TP : Programmation – analyse numérique
Dichotomie - Newton

TP Dichotomie-Newton

1. (a) Programmer les algorithmes de **Dichotomie et de Newton** appeler `dichotomie`, `newton_residu et newton_increment` suivant le test d'arrêt pour l'algorithme de Newton. Tester les sur la fonction $f(x) = x^2 - 2$ avec $[a, b] = [1, 2]$.

   À l'aide de Scipy : Importer la fonction `fsolve` de la bibliothèque `scipy.optimize`, la commande `fsolve`$(f, a_0, xtol)$ renvoie une solution approchée de $f(x) = 0$ pour la condition initiale a_0 dès que $|x_{n+1} - x_n| \leq xtol$ où (x_n) est la suite des valeurs approchées.

   (b) Afficher pour les trois méthodes et pour $\epsilon \in \{10^{-1}, 10^{-2}, \cdots, 10^{-10}\}$:

   Le nombre d'itérations, la valeur approchée $\tilde{x}$ trouvée d'un zéro de f, epsilon, $f(\tilde{x})$.

   Pour la méthode de dichotomie, afficher également les résultats pour $\epsilon \in \{2^0, 2^{-1}, 2^{-2}, \cdots, 2^{-10}\}$. Commenter.

   (c) Modifier les algorithmes précédents en substituant à la précision ϵ le nombre d'itérations n, on appellera les algorithmes `dichotomie_bis` et `newton_bis`.

   (d) Tracer dans une même fenêtre, la courbe reliant l'erreur $|u_n - \sqrt{2}|$ en fonction du nombre d'itérations pour chacun des deux algorithmes. Commenter.

   (e) En modifiant, les programmes élaborés à la question 1. (a), écrire un programme qui reproduit les dessins du cours. On les appellera : `dichotomie_graphe` et `newton_graphe`.

   (f) Observer le comportement de la méthode de Newton, à l'aide de la fonction `newton_graphe`, avec la fonction $f(x) = xe^{-x^2}$ pour $u_0 = 0.52$, puis $u_0 = 0.48$. Commenter.

TP : Programmation – analyse numérique
Dichotomie - Newton

2. **Dans l'algorithme de Lagrange**, on approxime le graphe de f sur $[a, b]$ par la sécante qui relie les points $A = (a, f(a))$ et $B = (b, f(b))$. L'abscisse du point d'intersection de la sécante (AB) avec l'axe des abscisses donne une approximation u_0 de la solution de l'équation $f(x) = 0$ sur $[a, b]$. Puis, on poursuit en considérant l'intervalle $[u_0, b]$ si $f(u_0)$ est du même signe que $f(a)$ et l'intervalle $[a, u_0]$ sinon. La suite (u_n) ainsi construite converge (sous certaines hypothèses sur f) vers un zéro de f sur $[a, b]$.
   (a) Écrire l'algorithme de Lagrange en pseudo-code.
   (b) Programmer l'algorithme de Lagrange, l'appeler `lagrange`.
   (c) Écrire un programme qui permet de visualiser l'évolution de la suite et les sécantes associées, l'appeler `lagrange_graphe`.