ENSAM PARIS

Dépliage de structures gonflables

ZAMBAITI Mario Sergio

17/06/2016

Tuteur de projet : M.Monteiro

TABLE DES MATIERES

INTRODUCTION

Le dépliage de structures gonflables est la méthode qui nous permet la division d'une structure gonflable en un nombre fini de laizes qui sont découpés sur des matériaux plats et qui sont plus tard assemblés entre elles pour reconstruire une fois l'assemblage est gonflé la structure initiale.

La construction de structures 3D avec l'assemblage d'éléments 2D très fins est présente depuis le début de l'histoire de l'humanité, une de ses applications les plus connues et anciennes est la construction de vêtements avec des tissus découpes convenablement. Bien que celle-ci soit l'application la plus développée de ces méthodes, nous allons orienter la démarche à des structures gonflables qui sont caractérisés par être des surfaces fermées, différence principale avec les vêtements.

Le but de ce projet est de réaliser une démarche laquelle à partir d'une structure gonflable 3D, réalisé sur un logiciel de CAO, nous donne la forme des laizes que nous devrons découper afin d'obtenir avec son assemblage la structure initiale.

Le phénomène de raides sur les ballons aérostatiques est provoqué par un dépliage non parfait de sa structure 3D crée à l'aide d'un logiciel CAO. Ce dépliage imparfait provoque que lors de l'assemblage des laizes, les longueurs et les angles entre deux laizes voisines ne sont pas exactement complémentaires. La méthode du WireWarping[1] développé par Charlie et Wang est intéressante parce qu'elle propose un dépliage en conservant les longueurs clés du modèle 3D à déplier. De cette façon il est possible que le résultat de cette méthode nous solutionne plus facilement le problème des raides pendant l'assemblage comparé à d'autres méthodes où la longueur du modèle 3D n'est pas conservée.

ETAT DE L'ART, le WireWarping

Le WireWarping[1] ou dépliage de courbes, est une méthode qui à l'aide d'un algorithme nous permet d'obtenir une approche des laizes 2D résultantes du dépliage d'une surface 3D, cette méthode garde constantes les longueurs de certaines courbes appelées courbes de forme.

Dans cette méthode nous allons partir d'une géométrie surfacique 3D connue et de laquelle nous disposons son fichier CAD. Cette surface 3D va être divisé en laizes 3D qui après seront dépliées en devenant des laizes 2D. Le but de cette méthode est de pouvoir reproduire la géométrie 3D après l'assemblage les laizes 2D et son gonflage.

Définitions

Pour la compréhension de cette méthode, nous allonsappeler courbes 3D les lignes sur la forme 3D, et courbes 2D celles qui peuvent être contenues dans un plan 2D (une fois réalise le dépliage des courbes 3D nous aurons des courbes 2D). Quelques termes qui seront utilisés dans l'explication de la méthode et qu'il faut expliquer sont:

- Les courbes de contour : les courbes en noir dans la figure 1, celles-ci sont les courbes qui délimitent les différentes laizes à déplier ainsi que les limites de la surface 3D et elles vont conserver sa longueur et sa forme (angle) après le dépliage.
- Les courbes de forme : ces courbes 3D sont des courbes représentatives de la surface 3D à déplier et qui vont devenir des courbes 2D après ce dépliage. Nous désignerons comme courbes de forme clés celles en rouge dans la figure 1, elles sont représentatives de la surface 3D et elles se caractérisent parce qu'elles vont conserver sa longueur et sa forme (angle) après le dépliage. Les courbes de contour auxiliaires en vert dans la figure 1, seront moins représentatives de la surface 3D et vont conserver sa longueur mais elles ne conserveront pas ses angles après le dépliage. La différence entre ces angles 3D et 2D sera optimisée.
- Les courbes de coupe : ces courbes qui sont en bleu dans la figure 1, sont des courbes qui sont coupés sur la forme de laize et donc doublées pour que le dépliage soit plus effectif. Elles deviendront des courbes de contour dans la séparation de la surface 3D en laizes 3D.

Toutes ces courbes seront appelées courbes ou « wires » pendant le reste du document et c'est au concepteur de les fixer avant le dépliage.

- Les wire-patchs : ou aussi appelés patchs, sont les surfaces qui se trouvent dans une laize et qui sont délimitées par des courbes ou « wires ». Il y a au moins un patch par laize et les patchs 3D seront dépliés en des patchs 2D, résultant en la forme finale de la laize 2D. Dans la figure 2, nous voyons une laize qui est divisé en différents patchs (chacun d'une couleur différente).
- Sur les courbes qui séparent les patchs nous retrouvons des vertex v, ces vertex sont la position qui correspond à un ou plusieurs nœuds et chacun d'un patch différent (omme on voit dans la figure 2),la distance entre ces vertex est fixée par la courbe 3D et restera constante.

- Tout le long des patchs, on retrouve des nœuds q_j^i ou l'indice i fait référence au patch P_i qui contient ce nœud et l'indice j sert comme index entre tous les points sur ce patch, à chaque nœud lui correspond son vertex noté par $v(q_j^i)$.
- Pour trois nœuds consécutifs $q_{j-1}^i, q_j^i, q_{j+1}^i$ dans le même patch nous retrouvons l'angle 3D α_j^i qui peut être calculé sur la courbe 3D, donc cet angle est une donnée de la méthode de dépliage. L'angle 3D α_j^i deviendra l'angle 2D θ_j^i sur le patch 2D une fois la structure dépliée.
- Les angles 2D θ_j^i sont le résultat de notre méthode dépliage et son calcul sera expliqué plus tard.
- Pour la distance entre deux nœuds q_j^i et q_{j+1}^i est appelé longueur l_i et est une donnée calculé à partir des courbes du modèle 3D. La particularité de cette méthode est que ces longueurs ou distance entre nœuds seront les mêmes avant et après le dépliage.

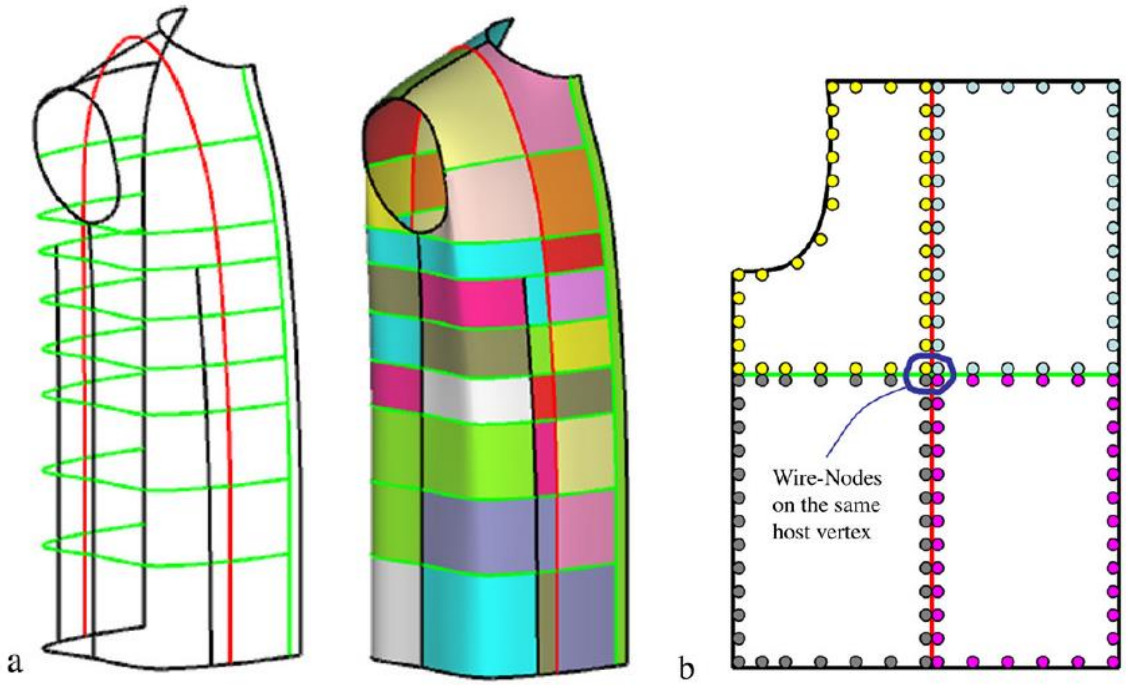

Figure 1 A gauche, représentation des courbes de contour sur une structure 3D, au milieu différents patchs et laizes sur la structure, à droite représentation d'une laize avec 4 patchs.

Le dépliage

Une fois que nous avons la structure 3D divisé en laizes et chaque laize en plusieurs patchs P_i et que nous avons calculé les distances l_i et les angles 3D α_j^i entre les nœuds, nous allons réaliser le dépliage de chaque patch pour avoir le dépliage final de chaque laize. Pour ce dépliage on trouve deux méthodes différentes : le dépliage progressif et le dépliage global, mais tout d'abord il faut comprendre en quoi consiste le dépliage d'un patch.

Pour qu'un patch déplié soit le plus représentatif de son patch avant dépliage, il faut qu'il conserve la longueur de son contour et la forme de son contour, ce qui veut dire de laisser les distances 2D et les angles 2D entre les nœuds avec la valeur des longueurs 3D et angles 3D. Dans cette méthode nous allons conserver les distances entre nœuds 3D en imposant que les distances entre nœuds 2D et 3D soient toujours les mêmes : l_i , tandis que les angles 2D θ_j^i vont être optimisés pour être le plus proches des angles 3D α_j^i. Le dépliage d'une surface va être réalisé par un algorithme d'optimisation qui minimisera la différence entre les angles 2D et 3D donc :

$$min \sum_{i=1}^{n} 1/2(\theta_i \ - \alpha_i \ )^2$$

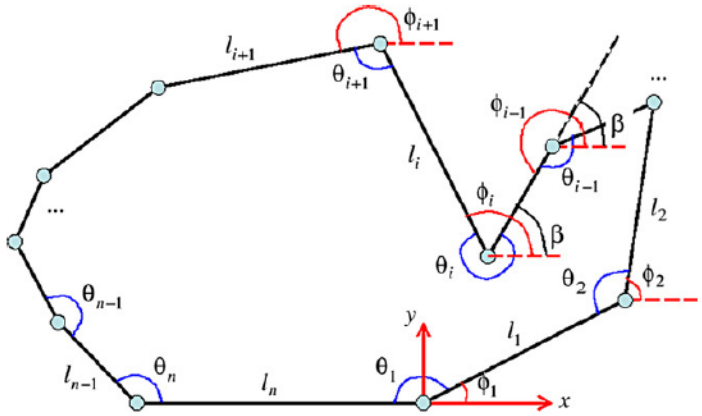

Figure 2Réprésentation du dépliage d'un patch et définition des différents angles 2D

Pour que les patchs soient des surfaces fermés nous devons imposer des contraintes de contour fermé :

$$n\pi - \sum_{i=1}^{n} \theta_i \ \equiv 2\pi \ \ \sum_{i=1}^{n} l_i \ cos\phi_i \ \equiv 0 \ \ \sum_{i=1}^{n} l_i \ sin\phi_i \ \equiv 0$$

$$\text{Avec } \ \phi_i = \pi - \theta_i \ + \phi_{i-1}$$

Le résultat de cette optimisation est un vecteur avec tous les θ_i , avec lesquels on obtient la forme du patch si on écrit chaque nœud sous la forme :

$$(x_i \ , y_i) = (l_i cos\phi_i \ , l_i sin\phi_i) + (x_{i-1}, y_{i-1})$$

Le dépliage progressif

Cette méthode de dépliage appliquée à une laize consiste à déplier individuellement chaque patch qui est contenu dans celle-ci. Selon la référence bibliographique [1], le dépliage progressif est plus effectif pour le dépliage de structures 3D qui ont une courbure hétérogènes ou localisées.

L'ordre de dépliage des patchs va être en commençant par celui qui a le plus de nœuds contraints et en finissant par celui qui a le moins de nœud fixés, avec nœud fixé ou contraint nous voulons dire que l'angle 2D est déjà calculé ou imposé. Pour cela nous allons définir un coefficient que nous allons appeler évidence de forme $\rho(P_i) = n'/n$ où n' est le nombre d'angles 2D fixés dans le patch P_i est n est le nombre de nœuds total du patch P_i . Ces coefficients sont recalculés après le dépliage de chaque patch de la laize, puisque après chaque dépliage le nombre d'angles 2D fixés pour chaque patch peut varier. On observe cette méthode dans la photo ci-dessus.

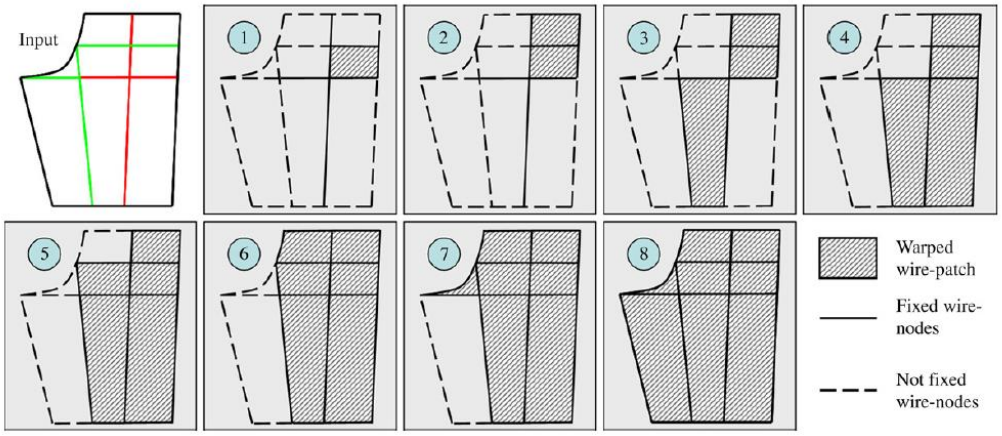

Figure 3 Pas d'un dépliage progressif

Le dépliage global

La méthode du dépliage global est une méthode dans laquelle nous déplions tous les patchs d'une laize d'une seule fois, pour cela nous allons optimiser tous les angles de tous patchs au même temps. Pour bien faire cela nous devons rajouter des contraintes puisqu'il faut que la somme des angles 2D θ_j^i dans un même vertex soit égale à 2π si le vertex n'est pas dans le contour du la laize(vertex intérieurs). Selon l'auteur, cette méthode fonctionne mieux dans les laizes qui ont une forme presque développable et dans quelques structures très courbées.

De manière générale, pour une laize avec m patches, on a $\sum_{i=1}^{m} \mathrm{n_i}$ nœuds qui doivent être optimisés, où $\mathrm{n_i}$ désigne le nombre de nœuds qu'il y a sur le patch P_i . La contrainte à rajouter pour chaque vertex intérieur est de la forme :

$$\sum_{q_{k\epsilon v}} \theta_k \equiv 2\pi$$

Puis une fois réalisé l'optimisation nous avons déjà tous les angles 2D pour construire les laizes.

Il faut noter que avec cette méthode on réalise une optimisation plus complexe que celle réalisée dans le dépliage progressif, nous aurons beaucoup plus de variables à optimiser d'une seule fois et avec plus de contraintes. Par conséquent le temps de calcul de cette méthode est beaucoup plus grand qu'avec la méthode progressive.

Calcul de l'erreur de dépliage

Pour pouvoir comparer deux dépliages et vérifier lequel représente le mieux la structure 3D dépliée, nous allons définir l'erreur de dépliage comme un terme, qui aurai une valeur 0 pour un dépliage d'une surface 3D développable. L'erreur est calculée avec la formule :

$$E_{ang} = \frac{1}{N(\Omega_a)} \sum_{\alpha \epsilon \Omega_a} \frac{|\alpha_a - \theta_a|}{\alpha_a}$$ où Ω_a est l'ensemble de nœuds optimisés dans la méthode

Analyse de la méthode WireWarping

Cette méthode expliquée ci-dessus part du principe de la conservation des longueurs clés du modèle 3D, cela se traduit en gardant constants les distances entre nœuds lors du dépliage.

Cette approche est très intéressante et nous fera avoir des résultats de dépliage qui pourront être très bons.

Il y a certaines parties de la méthode qui ne sont pas suffisamment décrites ainsi que d'autres qui sont impossibles de réaliser au moment de son application :

- Elle ne propose aucune méthode pour la séparation de la structure en laizes, ni aucune règle pour déterminer les courbes de forme clés et accessoires. Par contre, elle nous propose une méthode pour calculer l'erreur d'angle du dépliage et ainsi pouvoir comparer deux dépliages avec des divisions en laizes et patchs différentes.
- Il y a une paradoxe dans la conservation de la forme des courbes de contour et courbes de forme clés. Selon la méthode du WireWarping, la structure est séparée en laizes avant de la déplier, soit en 3D, et au moment du dépliage ces laizes doivent conserver la forme (angle 3D) et la longueur entre nœuds. Si on fait ceci, pour la plupart des formes 3D, la laize n'aura pas un contour fermé. En plus, si on a déjà fixé la forme de la laize, quand on fera le dépliage des patchs, la forme résultante de la laize des patchs dépliés sera la même qu'avant le dépliage, puisque elle a été déjà fixée.

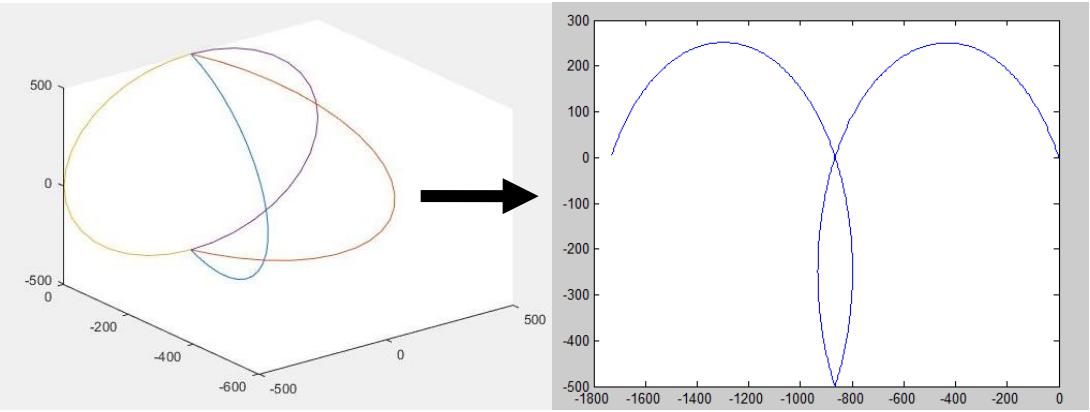

Figure 4 Dépliage d'une laize (sixième de sphère) si nous conservons les angles et longueurs 3D

Démarche de dépliage

Analysede la démarche

Le dépliage commence avec une structure 3D (il s'agit d'une surface géométrique) que nous avons construit à l'aide d'un logiciel de CAO et dont on va obtenir la forme des laizes à découper. Pour réaliser cela, j'ai divisé le dépliage en plusieurs étapes :

1. Division de la structure 3D, maillage et export.
2. Importation des laizes.
3. Calcul des nœuds et courbes.
4. Calcul données 3D nécessaires pour le dépliage.
5. Dépliage (deux méthodes).
6. Représentation des laizes dépliées.

Pour la réalisation de la structure 3D, sa division et son maillage j'ai utilisé le logiciel CATIA v5, pour l'export du maillage j'ai choisi de l'exporter en format « .stl » et pour l'écriture de la partie calcul, j'ai choisi le logiciel MATLAB dans sa version 2015.

Pour tester la démarche pendant que je serai en train de la développer, je vais modéliser sur Catia une géométrie simple qui va être une sphère de 1m de diamètre, laquelle je vais d'abbord diviser en 6 laizes identiques et ensuite on testera différentes divisions en patchs de la laize (sixième de sphere).

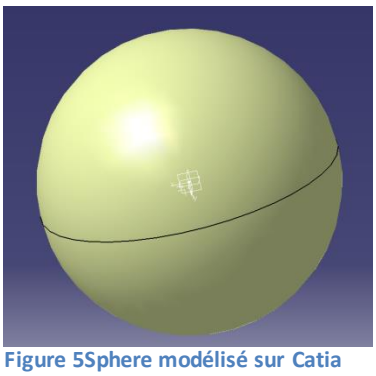

Figure 5Sphere modélisé sur Catia

Pendant toute la démarche sur Matlab, chaque variable sera mie sous la forme d'un vecteur de matrices où vecteur de vecteurs par example angle3d{1} et angle3d{2} seront les vecteurs avec les angles 3D pour le patch 1 et pour le patch 2, donc pour faire référence aux données d'un patch i il faudra inclure le suffixe $\{i\}$.

Division de la structure 3D, maillage et export

Comme dans la méthode du dépliage de courbes, nous devons tout en premier diviser le modèle dans les laizes ou parties à assembler. Ensuite nous devons tracer les courbes de forme clés et accessoires ainsi que les courbes de coupe et de contour, ceci nous divisera chaque laize en plusieurs patchs. Cette partie, je l'ai réalisé en divisant la géométrie à l'aide de l'outil coupe du module « Generativeshape design »dans « FORME ».

Ensuite je vais réaliser le maillage ou facettisation de chacun des patchs réalisés, pour cela j'utilise le module « STL Prototypage rapide » qui se trouve dans « USINAGE ». Dans la facettisation le logiciel nous demande de fixer une flèche de maillage, qui sera la distance max entre le modèle 3D et le maillage. Pour cette première structure test qui est la sphère, j'ai fixé un maillage très grossier pour minimiser le temps de calcul et faciliter la correction et identification d'erreurs.

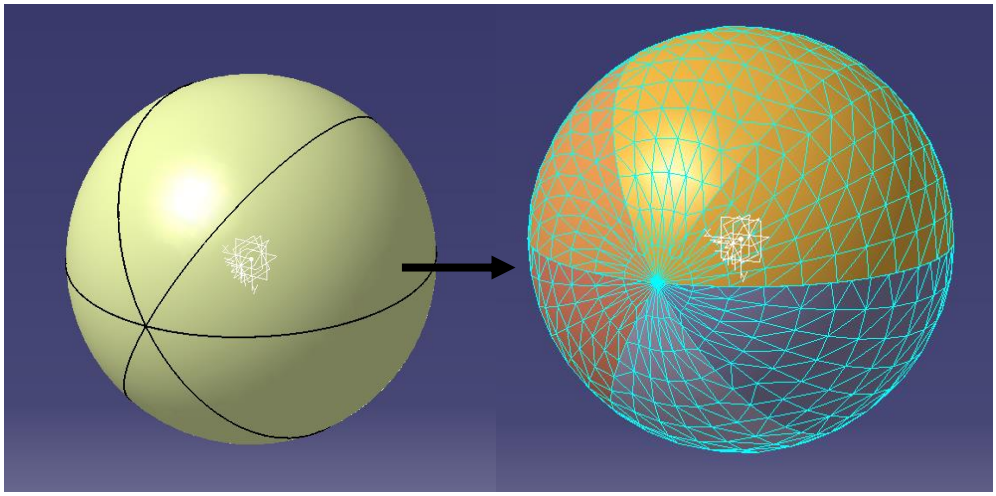

Figure 6 Sphère découpée en laizes d'un seul patch et puis maillée

Pour l'exportation, j'ai choisi le format .stl puisqu'il est le format plus utilisé pour l'export de maillages triangulaires, ainsi que le plus utilisé dans le domaine de la fabrication additive.

Importation des laizes

Dans cette partie on commence le traitement avec le logiciel Matlab, les fichiers .stl sont écris en binaire pour réduire sa taille, et une fois décodés en format .txt ils sont structurés de la façon suivante : une entête suivi de la description de chaque triangle, pour chaque triangle il donne le numéro du triangle, les coordonnées x,y,z de chacun des trois points du triangle et le vecteur normal à la surface extérieure du triangle.

Matlab ne dispose pas d'une fonction propre pour le décodage et lecture des fichiers .stl. Avec une recherche sur Internet j'ai trouvé deux fonctions Matlab pour la lecture et import des fichiers .stl :

- stlread.m : cette fonction qui a comme entrée le nombre d'un fichier, nous retourne trois matrices : une première matrice « v » de dimensions (3*n, 3) où « n » est le nombre de triangles du maillage, une deuxième matrice « f » de dimensions (n, 3) avec les postions des vertex de chaque triangle dans la matrice « v » et une troisième matrice « n » aussi de dimensions (n, 3) avec les vecteurs normaux aux triangles.
- patchslim.m : cette fonction qui a comme entrée les matrices « v » et « f » de sortie de la matrice précédente et nous retourne la matrice « v* » mais sans les vertex qui sont répétés, ce qui est très important pour le traitement de données au fur et à mesure que l'on réduit la taille du maillage et « f* » avec les nouveaux indices des vertex pour « v* ».

Une fois appelés les deux fonctions avec le nom du fichier, nous avons importé les maillages .Pour faciliter la démarche d'importation des laizes, j'ai créé une fonction (importerstl.m) pour exporter tous les patchs d'une laize. Pour cette fonction chaque patch doit être un fichier différent et chaque fichier s'appeller« nom de la laize » plus le suffixe« 1_n » où « n » est le numéro du patch. Quand on exporte les fichiers .stl de tous les patchs d'une même laize, Catia nous écrit ces suffixes automatiquement.

La fonction « importerstl.m » a comme entrée le nom du premier patch de la laize (qui finit par 1_1) et nous retourne les vertex non répétés, les positions des vertex de chaque triangle et les normales de chaque triangle (« v* », « f* » et « n ») pour chaque patch sous la forme de trois vecteurs de n matrices ou n est le nombre de patchs dans la laize.Cette fonction fait appel aux fonctions « stlread.m » et « Patchslim.m ».

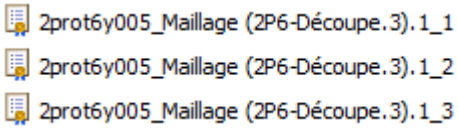

Figure 7Example de nomenclaturedes 3 patchs qui composent une laize pour utiliser la fonction importer.stl

Identification des nœuds et courbes

Le but de cette étape est d'identifier les nœuds du maillage qui se trouvent sur les courbes qui délimitent chaque patch de la laize à déplier. Pour calculer ces points je vais utiliser la commande Matlab « Intersect » avec l'option « 'rows' » qui nous trouve les nœuds coïncidants entre deux matrices de nœuds, ce qui nous donnera les nœuds dans les limites, donc les nœuds dans les courbes qui délimitent les patchs. Pour bien avoir tous les contours il faut « intersectioner » tous les patchs entre eux et les patchs extérieurs avec les laizes voisines. Cette identification avec intersections est très difficile à généraliser puisque pour chaque structure on aura des laizes avec des formes différentes et si on trouve les coïncidences entre tous les patchs, le temps de calcul devient vite très grand (combinatoire entre patchs). Le résultat de ce pas sont des morceaux de courbes (liste de nœuds) mais où les nœuds ne sont pas en ordre.

Une fois qu'on a identifié les nœuds de chaque morceau de courbe, il faut les Ordener, pour cela la démarche est la suivante : on prend trois morceaux de courbe qui donnent comme intersection un point, ensuite on va mettre en ordre les nœuds d'un morceau de courbe qui contient ce point. Pour les ordonner nous allons calculer la distance de chaque point de la courbe à ce point (fonction distance3d.m) et ensuite on va ordonner les nœuds par sa distance à ce point (voir code en-dessous).

```
d12=distance3d(I12,i123);
[~,is12]=sort(d12);
c12=I12(is12,:);
```

Figure 8 Code pour ordonner les points

La fonction « distance3d.m » a comme entrée une matrice (n,3) avec les coordonnées des nœuds et un vecteur (1,3) avec les coordonnés du point de départ.

Il faut remarquer que cette méthode d'ordonner les points n'est pas valide pour toutes les formes de courbe, mais avec la supposition depatchs très petits, ceux-ci ne devraient pas avoir beaucoup de courbure donc elle marchera correctement.

Le prochain pas est de recomposer les contours des patchs avec les morceaux de courbes ordonnés, pour cela nous allons « intersectioner » les courbes entre elles et si il y a deux points en commun, cela veut dire que ces deux morceaux de courbes délimitent eux seules un patch, si par contre il y a qu'un seul point en commun, cela veut dire que les deux morceaux sont reliés. On va créer une matrice de nœuds (nx3) en reliant les morceaux de courbes par les points coïncidents, il faut faire attention à relier les courbes dans le bon sens.

Cette étape peut être faite au même temps qu'on met en ordre les morceaux de courbe pour réduire le temps de calcul du code mais la complexité de la démarche est plus grande.

Finalement il ne nous reste que l'identification des vertex entre les patchs, comme précédemment nous allons trouver les nœuds coïncidents entre les différents patchs et faire une matrice de vertex avec le numéro des patchs de chaque vertex et l'index du nœud dans le repère du patch.

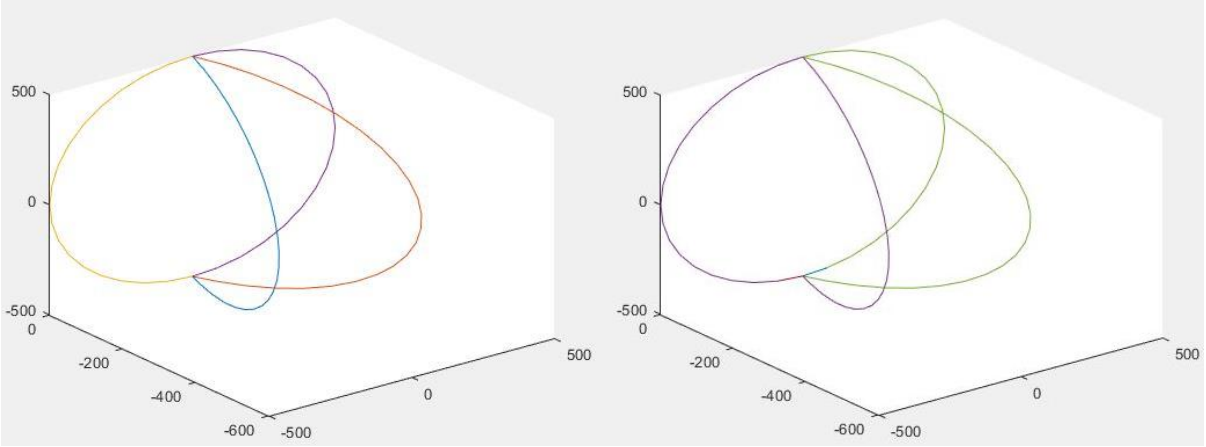

Figure 9Pour notre cas d'une sphère divisée en 6 laizes d'un seul patch chacune. A gauche: les courbes ordonnées chacune avec une couleur, à droite: les courbes de contour de chaque patch, chacune avec une couleur.

Calcul des données 3D nécessaires pour le dépliage

Une fois qu'on a les courbes de contour de chaque patch, on a besoin de calculer les angles 3D α_i et les longueurs 3D l_i nécessaires pour réaliser l'optimisation de dépliage, ces données géométriques qui ont été expliqués avant seront calculés chacun par une fonction :

- longueur3d.m : cette fonction a comme entrée une matrice (nx3) avec les coordonnées des nœuds et retourne une matrice (nx3) avec la longueur 3D, l_i pour chaque nœud q_i qu'il calcule de la forme suivante :

$$l_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}$$

- angle3d.m : celle-ci est une fonction qui calcule l'angle 3D α_i en utilisant le produit scalaire de deux vecteurs. Elle a comme entrées la longueur 3D l_i et les coordonnées de chaque nœudq_i, α_i est calculé sous la forme :

$$\alpha_i = \cos^{-1}\frac{((x_{i+1}-x_i),(y_{i+1}-y_i),(z_{i+1}-z_i)) \times ((x_i-x_{i-1}),(y_i-y_{i-1}),(z_i-z_{i-1}))}{l_i + l_{i-1}}$$

Une fois que nous avons calculé ces données on dispose déjà de tout ce qui est nécessaire pour réaliser le dépliage des laizes.

Dépliage

Dans cette étape nous allons réaliser le dépliage de chaque patch de la laize selon la méthode choisie, le dépliage progressif ou le dépliage global, au contraire que dans la méthode du WireWarping, dans ce dépliage nous n'allons pas fixer les angles 3D des lignes de contour et lignes de forme clés mais, au contraire, nous allons optimiser tous les angles de la laize. Le dépliage est une optimisation avec contraintes, pour cette optimisation on va utiliser la commande Matlab « fmincon ».

Tout d'abord il faut définir la fonction coût $min\sum_{i=1}^{n} 1/2(\theta_i - \alpha_i)^2$, qui est à minimiser. La fonction est déclaré dans le fichier « fonctioncout.m » qui a deux entrées le vecteur avec les angles 3D et le vecteur des variables à optimiser (angle 2D), le retour de la fonction est la différence entre les angles, valeur à minimiser.

Ensuite il faut aussi définir les contraintes :

- La première contrainte que nous allons déclarer est une contrainte qui indus les deux contraintes non linéaires $\sum_{i=1}^{n} l_i\, cos\phi_i \equiv 0$ et $\sum_{i=1}^{n} l_i\, sin\phi_i \equiv 0$, pour les contraintes linéaires on doit les déclarer comme une fonction, le script avec cette fonction s'appelle « nonlinconst.m » et a comme entrées le vecteur des longueurs 3D l_i et le vecteur des variables d'optimisation.
- La deuxième contrainte qu'on va créer est la contrainte linéaire $n\pi - \sum_{i=1}^{n} \theta_i \equiv 2\pi$ ou aussi écrite sous la forme normalisée $[A][\theta] = [B]$ donc a est un vecteur de uns avec longueur n et B est $(n-2)\pi$.

Maintenant, selon la méthode de dépliage nous devons rajouter une troisième contrainte qui est différente pour chaque méthode :

- Pour la méthode du dépliage global, on va déclarer la contrainte $\sum_{q_{k\epsilon v}} \theta_k \equiv 2\pi$ qui disait que la somme des angles des nœuds sur un même vertex intérieur est égale à2π, cette contrainte linéaire est en fait une série de m contraintes où m est le nombre de vertex intérieurs dans la laize. Cette contrainte va être écrite sous la forme [A][θ] = [B] est rajouté à la matrice A et B déjà existantes.
- Pour la méthode du dépliage progressif nous devons créer une boucle avec une fonction qui à chaque itération : calcule *l'évidence de forme* $\rho(P_i) = n'/n$ avec n' le nombre d'angles 2D fixés dans le patch P_i est n le nombre de nœuds total du patch P_i. Puis le patch à déplier est choisi et on crée la contrainte qui fixe l'angle pour les vertex intérieurs dont il ne reste qu'un angle à calculer sous la forme $\theta_i = 2\pi - \sum_{q_{k\epsilon v}} \theta_k$ où θ_i est l'angle du patch à déplier et θ_k sont les angles fixés qui ont le même vertex. Ses contraintes seront aussi mises sous la forme [A][θ] = [B] et rajoutés aux matrices des contraintes linéaires.

Dans notre cas, où on a une laize avec un seul patch, la méthode d'optimisation est égale puisqu'on n'a qu'un patch, par conséquent nous n'avons pas besoin de déclarer la troisième contrainte.

Ensuite on fixe les options de l'optimisation avec la commande « optimoptions » et on lance l'optimisation pour le cas du dépliage global ou sinon on lance la boucle des optimisations si c'est la méthode progressive. La fonction qui fait l'optimisation d'une laize a comme entrées les angles 3D et les longueurs 3D des nœuds.

Il faut aussi fixer les valeurs initiales des variables à optimiser, nous allons fixer $\theta_0 = \alpha$ puisque le but est que les valeurs de θ soient le plus proches possible des valeurs de α.

A la fin de l'optimisation, nous allons calculer l'erreur de dépliage :

$E_{ang} = \frac{1}{N(\Omega_a)} \sum_{a\epsilon\Omega_a} \frac{|\alpha_a - \theta_a|}{\alpha_a}$ oùΩ_a est l'ensemble de nœuds compris dans l'optimisation

Représentation des laizes

Dans cette dernière étape nous allons calculer la forme final de laize en 2D avec les angles 2D θ calculés dans l'étape précédente. Pour ceci, il faut calculer la position des nœuds dans le plan avec la fonction « plot2d.m » qui a comme entrées les angles 2D des nœuds et les longueurs entre nœuds d'un patch et retourne les coordonnées (x,y) pour chaque nœud.

Ensuite il faudrait réaliser une fonction qui relie tous les patchs pour former la forme finale 2D de la laize dépliée.

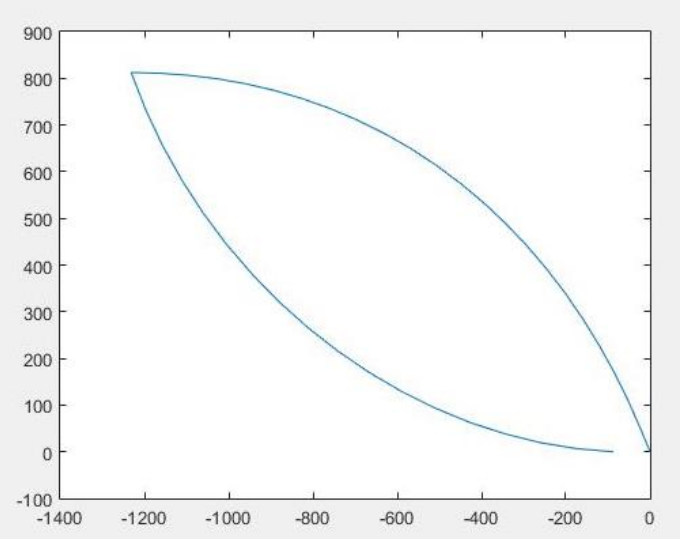

Figure 10 Forme finale de la laize dépliée. La laize ést un sixième de sphère de 1 m de diamètre

ANALYSE DE LA METHODE

Comme nous avons pu remarquer le long de l'explication de la démarche, cette démarche de dépliage est assez simple une fois que nous avons compris le fonctionnement de la méthode de dépliage que nous sommes en train d'optimiser.

La partie dans laquelle j'ai trouvé plus de problèmes est celle de la division de la structure 3D, maillage et export, réalisée sur Catiav5, puisque l'outil de maillage que j'ai utilisé n'est pas adapté pour le calcul, donc on trouvait des problèmes de maillage avec des modèles où il n'y a pas une correspondance des nœuds sur les laizes ou patchs adjacents, ceci on peut le voir dans la photo en-dessous, ces nœuds décalés ne nous permettent pas de bien composer les courbes, ni de bien calculer les vertex puisque nous avons un décalage entre les nœuds de deux patchs adjacents.

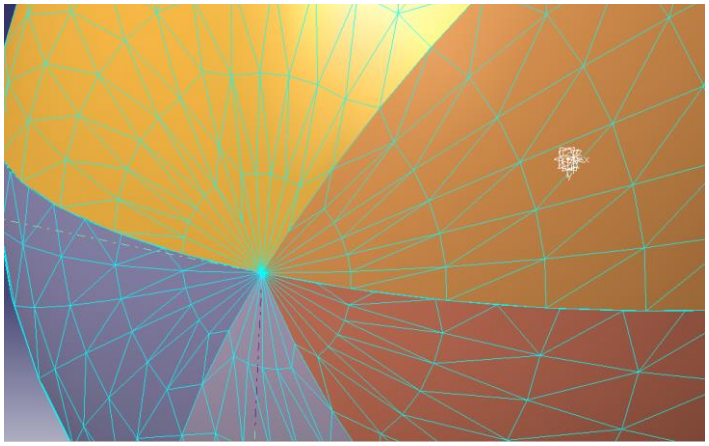

Figure 11 Zooms sur des parties de non correspondance des nœuds entres les laizes supérieures et inférieures.

Un autre point faible de la méthode est l'identification des nœuds et courbes, cette partie qui peut devenir tout de suite très lourde dès qu'on augmente le numéro des patchs, pourrait être évitée si nous utilisons une méthode d'importation des données avec les courbes déjà identifiées.

Le dépliage ce fait juste dans l'étape de l'optimisation, le reste de la démarche nous sommes en train d'importer où de mettre toutes les données sous la forme nécessaire pour réaliser l'optimisation. L'optimisation à elle, est assez bonne et rapide, puisque, comme nous partons des angles 3D comme points initiaux de l'optimisation, $\theta_0 = \alpha$, nous sommes déjà très proches de notre minimum, puisque la fonction à optimiser est la différence entre ces deux angles. Par conséquent, le nombre d'itérations n'est normalement pas très élevé.

Pour conclure, les résultats de cette démarche sont les attendus et en ce qui concerne le temps de calcul, il est plus court que celui prévu. Maintenant, je vais tester la méthode et analyser les résultats de celle-ci.

ANALYSE DES RESULTATS

Pour analyser les résultats de la démarche, je vais tester cette méthode en changeant quelques paramètres à chaque fois et ainsi voir le comportement pour des différents modèles et paramètres.

D'abord, nous allons étudier l'influence du point initial dans le résultat de l'optimisation, initialement nous pourrions penser que le meilleur point initial, où celui qui doit nous faire converger vers le minimum global est $\theta_0 = \alpha$, c'est pour ça que nous allons tester avec d'autres points. Comme nous savons que l'angle est plus petit que 2π, nous allons essayer avec d'autres points initiaux plus petits.

Points initiaux	α	α/1.2	α/1.4	α/2	0
Erreur de dépliage	0.012	0.657	0.657	--	--

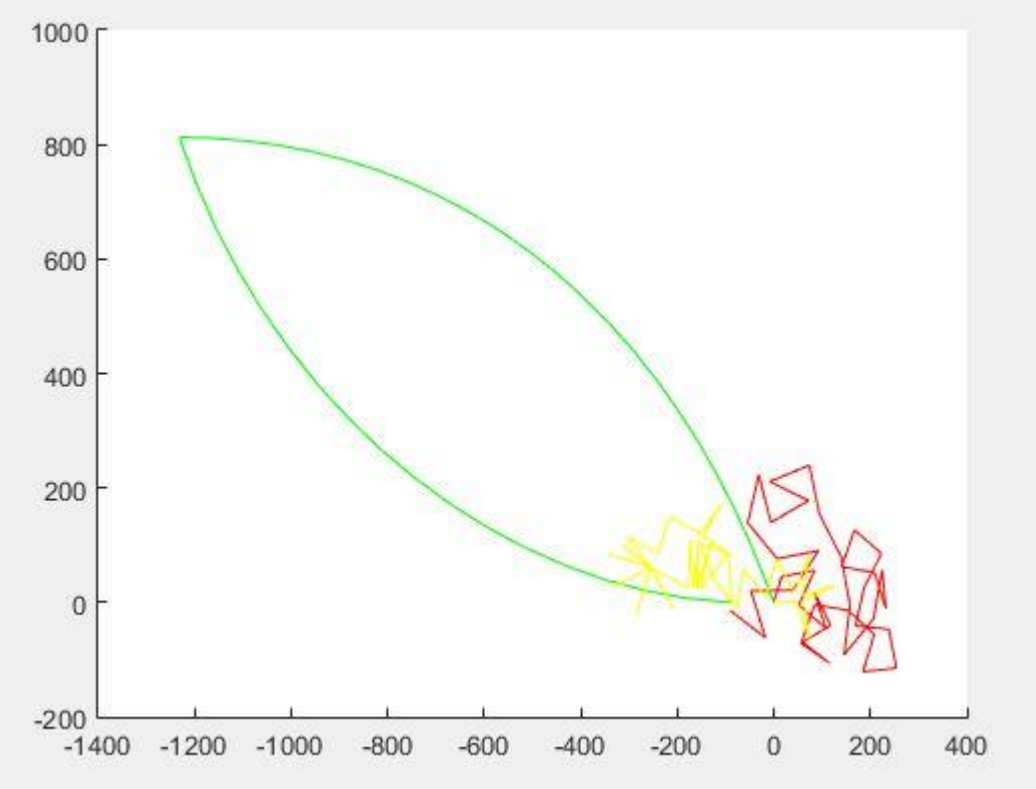

Figure 12 Laize déplie avec α (vert), α/1.2 (rouge) et α/1.4 jaune

On peut observer que si on a un point initial différent à $\theta_0 = \alpha$ l'optimisation donne des résultats incohérents comme dans l'image. Pour la valeur de α/2 et 0, l'optimisation ne finis même pas après 4100 évaluations. Il est prouvé que le meilleur point initial de l'optimisation c'est $\theta_0 = \alpha$ puisque c'est le seul qui nous donne un bon résultat final.

Ensuite, nous allons voir l'influence de la taille du maillage dans les résultats de la démarche, pour cela, nous allons faire le dépliage précédent mais avec des maillages qui ont une taille de flèche différente, pour une sphère de 1m de diamètre nous allons tester avec 0.2mm, 0.1mm, 0.05mm et 0.01mm et analyser les temps de calcul, les formes et l'erreur de dépliage. Nous pouvons voir les différents modèles maillés avec ces flèches dans l'annexe.

Taille de la flèche du maillage	0.2	0.1	0.05	0.01
Temps de l'optimisation (s)	0.63	0.97	1.62	11.02
Erreur de dépliage	0.012	0.0085	0.0059	0.0027

Comme on s'attendait, l'erreur de dépliage est plus petite au fur et au mesure qu'on réduit la flèche du maillage, au même temps, cette réduction de la taille du maillage produit un incrément du temps de calcul de l'optimisation.

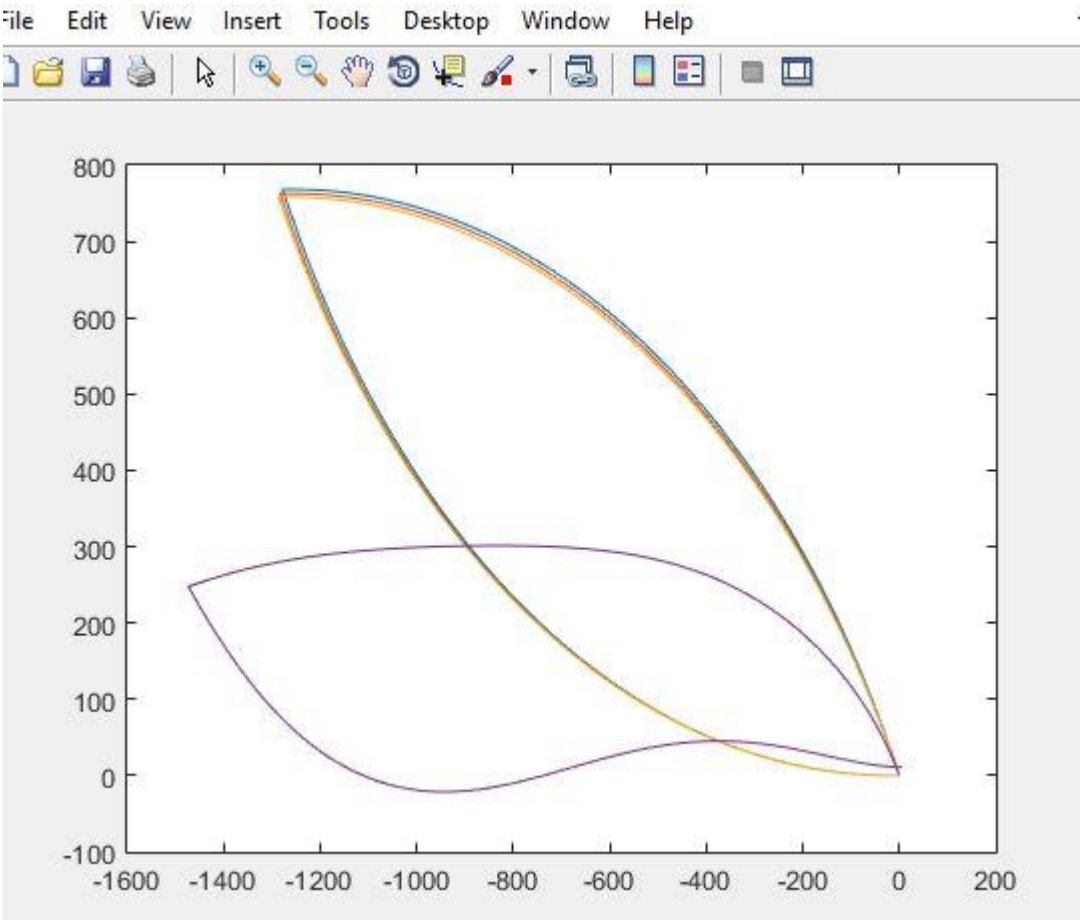

Figure 13 Laize déplie en bleu flèche 0.2, en rouge 0.1, en jaune 0.05 et on mauve 0.01

Nous pouvons apprécier que la différence entre les 3 premiers dépliages est très petite tandis que pour la laize avec une flèche de 0.01mm nous observons un phénomène qui à première vue semble un peu étrange et qui n'améliore pas en changeant le point initial (voir annexe), l'erreur de dépliage diminue pour chaque réduction du maillage mais nous ne pouvons pas assurer si les laizes sont bonnes. Pour ceci il faudrait faire l'expérience pratique ou essayer de trouver des solutions d'autres ressources bibliographiques.

A continuation, je voulais réaliser le dépliage de cette même laize (sixième de sphère) mais avec plusieurs patchs au lieu de seulement un, après écrire le script « directeurglobalw.m »,quand je l'ai lancé j'ai rencontré un très gros inconvénient, il avait un problème de non correspondance dans quelques uns des nœuds dans les deux frontières des 3 patchs (voir figure 14), ce problème je l'ai retrouvé qu'une fois écrit le programme puisqu'il était difficile de voir dans la CAO. Le script écrit ne marche pas sans un point de coïncidence entre les 3 laizes et je n'ai pas réussi à corriger ce défaut de maillage. Si le maillage est corrigé le script devrait marcher correctement.

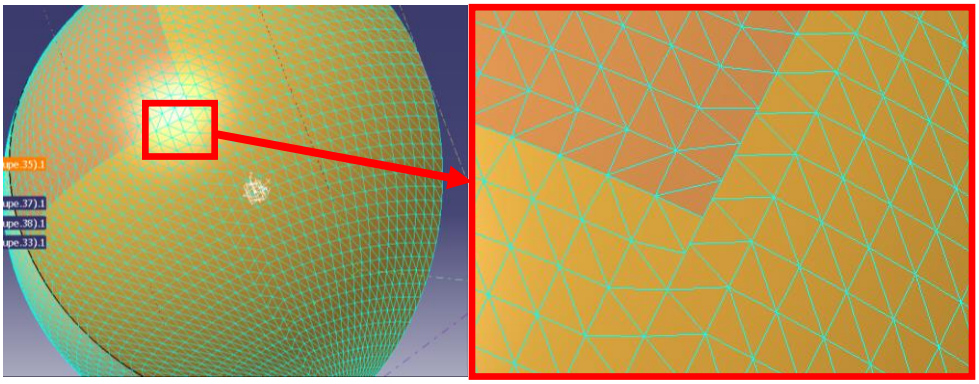

Figure 14 Découpe de la laize en 2 patchs où il y a un problème de maillage, à droite zoom sur le défaut

La seule forme à 4 patchs que j'ai réussis à mailler qui puisse être maillé est celle de la figure 15, le programme devait être modifié dans la partie qui identifie les courbes et qui prépare les contraintes de l'optimisation. Ces modifications étaient assez longues et malheureusement je n'ai pas eu le temps de les faire.

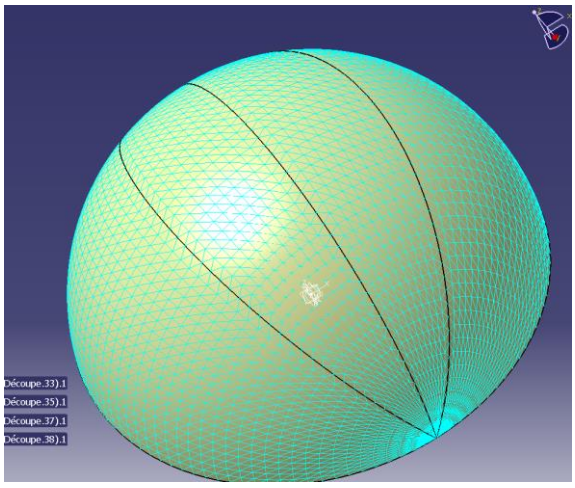

Figure 15 Forme des deux patchs sans problème de maillage.

CONCLUSION

Dans ce PJE « Dépliage de structures gonflables » j'ai crée une démarche pour le dépliage de structures 3D basé sur la théorie du Wirewarping[1], cette démarche part d'un modèle 3D crée dans un logiciel de conception CAO et finis par donner la liste des nœuds des laizes une foi s dépliés.

L'analyse des résultats récupérés de cette démarche nous fait penser que la méthode est bonne et surtout simple à programmer, puisque l'optimisation et les contraintes ne sont pas très complexes, en plus les temps de calcul sont assez performants.

Par contre, un désavantage qu'a la méthode est la difficulté de générer un code généraliste pour la première étape de la démarche en Matlab qui est l'identification des nœuds et courbes. En plus dans le cas de vouloir générer un code généraliste, il faudrait qu'il soit assez complexe pour prendre en compte toutes les formes possibles de géométrie que nous pouvons rencontrer.

Pour reprendre ce projet, tout d'abord il sera intéressant de lancer le code que j'ai écrit pour le dépliage de la même laize mais avec deux patchs différents pour pouvoir comparer ce dépliage avec celui d'un seul patch. En plus, un essai de reconstruction d'un modèle à l'échelle sera intéressent pour tester l'effectivité de la démarche.

Comme continuation de ce projet, il serait convenable de considérer l'utilisation d'un autre module du logiciel Catia pour le maillage. Il sera aussi intéressant de revoir la faisabilité de la création directe des courbes de contour sur ce logiciel pour pouvoir les discrétiser sur directement sur celui-ci et pouvoir exporter déjà les contours des patchs au lieu de devoir exporter toute la maille ce qui amène tout un processing derrière. Sinon, une démarche plus spécialisée vers des formes plus particulières peut être plus effective en ce qui concerne temps de calcul et complexité du code.

Pour conclure, je voudrais remercier M.Monteiro, mon tuteur de projet pour m'avoir permis d'effectuer ce projet avec ce sujet que j'ai trouvé très intéressant ainsi que pour m'encadrer et me guider tout au long de mon projet.

BIBLIOGRAPHIE

[1]: WireWarping: A fast surface flattening approach with length-preservedfeature curves Charlie C.L. Wang, Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong. 28 April 2007

[2]: Computing length-preserved free boundary for quasi-developablemesh segmentation. Wang CCL. EEE Transactions on Visualization andComputer Graphics 2008;14(1):25–36.

[3]: http://www.mathworks.com/matlabcentral/fileexchange

[4]: fr.mathworks.com/help/matlab/functionlist.html

ANNEXES

```
%maitre qui fonctionne pour tous

clear

%filename='Z:\Config\Bureau\PJE\matlab\6laises_01\2prot602_Maillage (2P6-Découpe.3).1_1.stl';
%filename='Z:\Config\Bureau\PJE\matlab\prot6en6_Maillage (ballons-quartSTLPROTOTYPE6-Découpe.3).1_1.stl';
filename='Z:\Config\Bureau\PJE\matlab\6laises_01\2prot6y001_Maillage (2P6-Découpe.3).1_1.stl';

tic
%[v,~,~]=importerstl(filename);
[v,f,n]=importerstl(filename);

toc

tic
[cs,ls,a3d,c,ds]=calculdonnes(v);
toc
%verification des courbes crées
tic
plot3(cs{2}(:,1),cs{2}(:,2),cs{2}(:,3))

toc

xy = plot2d(a3d,ls);
plot(xy(:,1),xy(:,2))

tic
[xy,E]=optimisation(a3d,ls);
toc
%verification des courbes crées
plot3(cs1(:,1),cs1(:,2),cs1(:,3))
```

Figure 16 Programme directeur final qui fait le dépliage de la laize à un patch

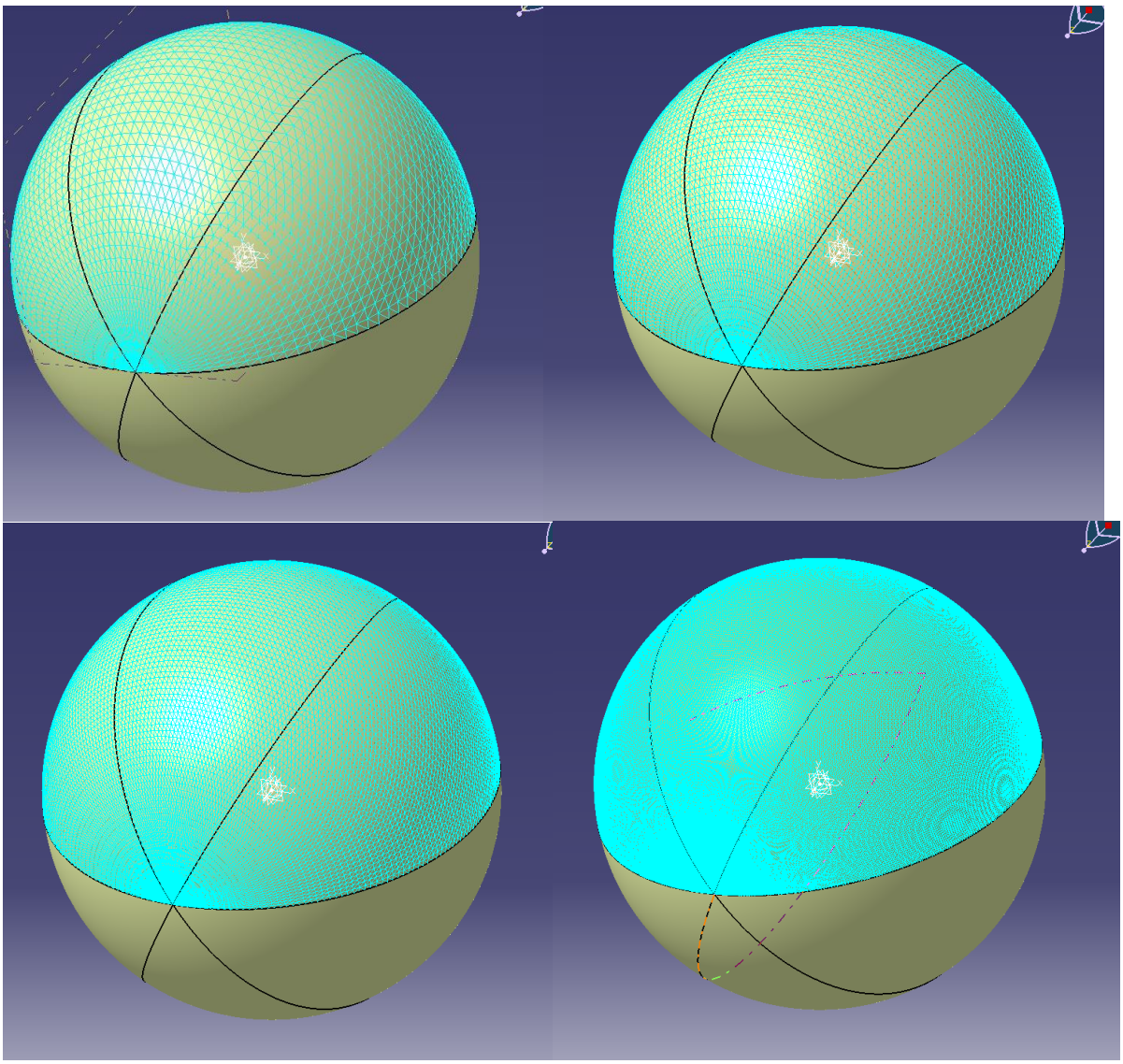

Figure 17 Maillages avec flèche 0.2, 0.1, 0.05, 0.01 de haut à droite à en bas à gauche

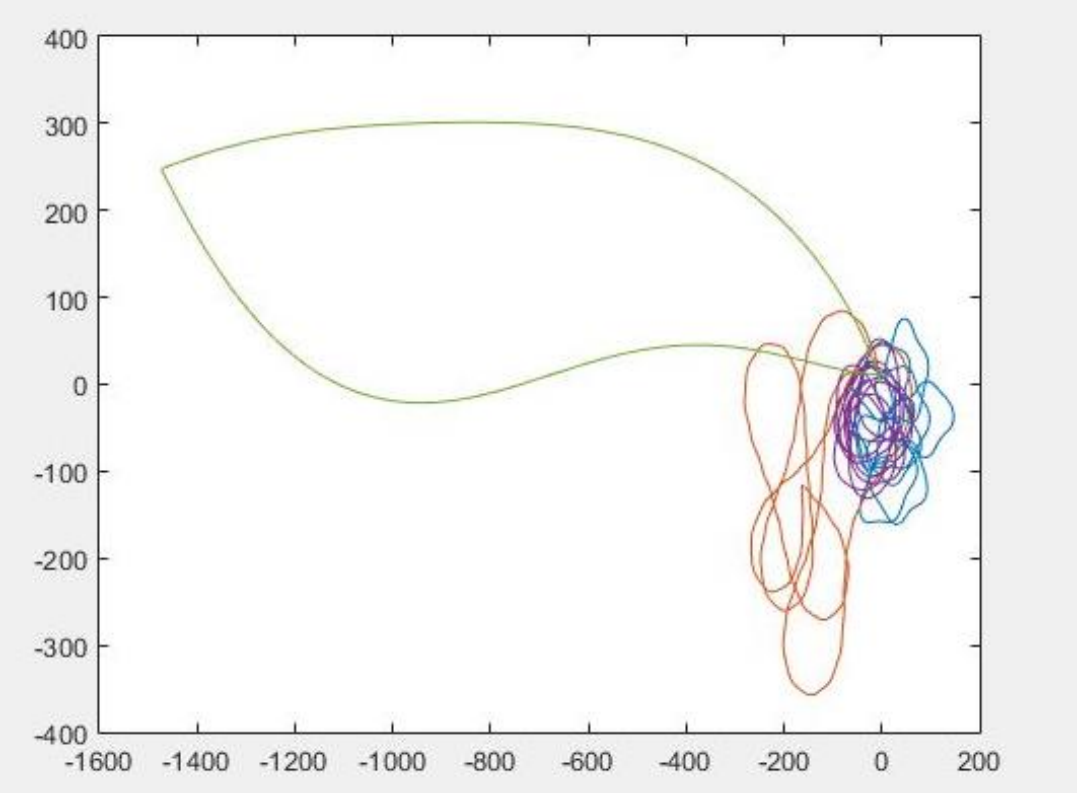

Figure 18 Laize du modèle avec flèche 0.05 déplie avec α (vert), α/1.2 (bleu), et α/1.4 (rouge) et α/2 (violet)