un tableau de Tête (de clés) : `Head`
- Le nombre d'objets : `NbrObj`
- Une liste d'objets (un ensemble de listes chaînées) : `LstObj`

Exemple dans le cas des arêtes:

- `SizHead = ?`
- `NbrObj = ?`

Liste chaînée

- Une structure de donnée de liste chaînée

$$\texttt{Head} = 2; \qquad \texttt{LstObj} = \begin{bmatrix} 10 & 1 & 5 & 7 \\ 3 & 1 & 4 & 0 \end{bmatrix}$$

- `Head` donne le debut de la liste : Ici la 2eme position
- `LstObj`[:][0] stocke les valeurs de la liste, *i.e.*, les objects :
  l'objet en position 3 est l'objet 5 : $\texttt{LstObj}[3][0] \to 5$
- `LstObj`[:][1] stocke le chaînage, *i.e.*, donne le suivant de la liste ou 0 si il n'y a pas d'autre objet :
  l'objet après le 5eme est l'objet 7 en position 4 : $\texttt{LstObj}[3][1] \to 4$
  et on a $\texttt{LstObj}\left[\texttt{LstObj}[3][1]\right][0] = \texttt{LstObj}[4][0] \to 7$

Question : Énumérer les objets de la liste dans l'ordre.

Table de hachage : Exercice

Le tableau `LstObj` suivant représente un *chaînage* dont le premier indice de ligne représente un numéro de point et le deuxième indice de ligne le lien vers le point suivant de la liste:

$$\texttt{LstObj} = \begin{bmatrix} 1 & 3 & 4 & 2 & 1 & 10 & 5 & 6 & 7 & 4 \\ 4 & 0 & 9 & 2 & 3 & 8 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Quels sont les derniers éléments de chaque liste ?
- En déduire combien de listes chaînées sont stockées dans ce tableau ?
- Énumérer les points de chaque liste chaînée dans l'ordre.

On souhaite désormais modifier `LstObj`. Écrire `LstObj` dans les cas suivants :

- après ajout d'une nouvelle liste composée des points $[4, 6, 5]$,
- après ajout du point 11 suivant le point 3.

Table de hachage : Exercice

On conserve `LstObj` qui représente un ensemble de listes chaînées.
On se donne le tableau de tête `Head` qui pour chaque indice (clé) donne l'indice dans `LstObj` du premier élément de la liste associée à cette clé.
Soit le tableau tête suivant:

$$\mathtt{Head} = \begin{bmatrix} 0 \\ 10 \\ 5 \\ 6 \\ 0 \\ 7 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Combien de points ont une clé de 2 ? De 5 ?
- Associer pour chaque clé, une des listes chaînées trouvées précédemment.

Table de hachage

Une table de hachage est représentée par un ensemble de listes chaînées

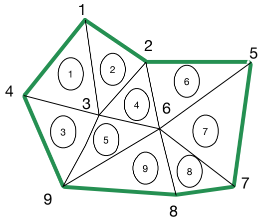
Pour cela, il faut les données:

- Taille du tableau de tête (de clés) : `SizHead`
- Un tableau de tête (de clés) : `Head`
- Le nombre d'objets : `NbrObj`
- Une liste d'objets (un ensemble de listes chaînées) : `LstObj`

Et, il faut les opérateurs:

- Initialisation et allocation : `hash_init`
- Trouve un objet dans la table de hachage : `hash_find`
- Ajoute un objet dans la table de hachage : `hash_add`
- Supprime un objet dans la table de hachage : `hash_suppr`

Table de hachage : le tableau des voisins en 2D



Les étapes sont les suivantes :

1. Choix de la clé de hachage : i_{min}, $i_1 + i_2$, une clé par arête, ...
2. Definition de la liste d'objets : 5 entrées $= (i_1, i_2, K_1, K_2, \texttt{Nxt})$
3. Initialisation et allocation de la table de hachage
4. Ajout des objets dans la tableau de hachage
   $\rightsquigarrow$ boucle sur les éléments puis les arêtes
5. Allocation du tableau des voisins : `TriVoi`
6. Construction des voisins
   $\rightsquigarrow$ boucle sur les éléments puis les arêtes