DEVOIR 2

Analyse Numérique pour l'ingénierie — Hiver 2025
MAT-2910

Consignes

- L'équipe doit être différente de celle du précédent devoir, sauf si vous êtes seul(e).
- Une fonction Python (pointfixe.py) accompagne le présent questionnaire (voir site web). Elle ne doit pas être modifiée.
- **À remettre :** Un fichier compressé contenant un rapport et 3 programmes Python : un script devoirXX.py où XX désigne le numéro de votre équipe, la fonction Python pointfixe.py et la fonction Python secante.py qui doit être créée.
- Lorsqu'on lance le script devoirXX.py, les 10 figures demandées (Figures 1 à 10) doivent apparaitre. Dans chaque figure, indiquer ce qu'il y a en abscisse et en ordonnée, et mettre un titre, voire une légende (commandes xlabel, ylabel, title, legend de matplotlib.pyplot).
- Si les figures n'apparaissent pas, le devoir se verra attribuer la note 0 aux questions correspondantes. De même si le numéro de la figure ne correspond pas à ce qui est demandé pour ce numéro. Si le programme ne fonctionne pas, le devoir se verra attribuer la note 0.
- Le rapport doit contenir les réponses aux questions et les 10 figures.
- ***Définissez vos fonctions à l'aide de numpy.***

Soit

$$f(x) = (x+1)(x-1)^2,$$

qui a deux racines, $r_1 = -1$ et $r_2 = 1$.
On va utiliser plusieurs méthodes pour voir comment elles se comportent pour obtenir des approximations de ces racines. Sauf indication contraire, on utilisera pointfixe.py avec $N = 50$ et $tol = 10^{-7}$. Pour trouver la racine r_1 on partira toujours de $x_0 = -1.5$, et pour trouver r_2, on partira toujours de $x_0 = 1.5$ (voir le début du programme pour la signification de tous ces paramètres). Notons que la sortie de pointfixe.py est un vecteur x dont les composantes sont les éléments de la suite (x_n) à partir de $n = 0$.

1. **Méthode du point fixe**.
   On se propose d'utiliser une méthode de point fixe avec la fonction

$$g(x) = x - f(x)/5.$$

   a) Dans le rapport, démontrer si la méthode converge ou non, à quel ordre et avec quel taux de convergence le cas échéant, pour chacune des racines.

   b) Dans le script Python *devoirXX.py* :
   - Appliquer la méthode de point de fixe pour r_1 en faisant appel à la fonction pointfixe.py.
   - Faire ensuite tracer la courbe de l'erreur $E_n = |x_n - r_1|$ en fonction de n en échelle semi-logarithmique avec la commande semilogy de matplotlib.pyplot (Figure 1), puis tracer le rapport E_{n+1}/E_n en fonction de n (Figure 2).
   - Faire de même pour r_2 et mettre les courbes correspondantes dans les figures 1 et 2 avec les courbes précédentes (utiliser la commande *matplotlib.pyplot.show()*).

   c) Dans le rapport, mettre les 2 figures et expliquer brièvement comment les figures confirment les résultats théoriques obtenus en a).

2. **Méthode de Newton**

   a) Dans le rapport, démontrer mathématiquement si la méthode de Newton converge linéairement ou au moins à l'ordre 2 pour chacune des racines. Pour r_2, autour de quelle valeur devrait se situer le rapport E_{n+1}/E_n ?

   b) Dans le script Python *devoirXX.py* :
   - Appliquer la méthode de Newton pour r_1 en faisant appel à la fonction pointfixe.py avec le choix approprié de g.
   - Faire ensuite tracer la courbe de l'erreur $E_n = |x_n - r_1|$ en fonction de n en échelle semi-logarithmique avec la commande semilogy de matplotlib.pyplot (Figure 3), puis tracer le rapport E_{n+1}/E_n^2 en fonction de n (Figure 4).

— Faire de même pour r_2 en mettant la courbe d'erreur dans la figure 3 précédente (utiliser la commande legend de matplotlib.pyplot pour indiquer quelle courbe est laquelle), tandis qu'on fera tracer le rapport E_{n+1}/E_n dans une nouvelle figure (Figure 5).

c) Dans le rapport, mettre les nouvelles figures (Figures 3, 4 et 5), et expliquer brièvement comment les figures confirment les résultats théoriques obtenus en a).

3. **Méthode de Steffenson :** On considère la méthode de point fixe de la question 1, pour appliquer la méthode de Steffenson.

   a) Dans le script *devoirXX.py*, faire appel à *pointfixe.py* avec le choix adéquat de g pour réaliser la méthode de Steffenson et pour trouver une approximation de r_1. Faire tracer la courbe de l'erreur $E_n = |x_n - r_1|$ (Figure 6), puis tracer E_{n+1}/E_n^2 (Figure 7).

   b) Dans le rapport, mettre les nouvelles figures et commenter les résultats obtenus (sont-ils fidèles aux propriétés connues de la méthode de Steffenson ?).

4. **Méthode de la sécante :**

   a) On veut appliquer la méthode de la sécante. À cette fin, on créée une fonction secante.py en dupliquant la fonction pointfixe.py et en la modifiant le moins possible. Les entrées de cette fonction sécante seront une fonction f, x_0, x_1, N et tol.

   b) Dans le script, appliquer cette méthode pour r_1, en choisissant $x_0 = -1.5$ et $x_1 = -1.25$, faire tracer la courbe de l'erreur dans la Figure 8 et faire tracer E_{n+1}/E_n^α, où α désigne le nombre d'or (Figure 9).

   c) Dans le rapport, mettre les nouvelles figures et commenter les résultats obtenus.