LES 4 FANTASTIQUES : Super « zéros » en action

Nous allons étudier dans ce projet quatre algorithmes de *résolution de racines*. Notre objectif est de trouver des solutions numériques à une équation et non des solutions analytiques via des *algorithmes de résolution itératifs*.

Ce problème est fortement lié à des problèmes communs en physique. Minimiser une fonction `f(x)` est par exemple équivalent à trouver les « *zéros* » de sa dérivée ou trouver un point fixe tel que `g(x)=x`, au final équivalent à trouver les « **zéros** » de `g(x)−x`, etc.

Si on sait résoudre assez facilement ce type de problèmes analytiquement lorsque la fonction f est un polynôme d'ordre inférieur ou égal à 4, la résolution numérique devient indispensable pour toutes les autres fonctions. Que x soit un scalaire ou un vecteur (à n dimensions), la méthode générale pour la recherche de racines est liée à la capacité d'encadrer numériquement la région où l'équation ou le système d'équations possède une racine particulière. Le principe dominant la recherche de racines d'équations est celui des méthodes itératives où, en partant d'une certaine valeur, on s'approche de plus en plus près de la solution exacte.

Vous allez durant ce projet explorer et comparer différentes méthodes de calcul de racines pour des fonction définies plus bas.

1. méthode de la dichotomie (vue en TP)

La clé de cette méthode repose sur l'existence d'un encadrement préalable de cette racine. S'il existe un couple `(a,b)` tel que le produit `f(a)f(b)<0` et si la fonction est continue, le théorème de la valeur intermédiaire nous indique que la fonction s'annule au moins une fois à l'intérieur de cet intervalle.

Coder la fonction **`dichotomie(f, a, b, tolerance)`** qui renvoie la racine de la fonction f avec la tolérance choisie.

2. méthode de Regula-Falsi (dite méthode de la position fausse)

La méthode de la position fausse approche la fonction de manière linéaire par la fonction `y = cx + d` passant par $f(a_i)$ et $f(b_i)$ en a_i et b_i dans l'intervalle (a_i, b_i) considéré.

On détermine la racine estimée par la racine de l'équation donnée par `cx + d = 0`. On reprend alors le principe de l'algorithme précédent.

Coder la fonction **`regula_falsi(f, a, b, tolerance)`** qui renvoie la racine de la fonction f avec la tolérance choisie.

3. méthode de Newton-Raphson

La méthode de Newton-Raphson nécessite en plus de la fonction f dont on cherche à déterminer une racine x, qu'elle soit dérivable au voisinage de celle-ci.

Les itérations successives de cette méthode sont basées sur le développement limité de la fonction autour d'un point.

$$f(x_{n+1}) = f(x_n) + f'(x_n)\delta + \sigma(\delta) + \ldots \text{ avec } \delta = x_{n+1} - x_n$$

La méthode consiste à choisir x_{n+1} tel que $f(x_{n+1}) \approx 0$ soit :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Géométriquement, cette méthode revient à calculer x_{n+1} comme le point où la tangente à f(x) en x_n coupe l'axe des abscisses.

***Nota** : si la dérivée de f n'est pas connue analytiquement, l'évaluation numérique est quand même possible par la formule d'accroissement (C'est alors la méthode de la sécante). L'initialisation nécessite deux point x_0 et x_1 proches si possible de la solution recherchée (c'est une généralisation de la méthode regula-falsi)*

On remplace alors la dérivée par la formule suivante : $f'(x_n) = \frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}$

Coder la fonction **`newton_raphson(f, a, b, tolerance)`** qui renvoie la racine de la fonction f avec la tolérance choisie.

4. méthode de Muller

La méthode de Muller est aussi basée sur la méthode de la sécante mais utilise une approximation quadratique d'une partie de la fonction et offre une convergence plus rapide.

La méthode nécessite 3 points x_0, x_1 et x_2 qui sont proches si possible de la solution recherchée.

On pose ainsi :

$$q = \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}$$

On définit ensuite trois termes :

$$A = qf(x_n) - q(1+q)f(x_{n-1}) + q^2 f(x_{n-2})$$
$$B = (2q+1)f(x_n) - (1+q)^2 f(x_{n-1}) + q^2 f(x_{n-2})$$
$$C = (1+q)f(x_n)$$

La relation de récurrence est finalement donnée par

$$x_{n+1} = x_n - (x_n - x_{n1})\frac{2C}{\max(B \pm \sqrt{B^2 - 4AC}}$$

Coder la fonction **`muller(f, a, b, tolerance)`** qui renvoie la racine de la fonction f avec la tolérance choisie.

Les calculs de racines se feront sur les fonctions suivantes (vous **déterminerez les intervalles de départ** pour chacune d'elles et **expliquerez** vos choix)

- `f(x) = 2e^x-10x^2`
- `g(x) = -20x.cos(-3x)`
- `h(x) = -2x^2.tan(x)+2`
- `l(x) = 2x-2/x^2`
- `m(x) = ln((3x^2-2)/sin(x^3))`
- `n(x) = (1/x^7)+2`

Affichage

Pour chacune des fonctions, écrire un **script/programme** permettant de choisir la fonction à afficher avec son intervalle et sa précision et permettant d'afficher graphiquement la racine calculée sous forme d'un '**+**' rouge pour les quatre méthodes.

Evaluation

Au-delà de l'aspect graphique, il est indispensable de **quantifier** la qualité des calculs effectués.

La **qualité** va s'évaluer quand cela est possible en comparant les résultats numériques obtenus avec une valeur de référence (*la racine calculée analytiquement par exemple*)

Les méthodes de calcul seront évaluées *a minima* (vous pouvez proposer d'autres variables) au moyen de trois variables : le **nombre d'itérations** utilisées pour effectuer pour le calcul, **l'erreur par rapport à la valeur de référence** et le **temps de calcul** de la racine pour les fonctions proposées.

Les résultats d'évaluation seront présentés :

1. **sous la forme d'une matrice** pour chaque fonction où pour chacune des méthodes utilisée, nous trouverons pour chacun des intervalles demandés : le résultat de référence de la racine, le résultat trouvé avec la méthode utilisée, la précision demandée et le temps de calcul.
2. **sous la forme de graphiques** représentant pour chaque méthode, le nombre d'itérations utilisées par rapport à la précision demandée.

Argumentez minutieusement vos critères de comparaison des méthodes et **conclure** sur la meilleure méthode à utiliser pour calculer une racine.

Travail à rendre

Vous devrez rendre pour le **dimanche 12 janvier 2025 23h55 GMT dernier délai** par voie électronique (à l'adresse **Philippe.Truillet@irit.fr ou Clement.Truillet@irit.fr**) :

- Les fichiers **.m, .py** ou .**ipynb** que vous aurez créés. Il vous est conseillé de faire un fichier pour chaque fonction de calcul et un fichier pour l'évaluation et la comparaison des méthodes de calcul ;
- Un rapport décrivant les algorithmes que vous aurez programmés en expliquant votre démarche de travail.

Une présentation orale de 10 minutes (avec documents pour la vidéo-projection) aura enfin lieu après la fin du projet dans la semaine du **13 au 17 janvier 2025**