**Description du KdTree**

**Contexte :**

* Après avoir calculé un modèle implicite sur le réservoir (fonction de potentiel f tel que l’horizon i est décrit par f(X) = ci avec ci la valeur de potentiel )
* Pour reconstruire les surfaces ,on a commencé par choisir les collocations avec une grille régulière
* Maintenant , on a passé à une structure plus adaptative qui est la Kd tree

**Schéma général :**

Le schéma général du code est composé de **4 parties** :

* + **Partie 1 : valeurs de potentiels**

On donne à chaque horizon i une valeur de potentiel ci = i

* + **Partie 2 : choix des collocations**

On choisit les collocations Avant par une grille régulière , maintenant avec KdTree

* + **Partie 3 : reconstruction de la fonction f**

On reconstruit notre fonction implicite f après résolution du système des moindres

carrées en utilisant les points de données et les collocations

* + **Partie 4 : représentation des surfaces**

On représente la surface correspondante à chaque horizon i ( f(X) = ci )

Diagram

Description automatically generated

Figure 1 : Schéma général

**Démonstration :**

(correspond à la partie 2 du schéma général)

A picture containing diagram

Description automatically generated

Figure 2 : Schéma de la partie 2

**Le travail du kd tree se fait par la fonction**

build\_kdtree\_size\_leafnode\_moins\_sv

**. Paramètres :**

- points : les points de données , sera modifié a chaqu4e appel de la fonction

- depth : la profondeur de l'arbre ,on commence par 0

- k : le nombre de dimensions (2 pour xy et 3 pour xyz) ,pour lesquelles on effectue la division

- seuil\_voxel : le nombre maximum de points qu'on peut avoir dans un voxel

- voxels\_var : un cell array qui contient lui même n\_horizon(le nombre

d'horizons) cell array qui contiendront les voxels de chaque horizon

,on commencent par des cells array vides {}

- nombre\_voxels\_var : un cell array qui contiendra le nombre de voxels

pour chaque horizon

- voxels\_index\_var :un cell array qui contiendra le nombre de voxels

pour chaque horizon

- M\_prime : les points de données ,restent inchangeable à chaque appel de la fonction ,servent pour savoir l'indice d'un point de données dans la

matrice initiale

- n\_horizons : le nombre d'horizons

**. Détails :**

- une fonction récursive qui construit pour chaque horizon des voxels dont le nombre des points est plus petit que seuil voxel (le nombre maximum de points qu'on peut avoir dans un voxel)

**. Valeurs de Retour :**

- tree : l'arbre obtenu par le kdtree effectue

- voxels\_par\_horizon : les voxels de chaque horizon

- nombre\_voxels\_par\_horizon : le nombre de voxels pour chaque horizon

- voxels\_index\_par\_horizon :les indices des points des voxels de chaque horizon

**Script à lancer­­ :** **script\_a\_lancer**.m

**. méthodes liées au Kdtree:**

* calculer\_collocations\_par\_horizon
  + paramètres :
    - n\_horizons : le nombre d'horizons

* + - voxels\_par\_horizon : un cell array contenant n\_horizons cell array qui contiennent les voxels de chaque horizon (structure arborescente)

* + - voxels\_index\_par\_horizon : un cell array contenant n\_horizons cell

array qui contiennent les indices des points de chaque voxel de chaque horizon (structure arborescente)

* + - nombre\_voxels\_par\_horizon : une liste qui contient le nombre de voxels dans chaque horizon
  + détails :
    - calculer\_collocations\_par\_horizon : calcule le barycentre de chaque

voxel de chaque horizon

* + valeurs de retour :
    - nombre\_collocations\_par\_horizon : le nombre de collocations de chaque horizon
    - collocations\_par\_horizon: les collocations de chaque horizon
    - voxels\_liste\_par\_horizon : la liste des voxels de chaque horizon
    - voxels\_index\_liste\_par\_horizon : la liste des indices des points des voxels de chaque horizon
* export\_voxels\_collocations\_xyz\_color
  + paramètres :
    - n\_horiz : le nombre d'horizons
    - collocations\_par\_horizon : les collocations de chaque horizon
    - voxels\_liste\_par\_horizon : la liste des voxels de chaque horizon
    - nombre\_voxels\_par\_horizon : le nombre de voxels de chaque horizon
    - offset : le vecteur minimum qu'on a soustrait des points de données au début
  + détails :
    - export\_voxels\_xyz\_color : exporte les voxels et les collocations de chaque horizon en fichiers xyzrgb en donnant une couleur différente à chaque voxel et une seule couleur (blanche) pour toutes les collocations
* coller\_collocations
  + paramètres :
    - n\_horiz : le nombre d'horizons
    - collocations\_par\_horizon : les collocations de chaque horizon
    - nombre\_collocations\_par\_horizon : le nombre de collocations de chaque horizon
  + détails :
    - coller\_collocations fait l'union de toutes les collocations des

differents horizons pour pouvoir reconstruire les surfaces

* + valeurs de retour :
    - n\_horiz : le nombre d'horizons
    - collocations\_par\_horizon : les collocations de chaque horizon
    - nombre\_collocations\_par\_horizon : le nombre de collocations de chaque horizon

**. L’enchainement des méthodes liées au Kdtree:**

**build\_kdtree\_size\_leafnode\_moins\_sv**

**donne**

*voxels\_par\_horizon*

*voxels\_index\_par\_horizon*

*nombre\_voxels\_par\_horizon*

**À**  **calculer\_collocations\_par\_horizon**

**Qui donne**

*collocations\_par\_horizon*

*nombre\_collocations\_par\_horizon(=* nombre\_voxels\_par\_horizon)

**À export\_voxels\_collocations\_xyz\_color et À coller\_collocations**