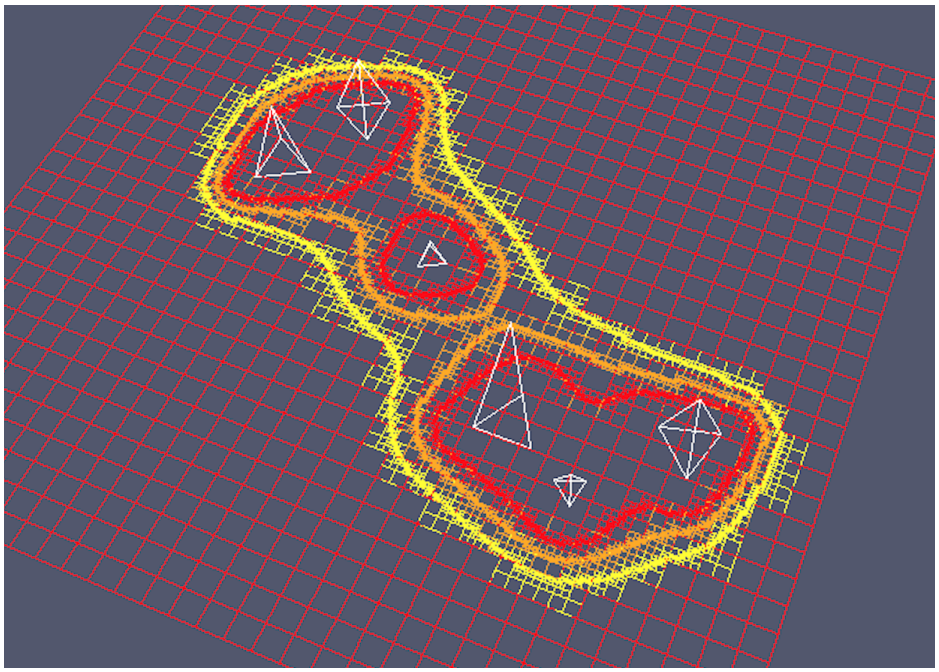

Figure 1: Résultat final du projet : un jeu de tétraèdres est affiché ainsi que trois lignes de niveau

Développement logiciel

Le but de ce projet est d'identifier des zones accessibles au sein d'un champ de tétraèdres. Ces tétraèdres sont supposés émettre un signal, S, qui décroit en $S \approx 1/r$, r étant la distance la plus courte entre un point de l'espace et le tétraèdre. Si il y a plusieurs tétraèdres on considère que le signal local en (x, y, z) est obtenu comme la somme des signaux émis par les différents tétraèdres.

$$S(x, y, z) = \sum_i \frac{1}{r_i(x, y, z)}$$

Une zone sera considérée comme accessible si le signal local est inférieur à une certaine valeur de référence S_{ref}. Pour délimiter les zones accessibles, il convient donc d'être capable d'identifier finement une ligne de niveau (en 2D) ou une isosurface (en 3D) $\mathcal{C} = \{(x, y, z)|S(x, y, z) = S_{ref}\}$ à une altitude donnée.
Le but de ce projet est de pouvoir :

- lire un fichier contenant une liste décrivant les tétraèdres qui pavent l'espace,
- identifier finement une ou plusieurs lignes de niveau à une altitude donnée à partir d'une approche basée sur des quad-trees,
- utiliser le logiciel Paraview pour visualiser les résultats.

Un exemple de résultat final est donné en figure 1.
Pour parvenir à ce résultat, le projet est découpé en différentes phases. Durant chacune de ces phases, vous allez coder des objets en python qui seront utilisés lors de séances qui suivront. Pour chaque objet un programme de test sera développé pour le valider indépendamment du programme final. L'ensemble des fichiers nécessaires à ce projet sont téléchargeables à l'adresse suivante : `https://centralesupelec-my.sharepoint.com/personal/jean-michel_dupays_centralesupelec_fr/_layouts/15/guestaccess.aspx?folderid=0f8dab2ff3e974271a54aaa8dc1835dd3&authkey=AeLwOKh5HIIZswUhIyMnMgk`

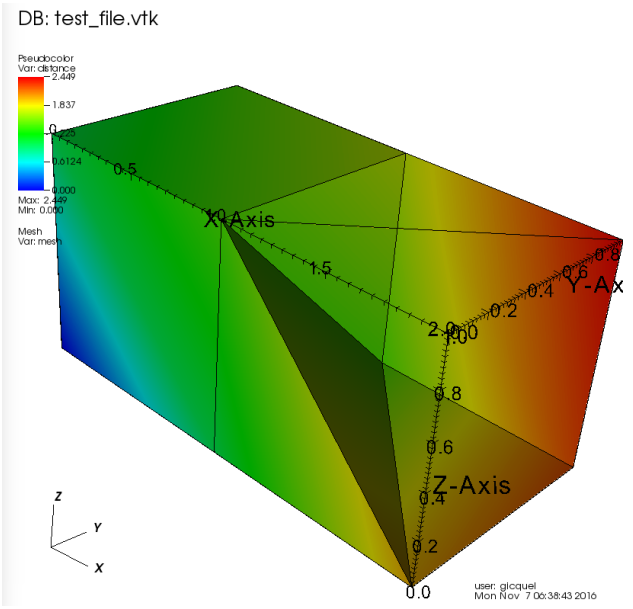

Figure 2: Visualisation avec Paraview du fichier de maillage ***exemple_cas1.vtk***.

Module 1 : génération d'un fichier vtk permettant de visualiser un maillage non structuré avec Paraview

Le format vtk permet de visualiser n'importe quel type de maillage avec un logiciel de visualisation comme Paraview, utilisé ici, Ensight, Tecplot ou Visit. Ce format bien que très pratique et couramment utilisé dans la communauté scientifique est peu documenté. Je vous propose donc ici, de faire comme le font beaucoup de chercheurs et d'ingénieurs, un peu de reverse engineering. Pour cela vous pouvez télécharger un fichier ***exemple_cas1.vtk*** sur le site mentionné en début de sujet. Le but de cette séance est de créer un objet python, **FileVTK** qui prenne comme entrée une liste de cellules pouvant inclure des segments, des quadrangles, des triangles, des tétraèdres et des hexaèdres et génère un fichier au format vtk permettant de visualiser le maillage ainsi que les valeurs qui seront stockées en chacun de ses points. Vous pouvez visualiser le fichier ***exemple_cas1.vtk*** avec Paraview, vous devez obtenir la même chose que sur la figure 2.
Pour coder l'objet **FileVTK** en python vous devez télécharger le fichier ***vtk.py***. Ce fichier contient la structure, les méthodes associées à cet objet, ainsi qu'un programme de test. Le but pour vous est de compléter les méthodes pour que le programme de test permette de générer un fichier identique au fichier ***exemple_cas1.vtk***. Dans la version initiale l'objet **FileVTK** ne fait rien, il se contente d'écrire une ligne de texte par méthode appelée. Pour exécuter le programme principal vous devez taper :

```
/TD$ python vtk.py
FileVTK.Init function to be coded
FileVTK.SavePoints function to be coded
FileVTK.SaveConnectivity function to be coded
FileVTK.CreatePointScalarSection function to be coded
FileVTK.SavePointScalar function to be coded
FileVTK.SavePointScalar function to be coded
FileVTK.Close function to be coded
```

Vous devez coder les méthodes une par une pour qu'une fois l'objet terminé vous obteniez un fichier identique (pour cet exemple) au fichier fourni. L'objet doit être générique pour pouvoir être

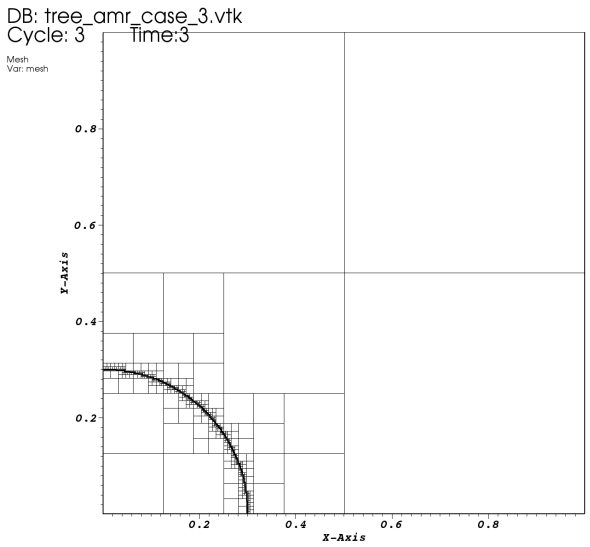

Figure 3: Exemple de maillage de type QuadTree raffiné autour d'une isoligne.

utilisé avec une liste quelconque de faces et de volumes. Cet objet sera utilisé par la suite pour créer les fichiers au format vtk qui vous permettront de visualiser vos résultats.

Module 2 : Création d'un objet QuadTree

Pour identifier finement une isoligne de niveau d'une fonction inconnue non linéaire il est nécessaire de pouvoir évaluer cette fonction sur un maillage très fin. Le coût d'une telle évaluation peut s'avérer extrêmement cher en temps de calcul. Pour remédier à ce problème nous allons utiliser une structure de donnée nous permettant d'avoir un maillage raffiné automatiquement autour de l'isoligne et grossier dans les autres zones. Le résultat que nous souhaitons obtenir est présenté sur la figure 3. Le type de structure de données retenue est de type QuadTree (`https://fr.wikipedia.org/wiki/Quadtree`).
Le développement de cet objet se fera en trois étapes :

- Pour commencer vous pouvez télécharger le fichier ***QuadTree1.py***, il vous permettra de définir un nouvel objet **Tree_amr**. Comme dans le cas précédent pour devez compléter l'objet. Pour vous y aider j'ai laissé les commentaires du code dans le fichier. Cette version de l'objet permet juste de construire l'ensemble des branches et des feuilles du QuadTree.

- La deuxième étape du développement consiste à rajouter une méthode qui renvoie une liste de quadrangles décrivant le QuadTree ce qui permettra en utilisant l'objet **FileVTK** de visualiser votre QuadTree. Télécharger le fichier ***QuadTree2.py***.

- La troisième et dernière étape consiste à fournir au QuadTree une fonction permettant de connaître la valeur d'intérêt aux noeuds et une autre permettant de savoir si une feuille doit être divisée en quatre ou non. Ces deux fonctions seront transmises au QuadTree grâce à un dictionnaire. Télécharger le fichier ***QuadTree3.py***. Le résultat que vous devez obtenir en faisant tourner le programme test est fourni pour illustration dans la figure 3.

Module 3 : Création d'un objet Tri

Les tétraèdres qui doivent paver l'espace sont composés de quatre surfaces triangulaires. Ces triangles étant à la base des tétraèdres nous allons commencer par définir un premier objet tri qui sera ensuite utilisé pour former les tétraèdres. La distance entre un point et le triangle est considérée comme définie uniquement si la normale de ce triangle pointe vers le point concerné. Le fichier python qui doit être modifié pour créer cet objet est : ***tri.py***.
Pour le calcul de la distance entre un point et un triangle en 3D, vous pouvez trouver l'algorithme par vous même ou bien profitez de la communauté Python sur Internet (le site https://github.com/ héberge beaucoup de codes opensource). Dans le second cas, veuillez à tester sur des cas simples que le code fonctionne correctement.
Note : Pour commencer, vous pouvez remplacer le calcul de la distance d'un point à un triangle par le calcul de la distance minimale d'un point aux sommets du triangle.

Module 4 : Création d'un objet Tetra

Nous allons maintenant créer un objet tétraèdre qui sera basé sur l'objet tri. Pour le faire vous pouvez compléter le fichier ***tetra.py***. Cet objet aura une méthode qui permettra de connaître la distance minimum entre un point de l'espace et celui-ci. Si le point se situe dans le tétraèdre on considérera que la distance n'est pas définie.

Module 5 : Création d'un objet Maillage

Pour pouvoir définir la zone d'accès, nous allons définir un objet maillage qui contiendra la liste de l'ensemble des tétraèdres et qui aura une méthode permettant de tracer l'ensemble des tétraèdres ainsi qu'une méthode calculant la valeur de S pour n'importe quel point de l'espace. Pour initialiser cet objet on lui fournira comme entrée un fichier texte qui contiendra les informations suivantes :

```
nombre de tétraèdres
coordonnées du centre de la base du tétraèdre
longueur du coté du triangle équilatéral servant de base
hauteur du tétraèdre
rotation du tétraèdre autour de sa hauteur
```

Nous nous limiterons à des tétraèdres à base équilatérale pour simplifier le problème. Vous pouvez télécharger le fichier descriptif d'exemple : ***defense_zone.txt***. Le fichier python à modifier est : ***mesh.py***

Module 6 : Application

Dorénavant, utilisez le travail précédent pour calculer certaines isolignes du signal

$$S(x, y, z) = \sum_i \frac{1}{r_i(x, y, z)}$$

émis par les tétraèdres définis par un fichier semblable à ***defense_zone.txt***. On pourra se restreindre à un plan fixé à priori (on cherche les lignes de niveaux dans ce plan). Le résultat obtenu doit être similaire à celui de la figure 1.
Si vous avez terminé, vous pouvez ajuster l'ensemble de votre code pour implémenter une géométrie 3D dans les Quadtrees (découpez l'espace avec des cubes) puis afficher des isosurfaces en 3D.
Vous pouvez également modifier la profondeur des Quadtrees comme suit: finaliser le niveau max de raffinement par une taille minimum de la maille et non un niveau de profondeur maximum.