MEC6212: GENERATION DE MAILLAGES

Travail pratique: Triangulation de Delaunay contraint

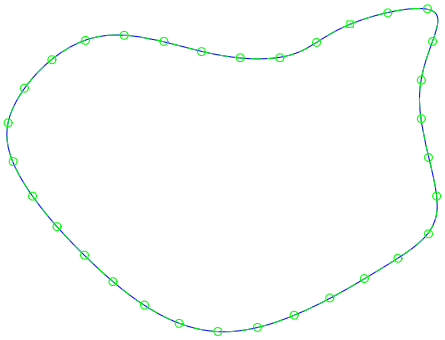
11 mars 2024

Énoncé

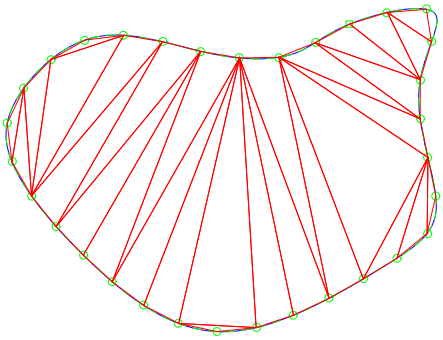
Dans ce travail on analyse un algorithme de triangulation basé sur la discrétisation de la frontière d'un domaine, c'est-à-dire une triangulation qui ne comprend que les sommets du bord du domaine. On obtient un recouvrement qui comprend explicitement ce bord, et qui pourra servir de maillage de départ pour un raffinement utilisant le noyau de Delaunay.

On appelle ce type de maillage un mailage Delaunay contraint, et ainsi on peut obtenir un maillage final qui comprendra les frontières du domaine.

Discrétisation du domaine



Recouvrement initial



Le point de départ est une discrétisation de la frontière par une approximation polygonale qui donne une liste d'arêtes en 2D, ou une liste de faces triangulaires en 3D. Ces arêtes constituent un front initial qui sera la base pour la construction des éléments (triangles ou tétrahèdres).

On fait progresser le front vers l'intérieur du domaine en ajoutant des triangles intérieurs jusqu'à la fermeture du front, soit jusqu'à ce qu'il ne reste plus dans le front que trois arêtes ou quatre arêtes, qui feront l'objet d'un traitement particulier.

Les éléments sont construits en utilisant seulement les noeuds et arêtes sur les frontières. L'ordre de la construction se fait selon l'angle le plus petit soutendu au noeud par les arêtes .

Le résulat est une triangulation qui :

— n'utilise que des noeuds (arêtes) frontière ;

— vérifie et inclut correctement la géométrie et topologie du domaine à mailler ;

— est un recouvrement du domaine ;

— est ou peut être transformé en un maillage Delaunay.

Le but de ce devoir est de **compléter** la fonction **delFRNT3** qui réalise cette triangulation.

1 Intégration de la fonction delFRNT3

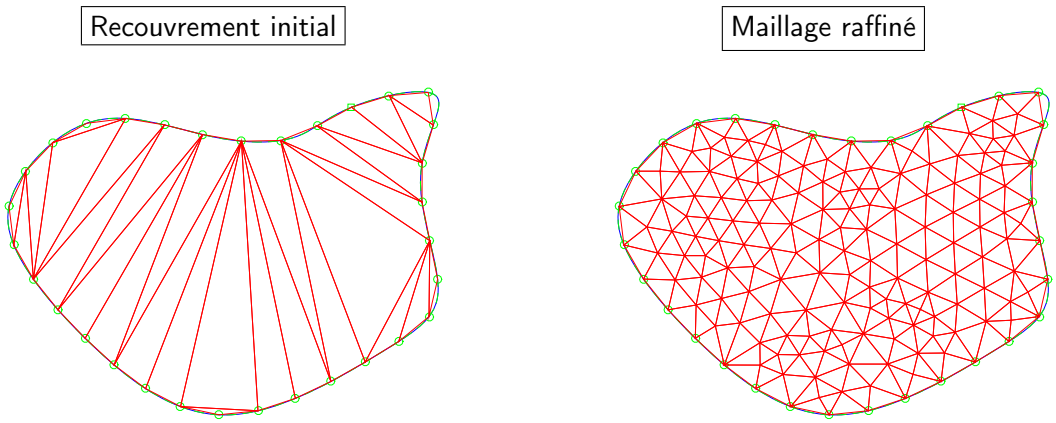
Les différentes étapes de la production d'un maillage Delaunay contraint sont :

Étape	**Fonction**	**Description**
Front initial	**FRNTconstr(iFACE)**	Pour chaque face du domaine, on construit un front constitué des arêtes de la discrétisation des frontières du domaine
Triangulation initiale	**delFRNT3**	Construit des triangles sur les sommets du front en choisissant le sommet soutendant le plus petit angle, ce qui donne une triangulation initiale de type Delaunay contraint.
Cible du raffinement	**cible=max(FRNTlong)**	La fonction `FRNTconstr` calcule la longueur de chacune des arêtes du front. Cette variable servira de critère de raffinement des triangles du maillage
Raffinement du maillage	Noyau de Delaunay	Récursivement, on raffine les éléments dont la taille dépasse la cible, et on applique le noyau de Delaunay.

```
%--------------------
%--------------------
% Initialisation
%--------------------
%--------------------
for iFACE=1:nbFACE
    FACE(iFACE,5)=nbELM+1;%------------------------ premier ELM de la FACE
    FACE(iFACE,7)=nbNOD+1;%------------------------ premier NOD de la FACE
    FACE(iFACE,9)=nbARE+1;%------------------------ premier ARE de la FACE

    FRNTconstr(iFACE)%---------------------- construction du FRNT initial

    cible=max(FRNTlong(1:FRNTactif));%------------ taille cible des ELM

    delFRNT3%---------------------- Construction du recouvrement initial

    [rayon,maxELM]=max(ELMgeo(1:nbELM,3));%-------------------- ELM cible

    while rayon > cible
        nbNOD=nbNOD+1;%---- insere un nouveau NOD au centre de maxELM
        S1 = E(maxELM,1);
        S2 = E(maxELM,2);
        S3 = E(maxELM,3);
        x(nbNOD)= .33333*(x(S1) +x(S2)+x(S3));
        y(nbNOD)= .33333*(y(S1) +y(S2)+y(S3));
        [CaviteELM,nbELMcavite,CaviteSOM,nbSOMcavite]=caviteEvS_SOM(nbNOD);

        bouleEvS(nbNOD,CaviteSOM,nbSOMcavite,CaviteELM,nbELMcavite);
        [rayon,maxELM]=max(ELMgeo(1:nbELM,3));
    end
    FACE(iFACE,6)=nbELM - FACE(iFACE,5)+1;%nb d'elements sur iFACE
    FACE(iFACE,8)=nbNOD - FACE(iFACE,7)+1;%-nb de noeuds sur iFACE
    FACE(iFACE,10)=nbARE - FACE(iFACE,9)+1;%-nb d'aretes sur iFACE
end
```

FIGURE 1 – Algorithme global d'un mailleur de type Delaunay contraint

Le maillage initial est un recouvrement utilisant uniquement les sommets et aêtres des frontières. Ce maillage est ensuite raffiné en insérant un sommet au centre de l'élément cible en appliquant le noyau de Delaunay. On applique une étape de lissage avec le bouton Lissage donnant le maillage final.



2 Les structures de données

2.1 Le front initial

A partir de la discrétisation de la géométrie, on construit un front, composé d'une liste de positions liées (chaînées), ordonnées dans le sens anti-horaire du parcourt du polygone (frontière). La structure de données décrivant le front est représentée par la structure suivante,

FRNT(1 :FRNTdim,1 :6)

tel que montré au Tableau 1, et où *FRNTdim* est égal au nombre de positions sur le front.

FRNT(posFRNT,1)	Sommet de la position	posSOM
FRNT(posFRNT,2)	Position prècèdente	preFRNT
FRNT(posFRNT,3)	Position suivante	suiFRNT
FRNT(posFRNT,4)	Sens de iARE	0 : contre 1 : dans le sens de FRNT 2 : de-activé
FRNT(posFRNT,5)	Arête correspondante	iARE
FRNT(posFRNT,6)	Référence à la liste des positions actives de *FRNT*	

TABLE 1 – Structure de l'entité *FRNT*

Pour une position donnée, *posFRNT*, les paramètres *FRNT(posFRNT,1 :6)* représentent la géométrie, et la connectivité de celle-ci relative à ses voisins (i.e. les positions précédente et suivante), et le maillage (l'arête). On illustre à la Fig.2.

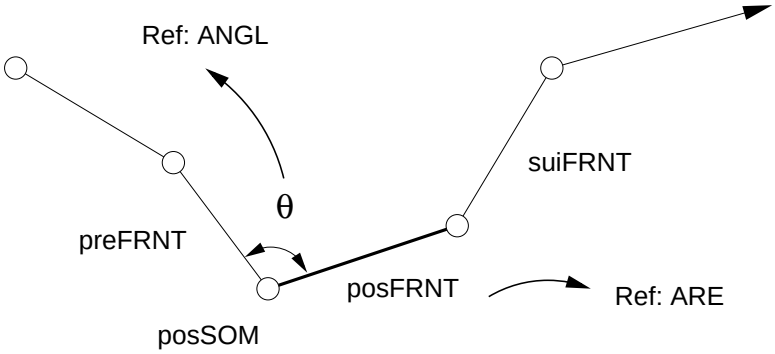


FIGURE 2 – Définition de l'entité *FRNT*

Le front initial est construit par la fonction **FRNTconstr(iFACE)** à partir de la discrétisation des frontières du domaine, tel qu'illustré à la Fig. 3.

$$faces => boucles ==> bords$$

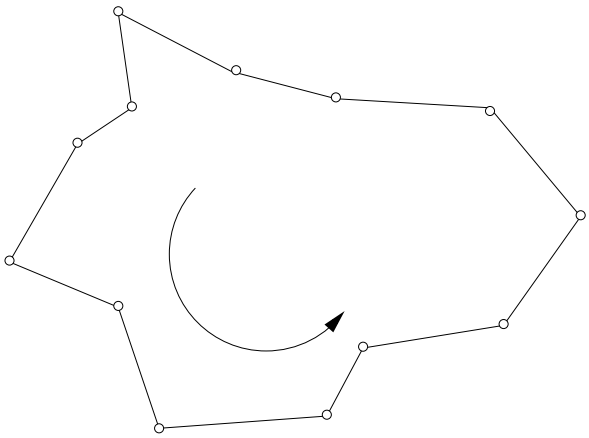


FIGURE 3 – Front initial

Initialement le front coincide avec le polygone représentant la frontière et comprend un nombre de positions égal à *FRNTdim*.

Le front dans sa progression laisse une trace qui n'est plus active car avec la création des nouvelles positions et les mises à jour des positions précédentes et suivantes font en sorte que certaines positions ne sont plus référencées (Voir Fig. 4). Sur le plan informatique, il n'est pas nécessaire de les effacer car elles sont effectivement désactivées. Donc, la dimension de *FRNT* ne change pas. Par contre la partie active du front qui au départ était égale à *FRNTdim*, diminue et tend vers zéro.

Les propriétés géométriques de la partie active du front, la longueur des arêtes et des angles soutendus aux sommets sont stockés dans les tableaux *FRNTlong*, *FRNTangl* et *FRNTindex*, tel que décrites au Tableau 2.

FRNTlong(iPOS)	Longueur de l'arête iARE à la position du *FRNT*
FRNTangl(iPOS)	Angle soutendu au sommet de la position du *FRNT*
FRNTindex(iPOS)	Référence à la position sur *FRNT*.

TABLE 2 – Géométrie du front actif

où *iPOS=1 :FRNTactif*, le nombre de positions actives du front.

On maintient dynamiquement ces listes,

FRNT, ***FRNTangl***, ***FRNTlong*** et ***FRNTindex***

au fur et à mesure de la progression du front.

Les éléments *FRNT(posFRNT,6)* et *FRNTindex* sont des références qui constituent le lien entre le front (*FRNT*) et le front actif (*FRNTangl*).

FRNT(posFRNT,6) ==> vers une position active de FRNT ==> *FRNTangl*

et vise-versa,

FRNTindex ==> vers la position sur FRNT ==> *FRNT(posFRNT,6)*

2.2 Triangulation du front

La construction du maillage se fait à partir du sommet avec le plus petit angle, dans la liste

FRNTangl(1 :FRNTactif)

donné par

[,posFRNT]=min(FRNTangl(1 :FRNTactif))

où *posFRNT* représente la position de cet angle et *FRNTactif* est le nombre de positions actives du front.

On construit un triangle avec,

```
nbELM=nbELM+1;
E(nbELM,1:3) = [preSOM,posSOM,suiSOM];
ELMconstr(nbELM)
```

en utilisant la localisation montrée à la Fig. 4.

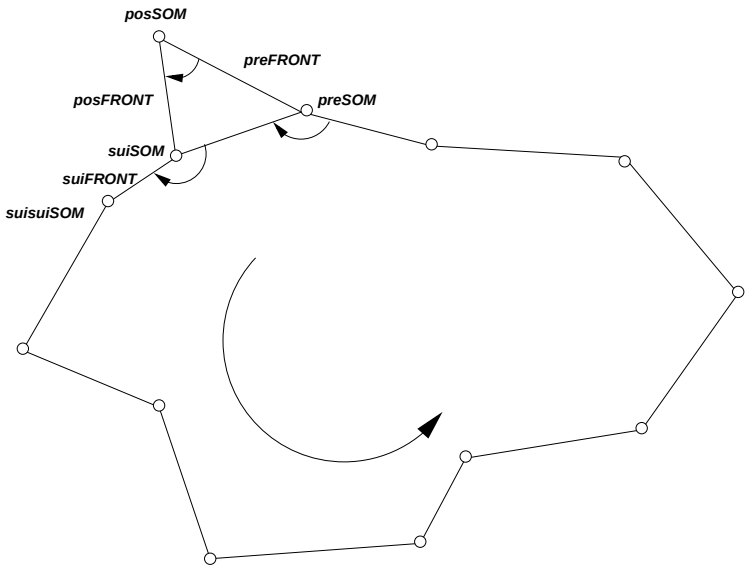


FIGURE 4 – Construction d'un élément sur le sommet avec le plus petit angle

Au départ, le front et le front actif sont identiques, et de dimension *FRNTdim* et *FRNTactif*, respectivement. Par la suite, alors que le front progresse, *FRNTdim* ne change pas, tandis que *FRNTactif* diminue et tend vers zéro ; on a alors obtenu le recouvrement initial.

On note qu'au fur et à mesure que des triangles sont contruits sur les positions du front, celles-ci demeurent dans la structure, mais ne sont plus référencées.

2.3 Mise à jour du front

Suite à la construction d'un triangle, on doit faire la mise à jour du front, c-à-d du tableau *FRNT*, qui consiste à enlever deux positions et à en insérer une nouvelle, comme montré à la Fig. 5. Sur le plan informatique, et en pratique, on recyclique une position et la seconde sera implicitement retirée car cette position ne sera plus référencée suite aux modifications suivantes.

1. On modifie (recyclique) la position *posFRNT* qui prend la place de la "nouvelle" position ;
   - Le sommet devient *preSOM* ;
   - La position précédente de *posFRNT* devient *prepreFRNT* ;

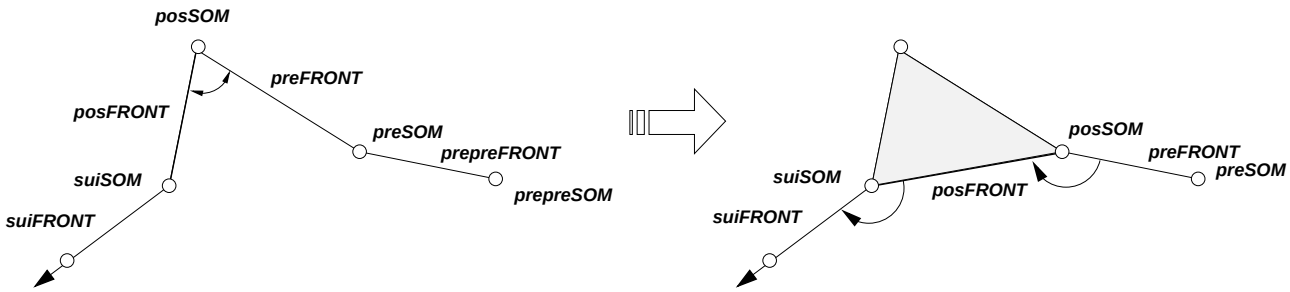


FIGURE 5 – Mise à jour du front, *FRNT*

2. La position suivante de *prepreFRNT* est modifiée par *posFRNT* ;
3. Mise à jour de la géométrie du front actif, c-à-d la liste des angles, *FRNTangl* :
   - on calcule les angles aux nouvelles positions avec la fonction **angleSOM**,
     angleSOM(som0,som1,som2)
     et on les insère dans la liste les angles du front actif. Par exemple, en référence à la Fig. 5,
     FRNTangl(posANGL)= angleSOM(preSOM,posSOM,suiSOM)
     et
     FRNTangl(suiANGL)= angleSOM(posSOM,suiSOM,suisuiSOM)
   - on retire un angle dans la liste des angles *FRNTangl* avec la fonction **retireANGL**,
     Spécifiquement, pour l'exemple de la Fig. 5,
     retireANGL(preANGL)
     La version *.p de ces fonctions peuvent être utilisées et ne sont pas à remettre
   - La gestion des références *FRNTindex* vers le *FRNT*
     FRNT(posFRNT,6) ==> FRNTangl

     et vise-versa,

     FRNTindex ==> FRNT(posFRNT,6)
     est réalisée par la fonction **retireANGL**, et n'est pas à remettre
   - On note que dans ce travail les longueurs, *FRNTlong* ne sont pas utilisées.

3 La fonction delFRNT3

L'algorithme de la fonction **delFRNT3** est décrit schématiquement à la Fig. 6 qui comprend essentiellement les étapes suivantes :

Localisation de posFRNT On localise le sommet avec le plus petit angle où sera construit l'élément ;

Partie finissante avec un triangle Analyse du front et test si celui-ci se referme avec trois positions. Alors, on construit un élément. Le script à la Fig. 7 donne le détail de cette étape.

Partie finissante avec un quadrangle Analyse du front et test si le front se referme avec quatre positions. Cette configuration donne lieu à deux possibilités de connectivité. On appliquera celle qui vérifie le critère de la sphère vide.

Construction d'un nouvel element Si le front n'est pas finissant, on construit un élément candidat. En appliquant le critère de la sphère vide, deux situations sont possibles,

- le triangle candidat est vide, alors le triangle est construit, suivi de la modification du front.
- le cercle du triangle circonscrit n'est pas vide, alors le front est scindé en deux parties.

Deux parties de cette fonction sont à compléter, comme indiqué dans le squelette de la Fig. 6.

Partie finissante avec un quadrangle

- Etudier à titre d'exemple, la Fig. 7 qui donne le détail pour la partie finissante (la fermeture du front) avec un triangle.
- Adapter pour la configuration d'une partie finissante avec un quadrangle en construisant deux triangles qui vérifient le critère de la sphère vide.

Construction d'un nouvel element La triangulation par avance de front pour construire le recouvrement consiste à former des triangles sur les positions du front.

On teste chaque triangle candidat pour s'assurer que le cercle circonscrit est vide, (Critère de Delaunay). On fera le cas où le cercle du triangle circonscrit est vide et on fait la mise à jour du front.

Le protocole d'appel ainsi que les variables en global pour **delFRNT3** se trouvent à la Fig. 6.

```
function delFRNT3(hObject, handles)
%-------------------------------------------------------------------------
global x y nbNOD ARE nbARE ARElong
global E nbELM ELMgeo
global FRNT FRNTdim FRNTactif FRNTlong FRNTangl  FRNTindex
%-------------------------------------------------------------------------
while FRNTactif > 0
    %------------------------------------------------- localisation de posFRNT
    [~,posANGL]=min(FRNTangl(1:FRNTactif));
    posFRNT=FRNTindex(posANGL);
    posSOM=FRNT(posFRNT,1);
    preFRNT=FRNT(posFRNT,2);
    preSOM=FRNT(preFRNT,1);
    prepreFRNT=FRNT(preFRNT,2);
    prepreSOM=FRNT(prepreFRNT,1);
    suiFRNT=FRNT(posFRNT,3);
    suiSOM=FRNT(suiFRNT,1);
    suisuiFRNT=FRNT(suiFRNT,3);
    suisuiSOM=FRNT(suisuiFRNT,1);
    if prepreSOM==suisuiSOM% partie finissante du FRNT forme un quadrangle
        ------------------------
        ------ votre code ------
        ------------------------
    elseif preSOM==suisuiSOM%- partie finissante du FRNT forme un triangle
        ------------------------
        ------ voir fig 6 ------
        ------------------------
    else%------------------------------------ cas general avec test de collision
        minFRNT=videELM2(preSOM,posSOM,suiSOM);
        if minFRNT==0%------------------------ le triangle candidat est vide
        ------------------------
        ------ votre code ------
        ------------------------
            afficheFRNT_EvS(hObject, handles)
            pause
        else%---- le triangle candidat non vide: on construit la diagonale
            %------------------------------------------ et le front est scinde
            FRNTdiag(minFRNT,preSOM,posSOM,suiSOM,posFRNT,preFRNT);
            afficheFRNT_EvS(hObject, handles)
            pause
        end
    end
end
```

FIGURE 6 – Squelette de la fonction **delFRNT3**

La Fig. 7 donne en exemple pour la partie finissante sur un triangle comme modèle pour réaliser la partie finissante avec un quadrangle.

```
elseif preSOM==suisuiSOM%partie finissante du FRNT forme un triangle
        nbELM=nbELM+1;
        E(nbELM,1:3) = [posSOM,suiSOM,preSOM];
        ELMconstr(nbELM)
        retireANGL(preFRNT)
        retireANGL(posFRNT)
        retireANGL(suiFRNT)
```

FIGURE 7 – Fonction **delFRNT3** : partie finissante avec un triangle

La Fig. 8 donne à titre indicatif la gestion de la liste du front actif.

```
function retireANGL(iFRNT)
global FRNT FRNTdim FRNTactif FRNTlong FRNTangl FRNTindex
FRNT(iFRNT,4)=2;%------------------------------- iFRNT de-active
iANGL=FRNT(iFRNT,6);%----deplace FRNTactif vers iFRNT==>iANGL
FRNTangl(iANGL)=FRNTangl(FRNTactif);
index=FRNTindex(FRNTactif);
FRNTindex(iANGL)=index;
FRNT(index,6)=iANGL;
FRNTactif=FRNTactif-1;
```

FIGURE 8 – Fonction **retireANGL** : Mise à jour du front actif

4 Mise en oeuvre

Les diverses fonctions constituant l'algorithme de Delaunay contraint sont appellées par le bouton **EvS ELMax** dans le panneau de la Fig. 9 lancé à partir du menu racine **Delaunay EvS**.

La variante qui est demandée pour ce travail est lancée par le bouton **Demo 3** qui exécute l'algorithme pas-à-pas.

Delaunay EvS => **Delaunay contraint EvS** =>	**Demo 1** **Demo2** **Demo 3** **EvS ELMax** **Lissage**
Informations =>	**Dessin**

FIGURE 9 - Panneau des boutons de fonctions pour l'algorithme de Delaunay contraint

1. Avec les modules **Géométrie** et **Domaine**, construire une configuration complexe qui serait difficile à mailler en structuré ; un domaine avec un trou.
2. Discrétiser la frontière avec la valeur uniforme par défaut.
3. Lancer l'algorithme (votre **delFRNT3**) avec le bouton **Demo 3**, et faire une copie pour fins de comparaison avec le bouton Dessin
4. Avec cette mème configuration, valider avec le bouton EvS ELMax.
5. Appliquer un lissage. Commenter le résultat.
6. Remettre dans un fichier identifiant.zip, le source *.m les résultats, comparaison et discussion