Cours-D?rivation-Dichotomie-Exemple12

November 6, 2019

0.1 Import des bibliothèques Python

```
In [1]: import numpy as np                #pour disposer des tableaux de type array
        import matplotlib.pyplot as plt   #pour les graphiques

In [3]: %matplotlib inline
        #pour l'affichage des graphiques dans la page et non pas dans une fenetre pop up

In [4]: import operator                   #pour utiliser les opérateurs de base sous forme de

In [5]: from sympy import *               #pour le calcul formel
        init_printing()
        t = symbols('t')

In [10]: def dérivée(exp, t):
             return diff(exp,t)

         def simplifier(exp):
             return simplify(exp)

         def factoriser(exp):
             return factor(exp)
```

0.2 Résolution approchée de $f(x) = 0$ par dichotomie, exemple 12 du cours

Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = 40x^3 - 561x^2 + 1917x - 200$ definie sur $\mathbb{R}$ et dérivable sur $\mathbb{R}$.

0.2.1 Question 1 : Calcul de dérivée

```
In [7]: #expression de f(x)
        fexp = 40 * t**3 - 561*t**2 + 1917 * t - 200
        fexp
```

Out[7]:

$$40t^3 - 561t^2 + 1917t - 200$$

```
In [8]: #expression de f'(x)
        fprimexp = dérivée(fexp, t)
        fprimexp
```

Out[8]:

$120t^2 - 1122t + 1917$

In [11]: `factoriser(fprimexp)`

Out[11]:

$3\,(4t - 9)\,(10t - 71)$

0.2.2 Questions 2 et 3 Etude des variations de f

In [12]:
```
f = lambdify(t, fexp,"numpy")
fprim = lambdify(t, fprimexp,"numpy")
```

In [13]:
```
#tracé des  courbes  de f et f'
#Message d'erreur pour f' pour le point d'abscisse 0 (division par 0)
xmin, xmax, ymin, ymax = 0, 9, -600 , 1800
plt.axis([xmin, xmax, ymin, ymax])
tx = np.linspace(xmin, xmax, 1001)
ty = f(tx)
tz = fprim(tx)
plt.axhline(color='blue')
plt.axvline(color='blue')
plt.grid(True)
plt.plot(tx, ty, linestyle='-', linewidth=2, color='red', label=r'$y=f(x)$')
plt.plot(tx, tz, linestyle='-', linewidth=1, color='green', label=r"$y=f'(x)$")
plt.legend(loc='lower right')
```

Out[13]: `<matplotlib.legend.Legend at 0x7f8c4f7c5390>`

0.2.3 Question 4 Existence de solutions de l'équation $f(x) = 0$ sur $[8;9]$

- $f : x \mapsto 40x^3 - 561x^2 + 1917x - 200$ est dérivable donc continue sur $[8;9]$
- $f(8) < 0$ et $f(9) > 0$
- f est strictement croissante sur $[8;9]$

D'après un corollaire du théorème des valeurs intermédiaires, l'équation $f(x) = 0$ possède donc une unique solution α dans l'intervalle $[8;9]$

0.3 Résolution approchée par balayage

```
In [15]: def balayage(g, a, b, pas, k):
             """Retourne un intervalle d'amplitude pas encadrant l'unique solution de g(x)= k
             dans l'intervalle [a,b]"""
             if g(a) < k:
                 comparaison = lambda u, v : operator.lt(u,v)
             else:
                 comparaison = lambda u, v : operator.gt(u,v)
             x = a
             #en-tete du tableau
             print('|{etape:^16}|{t:^12}|{ft:^12}|'.format(etape='Etape', t='t', ft='g(t)'))
             count = 1
             while comparaison(g(x), k):
                 print('|{etape:^16}|{t:^12.6f}|{ft:^12.6f}|'.format(etape=count,t=x, ft=g(x)))
                 x += pas
                 count += 1
             print('|{etape:^16}|{t:^12.6f}|{ft:^12.6f}|'.format(etape=count,t=x, ft=g(x)))
             return x - pas, x
```

```
In [16]: balayage(f, 8, 9, 0.1, 0)
```

```
|     Etape      |      t       |     g(t)     |
|       1        |  8.000000    |-288.000000   |
|       2        |  8.100000    |-221.870000   |
|       3        |  8.200000    |-147.520000   |
|       4        |  8.300000    | -64.710000   |
|       5        |  8.400000    | 26.800000    |
```

```
Out[16]:
```
$(8.299999999999999, 8.399999999999999)$

```
In [23]: balayage(f, 8.3, 8.4, 0.01, 0)
```

```
|     Etape      |      t       |     g(t)     |
|       1        |  8.300000    | -64.710000   |
|       2        |  8.310000    | -55.954460   |
|       3        |  8.320000    | -47.111680   |
|       4        |  8.330000    | -38.181420   |
```

```
|       5        |  8.340000  | -29.163440 |
|       6        |  8.350000  | -20.057500 |
|       7        |  8.360000  | -10.863360 |
|       8        |  8.370000  | -1.580780  |
|       9        |  8.380000  |  7.790480  |
```

Out[23]:
(8.37, 8.379999999999999)

```
In [22]: balayage(f, 8.36, 8.38, 0.001, 0)
```

```
|     Etape      |     t      |    g(t)    |
|       1        |  8.360000  | -10.863360 |
|       2        |  8.361000  | -9.939086  |
|       3        |  8.362000  | -9.013927  |
|       4        |  8.363000  | -8.087883  |
|       5        |  8.364000  | -7.160954  |
|       6        |  8.365000  | -6.233140  |
|       7        |  8.366000  | -5.304440  |
|       8        |  8.367000  | -4.374854  |
|       9        |  8.368000  | -3.444383  |
|       10       |  8.369000  | -2.513025  |
|       11       |  8.370000  | -1.580780  |
|       12       |  8.371000  | -0.647649  |
|       13       |  8.372000  |  0.286370  |
```

Out[22]:
(8.370999999999993, 8.371999999999993)

0.4 Résolution approchée par dichotomie

0.4.1 Fonctions Python

```
In [19]: def dicho(f,a,b,e, k):
             """valeur approchée à e près de la solution de f(x)=k
             sur [a,b]. On admet que l'utilisateur saisit des paramètres
             où la dichotomie peut s'appliquer. Construit une figure
             à chaque étape.
             Les valeurs de a et b affichées sont celles en sortie de boucle."""
             etape = 0 #nombre d'étapes
             if f(a) <= f(b):
                 #croissant ou du moins passage de - à +
                 croissant = True
             else:
                 croissant = False
             while b - a > e:
                 etape += 1
```

```
            #on calcule le milieu du segment [a,b]
            m = (a+b)/2
            #figure
            x = np.linspace(a,b,500)
            y = f(x)
            plt.xlim(a,b)
            if croissant:
                plt.ylim(max(f(a),-50),min(f(b),50))
            else:
                plt.ylim(max(f(b),-50),min(f(a),50))
            plt.plot(x,y,color='red')
            plt.grid(True)
            plt.axhline(k)
            plt.title('a=%.4f et m=%.4f et b=%.4f'%(a,m,b))
            plt.show()
            #fin de la figure
            s = (f(m) - k)*(f(a) - k)
            #si f(m) - k  et f(a) - k sont de meme signe, f(x)=k dans ]m,b[
            if s > 0:
                a = m
            #si f(m) - k  et f(a) - k sont de signes opposés, f(x)=k dans ]a,m]
            else:
                b = m
        return a, b, etape

    def dicho_tab(f,a,b,e, k):
        """valeur approchée à e près de la solution de f(x)=k
        sur [a,b]. On admet que l'utilisateur saisit des paramètres
        où la dichotomie peut s'appliquer.
        Ne retourne rien mais remplit un tableau avec les valeurs
        de a, b et m aux differentes etapes.
        Les valeurs de a et b affichées sont celles en sortie de boucle"""
        count = 0 #nombre d'étapes
        if f(a) <= f(b):
            #croissant ou du moins passage de - à +
            croissant = True
        else:
            croissant = False
        #en-tete du tableau
        print('|{etape:^16}|{median:^12}|{test:^10}|{binf:^12}|{bsup:^12}|'.format(etape=
                                                                                  binf='a', bsup='
                                                                                  test='Choix ?'))
        #première ligne du tableau
        #remplissage de la ligne du tableau
        print('|{etape:^16}|{median:^12}|{test:^10}|{binf:^12}|{bsup:^12}|'.format(etape=
                                                                                  binf=a, bsup=b,
                                                                                  test=str(None)))
        while b - a > e:
```

```
        count += 1
        #on calcule le milieu du segment [a,b]
        m = (a+b)/2
        s = (f(m) - k)*(f(a) - k)
        #si f(m) - k  et f(a) - k sont de meme signe, f(x)=k dans ]m,b[
        if s > 0:
            a = m
        #si f(m) - k  et f(a) - k sont de signes opposés, f(x)=k dans ]a,m]
        else:
            b = m
        #remplissage de la ligne du tableau
        print('|{etape:^16}|{median:^12}|{test:^10}|{binf:^12}|{bsup:^12}|'.format(eta
                                                                                    binf=a, bsup=b, m
                                                                                    test= 'gauche' i
```

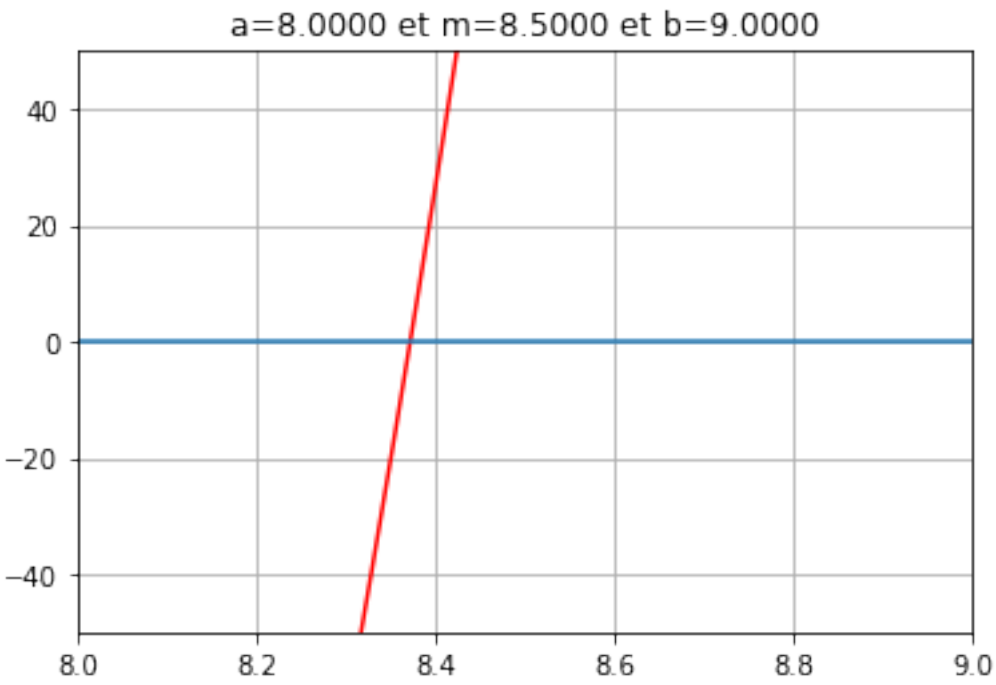
0.5 Résolution par dichotomie de l'équation $f(x) = 0$ dans l'intervalle $[8;9]$

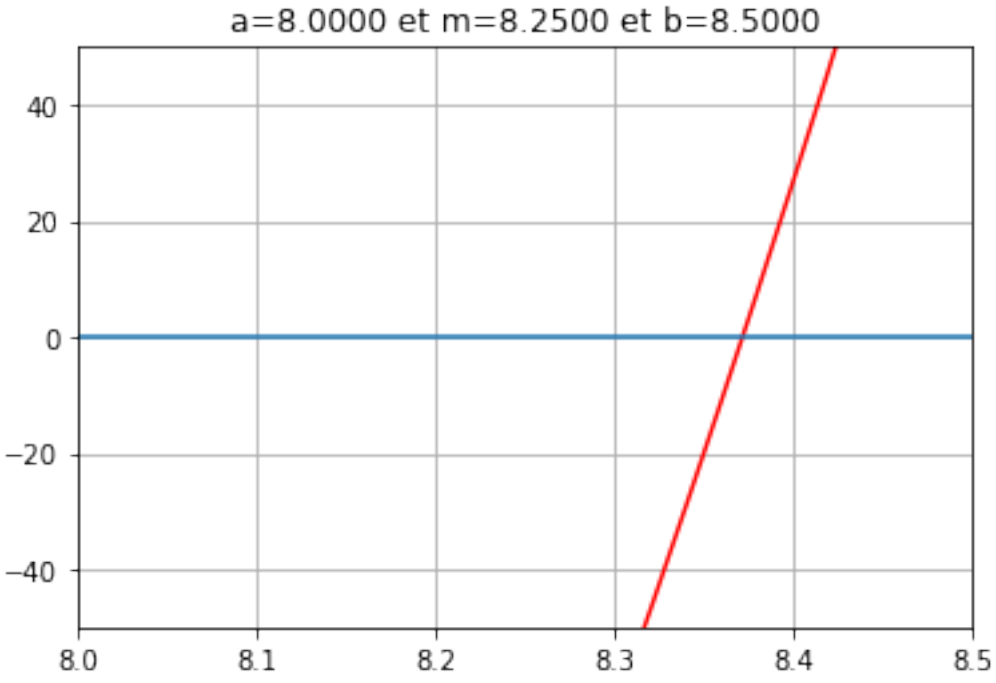
0.5.1 D'abord on s'arrete lorsque l'amplitude de l'intervalle $[a,b]$ est inférieure ou égale à $0,02$

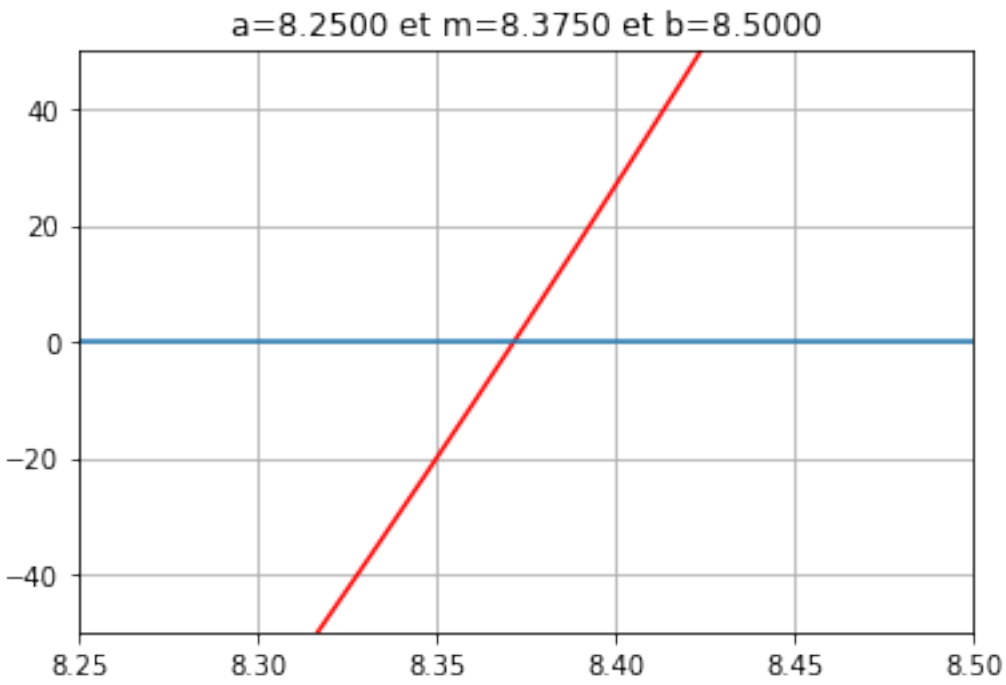
```
In [29]: dicho_tab(f, 8, 9, 0.02, 0)
```

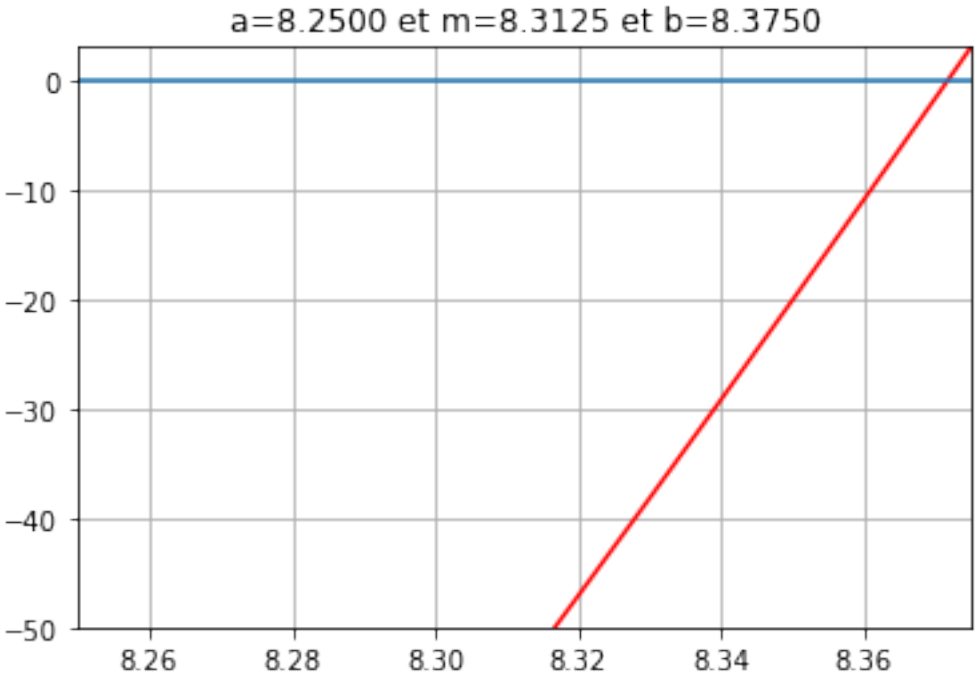
Etape	m	Choix ?	a	b
initialisation	None	None	8	9
1	8.5	gauche	8	8.5
2	8.25	droite	8.25	8.5
3	8.375	gauche	8.25	8.375
4	8.3125	droite	8.3125	8.375
5	8.34375	droite	8.34375	8.375
6	8.359375	droite	8.359375	8.375

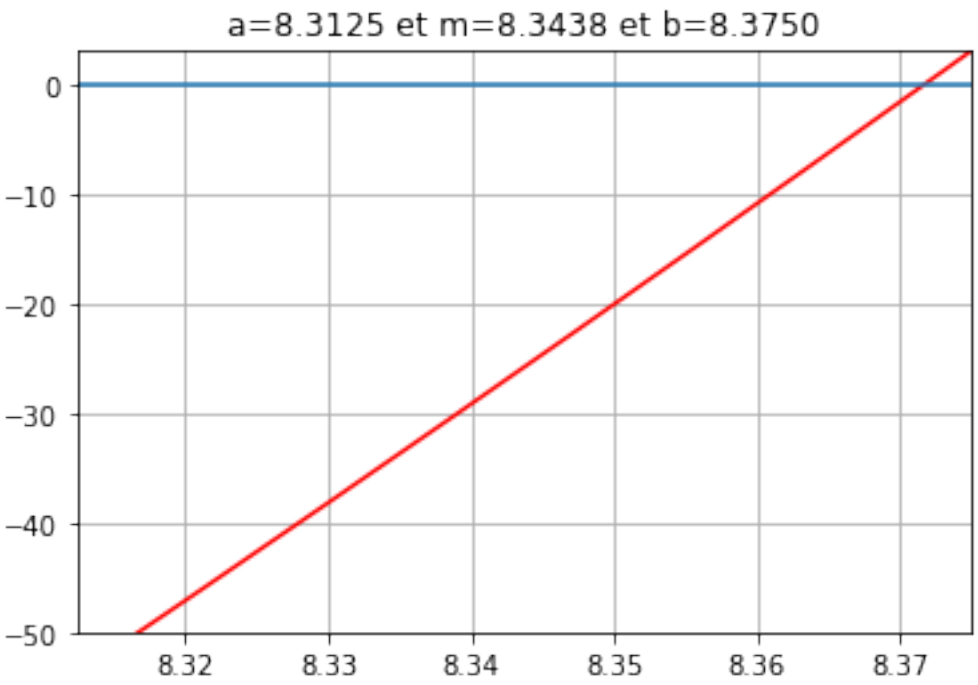
```
In [30]: dicho(f, 8, 9, 0.02, 0)
```

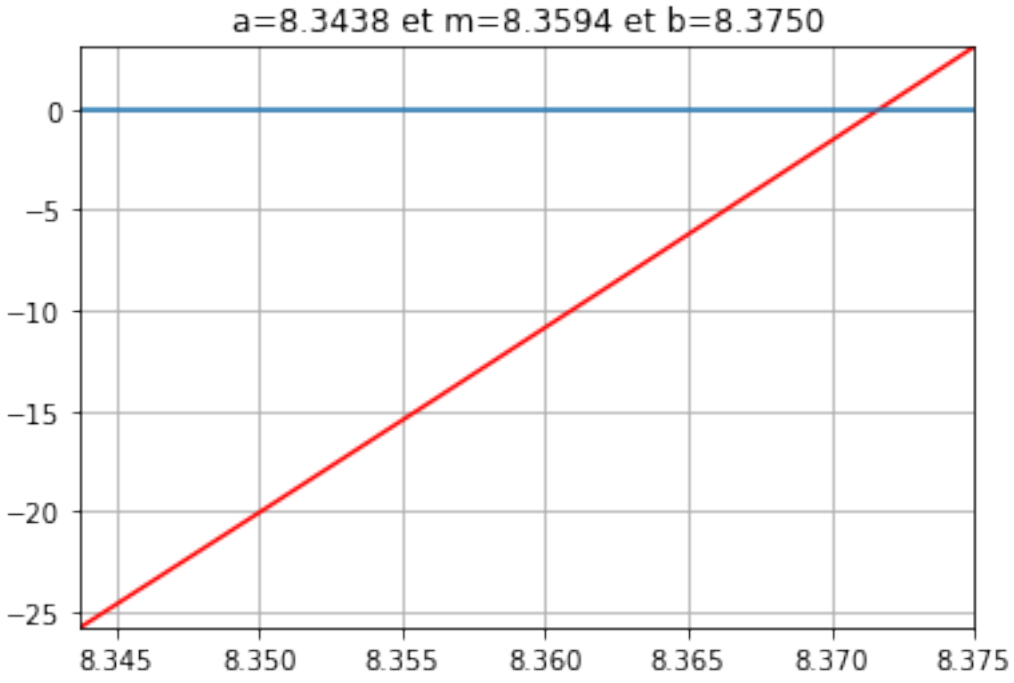

a=8.0000 et m=8.5000 et b=9.0000
40
20
0
−20
−40
8.0
8.2
8.4
8.6
8.8
9.0


a=8.0000 et m=8.2500 et b=8.5000
40
20
0
−20
−40
8.0
8.1
8.2
8.3
8.4
8.5


a=8.2500 et m=8.3750 et b=8.5000
40
20
0
−20
−40
8.25
8.30
8.35
8.40
8.45
8.50


a=8.2500 et m=8.3125 et b=8.3750
0
−10
−20
−30
−40
−50
8.26
8.28
8.30
8.32
8.34
8.36





Out[30]:

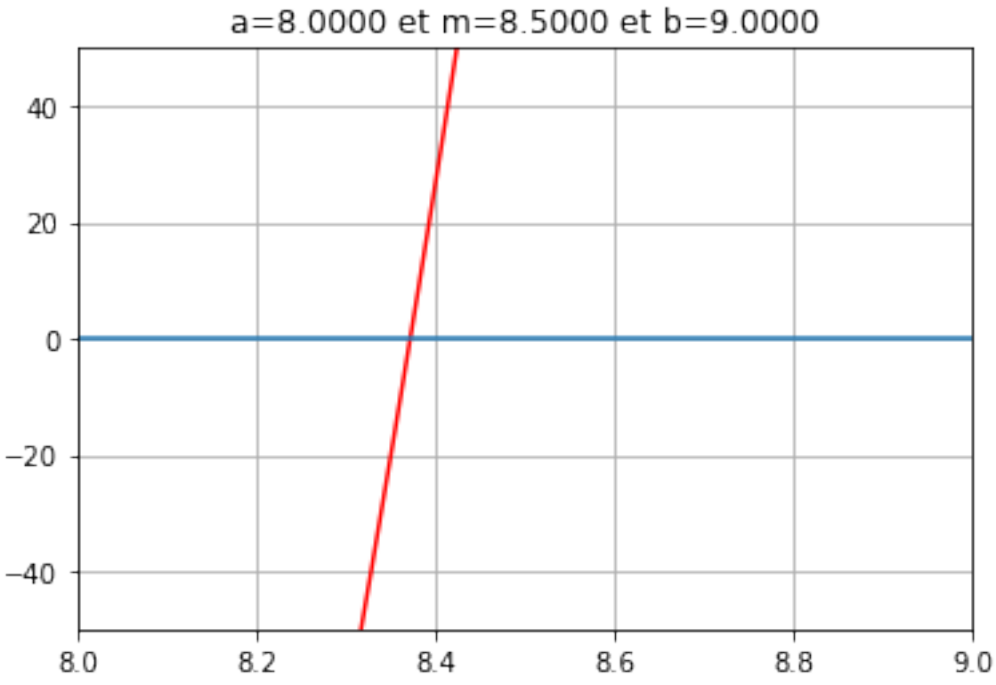
(8.359375, 8.375, 6)

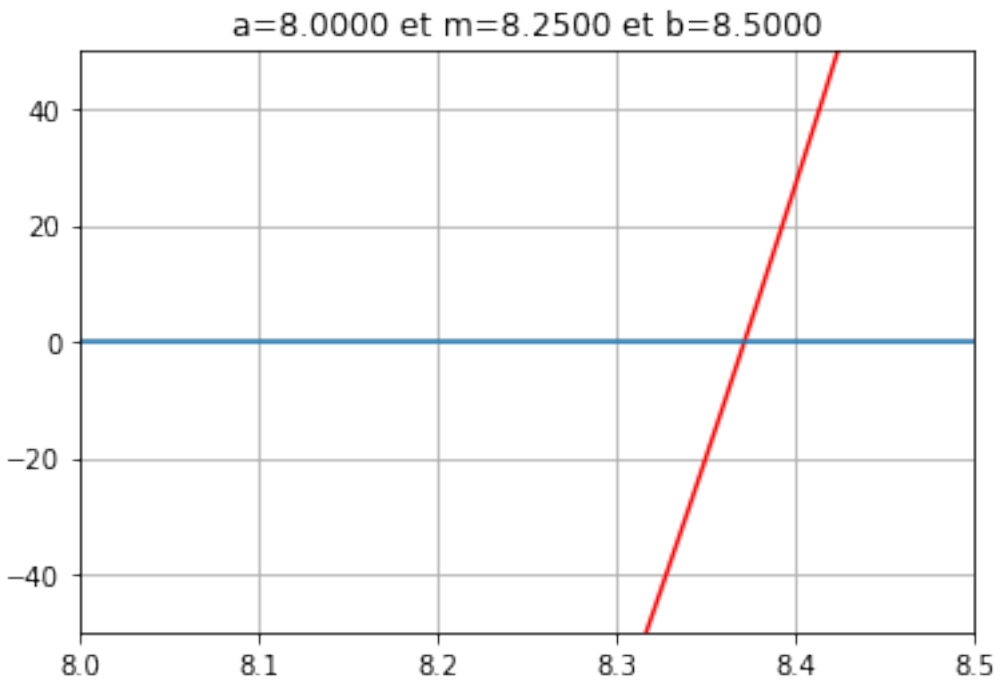
0.5.2 Ensuite on s'arrete lorsque l'amplitude de l'intervalle $[a, b]$ est inférieure ou égale à $0,002$

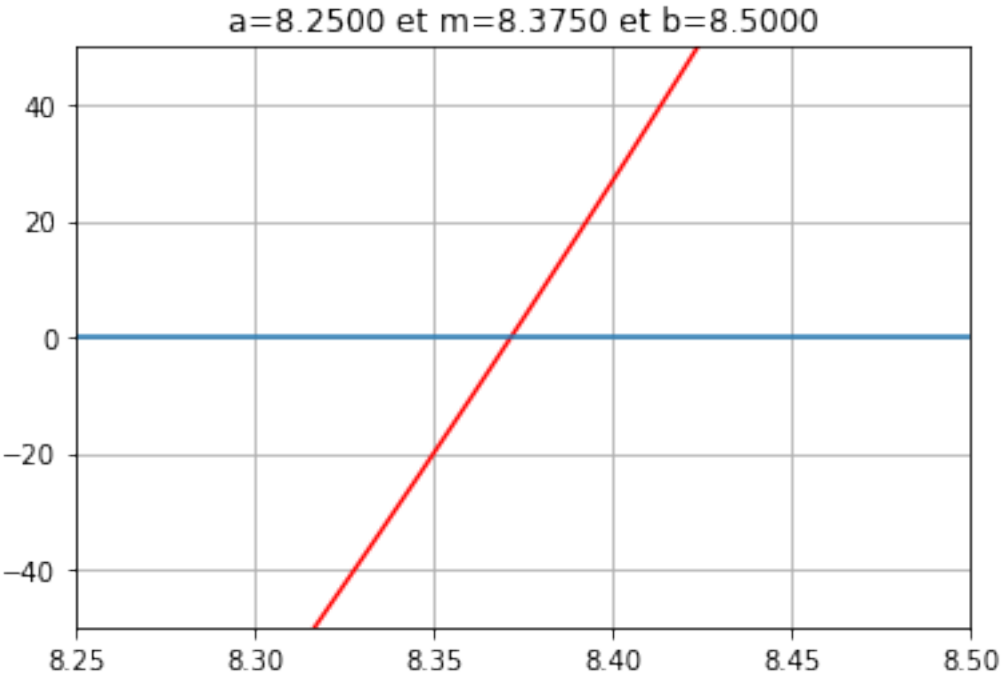
In [31]: `dicho_tab(f, 8, 9, 0.002, 0)`

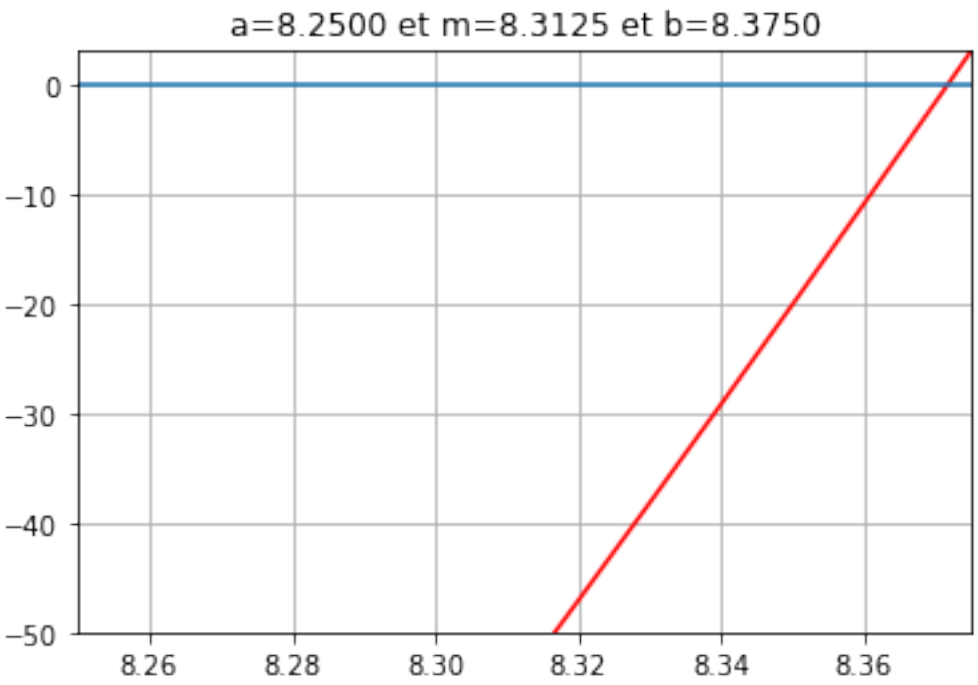
Etape	m	Choix ?	a	b
initialisation	None	None	8	9
1	8.5	gauche	8	8.5
2	8.25	droite	8.25	8.5
3	8.375	gauche	8.25	8.375
4	8.3125	droite	8.3125	8.375
5	8.34375	droite	8.34375	8.375
6	8.359375	droite	8.359375	8.375
7	8.3671875	droite	8.3671875	8.375
8	8.37109375	droite	8.37109375	8.375
9	8.373046875	gauche	8.37109375	8.373046875

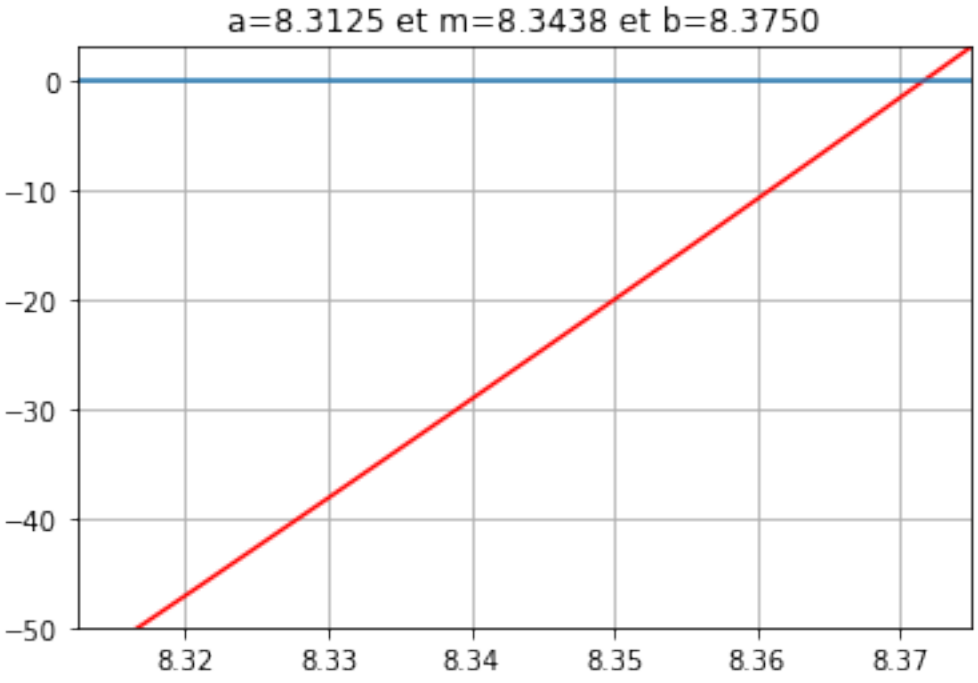
In [32]: `dicho(f, 8, 9, 0.002, 0)`

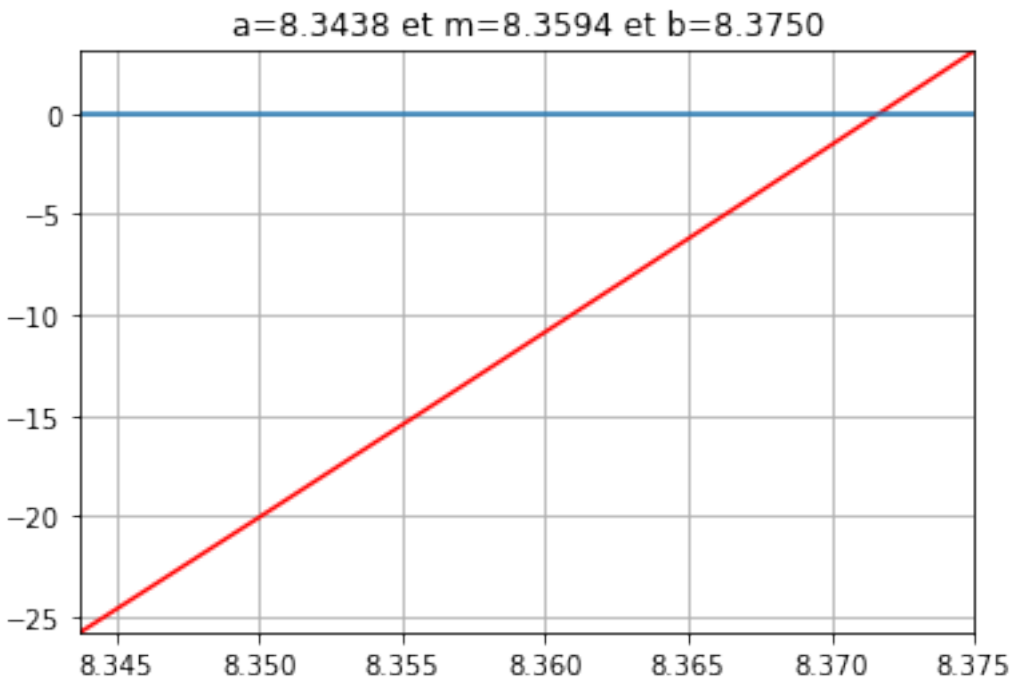


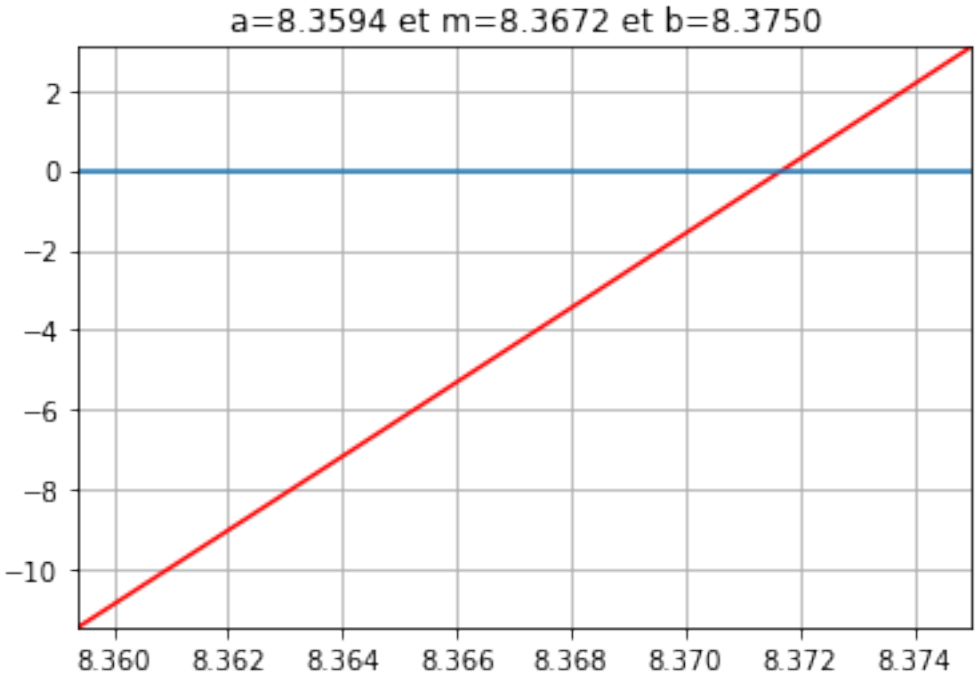

a=8.0000 et m=8.2500 et b=8.5000
40
20
0
−20
−40
8.0
8.1
8.2
8.3
8.4
8.5

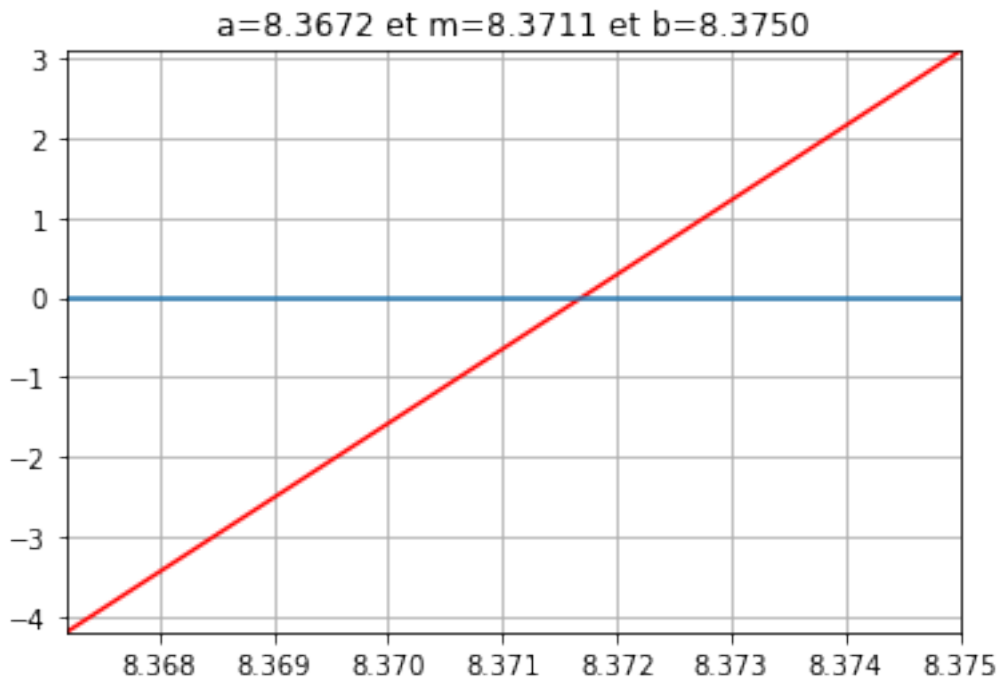

a=8.2500 et m=8.3750 et b=8.5000
40
20
0
−20
−40
8.25
8.30
8.35
8.40
8.45
8.50

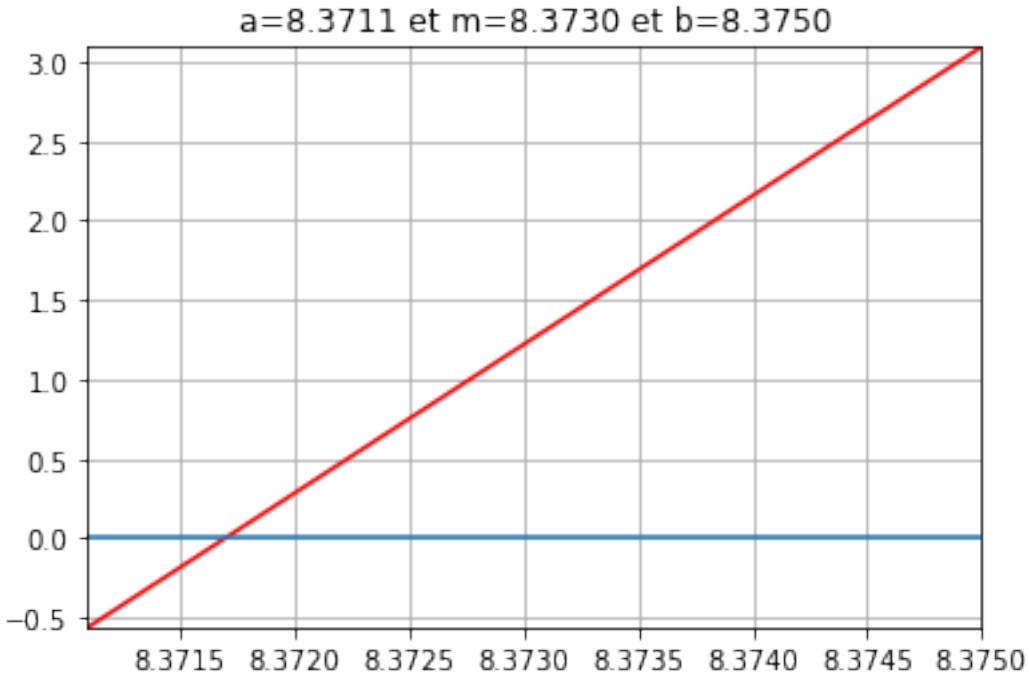

a=8.2500 et m=8.3125 et b=8.3750
0
−10
−20
−30
−40
−50
8.26
8.28
8.30
8.32
8.34
8.36


a=8.3125 et m=8.3438 et b=8.3750
0
−10
−20
−30
−40
−50
8.32
8.33
8.34
8.35
8.36
8.37


a=8.3438 et m=8.3594 et b=8.3750
0
−5
−10
−15
−20
−25
8.345
8.350
8.355
8.360
8.365
8.370
8.375


a=8.3594 et m=8.3672 et b=8.3750
2
0
−2
−4
−6
−8
−10
8.360
8.362
8.364
8.366
8.368
8.370
8.372
8.374





Out[32]:

(8.37109375, 8.373046875, 9)

```
In [28]: n = 0
         while 1/2**n > 0.001:
             n += 1
         print(n)
```

```
10
```

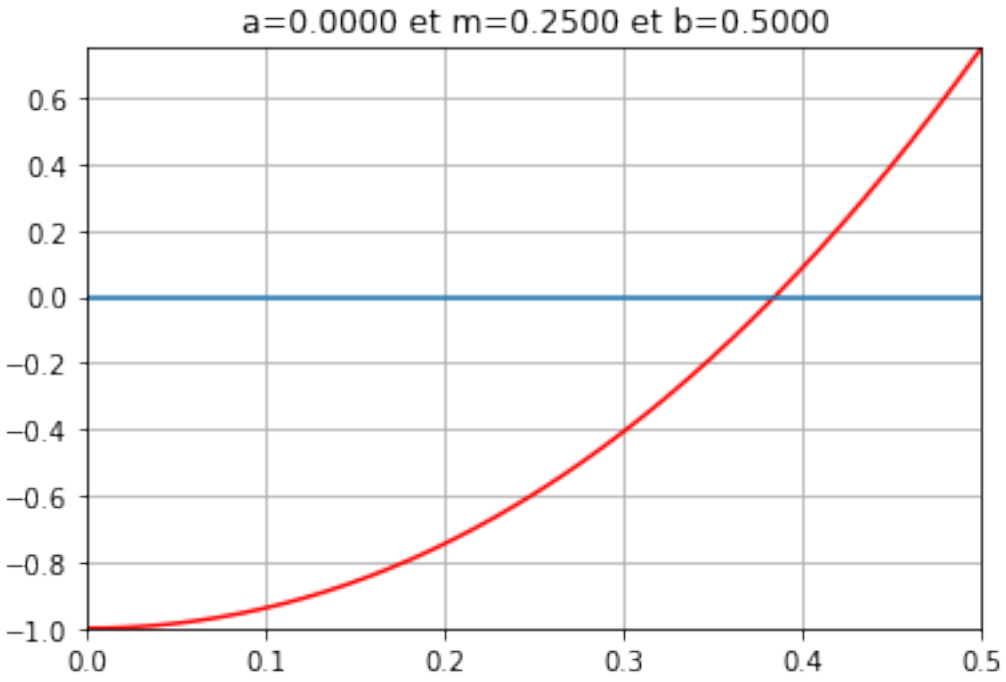
L'intervalle de départ a pour amplitude 1 ($a_0 = -2$ et $b_0 = -1$) A chaque étape l'amplitude de l'intervalle de recherche est divisée par 2 Au bout de n étapes, l'amplitude de la zone de recherche est de $\frac{b_0 - a_0}{2^n}$ soit $\frac{1}{2^n}$ ici. Avec la calculatrice (voir ci-dessus ou le logarithme népérien) on peut vérifier que le plus petit entier n tel que $\frac{1}{2^n} \leqslant 0,001$ est $n = 10$

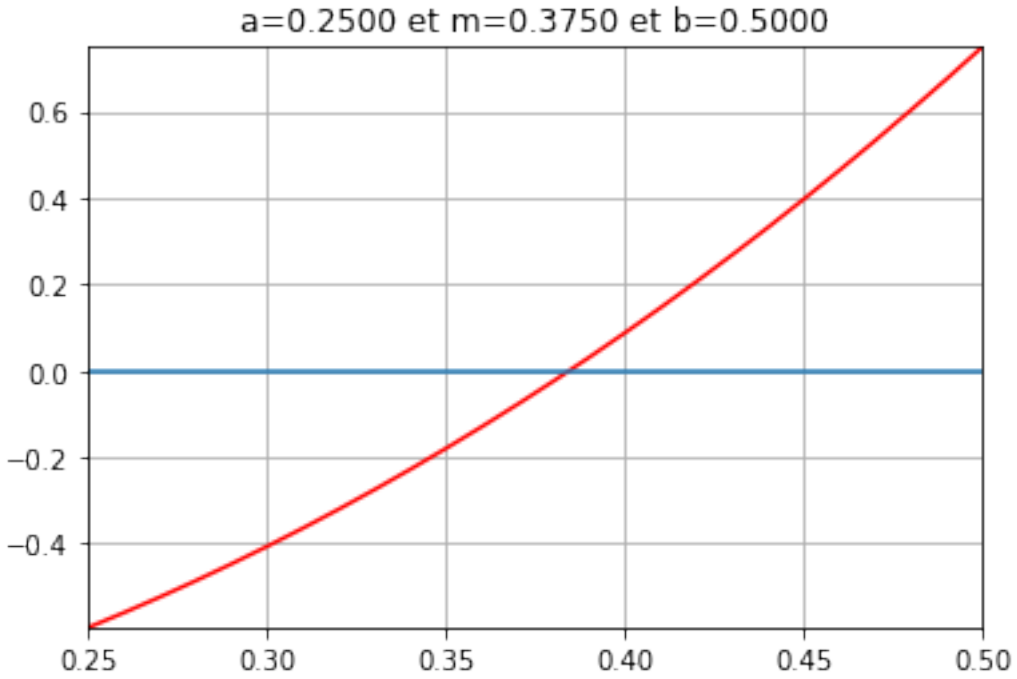
0.6 Exercice 3 TVI et dichotomie fiche d'exercices 2018

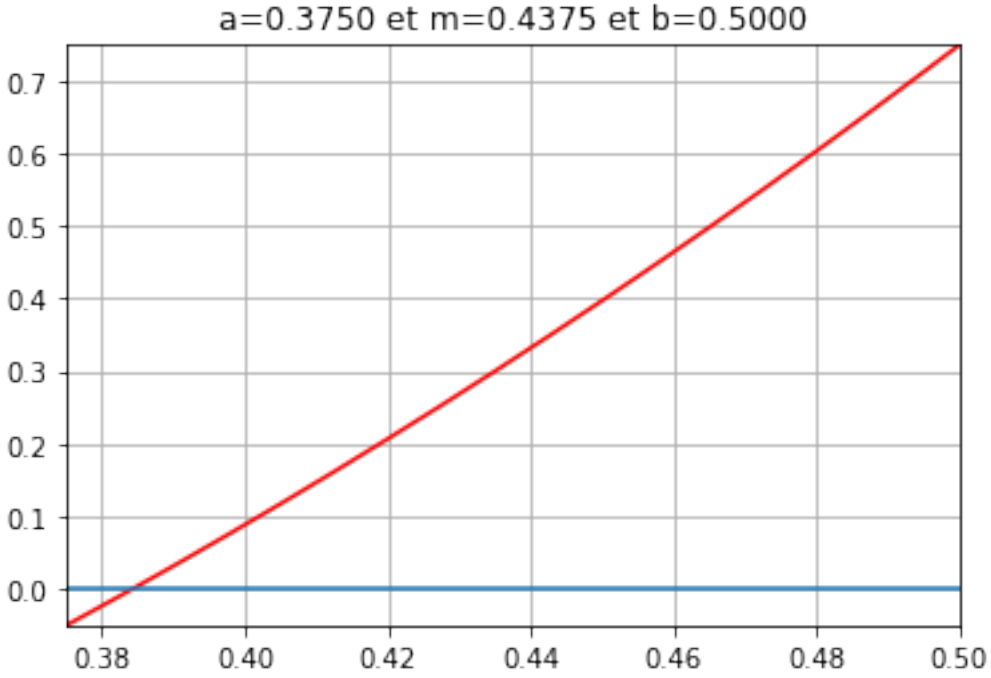
```
In [22]: dicho_tab(lambda x : 2 * x ** 3 + 6 * x ** 2 - 1, 0, 0.5, 0.01, 0)
```

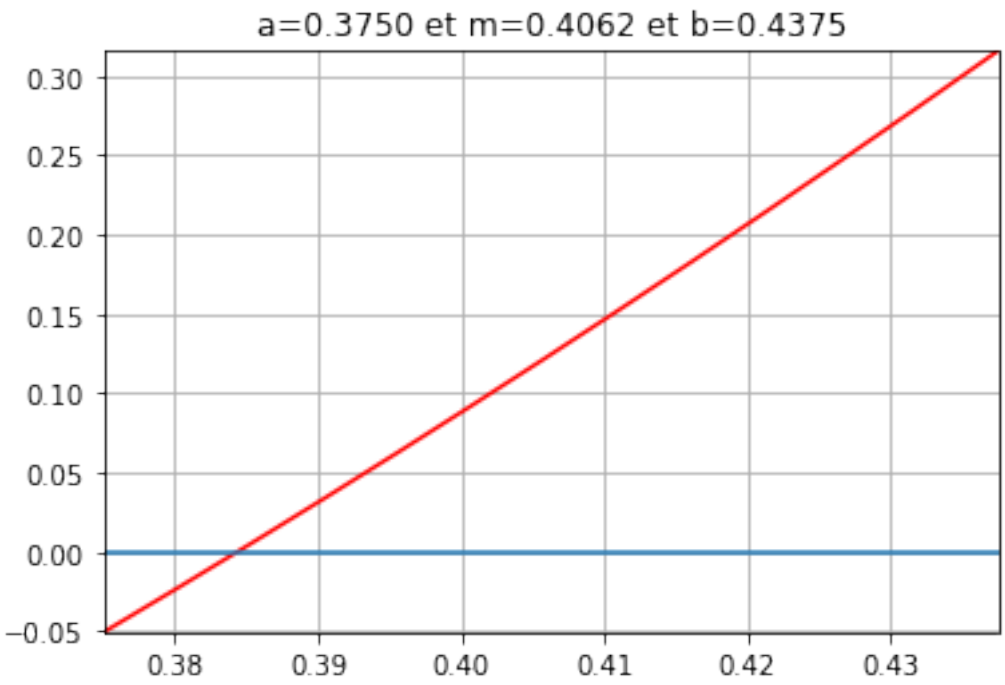
Etape	m	Choix ?	a	b
initialisation	None	None	0	0.5
1	0.25	droite	0.25	0.5
2	0.375	droite	0.375	0.5
3	0.4375	gauche	0.375	0.4375
4	0.40625	gauche	0.375	0.40625
5	0.390625	gauche	0.375	0.390625
6	0.3828125	droite	0.3828125	0.390625

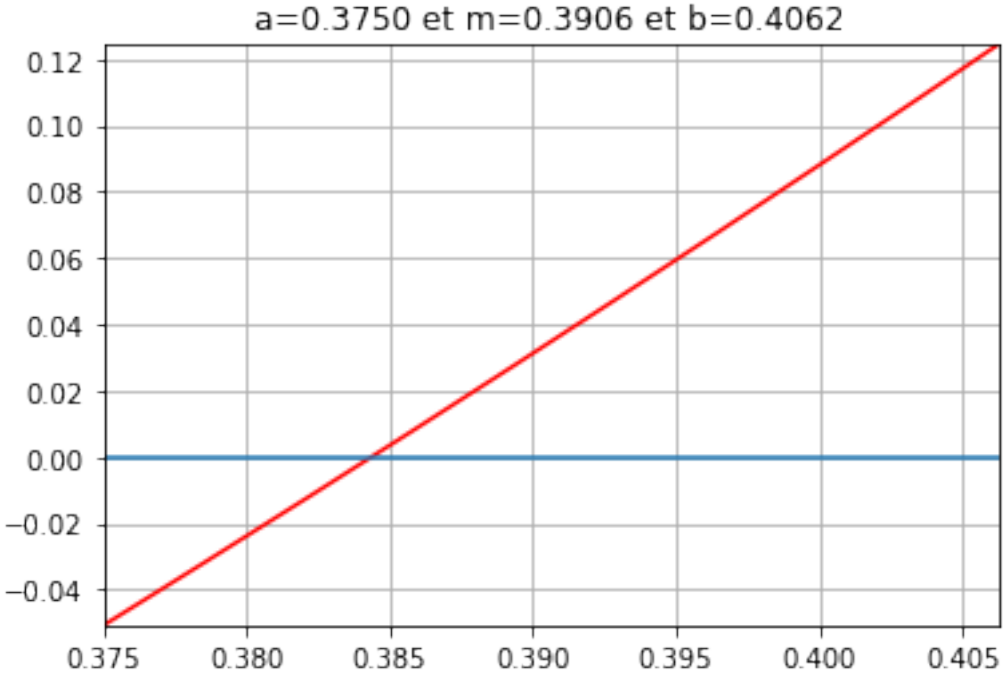
```
In [23]: dicho(lambda x : 2 * x ** 3 + 6 * x ** 2 - 1, 0, 0.5, 0.01, 0)
```

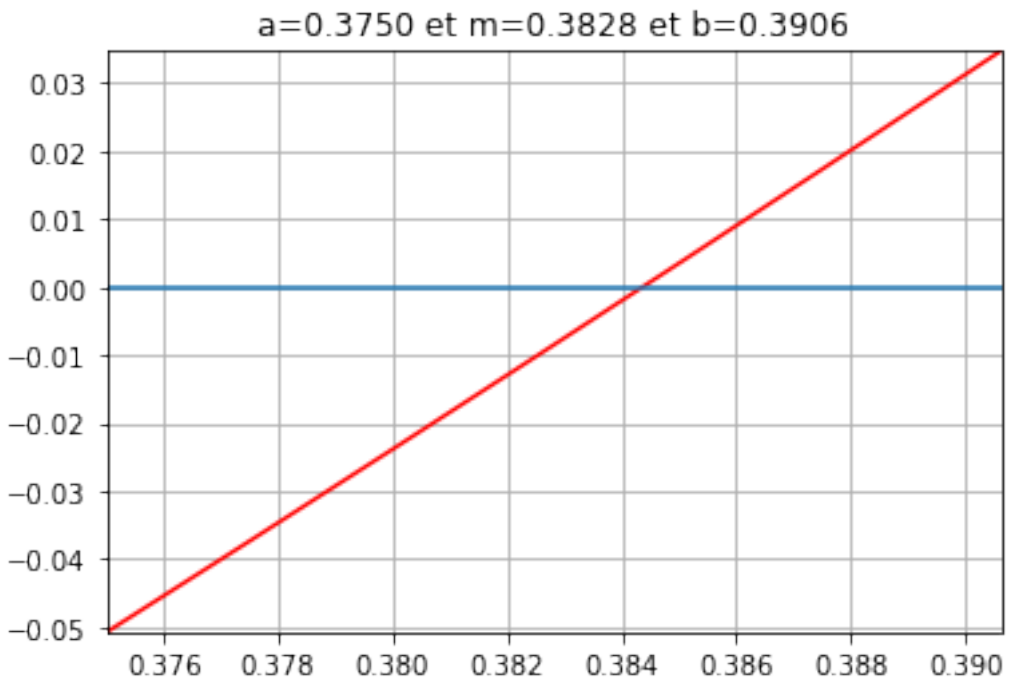



a=0.2500 et m=0.3750 et b=0.5000
0.6
0.4
0.2
0.0
−0.2
−0.4
0.25
0.30
0.35
0.40
0.45
0.50


a=0.3750 et m=0.4375 et b=0.5000
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.38
0.40
0.42
0.44
0.46
0.48
0.50


a=0.3750 et m=0.4062 et b=0.4375
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
0.38
0.39
0.40
0.41
0.42
0.43


a=0.3750 et m=0.3906 et b=0.4062
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
0.375
0.380
0.385
0.390
0.395
0.400
0.405



Out[23]:

$$(0.3828125,\quad 0.390625,\quad 6)$$