Projet 4
Systèmes d'équations non linéaires / Méthode de Newton-Raphson

Groupe 4 - Equipe 1

Responsable :
Secrétaire :
Codeurs :

Résumé : Ce projet a pour but de calculer la(les) racine(s) d'un système d'equations non-linéaires. La méthode proposée ici est celle de Newton-Raphson. L'objectif est d'évaluer les avantages et les inconvénients d'une telle solution. Cela sera fait en testant la méthode dans différents contextes.

1 Méthode de Newton-Raphson

Dans cette partie, la méthode de Newton-Raphson a été implémentée pour résoudre un système d'équations non linéaires. Cette méthode permet de donner une approximation de la racine de l'équation, qu'il s'agisse d'une équation unidimensionnelle ou multidimensionnelle.

1.1 Newton-Raphson standard

Soit $f : \mathbb{R}^n \to \mathbb{R}^n$ définie comme suit :
fonction ici
La jacobienne de cette fonction est la matrice des dérivées partielles selon chaque variable de cette fonction.
jacobienne ici
Le principe de la méthode peut être résumé comme suit :

— Étant donné un vecteur initial U_0, il faut trouver un vecteur U tel que pour tout ϵ petit $|f(U)| \leq \epsilon$.

Pour trouver ce vecteur, l'algorithme doit trouver un vecteur V qu'il faut ajouter au vecteur U pour approcher $f(U+V)$ de 0. Il est possible de remplacer $f(U+V)$ par $f(U)+J(U)\cdot V$, ainsi il faut résoudre l'équation $J(U)\cdot H = -f(U)$. Ceci va être fait en utilisant la fonction *numpy.linalg.lstsq*.

Ce processus sera répété au plus N_{max} fois. On peut donc conclure que cet algorithme est de complexité $O(N_{max} \times c)$ avec c la complexité de *numpy.linalg.lstsq*.

1.2 Newton-Raphson avec du backtracking

La méthode de Newton-Raphson permet bien de trouver la racine, mais pas dans tous les cas, voir figure 1, c'est pour cela qu'on va utiliser une méthode nommée du backtracking qui permet d'améliorer la convergence vers cette racine. Pour ceci, on ajoute des lignes de code qui permettent de vérifier si la condition $|f(U+\alpha \times V)| \leq |f(U)|$ est vérifiée ou pas. On a essayé de diviser α par deux à chaque itération mais on a rapidement remarqué que la convergence n'est pas assurée. On a donc dû diviser α par 1.1.

Pour conclure, on remarque bien que Newton-Raphson avec du backtracking converge vers la racine contrairement à celle standard selon la figure 1. Mais ceci nous coûte en complexité puisque la complexité de Newton-Raphson avec du backtracking est égale à $O(a \times N_{max} \times c)$ avec c la complexité de *numpy.linalg.lstsq*.

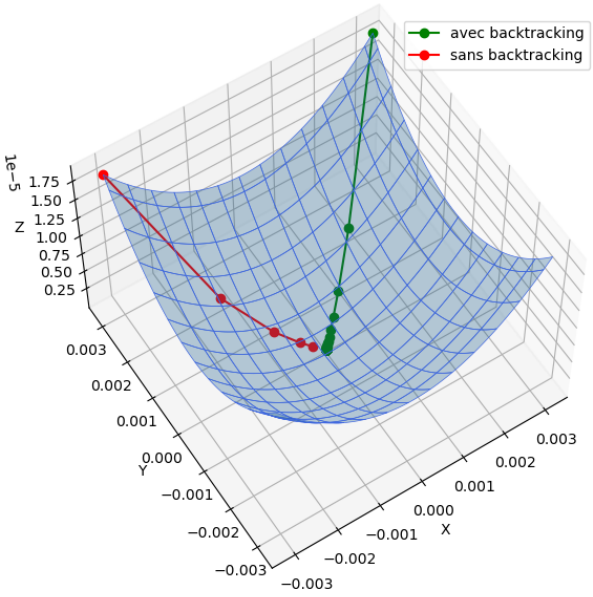

FIGURE 1 – Newton-Raphson avec et sans backtracking

2 Les points de Lagrange

Le but de cette partie est de trouver le point d'équilibre d'un objet sachant qu'il est soumis à plusieurs forces différentes.

2.1 Modélisation de différentes forces

Il a donc fallu dans un premier écrire des algorithmes qui vont permettre de calculer la force appliquer à un point en fonction de ses coordonées cartésiennes. Les différentes forces qui ont été choisies sont : la force élastique, la force centrifuge, et la force gravitationnelle. Les calculs qui suivent ont demandés aussi de calculer leur Jacobienne et de les implémenter en Python. Ce travail de modélisation a donc demandé plus de travail de calcul que de travail algorithmique.

Les forces sont représentées sous les formes suivantes :

$$f_e : \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} k(x-x0) \\ k(y-y0) \end{bmatrix}$$

$$f_g : \begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} -k \cdot \frac{x-x0}{((x-x0)^2+(y-y0)^2)^{\frac{3}{2}}} \\ -k \cdot \frac{y-y0}{((x-x0)^2+(y-y0)^2)^{\frac{3}{2}}} \end{bmatrix}$$

2.2 Recherche de points d'équilibre

Dans un premier nous avons recherché les points d'équilibres

3 Équilibre électrostatique

Dans cette partie, nous allons appliquer la méthode de Newton-Raphson à l'électrostatique. Nous allons calculer la Jacobienne de ∇E pour pouvoir ensuite chercher les positions d'équilibre d'un système et enfin voir si ces solutions correspondent à un maximum ou à un minimum de l'énergie.

3.1 Jacobienne de $\nabla E(x_1, x_2, ..., x_N)$

On considère qu'il existe N charges qui peuvent se déplacer dans l'intervalle [-1,1]. L'énergie électrostatique totale du système, E, se calcule avec la formule suivante.

$$E(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} \left(ln|x_i + 1| + ln|x_i - 1| + \frac{1}{2} \sum_{j=1, j\neq i}^{N} ln|x_i - x_j| \right)$$

Les dérivées partielles de E peuvent se calculer ainsi :

$$\frac{\partial E(x_1, ..., x_N)}{\partial (x_i)} = \frac{1}{x_i + 1} + \frac{1}{x_i - 1} + \sum_{j=1, j\neq i}^{N} \frac{1}{x_i - x_j}$$

On a alors ∇E tel que :

$$\nabla E(x_1, x_2, ..., x_N) = \left[\frac{\partial x_i}{\partial E(x_1, ..., x_N)} \right]$$

On veut calculer la Jacobienne, J, de ∇E. Pour ceci, on a implémenté les deux fonctions `compute_jacob_diag_coeff` et `compute_jacob_extra_coeff` qui calculent respectivement les coefficients diagonaux et les coefficients autres que ceux diagonaux de la matrice Jacobienne de taille n^2 à l'aide des formules suivantes :

$$J_{i,i} = \frac{\partial^2 E}{\partial x_i^2} = -\frac{1}{(x_i + 1)^2} - \frac{1}{(x_i - 1)^2} - \sum_{k=1, k\neq i}^{N} \frac{1}{(x_i - x_k)^2}$$

$$J_{i,j} = \frac{\partial^2 E}{\partial x_i \partial x_j} = -\frac{1}{(x_i - x_j)^2}$$

Après avoir implémenté ces deux fonctions, nous avons pu calculer la Jacobienne de ∇E à l'aide de la fonction `J_` qui utilise les deux fonctions précédentes.

3.2 Méthode de Newton-Raphson

Ensuite, dans cette section, nous allons chercher les positions d'équilibre. Pour ce faire, nous allons résoudre à l'aide de la méthode de Newton-Raphson, l'équation suivante :

$$\nabla E(x_1, x_2, ..., x_N) = \left[\frac{\partial x_i}{\partial E(x_1, ..., x_N)} \right] = 0$$

Nous avons pris pour X le tableau : [-0.8,-0.5,0.1,0.5]. Nous avons donc appliqué à ce tableau, la fonction `newton_raphson_backtracking` implémentée dans la Partie 1.2. Nous avons ainsi obtenu comme racines : [-0.92954991, -0.82677351, -0.58010903, -0.00758336] auxquelles nous avons rajouté

FIGURE 2 – Représentation des courbes des polynômes de Legendre et des racines de la Jacobienne de ∇E

les points -1 et 1. Afin de mieux visualiser les racines obtenues, nous avons décidé de placer les points sur un graphe, ce que l'on peut voir sur la Figure 2. Sur ce même graphe, nous avons tracé les courbes des polynômes de Legendre. On peut constater de manière graphique, sur la Figure 2, que la majorité des racines trouvées ressemblent aux racines des dérivées des polynômes de Legendre. En effet, on peut voir qu'une des racines obtenues est proche de 0 tout comme l'est la racine des dérivées des polynômes de Legendre de degré impair, c'est-à-dire, de degré 1,3 et 5. La racine proche de -0.58 est aussi proche de celle de la dérivée du polynôme de Legendre de degré 2. Enfin, celle proche de -0.93 est proche de celle de la dérivée du polynôme de Legendre de degré 6.

3.3 Maximum ou minimum de l'énergie électrostatique

Enfin, nous avons cherché à savoir si les solutions correspondent à un maximum ou à un minimum de l'énergie électrostatique. Pour cela, nous avons tracé l'énergie électrostatique en fonction de la position de la particule, x, ce que l'on peut voir sur la Figure 3. En effet, nous avons calculé E(x) pour des x allant de -1 à 1. On peut voir sur la Figure 3 que les solutions correspondent à un maximum de l'énergie électrostatique. En effet, le maximum est atteint en 0 sur le graphe.

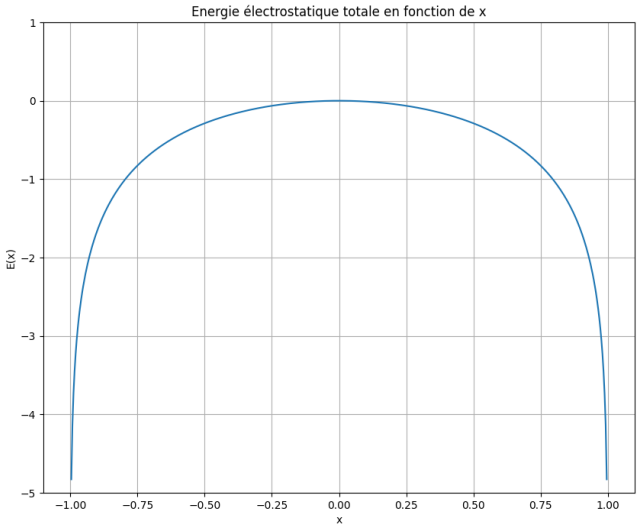

FIGURE 3 – Graphe de l'Energie électrostatique totale en fonction de x