TP14 Recherche dichotomique.

ECG1

Mars 2023

Exercice 1 (Modules utiles)**.**
Vous devez importer les modules

- matplotlib.pyplot avec l'alias plt
- numpy avec l'alias np

I Algorithme de dichotomie pour la résolution d'équation

On sait que pour une fonction continue sur un intervalle $[a,b]$ telle que $f(a)$ et $f(b)$ sont de signe opposé, il existe (au moins) un réel x_0 dans $[a,b]$ tel que $f(x_0)=0$, c'est le théorème des valeurs intermédiaires

Remarque importante

Pour savoir si deux réels u et v sont de signe opposé, il suffit de tester si $u\times v$ est négatif.

L'algorithme

1. On commence par tester si f change de signe
2. On calcule le milieu de a,b que l'on nomme c
3. Si f change de signe entre a et c revenir à l'étape 2 en ayant remplacé b par c. Dans le cas contraire c'est a que l'on remplace par c
4. La fonction s'arête quand $b-a$ est plus petit que la précision voulue, et la valeur renvoyée est a qui est une valeur approchée d'une solution x_0 de l'équation $f(x)=0$.

Exercice 2.
On vous demande de créer une fonction `dichotomie (f,a,b,eps)` qui calculer une valeur approchée d'une solution l'équation $f(x)=0$ sur l'intervalle $[a,b]$ à la précision **eps**. On supposera, sans le vérifier que f change de signe entre a et b.

Vous **DEVEZ** tester votre fonction. Par exemple essayer de calculer une valeur approchée de $\sqrt{2}$, $\sqrt{3}$.

II Rappels : algorithme de dichotomie pour la recherche d'un élément dans un tableau trié

Dans toute la suite on confondra tableaux et listes, les listes ne contiendront que des valeurs numériques (float or int) et ne seront pas vides.

II.1 Préliminaires

La méthode **.sort()** permet de trier un tableau ou une liste dans l'ordre croissant. Cette méthode effectue un tri en place, elle modifie la liste.

Exercice 3 (A essayer)**.**
Recopier et tester le code suivant.

```
l=[1,3,2,6,1]
l.sort()
```

Exercice 4 (Vérifier qu'une liste est triée)**.**
Écrire une fonction croissant(l) qui prend comme argument une liste l non vide et qui renvoie `True` si la liste est triée dans l'ordre croissant, `False` sinon.

Exercice 5 (Vérifier qu'une liste est triée (décroissant))**.**
Écrire une fonction decroissant(l) qui prend comme argument une liste l non vide et qui renvoie `True` si la liste est triée dans l'ordre décroissant, `False` sinon.

II.2 L'algorithme de recherche dichotomique

On s'intéresse à une liste triée **dans l'ordre croissant** et on étudie un algorithme de recherche dichotomique dont le principe est le suivant

1. Si la liste est constitué d'un seul élément il suffit de vérifier si cet élément est égal l'élément cherché
2. Sinon on observe le milieu de la liste. 1. Si l'élément cherché est égal à l'élément du milieu la recherche est terminée 2. Si l'l'élément cherché est strictement plus grand que l'élément du milieu alors l'élément recherché est éventuellement dans la deuxième moitié de la liste

3. Sinon on recherche l'élément dans la première moitié.
4. On recommence l'étape précédente

INDICATIONS

1. pour obtenir l'indice milieu entre i et j on doit utiliser l'expression $(i+j)//2$, où // est l'opérateur quotient dans la division euclidienne
2. On utilise une boucle
3. while

Exercice 6 (présence)**.**
Écrire une fonction `recherche_dichotomique(lt,x)` qui renvoie `True` si x est dans `lt` une liste triée, `False` sinon :

Exercice 7 (position)**.**
Écrire une fonction `recherche_dichotomique_indice(lt,x)` qui renvoie la position de x si x est dans la liste triée, `-1` sinon :