**Géométrie numérique**
**M1 I3D et IIRVIJ - 2018–2019**
**Travaux Pratiques**

# TP 3 : Lissage de maillages

## Exercice 1 : Lissage laplacien

Reprenez la structure de données *Half-Edge* des TP précédents. Ajoutez à la classe `Vertex` une méthode `uniformLaplacian()` qui calcule l'opérateur Laplacien avec poids uniformes. Le Laplacien en un sommet $x_i$ est un vecteur défini par la formule :

$$L(x_i) = \frac{1}{\|N_1(i)\|} \sum_{j \in N_1(i)} (x_j - x_i) \tag{1}$$

Ajoutez à la classe `Mesh` une méthode `laplacianSmoothing()` qui lisse le maillage instancié en appliquant $k$ fois la formule suivante, $k$ et $\lambda$ étant des paramètres choisis par l'utilisateur (à initialiser à 1 et 0.5 respectivement), pour tout sommet $x_i$ du maillage :

$$x_i \leftarrow x_i + \lambda L(x_i) \tag{2}$$

Attention à ne pas écraser les valeurs courantes des coordonnées des sommets lors du calcul séquentiel des nouvelles valeurs.

Testez votre méthode et regardez ce qui se passe quand on applique un nombre croissant d'itérations $k$. Vous pouvez aussi faire varier $\lambda$.

## Exercice 2 : Lissage de Taubin

L'algorithme de lissage $\lambda|\mu$, proposé par Gabriel Taubin en 1995 [1], a pour but d'éviter cette perte de volume. Il alterne une passe de lissage laplacien et une passe de "contre-lissage" :

$$x_i \leftarrow x_i - \mu L(x_i), \tag{3}$$

avec des valeurs $\lambda$ et $\mu$ bien choisies : $\mu$ doit être supérieur à $\lambda$ mais pas trop éloigné.
Ajoutez à la classe `Mesh` une méthode `taubinSmoothing()` qui lisse le maillage instancié selon cet algorithme.

Testez votre méthode. Que se passe-t-il si on fait trop d'itérations ?

## Exercice 3 : Bonus : flot de courbure moyenne

Ajoutez à la classe `Vertex` une méthode `cotangentLaplacian()` qui calcule l'opérateur Laplacien avec poids cotangents. Le Laplacien en un sommet $x_i$ est maintenant défini par la formule :

$$L(x_i) = \frac{1}{\sum_{j \in N_1(i)} cotan\alpha_{ij} + cotan\beta_{ij}} \sum_{j \in N_1(i)} (cotan\alpha_{ij} + cotan\beta_{ij})(x_j - x_i) \tag{4}$$

avec $\alpha_{ij}$ et $\beta_{ij}$ les angles comme sur la figure 1. Vous pouvez vous inspirer du code écrit lors du TP précédent.

Modifiez la méthode `taubinSmoothing()` pour prendre en compte cette nouvelle version du Laplacien. Testez et comparez avec la version précédente.

1. Gabriel Taubin, "A signal processing approach to fair surface design". In ACM SIGGRAPH, 1995.

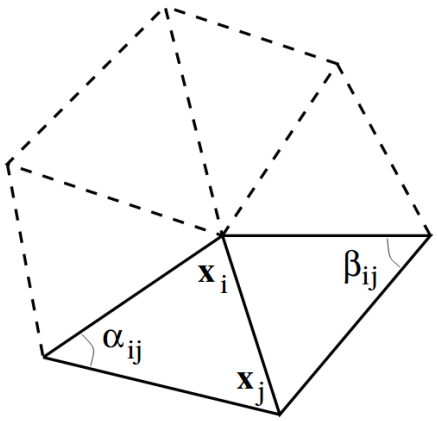


FIGURE 1 – [Meyer et al. 2003]