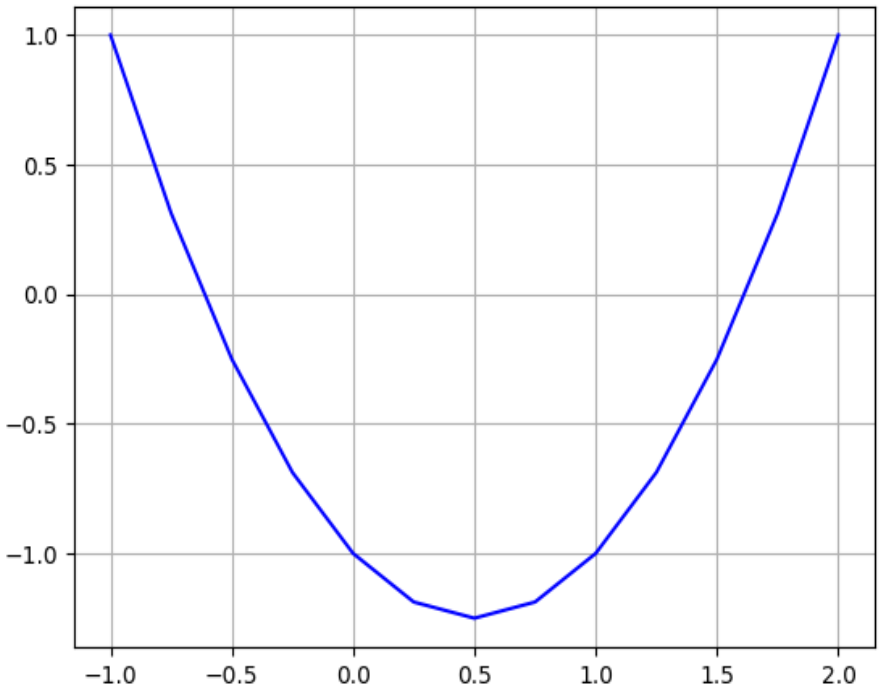
TP1 :
Résolution numérique de f (x) = 0



Nom :

Hamdane Amine 11606514
Kentsa Habib Edgar 11607247

Groupe :

L2 Mathématiques

Nom du professeur :

Cardinal Jean-Paul

Année 2017-2018

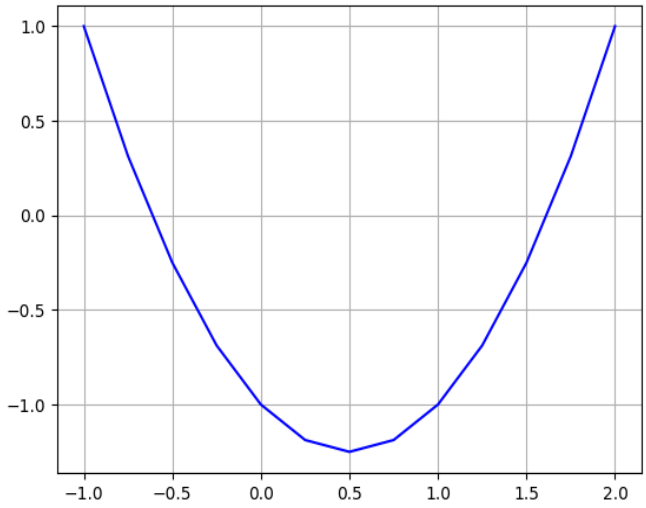
Introduction :

Nous avons créés un ficher tp1.py a partir du terminal avec la fonction « cd »
Ensuite nous avons basculer vers un terminal python grâce a la fonction « ipython »
Enfin nous avons pu commencer à coder dans notre fichier puis compiler à partir du terminal grâce a la fonction « run tp1,py »

Question 1 : Création d'une fonction simple et représentation graphique.

a) Nous avons créé une fonction f(x) qui renvoie x^2-x-1 selon les valeurs de x.

b) Nous avons créé deux liste de réel grâce a une boucle while une contenant les valeur de x et l'autre contenant les valeurs de y=f(x).
Ensuite en ouvrant la bibliothèque matplotlib.pyplot nous traçons la courbe f(x) en utilisant les valeurs stockées dans les liste X et Y.



c) A partir du graphique créé à la question B nous pouvons voir les différentes racines de la f onction f(x) qui sont environs -0,6 et 1,6.

<u>Question 2 :</u> Recherche d'un zéro d'une fonction au moyen de la méthode du point fixe

a) **.**On créé une fonction g(x) qui renvoie 1+1/x selon les valeurs de x.
Ensuite on trace la fonction g(x) et une droite affine x=y, l'intersection nous donne le point fixe de g(x) qui correspond au zéro de f(x).

b) A l'aide d'une boucle for on calcule les 25 premiers thermes de la fonction g(x) avec x=1 en premier thermes et x=g(x) en thermes suivant, On stock ensuite la suite dans une liste X.

c) On remarque que cette suite converge vers le point fixe de g(x) soit 1.6180339886704433

<u>Question 3 :</u> implémentation de la fonction « point_fixe »

a)

Durant l'écriture de la fonction point_fixe, nous avons rencontré plusieurs problèmes notamment comment poser la condition dans le « while ». Nous pensions que tout comme dans le langage « c » il était possible de l'écrire ainsi :

```
def point_fixe(g, X[0], e):

    while X[i+1]-X[i]> e:

        r = g(X[I+1])

    return r
```

b)

Nous avons aussi rencontré des problèmes dans la façon dont doit être présenté le programme pour qu'il puisse être exécuté sans envoyer de messages d'erreur. Mais avec un raisonnement logique et l'aide du professeur nous avons pu écrire une fonction qui donne approximativement les bonnes valeurs dont :

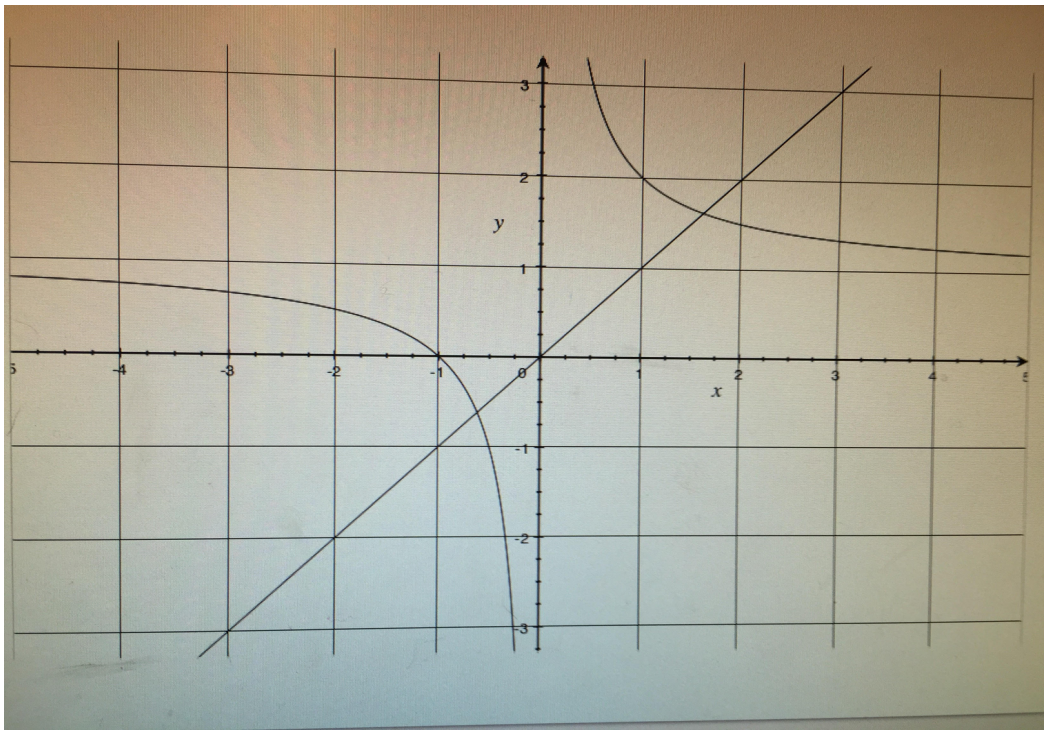
- pour x0 = 1,0 et e = 10^{-12} on obtenait r=1.6180339887496482 et ceci après 30 itérations

- pour x0 = 1,0 et e = 10^{-12} on obtenait r=1.6180339887496482 et ceci après 36 itérations

Ces résultats nous montrent que plus la valeur initiale est petite plus il faut d'itérations pour trouver le « r » qui convient.

c)

Voici le graphe de la fonction g ainsi que la première bissectrice avec g(x) = 1 + 1/x.
Nous pouvons remarquer que les points d'intersections entre la courbe de g et de la bissectrice correspondent aux 0 de la fonction f(x).



<u>Question 4 :</u> La méthode de Newton

a)

Après vérification à la main nous sommes tentés de dire que les points fixes de g sont bien les zéros de la fonction f et convergent vers 1,6.

b)

Après avoir longuement cogité quand a la méthode d'écriture de la méthode de Newton en python nous somme venus à un résultats satisfaisant que vous pouvez observer dans le code à la partie « def newton ».
La méthode de Newton se fait en cherchant le point fixe de g à partir de la fonction f et de sa dérivé df, cette méthode semble plus rapide que la méthode du point fixe, celle ci ce fait en se référant à la valeur précédente tout comme la méthode du point fixe.

c)

Nous avons tester notre fonction avec les caractéristiques suivant dans un premiers temps :
f (x) = x 2 − x − 1, x 0 = 1.0, # = 10 −12
Cela nous donne une valeur du point fixe qui est r=1.618033988749895 en rentrant seulement 6 fois dans la boucle alors que pour la méthode précédente nous somme passé 30 fois dans la boucle.

Nous avons tester notre fonction avec les caractéristiques suivant dans un second temps :
f (x) = x 2 − x − 1, x 0 = −1.0, # = 10 −12
Cela nous donne une valeur du point fixe qui est r=-0.6180339887498948 en rentrant seulement 5 fois dans la boucle.

<u>Question 5 :</u>La méthode de la sécante

a)

La méthode de la sécante est une méthode itérative où chaque

approximation est construite à partir des deux approximations précédentes. On doit donc partir de deux valeurs initiales distinctes, x 0 , x 1 puis on calcule par récurrence les termes de la suite x n+1= x n − f (x x n n)−f(x n−1) f (x n). L'avantage sur la méthode
de Newton est qu'on n'a pas besoin de la dérivée de f.

La méthode de la sécante nous était totalement inconnu nous avons du donc comprendre dans un premier temps son fonctionnement afin de pouvoir la coder en python. Une fois cela fait, grâce à l'expérience acquise en python au court du TP nous avons pu coder facilement cette méthode.

b)

Après avoir fait tourner la fonction avec deux valeurs précises et on a obtenu :
- pour x0 = 1.5, x 1 = 2.0, epsi = 10 −12 on obtient 1.6180339887498947, avec au total 6 itérations
- pour x0 = −1.0, x 1 = −0.5, epsi = 10 −12 on obtient -0.6180339887498948, après 8 itérations

<u>Question 6 :</u>La méthode de dichotomie

a)

La méthode de dichotomie est une méthode qui consiste à déterminer le(s) plus petit(s) intervalle dans lequel se trouve le(s) zéros d'une fonction. Par cette méthode, nous pourons déterminer la valeur par défaut ou par excès.

Voir l'implémentation dans le fichier tp1.py

b)

Après avoir fait tourner la fonction avec deux valeurs précises et on a obtenu :
- pour a = 1.5, b = 2.0, epsi = 10 −12 on obtient [1.6180339887496302, 1.6180339887505397]
- pour a = −1.0, b = 0.0, epsi = 10 −12 on obtient [-0.6180339887505397, -0.6180339887496302]