UNIVERSITÉ LAVAL

GIREF
Groupe interdisciplinaire de recherche en éléments finis

Algorithme de détection de cassures

RAPPORT DE STAGE DE :

Marc-Antoine Massicotte
Département de génie électrique

Giref-rap année 2000

Marc-Antoine Massicotte
Génie électrique

Rapport de stage

Algorithme de détection de cassures

Directeurs de stage

Michel Fortin
Éric Chamberland

Année 2000

1. Algorithme de détection des cassures

1.1. Description

L'algorithme de détection des cassures doit trouver les changements de courbure sur un maillage de peau composé d'éléments triangulaires. L'objectif est donc d'identifier toutes les arêtes qui sont situées sur une cassures. L'algorithme effectue un test sur chacune des aretes du maillages pour determiner si l'arête est une cassure ou non.

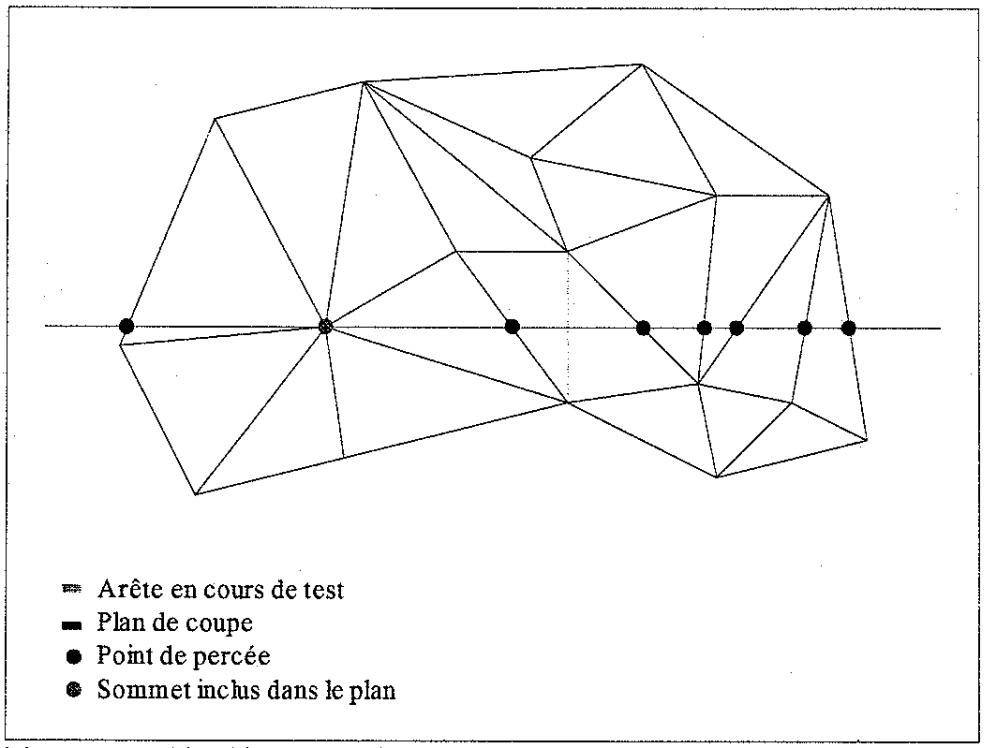


Figure 1 – Objectif de l'algorithme

La technique utilisée pour estimer la courbure autour d'une arête est illustrée à la figure 1. Dans un premier temps, un plan ayant comme normale l'arête en cours de test est construit. Ce plan passe par le centre de l'arête en cours de test. Ensuite, les éléments sont parcourus de chaque côté de l'arête en cours de test, à la recherche des éléments (arêtes) qui coupent le plan.

1.2. Algorithme

La première étape est de trouver l'équation du plan de coupe. Si P1 et P2 sont les sommets de l'arête en cours de test, alors :

$\vec{p}=\frac{(\vec{P}_1-\vec{P}_2)}{2}$ est le point central de l'arête en cours de test

$\vec{n}=(\vec{P}_1-\vec{P}_2)$ est la normale du plan de coupe

L'équation d'un plan est $\vec{n}\cdot\vec{x}+d=0$, le d est obtenu par l'équation :

$$d=\vec{n}\cdot\vec{p}$$

Lorsqu'on a l'équation du plan, on peut déterminer si le point $\vec{x}$ est inclus dans le plan, d'un côté du plan ou de l'autre :

$\vec{n}\cdot\vec{x}+d<0$ le point $\vec{x}$ est d'un côté du plan
$\vec{n}\cdot\vec{x}+d=0$ le point $\vec{x}$ est inclus dans le plan
$\vec{n}\cdot\vec{x}+d>0$ le point $\vec{x}$ est de l'autre côté du plan

Bref, on a les outils nécessaires pour déterminer si une arête perce le plan de coupe, il faut que les sommets de cette arête donnent des signes différents lorsqu'ils sont calculés à l'aide de l'équation du plan. À la figure 2, on voit l'analyse de la partie droite de l'exemple illustré à la figure 1, étapes par étapes :

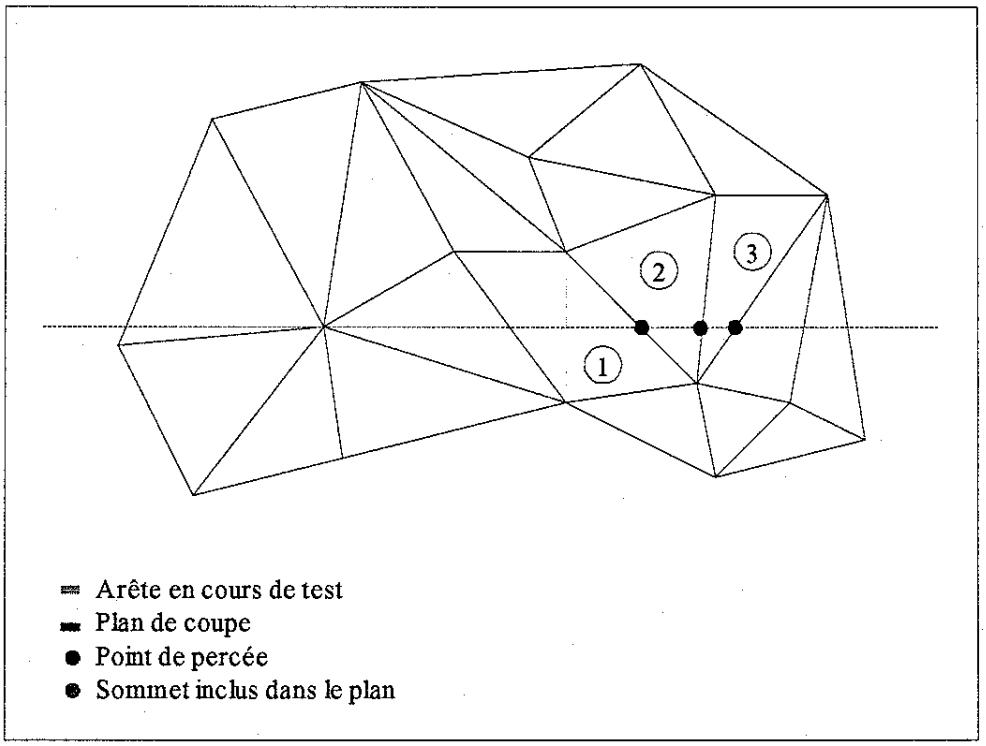


Figure 2 – Estimation de la courbure autour d'une arête

Premièrement, on récupère les deux éléments adjacents à l'arête en cours de test par connectivité inverse. Sur la figure 2, l'élément adjacent, noté par 1, permet d'avoir accès aux trois arêtes qui le compose. Il suffit de tester chacune des arêtes (sauf l'arête en cours de test) dans l'équation du plan pour obtenir celle qui perce le plan de coupe. Grâce à cette arête qui traverse le plan de coupe, il est possible d'obtenir l'élement 2 par connectivité inverse. De la même manière, on a accès aux trois arêtes de l'élément 2. Si on test les arêtes dans l'équation du plan, il est possible de trouver l'arête entre l'élément

2 et 3. Du même coup il est possible d'obtenir l'élément 3 par connectivité inverse. Bref, on peut trouver tous les éléments qui traversent le plan de coupe. Lorsqu'on visite un élément, il suffit de conserver sa normale dans une liste. C'est grâce à ces normales qu'il sera possible, plus tard, de déterminer l'angle entre les éléments.

Cette méthode marche bien pour les maillages non–réguliers, mais lorsqu'il y a symétrie cela cause un problème, car il arrive fréquemment qu'un sommet soit inclus dans le plan.

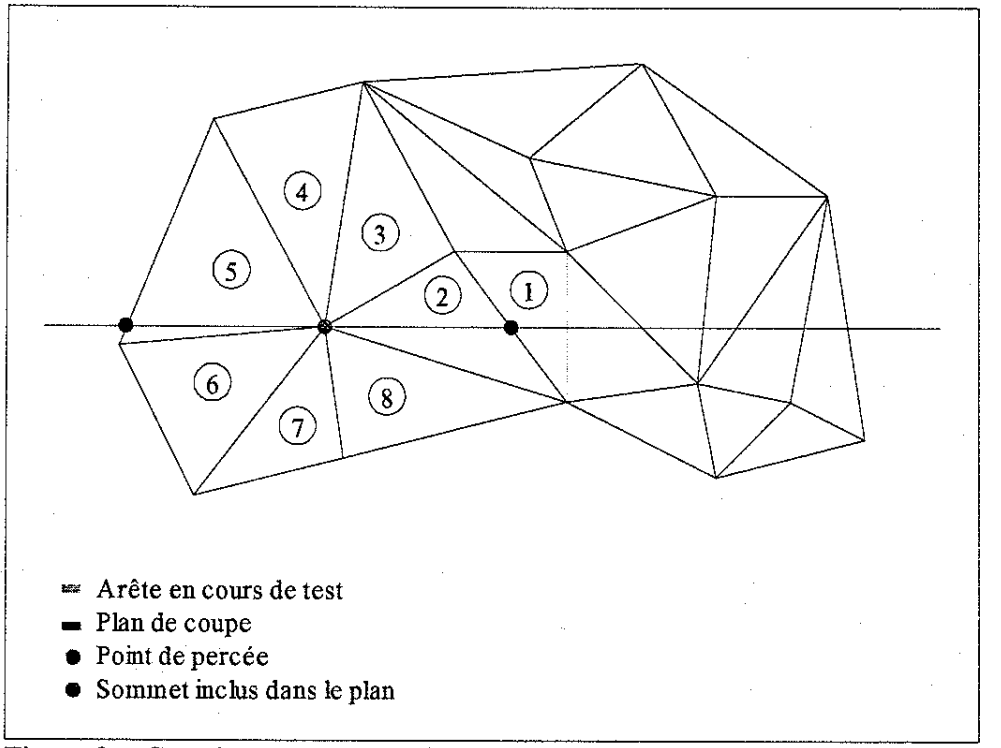


Figure 3 – Cas où un sommet est inclus dans le plan de coupe

Donc, il faut gèrer ce cas d'exception. Si aucune arête ne traverse le plan de coupe, il faut vérifier si le sommet de l'élément testé est inclus dans le plan. Ce cas est illustré à la figure 3. On voit bien que les arêtes de l'élément 2 ne permettent pas de continuer la recherche d'éléments, ce n'est possible qu'en balayant tous les éléments (3,4,5,6,7,8) autour du sommet inclus dans le plan. Encore une fois, il suffit de tester les arêtes de chacun des éléments dans l'équation du plan (sauf pour l'élément 2). Ainsi, il sera possible de continuer la recherche des éléments. À chaque élément trouvé, on garde sa normale dans une liste.

1.3. Transformation linéaire

Une fois que tous les éléments ont été trouvés, on a une liste de toutes les normales des éléments perpendiculaires à l'arête en cours de test. Ces normales permettront de calculer les angles entre chacun des éléments traversés par le plan de coupe. Pour calculer ces angles, il suffit de projeter les normales des éléments sur le plan de coupe. Cette projection est réalisée à l'aide d'une transformation linéaire. Dans un

premier temps, il faut se créer une base orthonormé dans le plan de coupe. Si $\vec{n}_1$, $\vec{n}_2$ sont les normales des éléments adjacents à l'arête en cours de test et $\vec{n}$ la normale du plan de coupe, alors on peut définir la base orthonormée suivante :

$$\vec{w}_x = \frac{\vec{n}_1 + \vec{n}_2}{|\vec{n}_1 + \vec{n}_2|}$$ coordonée *x* de la base orthonormale *w*

$$\vec{w}_y = \frac{\vec{n}}{|\vec{n}|} \cdot \frac{\vec{n}_1 + \vec{n}_2}{|\vec{n}_1 + \vec{n}_2|}$$ coordonée y de la base orthonormale *w*

La nouvelle normale $\vec{n}_{proj}$ se calcule à l'aide de la transformation linéaire suivante :

$$\vec{n}_{proj} = (\vec{n}_i \cdot \vec{w}_x)\vec{w}_x + (\vec{n}_i \cdot \vec{w}_y)\vec{w}_y$$

1.4. Calcul de l'angle

Une fois que toutes les normales ont été projetées, on calcule l'angle α entre les normales projetées de deux éléments adjacents ($\vec{n}_i$, $\vec{n}_{i+1}$). Cet angle ce calcule de deux façons :

1. $\alpha = \arccos(\vec{n}_i \cdot \vec{n}_{i+1})$ donne toujours le plus petit angle (sans signe)

2. $\alpha = \left|\arcsin\left(\left|\vec{n}_i \times \vec{n}_{i+1}\right|\right)\right|$ et le signe est donné par :

   $sign(\vec{n}_n \times (\vec{n}_g \times \vec{n}_d))$ où $\vec{n}_g$ et $\vec{n}_d$ sont respectivement les normales projetées des premiers éléments à gauche et à droite de l'arête en cours de test. $\vec{n}_n$ est la normale de l'élément le plus près de l'arête en cours de test entre $\vec{n}_i$ et $\vec{n}_{i+1}$.

2. Recherche des arêtes vives

Une fois que tous les angles sont calculés on doit déterminer si l'arête est vive ou non. Pour ce faire, plusieurs critères ont été développés :

- Recherche des courbures régulières (inverse d'une cassure)
- Recherche des cassures
- Algorithme de nettoyage
- Note sur les autres algorithmes envisagés

Toutes les méthodes se trouvent dans le fichier *PreTraitementGeometrie.cc* dans la fonction *detectionInitiale()*.

2.1. Recherche des courbures régulières

2.1.1. Description

La première méthode qui a été développée vise à isoler les cassures en détectant tout ce qui n'est pas une cassure, c'est à dire toutes les courbures régulières. Ce principe fonctionne relativement bien, mais détecte difficilement les très petits angles. C'est la méthode METHODE_COURBURE_REGULIERE implanté dans le code. Les calculs sont relativement simples :

1. On trouve les angles à l'aide des normales projetées :

$\alpha_{g_2}=\arccos(\vec{n}_{g_2}\cdot\vec{n}_{g_1})$	angle gauche 2
$\alpha_{g_1}=\arccos(\vec{n}_{g_1}\cdot\vec{n}_{g_0})$	angle gauche 1
$\alpha_c=\arccos(\vec{n}_{g_0}\cdot\vec{n}_{d_0})$	angle central
$\alpha_{d_1}=\arccos(\vec{n}_{d_0}\cdot\vec{n}_{d_1})$	angle droit 1
$\alpha_{d_2}=\arccos(\vec{n}_{d_1}\cdot\vec{n}_{d_2})$	angle droit 2

2. On cherche les angles les plus significatifs (le crititère du 75%) :

$si\left(\lvert\alpha_{g1}\rvert<0.75\lvert\alpha_{g2}\rvert\right)alors\,\alpha_G=\alpha_{g_2},sinon\,\alpha_G=\alpha_{g_1}$	angle GAUCHE
$dans\,tous\,les\,cas\,\alpha_C=\alpha_c$	angle CENTRAL
$si\left(\lvert\alpha_{d1}\rvert<0.75\lvert\alpha_{d2}\rvert\right)alors\,\alpha_D=\alpha_{d_2},sinon\,\alpha_D=\alpha_{d_1}$	angle DROIT

3. On calcule les angles moyens :

$$\hat{\mu}=\frac{\alpha_G+\alpha_C+\alpha_D}{3}$$

$$\hat{\mu}_G=\frac{\alpha_G+\alpha_C}{2}$$

$$\hat{\mu}_D=\frac{\alpha_C+\alpha_D}{2}$$

4. On évalue l'erreur par rapport aux angles moyens :

$$\Delta\varepsilon_G=\lvert\hat{\mu}-\alpha_G\rvert$$
$$\Delta\varepsilon_C=\lvert\hat{\mu}-\alpha_C\rvert$$
$$\Delta\varepsilon_D=\lvert\hat{\mu}-\alpha_D\rvert$$

5. On evalue l'erreur par rapport à une demi-courbure :

$$\Delta\varepsilon_{max_G} = max(|\hat{\mu}_G - \alpha_G|, |\hat{\mu}_G - \alpha_C|)$$
$$\Delta\varepsilon_{max_D} = max(|\hat{\mu}_D - \alpha_C|, |\hat{\mu}_D - \alpha_D|)$$

6. Ensuite on évalue les conditions nécessaires à l'identification des courbures régulières. Si δ est le seuil acceptable, on obtient les conditions booléennes suivantes :

$\beta_{reguliere} = (\Delta\varepsilon_G < \delta) \& (\Delta\varepsilon_C < \delta) \& (\Delta\varepsilon_D < \delta)$, courbe régulière
$\beta_G = (\alpha_G < \hat{\mu}) \& (\alpha_C < \hat{\mu})$, sous-moyenne gauche
$\beta_D = (\alpha_C < \hat{\mu}) \& (\alpha_D < \hat{\mu})$, sous-moyenne droite
$\beta_{demi} = (((\Delta\varepsilon_{max_G} < \delta) \& \beta_G) | (((\Delta\varepsilon_{max_D} < \delta) \& \beta_D)$, demi courbe régulière
$\beta_{petit\ angle} = (0.75|\alpha_C|) < min(|\alpha_G|, |\alpha_D|)$, petit angle central relatif
$\beta_{>90^\circ} = (\vec{n}_g \cdot \vec{n}_d) \leq 0$, angle plus grand que 90 degrée.

7. Décision finale, si la condition suivante est vraie on marque l'arête vive :

$$\beta_{vive} = (\beta_{reguliere}) \& (\beta_{demi}) \& (\beta_{petit\ angle}) \& (\beta_{>90^\circ})$$

2.1.2. Discussion

La méthode de détection des courbures identifie généralement les bonnes cassures. Par contre l'algorithme tend à ignorer les petites cassures, ce qui, dans certains cas, peut être bénéfique et dans d'autres cas légèrement insuffisant. Mais en général cette méthode donne de bons résultats.

2.1.3. Conclusion

Cette méthode peut être utilisée lorsque les maillages de peau sont bruité et qu'on ne désire pas identifier des très petits angles qui peuvent être associés au bruit. La méthode fonctionne et est particulièrement rapide.

2.2. Recherche des cassures

2.2.1. Description

C'est la dernière méthode qui a été développé et celle qui semble fonctionner le plus correctement. Cette méthode recherche un seuil linéaire relatif pour déterminer si l'arête en cours de test est vive ou non. Cette méthode à été développée par essais et

erreurs, les quelques paramètres qui la définie pourraient être sujet à optimisation. Les présents paramètres donnent de bons résultats. Cette méthode porte le nom de METHODE_CASSURE_INCLUSIVE dans le code. Les calculs sont relativement simples:

2. On trouve les angles à l'aide des normales projetées :

$\alpha_{g_2} = \arccos(\vec{n}_{g_2} \cdot \vec{n}_{g_1})$ angle gauche 2

$\alpha_{g_1} = \arccos(\vec{n}_{g_1} \cdot \vec{n}_{g_0})$ angle gauche 1

$\alpha_c = \arccos(\vec{n}_{g_0} \cdot \vec{n}_{d_0})$ angle central

$\alpha_{d_1} = \arccos(\vec{n}_{d_0} \cdot \vec{n}_{d_1})$ angle droit 1

$\alpha_{d_2} = \arccos(\vec{n}_{d_1} \cdot \vec{n}_{d_2})$ angle droit 2

3. On cherche les angles les plus significatifs (le crititère 95%) :

$si\left(|\alpha_{g1}| < 0.95|\alpha_{g2}|\right) alors\, \alpha_G = \alpha_{g_2}, sinon\, \alpha_G = \alpha_{g_1}$ angle GAUCHE

$dans\, tous\, les\, cas\, \alpha_C = \alpha_c$ angle CENTRAL

$si\left(|\alpha_{d1}| < 0.95|\alpha_{d2}|\right) alors\, \alpha_D = \alpha_{d_2}, sinon\, \alpha_D = \alpha_{d_1}$ angle DROIT

4. On calcule les conditions booléenes nécessaires pour déterminer si l'arête est vive ou non. Si δ est la tolérance acceptable, $\rho_{diviseur}$ le premier paramètre d'ajustement et $\rho_{plancher}$ le deuxième paramètre on obtient les conditions suivantes :

$\rho_{diviseur} = 10.0$ $\rho_{diviseur} \in]0, \infty$, valeur du paramètre 1 par défaut

$\rho_{plancher} = 0.02$ $\rho_{plancher} \in \left[0, \frac{\pi}{2}\right]$, valeur du paramètre 2 par défaut

$$\beta_G = \alpha_C > \delta\left(\alpha_G + \frac{\rho_{plancher}}{\delta} + \frac{1}{\rho_{diviseur}}\right)$$

$$\beta_D = \alpha_C > \delta\left(\alpha_D + \frac{\rho_{plancher}}{\delta} + \frac{1}{\rho_{diviseur}}\right)$$

7. Décision finale, si la condition suivante est vraie on marque l'arête vive :

$$\beta_{vive} = \left(\beta_G\right) | \left(\beta_D\right)$$

2.2.2. Discussion

Cette méthode tente d'ajuster avec une fonction linéaire le seuil de tolérance pour déterminer si l'angle central est plus grand que un des angles adjacents. C'est ce qui se rapproche le mieux de la définition qu'on se fait d'une cassure en tant qu'être humain. À la limite, la fonction d'ajustement pourrait être beaucoup plus complexe et se baser sur une définition plus précise, par exemple on pourrait se baser sur des statistiques moyennes issues de ce que l'humain considère comme étant une cassure ou non. À partir de ces résultats, on pourrait faire une meilleure modélisation.

2.2.3. Conclusion

La méthode fonctionne dans la plupart des cas, elle détecte généralement les petits angles que la méthode précédente ne détectait pas. Par contre, l'inconvenient c'est qu'il apparait des cassures là où il y a des "patches" générée par le logiciels de CAD. Présentement, les paramètres ont été ajustées de manière *ad hoc*, il serait intéressant d'approfondir sur ce type de modélisation des cassures.

3. Algorithme de nettoyage

3.1. Description

Les algorithmes de détection de courbure ne sont pas parfaits, il arrive souvent que certaines arêtes qui sont seules et sans voisin, sont détectées. L'algorithme de nettoyage balaye toutes les arêtes et vérifie si elle ont des voisins. Si une arête est marquée vive et n'a aucun voisin, elle est supprimée de la liste des arêtes vives.

3.2. Note

L'algorithme devrait être optimisé, présentement il balaye toutes les arêtes ~~vives~~, alors que logiquement il devrait uniquement boucler sur la liste des arêtes vives.

4. Note sur les autres algorithmes envisagés

4.1 Modélisation de la courbure par un cercle ou une ellipse

Plusieurs essais ont été tentés pour modéliser la courbure à l'aide de l'équation paramétrique du cercle. Les équations pertinentes ont été trouvées à l'aide de Maple 6.0.

Solution pour un cercle (3 points)

$$x_0 = r\cos(t_0) + O_x$$
$$y_0 = r\sin(t_0) + O_y$$
$$x_1 = r\cos(t_1) + O_x$$
$$y_1 = r\sin(t_1) + O_y$$

$$x_2 = r\cos(t_2) + O_x$$
$$y_2 = r\sin(t_2) + O_y$$

Code C généré par Maple 6.0 :

```
t1 = x2*x2;
t4 = y0*y0;
t7 = x1*x1;
t9 = y2*y2;
t11 = y1*y1;
t14 = x0*x0;
t19 = y0*t1-t1*y1-t4*y2+t4*y1-y0*t7+y0*t9-y0*t11+t11*y2-y2*t14-*t9+t14*
y1+t7*y2;
t27 = 1/(x2*y0-y0*x1-y1*x2+y2*x1-y2*x0+y1*x0);
t42 = -x1*t9+t9*x0-t4*x2+t4*x1+t11*x2+t14*x1-t14*x2-x0*t11+x2*t7-*t7+x0
*t1-t1*x1;
Ox = t19*t27/2.0;
Oy = -t42*t27/2.0;
```

Solution pour un cercle (tangente + 2 points)

$$\vartheta_0 = -\cot(t_0)$$

$$x_0 = a\cos(t_0) + O_x$$
$$y_0 = a\sin(t_0) + O_y$$
$$x_1 = a\cos(t_1) + O_x$$
$$y_1 = a\sin(t_1) + O_y$$

Code C généré par Maple 6.0 :

```
t3 = 2.0*theta0*y1*y0;
t4 = y1*y1;
t5 = theta0*t4;
t6 = x0*x0;
t7 = t6*theta0;
t8 = y0*y0;
t9 = theta0*t8;
t10 = x1*x1;
t11 = t10*theta0;
t12 = x0*y0;
t13 = 2.0*t12;
t14 = x0*y1;
t15 = 2.0*t14;
t20 = 1/(-x1*theta0+x0*theta0+y1-y0);
t23 = x1*y0;
t25 = 2.0*t23*theta0;
t27 = 2.0*t12*theta0;
t28 = x1*x0;
t29 = 2.0*t28;
t33 = theta0*theta0;
t45 = 2.0*y1*y0;
t46 = t6+t33*t4+t4+t33*t6+t8-2.0*t33*x1*x0-
          2.0*t33*y1*y0+t33*t8+t33*t10+t10-t29-t45;
t50 = sqrt(1/(1.0+t33));
t55 = x1*y1;
t60 = 1/(t6+t10+t4-t45+t8-t29);
t70 = atan2(-(-2.0*t14*theta0+2.0*t55*theta0-t25+t27-t29+t6+t10-t8+t45-
          t4)*t60*t50,-(-2.0*t28*theta0+t7+t11-t9+t3-t5+t15-
               2.0*t55+2.0*t23-t13)*t60*t50);
Ox = (t3-t5+t7-t9-t11-t13+t15)*t20/2.0;
Oy = (-t25+t27-t29+t6-t8+t4+t10)*t20/2.0;
a = t46*t20*t50/2.0;
t1 = t70;
```

Malgré plusieurs efforts, le problème suivant persiste toujours (définition d'une cassure) :

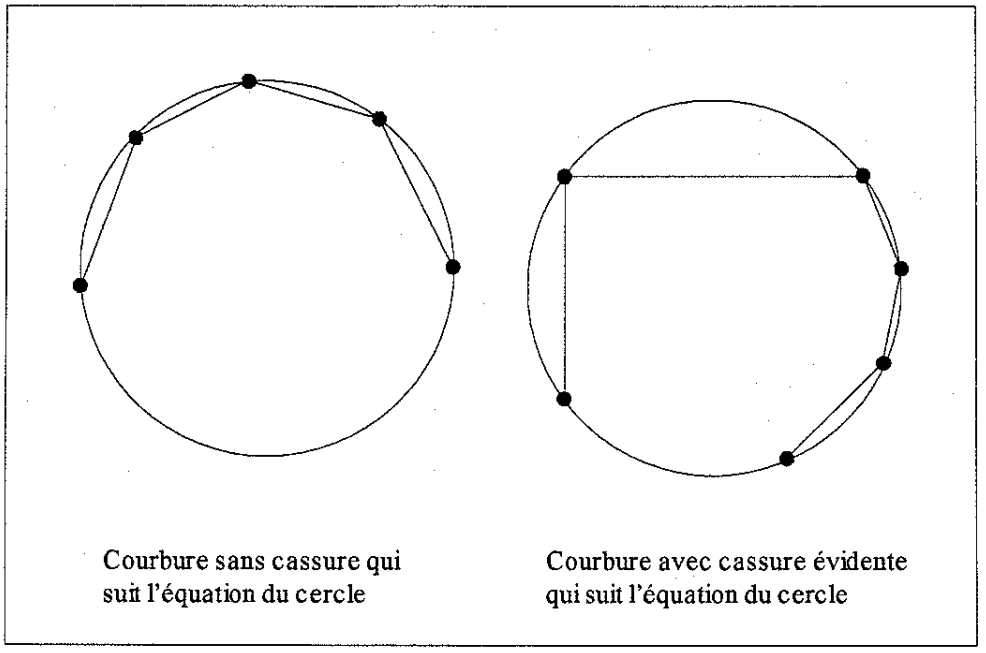


Figure 5 – Problème de modélisation avec l'équation du cercle.

On voit bien à la figure 5 qu'il y a des cas où le cercle ne peut pas nécessairement modéliser correctement une courbure. Il est assez difficile de définir ce qu'est exactement un cassure à l'aide d'un cercle.

4.2. Problème lié à l'évaluation de la courbure dθ/dS

La défintion mathématique d'une courbure est ~~défini par~~ dθ/dS. Il est donc logique de vouloir utiliser cette définition pour estimer le changement de courbure aux environs d'une cassure. Mais le problème illustré à la figure 6 ~~(sur la page suivante)~~ survient. En effet, lorsque dS tend vers zéro, dθ/dS devient très grand, même lorsque la courbure se rapproche de celle d'un plan. Les résultats sont bien en général, mais le taux de mauvaises détections est de beaucoup supérieur à la méthode de détection des courbures (première méthode).

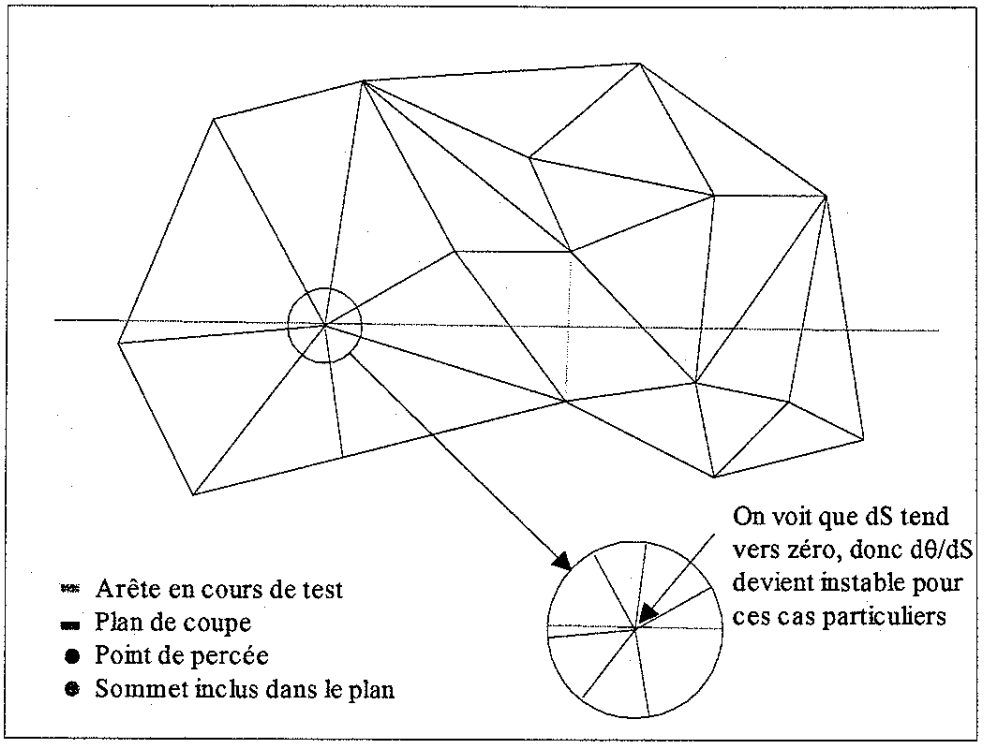


Figure 6 – Problème pour l'évaluation de dθ/dS

4.3. Problème lié à l'évaluation des dérivées et dérivées secondes

Il est difficile d'évaluer les dérivées, car la transformation des éléments quads en éléments triangles ajoute des arêtes qui contribuent à ajouter des angles plats qui sont habituellement très près de zéro. Il ne faut pas les détecter comme étant un changement de courbure, cela complique la tâche de détection des angles pertinants. Plusieurs test ont été tentés, même en éliminant en majeur partie les angles plats parasites (voir figure 7). Ces angles parasites peuvent être en grande partie éliminés par les relation suivante :

$si\left(\left|\alpha_{g1}\right|<0.75\left|\alpha_{g2}\right|\right) alors\, \alpha_G=\alpha_{g_2}, sinon\, \alpha_G=\alpha_{g_1}$ angle GAUCHE

$dans\ tous\ les\ cas\ \alpha_C=\alpha_c$ angle CENTRAL

$si\left(\left|\alpha_{d1}\right|<0.75\left|\alpha_{d2}\right|\right) alors\, \alpha_D=\alpha_{d_2}, sinon\, \alpha_D=\alpha_{d_1}$ angle DROIT

Bien que la plupart des angles parasites aient été éliminés, les résultats étaient plutôt étranges, c'est–à–dire très bruités avec peu de cohérence.

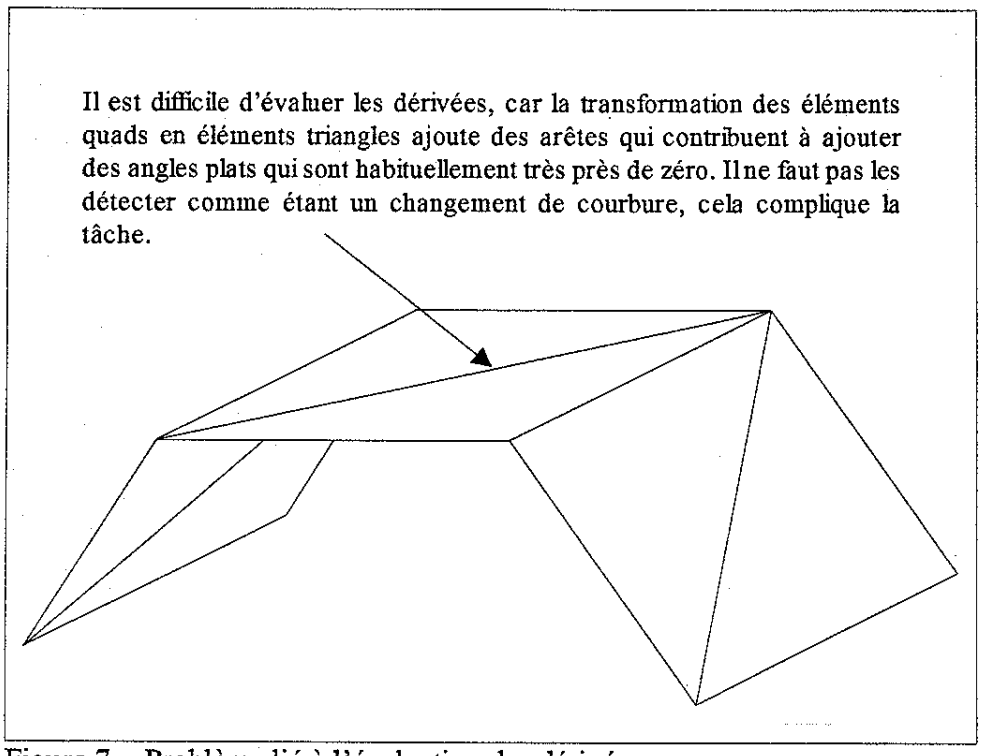


Figure 7 – Problème lié à l'évaluation des dérivées

4.4. Autre méthode d'estimation de la courbure

La méthode est intéressante et donne des résultats relativement bon, mais elle est moins bonne que les méthodes de détection des courbures/cassures (voir section 2.1 et section 2.2). L'évaluation de la courbure se fait avec la méthode suivante :

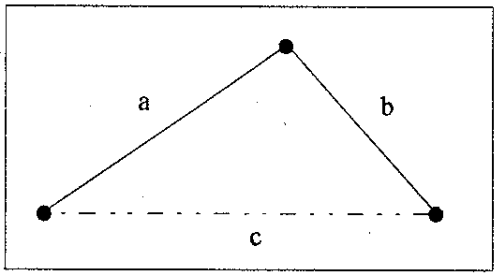


Figure 8 – Estimation de la courbure

L'estimation de la courbure se fait à l'aide de l'équation suivante:

$$\gamma = \left[1 - \left(\frac{c}{a+b} \right) \right]$$, 0 = Pas de courbure, 1 = Courbure infinie

4.5. Estimation de courbure à l'aide de splines

Plusieurs tests ont été tenté pour utiliser les splines dans l'estimation de la courbure, sans succès. Principalement à cause d'un problème conceptuel relié à l'estimation de l'erreur ou du changement de courbure. Un outil visuel intéressant sur le web permet de tester différentes situations:

http://www.eml.hiroshima-u.ac.jp/~nis/javaexampl/java_eng.html

Cette alternative a été mise de côté, car il n'est pas évident de trouver le critère pour déterminer si l'arête en cours de test est vive ou non. Toutes les tentatives ont donné des résultats de peu d'intérêt et moins bon que les algorithmes décrits aux sections 2.1 et 2.2.

4.6. Estimation à l'aide de l'algorithme de cabossage

Idée abandonnée. Semble trop complexe et temps de développement trop important.

4.7. Estimation à l'aide d'une surface

Temps d'exécution beaucoup trop lent. Problème conceptuel dans la détermination du critère pour marquer les arêtes vives.

5. Calcul des points de percé dans le plan de coupe

$$P_1=(x_1,y_1,z_1)$$
$$P_2=(x_2,y_2,z_2)$$
$$\overrightarrow{P_1P_2}=(x_2-x_1,y_2-y_1,z_2-z_1)$$

$$x=x_1+(x_2-x_1)t$$
$$y=y_1+(y_2-y_1)t$$
$$z=z_1+(z_2-z_1)t$$

$P=(x,y,z)$, point de percé, l'inconnue est t

$$ax+by+cz+d=0$$
$$(ax_1+by_1+cz_1)+[a(x_2-x_1)+b(y_2-y_1)+c(z_2-z_1)]t+d=0$$

$$t=\frac{-d-(ax_1+by_1+cz_1)}{a(x_2-x_1)+b(y_2-y_1)+c(z_2-z_1)}$$

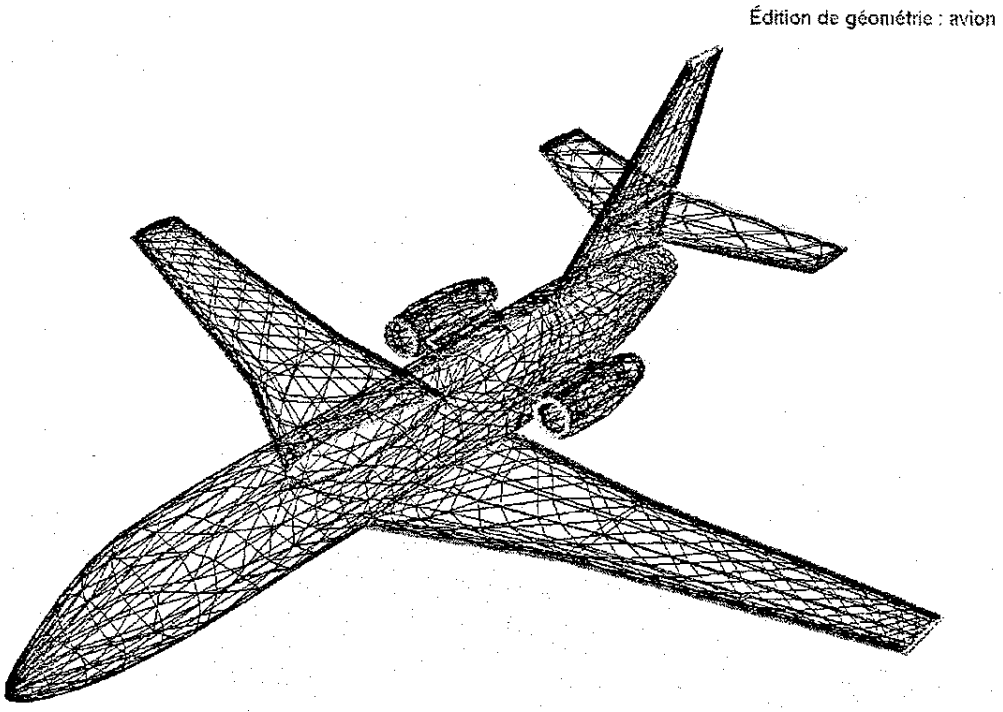
Le point de percé peut être convertit en coordonnée du plan de coupe à l'aide de la transformation linéaire décrite à la section 1.3.

6. Discussion

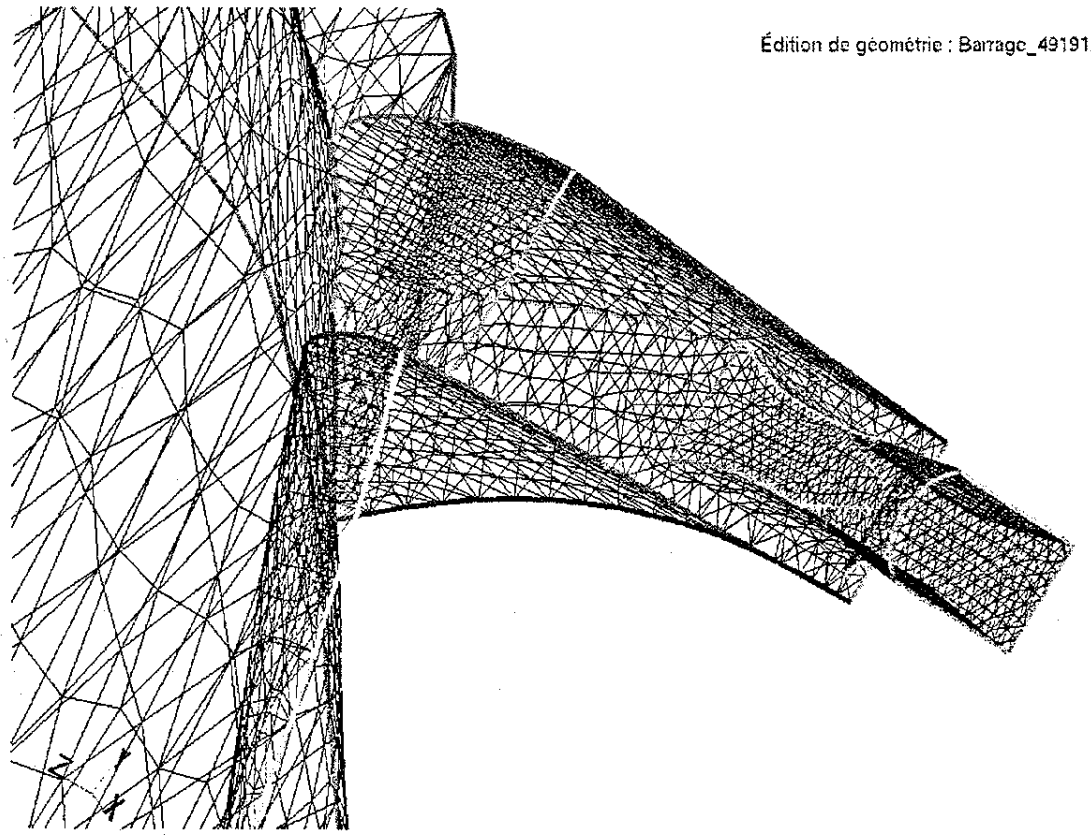
La méthode retenue est celle décrite à la section 2.2. L'algorithme de détection des cassures basé sur un seuil de tolérance qui varie selon une fonction linéaire pour déterminer si l'angle central est plus grand que un des angles adjacents. C'est la méthode, qui présentement, fonctionne le mieux dans la plupart des cas. Cependant, pour certains maillages, il est préférable d'utiliser une variante. Cette variante peut être appellé en modifiant la variable *lMethode* dans la fonction *detectionInitiale()* par METHODE_CASSURE_EXCLUSIVE. L'inconvenient à cette variante c'est qu'elle ne détecte pas deux cassures distantes d'un seul élément. En général, la méthode est appropriée.

7. Conclusion

Après quelques tests et en moyenne 5 à 10 minutes de nettoyage manuel sur les maillages, on obtient les résultats suivants :

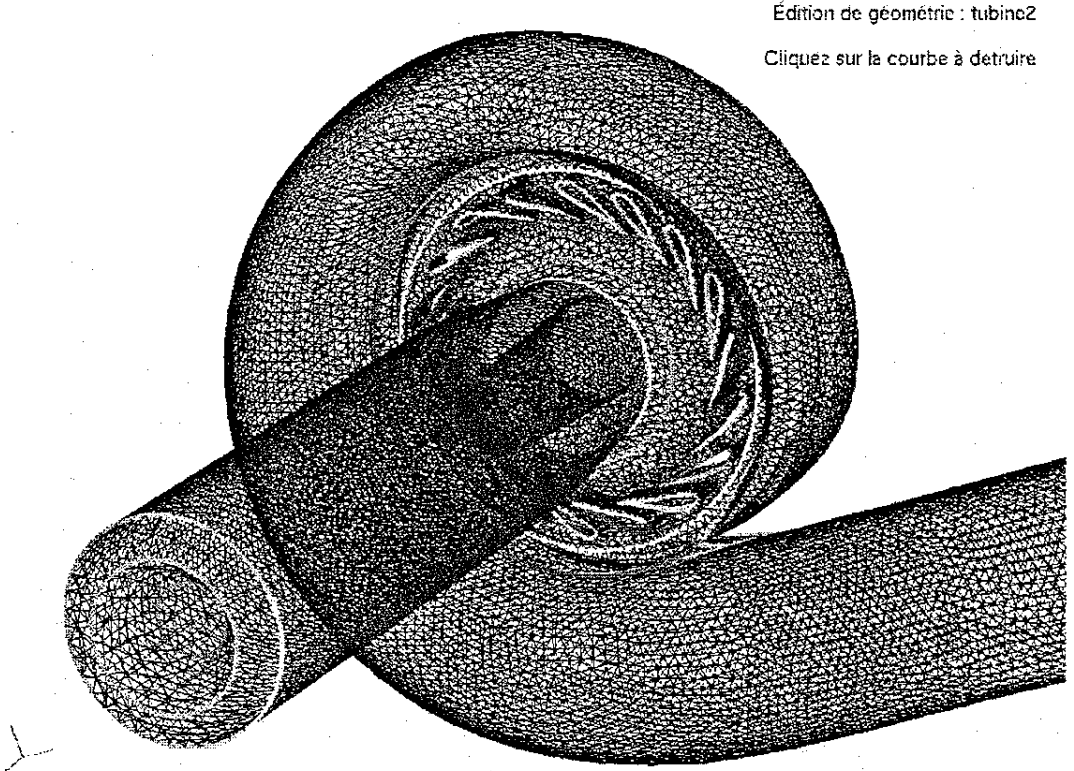


Édition de géométrie : Barrage_49191E

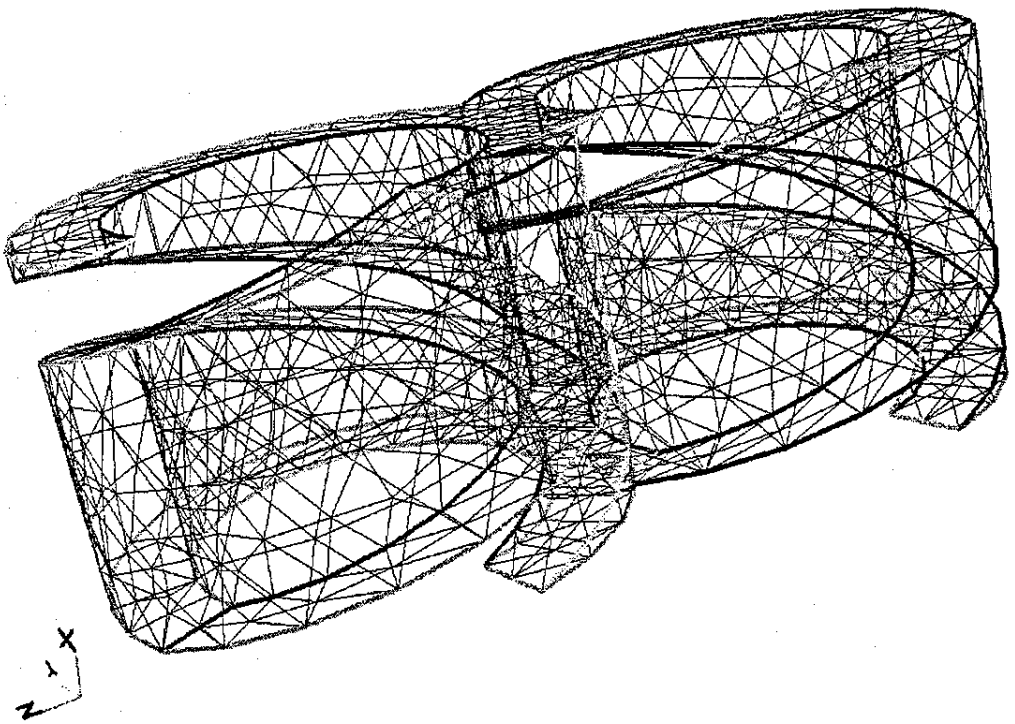


Édition de géométrie : tubine2

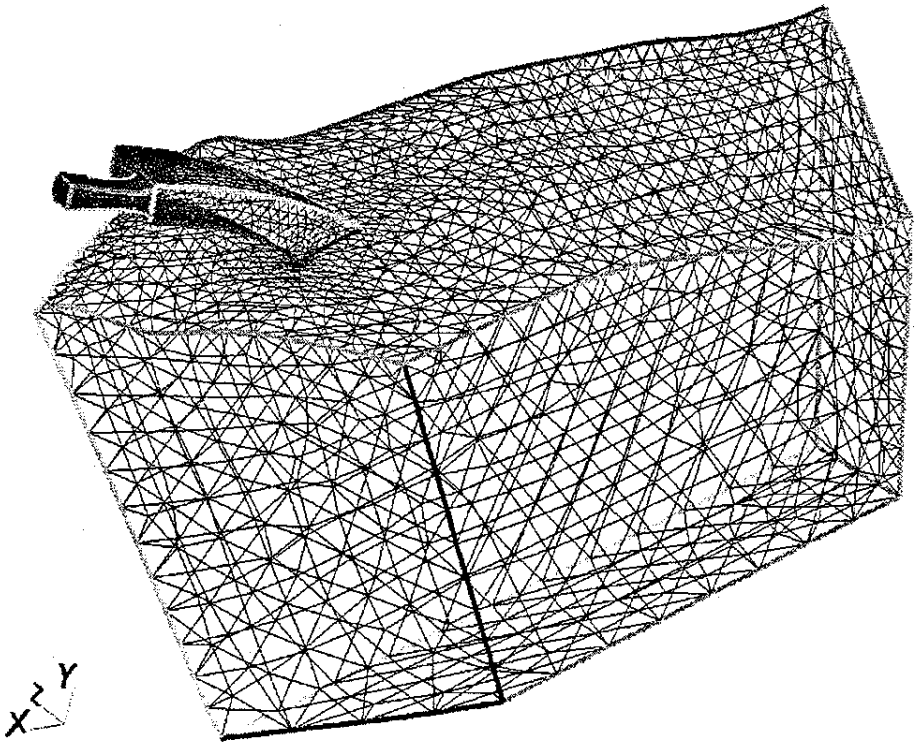
Cliquez sur la courbe à detruire

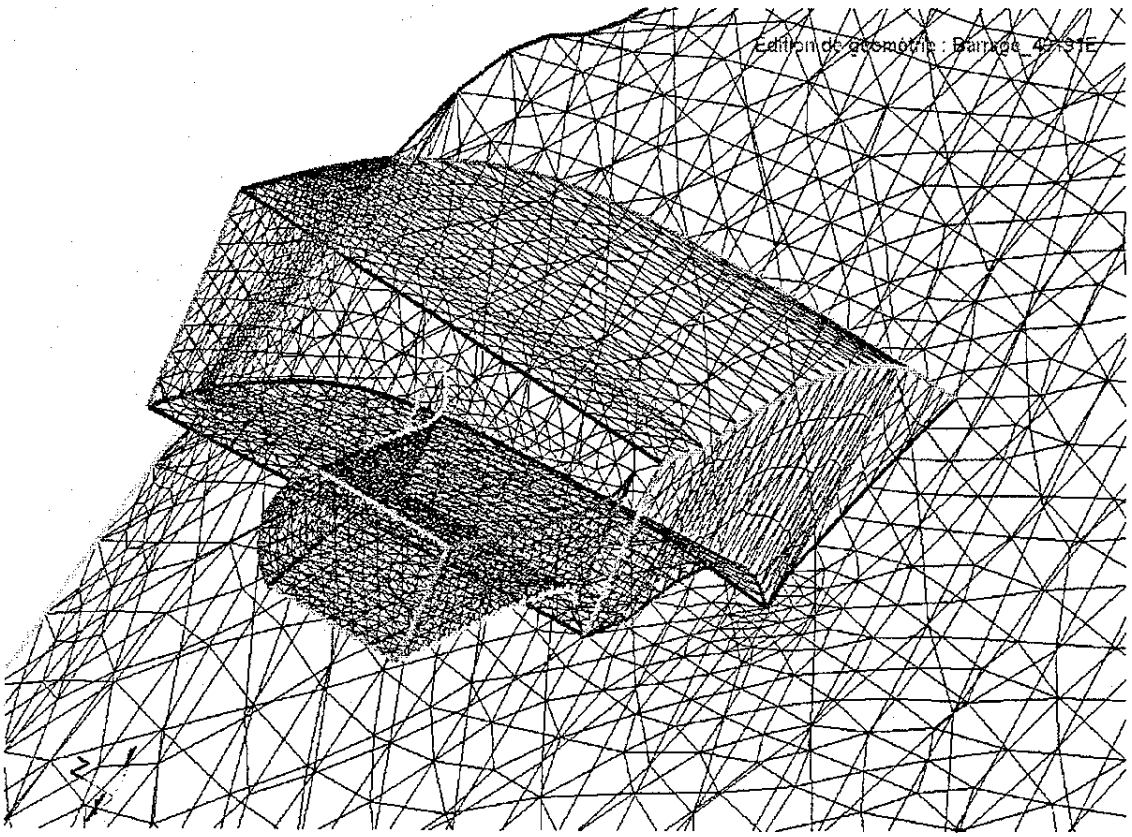


Édition de géométrie : aDim_26

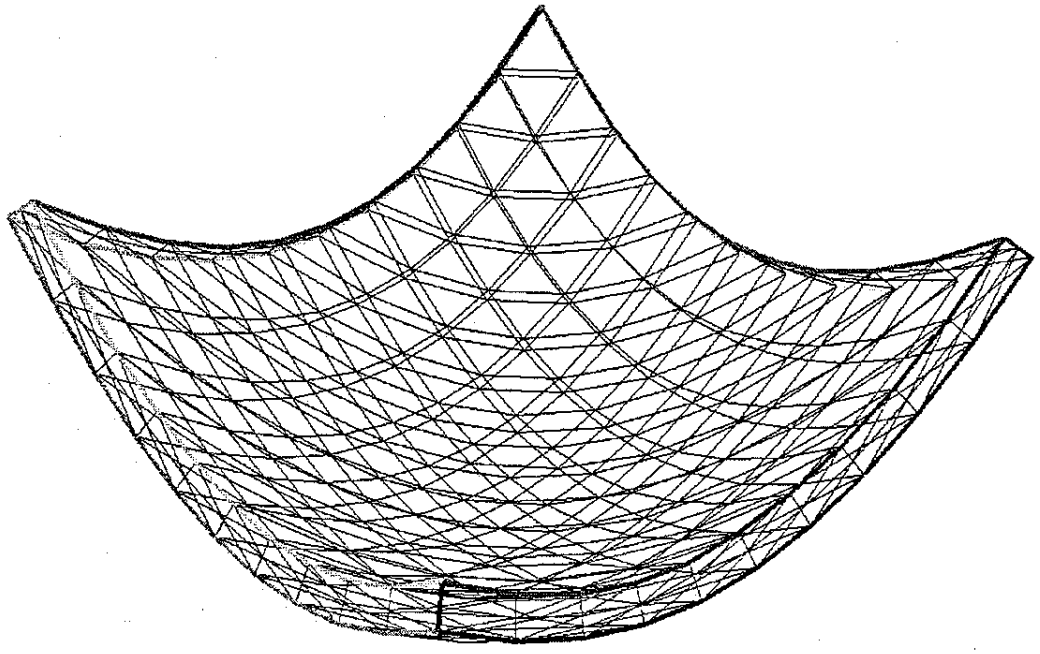


Édition de géométrie : Barrage_49191E

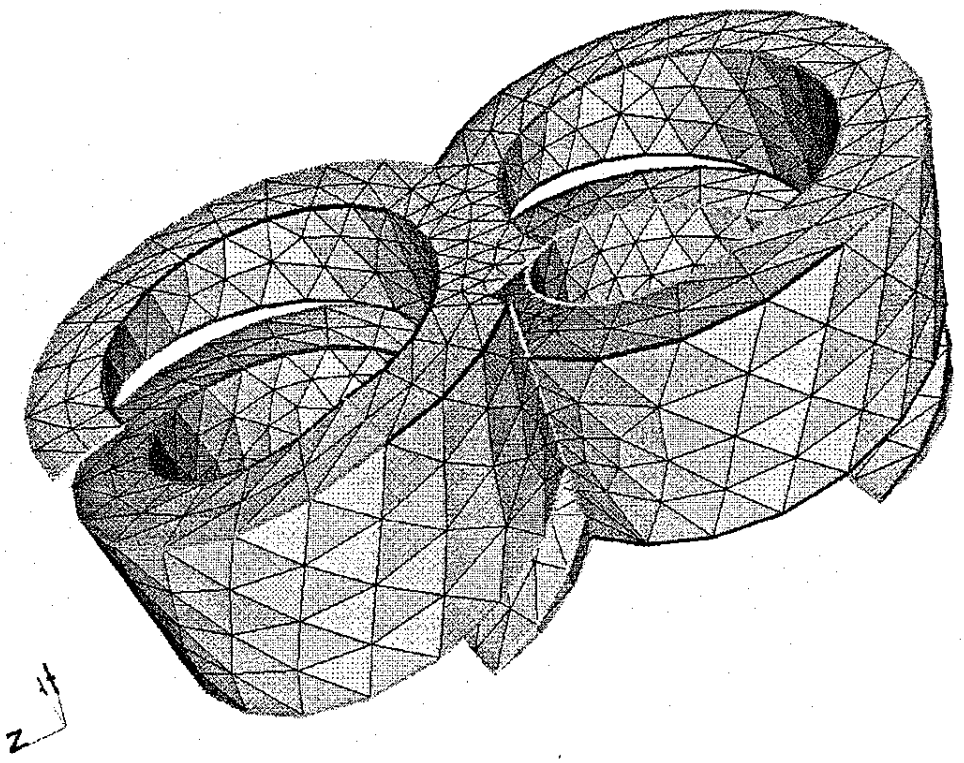




Édition de géométrie : cloche (extrude)



Édition de géométrie : aDim_26



Édition de géométrie : tubine2