Projet : phases 1 et 2

OBJECTIFS :

- Se familiariser avec les hyperquadriques : les visualiser, comprendre le rôle des différents paramètres
- Comprendre le principe du fit d'un nuage de points par une HQ en étudiant un cas simple
- Mettre en œuvre une recherche de minimum sans contrainte par la méthode du gradient à « pas fixe »
- Mettre en œuvre une recherche de minimum sans contrainte par la méthode de Newton
- Analyser le comportement et les conditions de convergence de ces méthodes

RAPPEL - DEFINITION D'UNE HYPERQUADRIQUE (HQ)

Une hyperquadrique (HQ) est un contour défini dans le plan (x, y) par l'équation implicite :

$$\psi(x, y, \lambda) = 0$$

avec : $\psi(x, y, \lambda) = \varphi(x, y, \lambda) - 1$

et : $\varphi(x, y, \lambda) = \sum_{k=1}^{Nh} |A_k . x + B_k . y + C_k|^{\gamma_k}$

N_h est le nombre de termes de la l'hyperquadrique HQ, supérieur ou égal à 2, et $\lambda = \{A_k, B_k, C_k, \gamma_k, \ \ \forall k = 1, N_h\}$ est le vecteur des paramètres de HQ, avec $\gamma_k > 0$.

1 - VISUALISER UNE HQ

1.1 Principe

Une façon simple de visualiser une hyperquadrique de paramètres λ est de tracer l'isovaleur de niveau 0 de la fonction $\boldsymbol{\psi(x, y, \lambda) = 0}$.

Pour guider le tracé, il est utile de déterminer les droites englobantes. En effet, en tout point (x,y) du contour HQ, la relation :

$$\sum_{k=1}^{Nh} |A_k . x + B_k . y + C_k|^{\gamma_k} = 1$$

implique que chacun des termes de la somme est inférieur à 1:

$$\forall k \qquad -1 \leq A_k . x + B_k . y + C_k \leq 1$$

Chacune de ces condition définit une bande dans le plan (x, y) :

Si $B_k \neq 0$ $\qquad -\frac{A_k}{B_k} . x - \frac{C_k}{B_k} - \frac{1}{B_k} \leq y \leq -\frac{A_k}{B_k} . x - \frac{C_k}{B_k} + \frac{1}{B_k}$

Si $B_k = 0$ $\qquad -\frac{C_k}{A_k} - \frac{1}{A_k} \leq x \leq -\frac{C_k}{A_k} + \frac{1}{A_k}$

Le contour HQ se situe donc dans le domaine défini par l'intersection de ces N_h bandes. Sur un plan pratique, l'analyse des intersections entre les différentes droites englobantes permet de définir la zone du plan dans laquelle l'isovaleur $\psi(x, y, \lambda) = 0$ se situe.

1.2 Travail à réaliser

Dans cette phase du projet, il est demandé d'écrire un programme qui permet de visualiser une hyperquadrique de paramètres donnés, ainsi que les droites englobantes, puis d'analyser l'influence des différents paramètres, en s'inspirant des slides 8 et 9 de la présentation du projet. Les résultats seront présentés de préférence sous la forme d'un notebook.

NB : Un notebook peut faire appel à des fonctions écrites dans des fichiers python classiques. Le notebook joue alors le rôle de programme principal, avec la possibilité d'associer texte, commentaires et résultats de code.

2 - FITTER UN NUAGE DE POINTS PAR UNE HQ

L'objectif final du projet est d'identifier les paramètres d'une hyperquadrique approchant au mieux un contour défini par un nuage de points. Dans un premier temps, il vous est proposé de travailler sur un problème simplifié ne comportant que deux paramètres à identifier. Cela permet de comprendre la démarche à mettre en œuvre plus facilement que pour le problème général. Par ailleurs, travailler sur un problème à deux inconnues permet de visualiser le fonctionnement de la méthode de recherche de ces deux paramètres.

2.1 Problème simplifié

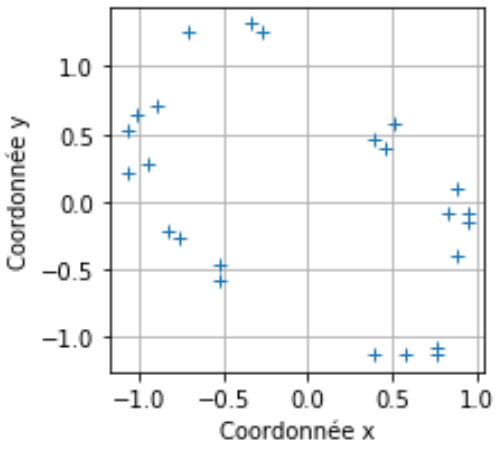
Le problème considéré est un cas particulier du problème général, avec $N_h = 2$. De plus, 6 des 8 paramètres sont connus. L'HQ est définie par l'équation ci-dessous, dans laquelle il n'y a que deux paramètres : a et b.

$$\psi(x, y, a, b) = 0$$

avec $\psi(x, y, a, b) = (a\,x + b\,y)^4 + (x + y)^4 - 1$

Les données à fitter sont un ensemble de points $\{(x_i, y_i)\}_{i=1,N}$. Le but du problème est de trouver le jeu de coefficients a et b tels que l'hyperquadrique passe au mieux par le nuage de points. Pour cela, il faut définir un critère de distance entre un point (x_i, y_i) et l'hyperquadrique HQ, dont on déduit un critère de distance entre l'ensemble des points et HQ. On propose de travailler avec le critère quadratique ci-dessous.

$$J(a, b) = \sum_{i=1}^{N} [\psi(x_i, y_i, a, b)]^2$$



Dans cette partie préliminaire du projet, un fichier comportant les coordonnées de 24 points (figure ci-contre) est fourni, et il faut déterminer les coefficients a et b qui minimisent le critère de distance $J(a, b)$.

Deux méthodes seront implémentées et testées : une descente de gradient à pas fixe, puis la méthode de Newton. Les algorithmes de ces méthodes dont donnés dans les sections 2.2 et 2.3.

2.2 Méthode du gradient

Le problème traité est de minimiser une certaine fonction $J(X)$, avec $X = (x_1, x_2) \in \mathbb{R}^2$. La fonction J est supposée différentiable à l'ordre 2[1] sur $\mathbb{R}^2$, et on suppose également que l'on connait l'expression analytique du gradient.

Principe des méthodes de gradient (ou méthodes de descente) :

De manière générale, les méthodes de descente sont des méthodes itératives qui génèrent une suite de points X_n tels que : $\forall n \geq 0 : J(X_n) < J(X_{n-1})$. Pour cela, une technique simple consiste à faire diminuer la valeur de J en se déplaçant d'une certaine quantité dans la direction de plus grande pente, donnée par le gradient au point considéré et normale à l'isovaleur passant par ce point.

Le schéma général de la méthode du gradient est donc le suivant :

$$X_n = X_{n-1} - \alpha . \nabla J(X_{n-1}) \qquad \text{avec} \quad \alpha > 0 \text{ choisi pour avoir : } J(X_n) < J(X_{n-1})$$

Selon la valeur choisie de α choisie, on se déplace plus ou moins loin dans la direction opposée au gradient.

Au voisinage de X_{n-1}, la fonction J est décroissante dans la direction opposée au gradient. En choisissant une valeur de α « très petite », on est sûr de faire diminuer J, ce qui garantit la convergence de l'algorithme, mais le déplacement à chaque itération, $\|X_n - X_{n-1}\|$, est petit et la convergence risque d'être longue. Il y a donc un compromis à trouver afin d'utiliser avec une valeur de α qui assure la convergence, sans trop la ralentir.

Gradient à pas fixe :

La méthode dite « à pas fixe » consiste à prendre une valeur de α constante. On note que le déplacement à chaque itération est proportionnel au module du gradient. La longueur du déplacement se réduit donc automatiquement au voisinage du minimum, assurant la convergence. Le problème est de choisir une valeur de α qui garantisse à la fois la convergence et l'efficacité de l'algorithme. En pratique, cette recherche est empirique (essai-erreur), à moins que l'on dispose d'informations sur la concavité de la fonction.

Gradient à pas optimal :

La méthode dite « à pas optimal » consiste à déterminer à chaque itération la valeur de α qui minimise la fonction J dans la direction du gradient. On traite donc un problème de minimisation par rapport à la variable α à chaque itération. Cette méthode est conceptuellement plus satisfaisante que la méthode à pas fixe, car elle évite la détermination empirique de α. En pratique, elle n'est pas vraiment plus efficace que la méthode à pas fixe, et ce pour deux raisons. La première raison est que la minimisation de J dans une direction à chaque itération a un certain coût de calcul. La deuxième est que les directions de recherche utilisées au cours de l'algorithme sont deux à deux orthogonales. Elles sont donc entièrement déterminées par le point initial et ne sont pas nécessairement adaptées au relief de la fonction dans le voisinage du minimum.

Schéma général de l'algorithme à implémenter :

Dans le cadre de ce projet, seule la méthode du gradient à pas fixe sera implémentée.

- Notations :
  - X_{n-1} et X_n : point initial et point final de l'itération n.
  - dX : longueur du déplacement pour une itération
  - n et n_{max} : compteur d'itérations et nombre maximal d'itérations autorisé
  - X_0 : point de départ de l'algorithme
  - α : pas de recherche

[1] C'est-à-dire que le vecteur gradient et la matrice hessienne sont définis en tous points

- ε : précision sur la position du minimum
- $converge$: indicateur booléen de convergence

- Algorithme :
  - Choix des paramètres de l'algorithme : X_0, α, ε, n_{max}
  - Initialisation : $X_{n-1} \leftarrow X_0$, $dX \leftarrow 1, n \leftarrow 0$
  - Tant que $dX > \varepsilon$ et $n < n_{max}$:
    - $X_n \leftarrow X_{n-1} - \alpha . \nabla J(X_{n-1})$
    - $dX \leftarrow \alpha . \|\nabla J(X_{n-1})\|$
    - $n \leftarrow n + 1$
  - $converge \leftarrow (dX \leq \varepsilon)$

2.3 Méthode de Newton

Quand elle est bien implémentée, la méthode des gradients est robuste. Toutefois sa vitesse de convergence est limitée (convergence d'ordre 1). Au voisinage de la solution, quand la fonction à minimiser est localement convexe, il peut être avantageux de mettre en place la méthode de Newton afin de converger plus rapidement (convergence d'ordre 2).

Principe de la méthode de Newton :

On souhaite trouver le minimum de la fonction J, supposée convexe sur $\mathbb{R}$, ou au moins dans le voisinage étudié. On suppose que l'on dispose du gradient ∇J et de la matrice hessienne HJ.

Soit X_{n-1}, une solution approchée. La fonction est approximée par $\tilde{J}_{n-1}$ son développement limité d'ordre 2 en X_{n-1}:

$$\widetilde{J_{n-1}}(X) = \tilde{J}_{n-1}(X_{n-1}) + \nabla J_{n-1}(X_{n-1})(X - X_n) + \frac{1}{2}(X - X_{n-1})^T . HJ_{n-1}(X_{n-1}) . (X - X_n)$$

La nouvelle solution approchée X_n est le minimum de $\widetilde{J_{n-1}}$, obtenu en annulant le gradient de $\widetilde{J_{n-1}}$.

$$\nabla \widetilde{J_{n-1}} = \nabla J_{n-1}(X_{n-1}) + HJ_{n-1}(X_{n-1})(X - X_n)$$

$$\nabla \widetilde{J_{n-1}} = 0 \iff \nabla J_{n-1}(X_{n-1}) + HJ_{n-1}(X_{n-1})(X - X_n) = 0$$

d'où : $X_n = X_{n-1} + \Delta X$, où ΔX est solution du système linéaire : $HJ_{n-1}(X_{n-1}) . \Delta X = -\nabla J_{n-1}(X_{n-1})$

Algorithme :

- Choix des paramètres de l'algorithme : X_0, ε, n_{max}
- Initialisation : $X_{n-1} \leftarrow X_0$, $dX \leftarrow 1, n \leftarrow 0$
- Tant que $dX > \varepsilon$ et $n < n_{max}$:
  - Calculer $\nabla J_{n-1}(X_{n-1})$
  - Calculer $HJ_{n-1}(X_{n-1})$
  - $\Delta X \leftarrow$ solution du système $HJ_{n-1}(X_{n-1}) . \Delta X = -\nabla J_{n-1}(X_{n-1})$
  - $X_n \leftarrow X_{n-1} + \Delta X$
  - $dX \leftarrow \|\Delta X\|$
  - $n \leftarrow n + 1$
- $converge \leftarrow (dX \leq \varepsilon)$

Remarque :

La méthode de Newton recherche des points critiques et peut donc converger vers un maximum. Il convient donc d'analyser le résultat obtenu par l'algorithme.

2.4 Travail demandé

Implémenter la méthode du gradient à pas fixe, puis la méthode de Newton afin de déterminer les coefficients a et b qui minimisent le critère de distance $J(a,b)$ défini dans la section 2.1.

Les isovaleurs de J seront tracées dans le plan (a,b), avec $-1. \leq a \leq 1.$ et $-1. \leq b \leq 1.$, de même que la suite de points intermédiaires (a_n, b_n). Le programme indique à l'utilisateur si la recherche de minimum a convergé, le nombre d'itérations réalisées et le dernier point obtenu (voir exemple de sortie en bas de page).

L'hyperquadrique ainsi obtenue sera alors superposée au nuage de points pour vérifier le résultat (voir exemple de sortie en bas de page).

Le test des algorithmes devra être fait très progressivement, avec quelques itérations de calcul pour commencer. En effet, il sera nécessaire de faire quelques ajustements pour éviter que les algorithmes « partent dans les décors ».

Une fois l'algorithme mis au point sur quelques itérations, les méthodes seront appliquées avec un critère d'arrêt $\varepsilon = 10^{-6}$.

Vous montrerez que la solution (a,b) n'est pas unique et dépend du point initial de l'algorithme. Vous mettrez en évidence les complémentarités entre méthode du gradient et méthode de Newton.

Documents à rendre :

- Présentation de la mise en œuvre des algorithmes de minimisation, avec en particulier le calcul du gradient et de la matrice hessienne : $\mathbf{\nabla J}(a,b) = \begin{bmatrix} \frac{\partial J}{\partial a}(a,b) \\ \frac{\partial J}{\partial b}(a,b) \end{bmatrix}$ et $H_J(a,b) = \begin{bmatrix} \frac{\partial^2 J}{\partial a^2}(a,b) & \frac{\partial^2 J}{\partial a \partial b}(a,b) \\ \frac{\partial^2 J}{\partial b \partial a}(a,b) & \frac{\partial^2 J}{\partial b^2}(a,b) \end{bmatrix}$.

- Code python (fichiers .py)

- Résultats commentés, sous forme d'un notebook ou d'un rapport classique, à votre convenance.

Exemples de sortie graphique, pour inspiration : isovaleurs du critère J – résultat intermédiaire du fit (la convergence n'est pas obtenue à 10^{-6} près et le fit, bien que satisfaisant, peut être amélioré.

```
Recherche de a et b par descente de gradient :
==============================================
Point de départ : a_ini = 0.000 - b_ini = -0.250
Point d'arrivée : a_fin = 0.733 - b_fin = -0.416
Convergence = False
n_iter = 31
Valeur finale du critère : 2.262
```