MEC6212: GENERATION DE MAILLAGES

Travail pratique: Triangulation de Delaunay

29 février 2024

Énoncé

Soit une distribution régulière de N sommets $\mathcal{S} = \{S_i\}_{i=1,\dots,N}$, et on suppose qu'il n'y a pas quatre sommets cocycliques, ni trois sommets colinéaires comme illustré à la Fig. 1.

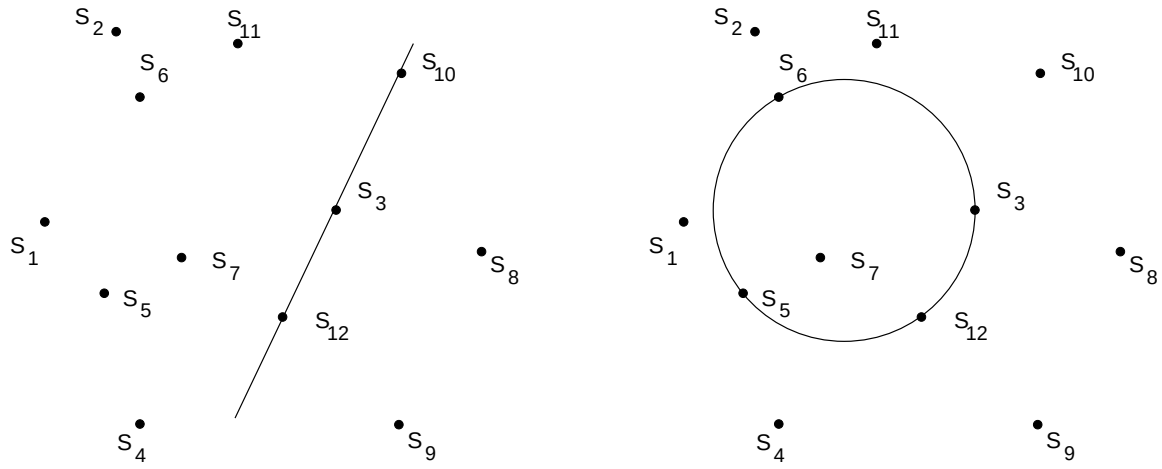


FIGURE 1 – Tessellation régulière en 2d

Le but de ce travail est de rédiger un algorithme de triangulation basé sur le critère de la sphère vide.

Spécifiquement, il s'agit de rédiger les quatre fonctions qui constituent le noyau de Delaunay :

Étape	**Fonction**	**Description**
Triangulation initiale	`boiteConstr`	Ajoute les 4 coins de la boîte à la liste des sommets du nuage de points à trianguler, et construit deux triangles.
Construit la cavité	`caviteEvS_SOM`	Pour chaque sommet `iNOD` qui s'ajoute à la triangulation, on forme la liste `CaviteELM` des `nbELMcavite` éléments de la cavité en appliquant le test de la sphère vide
Construit la boule	`bouleEvS`	On ajoute les éléments de la boule, en reliant les sommets du périmètre de la cavité au sommet `iNOD`
Triangulation finale	`boiteEnleve`	On enlève les éléments de la triangulation qui partagent un sommet avec la boîte initiale

FIGURE 2 – Étapes du noyau de Delaunay

1 Méthode incrémentale

On introduit une boîte qui englobe les sommets du nuage $\mathcal{S} = \{S_j\}_{j=1,\dots,N}$. On fait une triangulation de Delaunay initiale $\mathcal{T}_0$ avec les quatre sommets de la boîte englobante. Alors, les sommets du nuage $\mathcal{S}$ sont toujours inclus dans la triangulation courante.

La triangulation initiale La triangulation de Delaunay initiale $\mathcal{T}_0$ est composée de deux triangles formés avec les sommets de la boîte englobant le nuage de sommets $\mathcal{S} = \{S_j\}_{j=1,\dots,N}$.

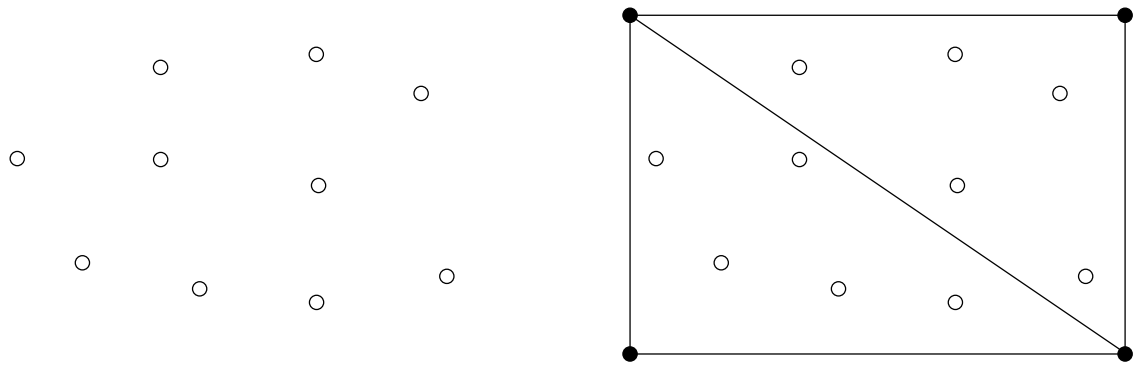


FIGURE 3 – Boîte et maillage de départ

Insertion d'un point : la cavité On forme la cavité en appliquant le test de la sphère vide à chaque élément de la triangulation courante $\mathcal{T}_i$. Le résulat est une liste d'éléments et une liste d'arêtes qui sont stockés dans les tableaux *CaviteELM* et *CaviteSOM*, respectivement.

Le tableau *CaviteELM(1 :nbELMcavite)* contient les indices des *nbELMcavite* éléments, et le tableau *CaviteSOM(1 :nbSOMcavite,1 :2)* contient les indices des sommets formant les arêtes des triangles précédents. Ce tableau contient des arêtes en double (celles qui sont partagées par deux triangles de la cavité) qu'il faut retirer de la liste de sorte que le résultat soit une liste d'arêtes qui forment le périmetre de la cavité.

La boule : La boule $\mathcal{B}_P$ est créée en connectant le point P aux sommets du périmètre de la cavité donnant la nouvelle triangulation $\mathcal{T}_{i+1}$.

Triangulation finale : Le noyau de Delaunay est appliqué de façon incrémentale jusqu'à l'épuisement des points $S_{j\,j=1,\ldots,N}$.

Le résultat est une triangulation qui contient, en plus des points du nuage à connecter, les sommets de la boîte initiale. Cet artifice simplifie l'implémentation du noyau de Delaunay sans perte de généralité.

À la fin, on enlève de cette triangulation les éléments dont au moins un des sommets est un sommet de la boîte.

Cette triangulation n'est pas nécessairement l'enveloppe convexe de $\mathcal{S}$. On s'assure que la triangulation sera Delaunay en prenant une boîte de dimensions suffisament grande.

2 Le noyau de Delaunay

Ces quatre fonctions constituant l'algorithme de Delaunay sont appellées par le bouton E ==> S dans le panneau de la Fig. 4 lancé à partir du menu racine **Triangulation**.

Triangulation =>	Tesselation =>	Créer	Nuage
		Déplacer	Déplacer
		Ajouter	Editer
	Triangulation Delaunay =>	Démo EvS	E ==> S
		Démo AvA	A ==> A
		Effacer	Voronoy
		Analyse	
	Traitement/Information =>	Imprime	Dessin

FIGURE 4 - Panneau des boutons de fonctions pour l'algorithme de Delaunay

Ce bouton exécute la fonction **TrgItnEvS_Callback** montrée à la Fig. 5, et applique l'algorithme du noyau de Delaunay tel que décrit à la Section 1, qui gère les appels aux différentes fonctions correspondant aux étapes suivantes,

— la construction de la boîte, **boiteConstr** ;
— l'insertion d'un nouveau sommet et l'application du noyau de Delaunay, **trgltn_EvS** ;
— le retrait de la boîte, **boiteEnleve**.

```
function trgltn_EvS_Callback(hObject,~, handles)
%-------------------------------------------------------------------
% Structure: EvS
% Mode; automatique
%-------------------------------------------------------------------
boite=[0,0,0,0];
iNOD=boiteConstr(boite,'auto','EvS');
trgltn_EvS(hObject, handles,'auto',iNOD)
boiteEnleve('EvS')
.......
```

FIGURE 5 - Algorithme global pour la triangulation d'un nuage de points

La fonction **TrgltnEvS** gère l'application du noyau de Delaunay avec les fonctions **caviteEvS_SOM** et **bouleEvS** pour l'insertion d'un nouveau sommet dans la triangulation existante.

```
function trgltn_EvS(hObject, handles,mode,iNOD)
global x y nbTSSLTN
........
........
while iNOD>0
 [CaviteELM,nbELMcavite,CaviteSOM,nbSOMcavite]=caviteEvS_SOM(iNOD);
 bouleEvS(iNOD,CaviteSOM,nbSOMcavite,CaviteELM,nbELMcavite)
 iNOD=iNOD-1;
end
.......
end
```

FIGURE 6 - Noyau de Delaunay: la cavité et la boule

2.1 Construction de la tessselation de points à trianguler

Le nuage de points est créé avec le bouton,

Triangulation ==> [Tesselation] ==> Creer

avec les diverses fonctions d'édition montrées à la Fig. 4.

Attention Ne pas utiliser le bouton **Nuage**.

2.2 Construction de la boîte initiale

À cette étape, on construit un maillage initial comprenant deux éléments qui englobent le nuage de points à trianguler.

iNOD=boiteConstr(hObject,handles,auto,'EvS')

qui prend en argument,

auto : le mode automatique de la construction de la boîte,
'EvS' : indique la structure de données utilisée.

et retourne,

iNOD, le premier sommet de la liste à insérer dans la triangulation

```
function iNOD=boiteConstr(boite,mode,struct)
%-------------------------------------------------------------------
%   CONSTRUCTION DE LA LA BOITE QUI ENGLOBE LES SOMMETS
%-------------------------------------------------------------------
global E nbELM
global x y nbNOD nbTSSLTN
global ARE AREbcl nbARE nbAREbrd ARElong
global PRM
nbELM=uint16(0);
nbARE=uint16(0);
PRM   = ones(10,6,'uint16');
% Calcul des coordonnees des quatre coins de la boite
% xg,xd,xd,xg et yg,yg,yd,yd
%-------------------------------------------------------------------
% a vous
%--------------- Ajoute les quatre sommets a la tesselation (x,y)
x=[x,xg,xd,xd,xg];
y=[y,yg,yg,yd,yd];
nbNOD=nbTSSLTN+4;
iNOD=nbTSSLTN+1;
%----------------- Maillage initial: constructioin et ajout 2 ELM
% a vous
%-------------------------------------------------------------------
```

FIGURE 7 - Squelette et protocole d'appel de la fonction **boiteConstr**

La taille de la boîte est calculée à partir de la différence des coordonnées des sommets extrèmes en x et en y, multipliée par des facteurs Fx et Fy, au choix.

A cette étape, on dispose des coordonnées ***x(1 :nbTSSLTN)*** et ***y(1 :nbTSSLTN)*** des ***nbTSSLTN*** sommets du nuage à trianguler. On initialize les variables suivantes :

— le nombre d'éléments : ***nbELM=uint16(0)*** ;

— le nombre de sommets : ***nbNOD=nbTSSLTN+4*** ;

— le premier sommet à insérer dans la triangulation : ***iNOD=nbTSSLTN-1***.

Cette fonction doit,

— Trouver les sommets extrêmes de la boîte rectangulaire qui englobe le nuage ;

— Ajouter ceux-ci à la liste des sommets x et y du nuage, saisis au curseur ;

— Construire le maillage initial comprenant deux ELM avec les quatre coins de la boîte.

2.3 Construction de la cavité

On construit la cavité en appliquant le critère de la sphère vide par la fonction **caviteEvS_SOM(iNOD)** qui consiste à trouver les triangles qui violent ce critère, c-à-d qui forment la cavité.
Ce qui donne :

liste des triangles , *ELM*, dont le cercle circonscrit inclue le sommet, *iNOD*, dans la variable :

CaviteELM(1 :nbELMcavite)

Comme illustré à la Fig. 8 ci-dessous.

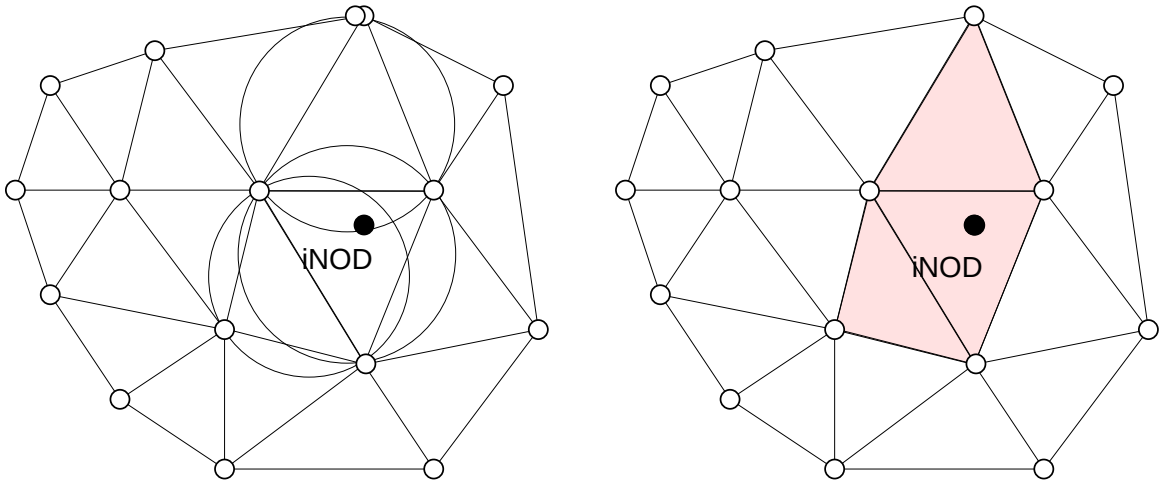


FIGURE 8 – Insertion d'un nouveau point dans le maillage et identification des éléments de la cavité :

liste des cotés , $S_i - S_j$ *de tous les triangles formant la cavité, dans la variable,*

CaviteSOM(1 :nbSOMcavite,1 :2)

CaviteSOM(1 :nbSOMcavite,1) pour le sommet S_i, et
CaviteSOM(1 :nbSOMcavite,2) pour le sommet S_j.

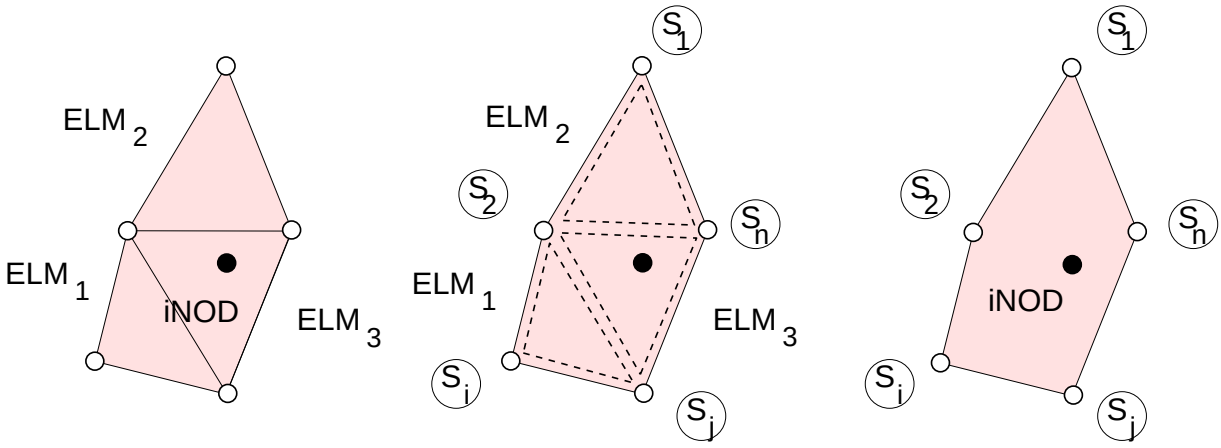


FIGURE 9 – Construction de la cavité

La cavité illustrée à la Fig. 9, il y a 9 cotés (3 par triangles). Pour la construction de la boule, on a besoin des cotés du périmètre de la cavité. Ceux-ci peuvent être obtenus en éliminant les doublons $S_2 - S_n$ et $S_2 - S_i$ de la liste des cotés ***CaviteSOM***.

L'ensemble de ces actions est réalisé par la fonction :

[CaviteELM,nbELMcavite,CaviteSOM,nbSOMcavite]= **caviteEvS_SOM(iNOD)**

qui prend en argument, ***iNOD***, l'indice du point du nuage à insérer dans la triangulation, et retourne :

CaviteELM : la liste des éléments formant la cavité ;

nbELMcavite : le nombre d'éléments de la cavité ;

CaviteSOM : la liste des sommets des cotés formant la cavité (c-à-d le polygone formant la cavité) ;

nbSOMcavite : le nombre de cotés de la cavité ;

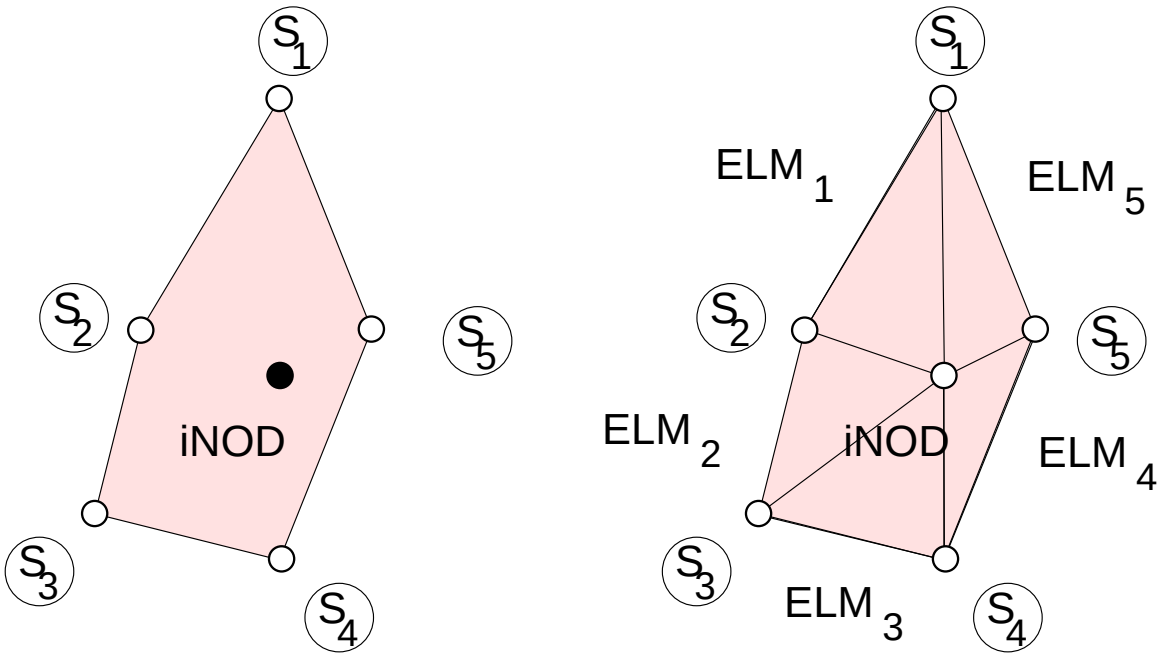
La Figure 10 donne un squelette de la fonction pour la construction de la cavité.

```
function [CaviteELM,nbELMcavite,CaviteSOM,nbSOMcavite]=caviteEvS_SOM(iSOM)
%-------------------------------------------------------------------------
%  Pour tous les elements actifs, on teste si le point iSOM est a l'interieur
%-------------------------------------------------------------------------
global E nbELM ELMgeo
global x y nbNOD

CaviteSOM    = ones(10,2,'uint16');%-------------- initialize la Cavite
nbSOMcavite  = uint16(0);%--------------- nombre de  cotes de la cavite
CaviteELM    = ones(10,'uint16');
nbELMcavite  = uint16(0);%------------------ nombre de ELM de la cavite

for iELM=1:nbELM
        ..... a vous de jouer ........
end
```

FIGURE 10 - Construction de la cavité

2.4 Construction de la boule

Les éléments de la boule sont construits sur chaque coté, $S_i - S_j$, du périmètre de la cavité en connectant ces deux sommets au sommet ***iSOM*** à insérer



Dans cet exemple, nous avons ***nbSOMcavite=5*** et on forme l'élément ***nbELM*** sur le 3ième coté,

E(nbELM,1 :3) = [iNOD,S3,S4] ;

En principe, en appliquant le noyau de Delaunay, on retire les éléments de la cavité. Ceci donnerait des trous dans le tableau des éléments. En pratique, on recycle ces espaces mémoire pour stocker les nouveaux éléments de la boule, et seulement lorsqu'on aura utilisé tous les espaces mémoire de la cavité, on pourra ajouter des nouveaux triangles à la liste.

La construction de la boule est réalisée par la fonction,

bouleEvS(iNOD,CaviteSOM,nbSOMcavite,CaviteELM,nbELMcavite)

qui prend en arguments,

iNOD, ***CaviteSOM***,***nbSOMcavite***, ***CaviteELM***, et ***nbELMcavite***
calculés par la fonction précédente.

La Figure 11 donne un squelette de la fonction pour la construction de la boule.

```
function bouleEvS(iNOD,CaviteSOM,nbSOMcavite,CaviteELM,nbELMcavite)
%-------------------------------------------------------------------------
%  nbSOMcavite nouveaux elements sont inseres dans la liste
%  Ces ELM sont construits en reliant les  segments du
%  polygone de la cavite et l'insertion d'un nouveau sommet  iNOD
%-------------------------------------------------------------------------
global E nbELM
for iC=1:nbSOMcavite
    ......................
    votre code
    ......................
end
```

FIGURE 11 - Construction de la boule

2.5 Retrait de la boîte initiale

A cette étape, tous les points sont insérés dans la triangulation, qui comprend ***nbNOD=nbTSSLTN+4***. On retire tous les éléments qui contiennent un des quatre sommets de la boîte initiale, sachant que ceux-ci ont été ajoutés à la fin de la liste des sommets. La triangulation comprend alors ***nbNOD=nbTSSLTN***.

La fonction **boiteEnleve('EvS')**, prend en argument,

'EvS' qui indique la structure de données utilisée.

```
function boiteEnleve(struct)
%-------------------------------------------------------------------------
%  On visite chaque element actif et on verifie les 3 sommets
%  en les comparant avec les indices des 4 sommets de la boite.
%  Si on trouve, alors l'element est desactive/retire
%-------------------------------------------------------------------------
global E nbELM ELMgeo
global x y nbNOD nbTSSLTN
global ARE AREbcl nbARE nbAREbrd ARElong
              ....................................
              votre code
              ....................................
```

FIGURE 12 - Protocole d'appel de la fonction boiteEnleve

2.6 Construction d'un élément

La formation d'un nouvel élément, se fait en deux étapes :

1. *Création d'un nouvel élément nbELM avec les sommets Si,Sj et Sk*

```
nbELM=nbELM+1;
E(nbELM,1:3) =  [Si,Sj,Sk];
```

2. *Calcul des propriétés du cercle circonscrit par la fonction* ELMconstr

```
ELMconstr(nbELM)
```

Les sommets sont donnés dans le sens trigonométrique, et la fonction **ELMconstr(nbELM)** calcule diverses propriétés géométriques qui sont stockées dans le tableau **ELMgeo**

```
ELMgeo(iELM,1) = abcisse du centre
ELMgeo(iELM,2) = ordonnee du centre
ELMgeo(iELM,3) = rayon du cercle circonscrit
```

A titre d'information, la Fig. 13 donne le source pour le calcul des propriétés géométriques du cercle circonscrit d'un élément en fonction de ses trois sommets.

```
function ELMconstr(iELM)
%-------------------------------------------------------------------------
% Calcule le centre et rayon du cercle circonscrit du triangle iELM
%-------------------------------------------------------------------------
global E ELMgeo x y
S1=E(iELM,1);S2=E(iELM,2);S3=E(iELM,3);
x1 = x(S1);x2 = x(S2);x3 = x(S3);
y1 = y(S1);y2 = y(S2);y3 = y(S3);
matA = [2*x1 2*y1 1;
    2*x2 2*y2 1;
    2*x3 2*y3 1];
matB = [x1^2 + y1^2;
    x2^2 + y2^2;
    x3^2 + y3^2];
matX = -matA\matB;
ELMgeo(iELM,1) =-matX(1);%--------------------------- abcisse du centre
ELMgeo(iELM,2) =-matX(2);%--------------------------- ordonnee du centre
ELMgeo(iELM,3) =sqrt(matX(1)^2 +matX(2)^2-matX(3));%------------- rayon
```

FIGURE 13 - Fonction ELMconstr

3 Exécution de l'algorithme

1. On utilise la structure $E \longrightarrow S$;
2. On créé un nuage de points, avec

   Menu : Triangulation ==> [Points de controle]==>[Creer]

   Les $nbTSSLTN$ points sont à insérer dans la triangulation avec le programme de la Fig. 5, exécuté par le bouton,

   [Triangulation Delaunay] ==> [E==>S]

   en mode automatique, ou par,

   [Triangulation Delaunay] ==> [Demo EvS]

   en mode pas-à-pas de l'interface usager.
3. Dans la formation de la boule, le point à insérer est connecté à chaque sommet du périmètre de la cavité formant ainsi de nouveaux éléments dont certains vont remplacer les éléments de la cavité, et les autres s'ajouteront au maillage. Il faudra coder soigneusement afin d'éviter de laisser des éléments de la cavité (Ils doivent tous être remplacés par les éléments de la boule).

4 Critique de l'algorithme

Rédiger une analyse critique de la méthode de Delaunay :

1. en identifiant la fonction parmi celles à développer dans ce travail qui serait la plus exigeante en fonction de temps de calcul.
2. Proposer des piste(s) de solution ?
3. Pourquoi dans l'application du nouyau de Delaunay n'a-t-on pas formé un domaine ?
4. Tout autre sujet qui vous paraît pertinent.

Remettre le document *.pdf avec quelques exemples, ainsi que les quatre fonctions *.m dans un fichier *.zip.