ananu2020_TD2

May 14, 2020

1 Exos Analyse mai 2020 - TD2 Recherche de zéros

version du 13 mai 2020

Outils implémentés à l'aide de `sympy`

1.1 Exercice 2 : Nombre d'entiers

Combien y a-t-il de nombres impairs entre 179 et 1243 ? De nombres pairs ?

[1]:
```
# Pour tracer les courbes
import numpy as np
import matplotlib.pyplot as plt
# Pour les calculs formels
from sympy import *
#  pour automatiquement afficher les résultats avec jsmath
init_printing()
```

[2]:
```
a,b=179,1243
print("nb d'entiers         entre %s et %s : %s"%(a,b,len([i for i in␣
 ↪range(a,b+1)])))
print("attention : %s-%s=%s"%(b,a,b-a))
print("nb d'entiers impairs entre %s et %s : %s"%(a,b,len([i for i in␣
 ↪range(a,b+1) if i%2==1])))
print("nb d'entiers   pairs entre %s et %s : %s"%(a,b,len([i for i in␣
 ↪range(a,b+1) if i%2==0])))
```

```
nb d'entiers         entre 179 et 1243 : 1065
attention : 1243-179=1064
nb d'entiers impairs entre 179 et 1243 : 533
nb d'entiers   pairs entre 179 et 1243 : 532
```

1.2 Exercice 3: Sommes des termes d'une suite arithmétique

Calculer S0 = 1 + 3/2 + 2 + ... + 29/2

Calculer S1 = 1/3 + 1 + 5/3 + ... + 19/3 + 7

Calculer la somme des entiers multiples de 5 qui sont plus grands que 99 et plus petits que 999.

Une suite arithmétique est telle que $u_0 = 2$ et $\sum_{i=3}^{i=n} u_i = 6456$. Sa raison est 5, que vaut n ?

Une suite arithmétique est telle que u0 = 2 et ??? ui = 1457. Que vaut sa raison ?

[3]:
```
def resumeliste(l) :
    if len(l)<2 :
        return "%s (%s terme)"%(l,len(l))
    elif len(l) < 11 :
        return "%s (%s termes)"%(l,len(l))
    else :
        return "[%s, %s, %s, %s, ... , %s, %s, %s, %s] (%s␣
 ↪termes)"%(l[0],l[1],l[2],l[3],l[-4],l[-3],l[-2],l[-1],len(l))
print(resumeliste([]))
print(resumeliste([1]))
print(resumeliste([2,3,4]))
print(resumeliste([i for i in range(1000,10000)]))
```

```
[] (0 terme)
[1] (1 terme)
[2, 3, 4] (3 termes)
[1000, 1001, 1002, 1003, … , 9996, 9997, 9998, 9999] (9000 termes)
```

[4]:
```
l=[Rational(i,2) for i in range(2,30)]
#print(' + '.join([str(t) for t in l]))
print(resumeliste(l))
print(sum(l))
```

```
[1, 3/2, 2, 5/2, … , 13, 27/2, 14, 29/2] (28 termes)
217
```

[5]:
```
l=[Rational(1+2*i,3) for i in range(0,11)]
print(resumeliste(l))
print(sum(l))
```

```
[1/3, 1, 5/3, 7/3, … , 5, 17/3, 19/3, 7] (11 termes)
121/3
```

[6]:
```
l=[5*i for i in range(20,200)]
print(resumeliste(l))
print(sum(l))
```

```
[100, 105, 110, 115, … , 980, 985, 990, 995] (180 termes)
98550
```

[7]:
```
u = lambda i : 2+5*i
n,s=3,u(3)
while s< 6456 :
    n+=1
    s+=u(n)
print(n,s)
```

```
50 6456
```

[8]:
```
l=[2+5*i for i in range(3,51)]
print(resumeliste(l))
print(sum(l))
```

```
[17, 22, 27, 32, … , 237, 242, 247, 252] (48 termes)
6456
```

[9]:
```
r=Symbol('r')
l = [2+r*i for i in range(0,31)]
print(resumeliste(l))
print(sum(l))
```

```
[2, r + 2, 2*r + 2, 3*r + 2, … , 27*r + 2, 28*r + 2, 29*r + 2, 30*r + 2] (31
termes)
465*r + 62
```

[10]:
```
solve(465*r+62-1457,r)
```

[10]:
[3]

[11]:
```
r = 3
l = [2+r*i for i in range(0,31)]
print(resumeliste(l))
print(sum(l))
```

```
[2, 5, 8, 11, … , 83, 86, 89, 92] (31 termes)
1457
```

1.3 Exercice 5: Sommes des termes d'une suite géométrique

- Calculer S1 = 18+54+162+…+39366.
- Calculer S2 = $\sqrt{2}-2+2\sqrt{2}\ldots-64+64\sqrt{2}-128$.
- Calculer S3=$2^{7+2}8+2^{9+\ldots+2}21$

[12]:
```
i=Symbol('i')
solve(6*3**i-39366,i)
```

[12]:
[8]

[13]:
```
l = [6*3**i for i in range(0,9)]
print(resumeliste(l))
print(sum(l))
```

```
[6, 18, 54, 162, 486, 1458, 4374, 13122, 39366] (9 termes)
59046
```

[14]:
```
rac2=-sqrt(2)
l = [-rac2**i for i in range(1,15)]
print(resumeliste(l))
print(sum(l))
```

```
[sqrt(2), -2, 2*sqrt(2), -4, … , 32*sqrt(2), -64, 64*sqrt(2), -128] (14
termes)
-254 + 127*sqrt(2)
```

[15]:
```
l = [2**i for i in range(7,22)]
print(resumeliste(l))
print(sum(l))
```

```
[128, 256, 512, 1024, … , 262144, 524288, 1048576, 2097152] (15 termes)
4194176
```

[16]:
```
a,b=2**7-1,2**22-1
print("%s - %s = %s"%(b,a,b-a))
```

```
4194303 - 127 = 4194176
```

1.4 Exercice 7 : Classique 1 - la légende de l'échiquier

Le roi de Perse Sargon voulut récompenser l'inventeur du jeu d'échecs. Celui-ci demanda alors au roi de déposer un grain de blé sur la première case, 2 grains sur la seconde, 4 grains sur la troisième et ainsi de suite en doublant à chaque fois le nombre de grains jusqu'à la 64e case.

- Combien de grains de blé devront être posés sur l'échiquier ?
- En admettant que 1024 grains de blé pèsent 100 grammes, calculer la masse de ces grains de blé.
- En supposant que la production française de blé est de 30 millions de tonnes par an, combien d'années de production faudrait-il pour remplir l'échiquier ?
- Sachant que le roi pose un grain à la seconde, et qu'il commença lors du big-bang (il y a environ 13,7 milliards d'années), a-t-il aujourd'hui terminé ?

[17]:
```
l = [2**i for i in range(0,64)]
print(resumeliste(l))
print(sum(l))
nbg=2**64-1
print("%s   %6g"%(nbg,nbg))
```

```
[1, 2, 4, 8, … , 1152921504606846976, 2305843009213693952,
4611686018427387904, 9223372036854775808] (64 termes)
18446744073709551615
18446744073709551615   1.84467e+19
```

[18]:
```
print("nombre de grains          : %6g"%nbg)
hg=nbg/1024
print("en hectogrammes           : %6g"%hg)
mt=hg/10/1000/10**6
print("en millions de tonnes     : %6g"%mt)
ap=mt/30
print("en années de production   : %6g"%ap)
auens=13.7e9*364.25*24*60**2
print("âge univers en secondes   : %6g"%auens)
print("ng grains / age univers en secondes : %6g"%(nbg/auens))
```

```
nombre de grains          : 1.84467e+19
en hectogrammes           : 1.80144e+16
en millions de tonnes     : 1.80144e+06
en années de production   :  60048
âge univers en secondes   : 4.31155e+17
ng grains / age univers en secondes : 42.7844
```

1.5 Exercice 9: Capital acquis

Dans le but de constituer un capital pour son petit-fils, la grand-mère de Félix décide de verser sur un compte rémunéré à intérêts composés au taux de 2% la somme de 1 000€ par an. Elle effectue son premier versement le jour de la naissance de Félix.

- Quel âge aura Félix quand son capital dépassera 5000€?
- Calculer le capital dont disposera Félix le jour de son 18e anniversaire.

Première version récursive, deuxième avec *memoization*

[19]:
```
def capital(n,taux=1.02) :
    if n==0 :
        return 1000
    else :
        return 1000+taux*capital(n-1)
for n in [0,1,2,3,4,5,18] :
    print("%2s %10.2f"%(n,capital(n)))
```

```
 0    1000.00
 1    2020.00
 2    3060.40
 3    4121.61
 4    5204.04
 5    6308.12
18   22840.56
```

[20]:
```
memo={0:1000}
def capital(n,taux=1.02) :
    if n in memo :
```

```
        return memo[n]
    else :
        reponse = 1000+taux*capital(n-1)
        memo[n] = reponse
        return reponse
print(capital(18))
print(memo)
```

```
22840.55862638992
{0: 1000, 1: 2020.0, 2: 3060.4, 3: 4121.608, 4: 5204.0401600000005, 5:
6308.120963200001, 6: 7434.283382464001, 7: 8582.96905011328, 8:
9754.628431115547, 9: 10949.720999737858, 10: 12168.715419732614, 11:
13412.089728127266, 12: 14680.331522689812, 13: 15973.938153143608, 14:
17293.41691620648, 15: 18639.28525453061, 16: 20012.070959621222, 17:
21412.312378813647, 18: 22840.55862638992}
```

Idem mais avec un taux de 1% au lieu de 2%…

[21]:
```
memo={0:1000}
def capital(n,taux=1.01) :
    if n in memo :
        return memo[n]
    else :
        reponse = 1000+taux*capital(n-1)
        memo[n] = reponse
        return reponse
print(capital(18))
print(memo)
```

```
20810.895044353154
{0: 1000, 1: 2010.0, 2: 3030.1, 3: 4060.401, 4: 5101.00501, 5: 6152.0150601, 6:
7213.535210700999, 7: 8285.670562808009, 8: 9368.52726843609, 9:
10462.21254112045, 10: 11566.834666531655, 11: 12682.50301319697, 12:
13809.32804332894, 13: 14947.42132376223, 14: 16096.895536999853, 15:
17257.864492369852, 16: 18430.44313729355, 17: 19614.747568666487, 18:
20810.895044353154}
```

Idem avec un taux de 10% :

[22]:
```
memo={0:1000}
def capital(n,taux=1.10) :
    if n in memo :
        return memo[n]
    else :
        reponse = 1000+taux*capital(n-1)
        memo[n] = reponse
        return reponse
print(capital(18))
print(memo)
```

```
51159.09044841459
{0: 1000, 1: 2100.0, 2: 3310.0, 3: 4641.0, 4: 6105.1, 5: 7715.610000000001, 6:
9487.171000000002, 7: 11435.888100000004, 8: 13579.476910000005, 9:
15937.424601000006, 10: 18531.167061100008, 11: 21384.28376721001, 12:
24522.712143931014, 13: 27974.983358324116, 14: 31772.48169415653, 15:
35949.72986357218, 16: 40544.7028499294, 17: 45599.17313492235, 18:
51159.09044841459}
```

1.6 Exercce 14

Résolution par dichotomie de $x^{3+x}2+2x-6=0$

On remarque que pour un polynôme de degré au plus 4 une résoltion par radicaux est en théorie accessible… mais pltôt indigeste :

[23]:
```
f = lambda x : x**3+x**2+2*x-6
x=Symbol('x')
solve(f(x),x)
```

[23]:
$$\left[-\frac{1}{3}+\left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{89}{27}+\frac{\sqrt{894}}{9}}-\frac{5}{9\left(-\frac{1}{2}-\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{89}{27}+\frac{\sqrt{894}}{9}}},\ -\frac{1}{3}-\frac{5}{9\left(-\frac{1}{2}+\frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{89}{27}+\frac{\sqrt{894}}{9}}}+\left(-\frac{1}{2}+\sqrt{}\right.$$

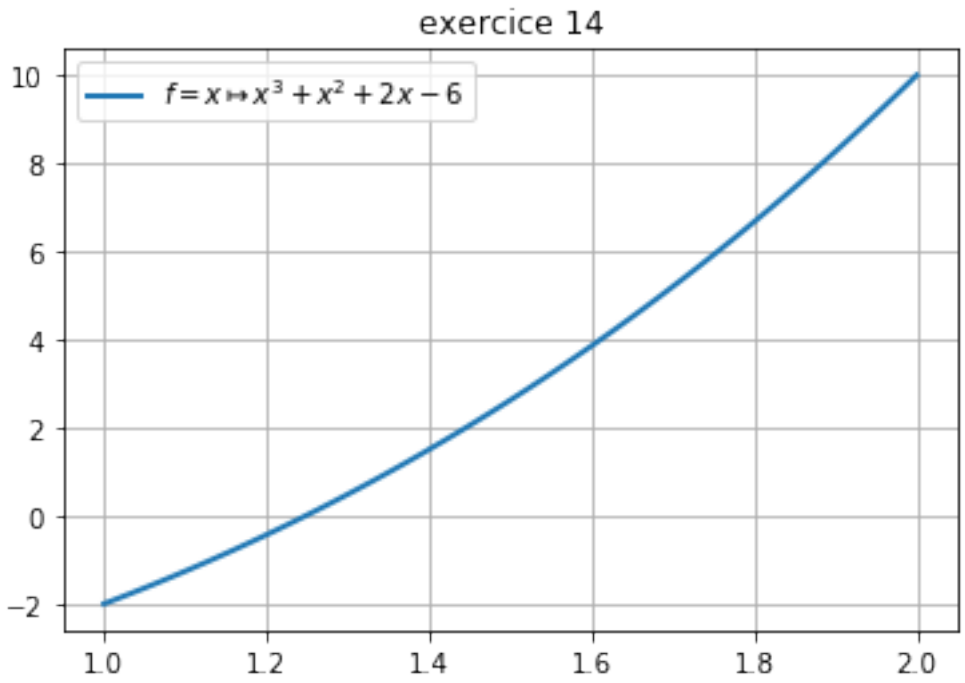
[24]:
```
[N(_) for _ in solve(f(x),x)]
```

[24]:
$$[-1.12414891114227 - 1.88224188163544i,\ -1.12414891114227 + 1.88224188163544i,\ 1.24829782228455]$$

Une étude de la fonction $f = x \mapsto x^3 + x^2 + 2x - 6$ permet de prouver que

- il y a une seule solution réelle ,
- que cette solution (que nous noterons α) est comprise entre 1 et 2
- que sur cet intervalle la fontion est croissante.

[25]:
```
x=Symbol('x')
t = np.linspace(1,2, 120)
plt.plot(t, f(t), linewidth=2,label="$f = x \mapsto %s$"%latex(f(x)))
####
plt.title('exercice 14')
plt.grid()
#plt.axis('equal')
plt.legend()
####
plt.savefig('courbesexo14a.pdf')
plt.show()
```



[26]:
```
def dicho(f,a,b,nbpas,format="%+.6f",formatN="%9s",formatI="%9i") :
    '''Donne un encadrement pour une solution
    de f(x) == 0
    avec f fonction continue et f(a) et f(b) de signes distincts'''
    i=0
    print(((formatN+"  ")*6)%('i','a','b','c=(a+b)/2','f(c)','decision'))
    if f(a)<0 : # signe f de a
        sgnfdea = -1
    else :
        sgnfdea = +1
    # donc sgnfdea*f(a)>0
    for i in range(nbpas) :
        c=(a+b)/2
        fdec=f(c)
        if sgnfdea * fdec > 0 : # f(c) et f(a) de meme signe
            decision = 'a=c'
            print((formatI+"  "+format+"  "+format+"  "+format+"  "+format+"
↪ %s")%(i,a,b,c,fdec,decision))
            a=c
        else :
            decision = 'b=c'
            print((formatI+"  "+format+"  "+format+"  "+format+"  "+format+"
↪ %s")%(i,a,b,c,fdec,decision))
            b=c
```

```
    return(a+b)/2
####
dicho(lambda x : x**3+x**2+2*x-6,1,2,20)
```

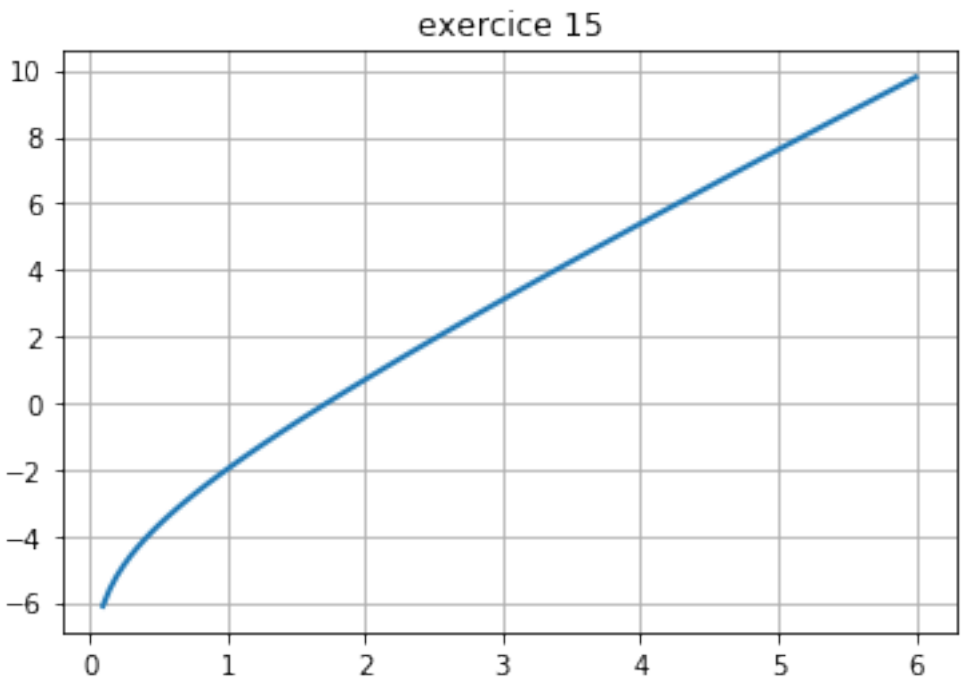
i	a	b	c=(a+b)/2	f(c)	decision
0	+1.000000	+2.000000	+1.500000	+2.625000	b=c
1	+1.000000	+1.500000	+1.250000	+0.015625	b=c
2	+1.000000	+1.250000	+1.125000	-1.060547	a=c
3	+1.125000	+1.250000	+1.187500	-0.540283	a=c
4	+1.187500	+1.250000	+1.218750	-0.266876	a=c
5	+1.218750	+1.250000	+1.234375	-0.126774	a=c
6	+1.234375	+1.250000	+1.242188	-0.055863	a=c
7	+1.242188	+1.250000	+1.246094	-0.020191	a=c
8	+1.246094	+1.250000	+1.248047	-0.002301	a=c
9	+1.248047	+1.250000	+1.249023	+0.006657	b=c
10	+1.248047	+1.249023	+1.248535	+0.002177	b=c
11	+1.248047	+1.248535	+1.248291	-0.000062	a=c
12	+1.248291	+1.248535	+1.248413	+0.001057	b=c
13	+1.248291	+1.248413	+1.248352	+0.000497	b=c
14	+1.248291	+1.248352	+1.248322	+0.000217	b=c
15	+1.248291	+1.248322	+1.248306	+0.000078	b=c
16	+1.248291	+1.248306	+1.248299	+0.000008	b=c
17	+1.248291	+1.248299	+1.248295	-0.000027	a=c
18	+1.248295	+1.248299	+1.248297	-0.000010	a=c
19	+1.248297	+1.248299	+1.248298	-0.000001	a=c

[26]: 1.248298168182373

1.7 Exercice 15 : résolution approchée de ln(x)+2x-4=0

Résolution approchée de ln(x)+2*x-4=0

[27]:
```
x=Symbol('x')
#f = lambda x : np.exp(-x)
t = np.linspace(0.1,6, 120)
plt.plot(t, np.log(t)+2*t-4, linewidth=2)
####
plt.title('exercice 15')
plt.grid()
#plt.axis('equal')
#plt.legend()
####
plt.savefig('courbeexo16a.pdf')
plt.show()
```



[28]:
```
dicho(lambda x : log(x)+2*x-4,1.0,2.0,20)
```

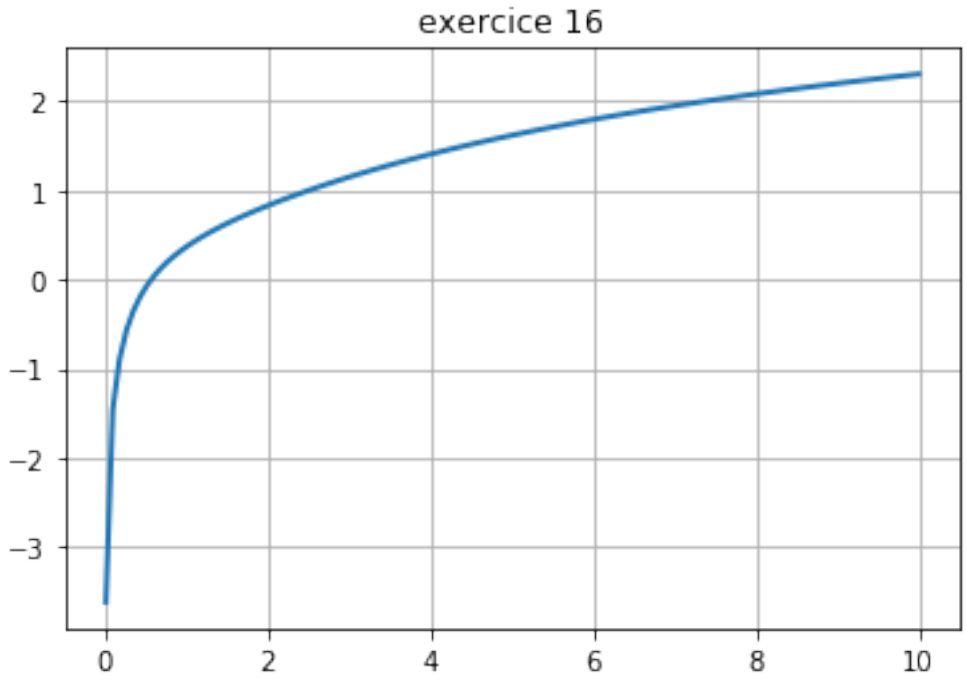
i	a	b	c=(a+b)/2	f(c)	decision
0	+1.000000	+2.000000	+1.500000	-0.594535	a=c
1	+1.500000	+2.000000	+1.750000	+0.059616	b=c
2	+1.500000	+1.750000	+1.625000	-0.264492	a=c
3	+1.625000	+1.750000	+1.687500	-0.101752	a=c
4	+1.687500	+1.750000	+1.718750	-0.020903	a=c
5	+1.718750	+1.750000	+1.734375	+0.019397	b=c
6	+1.718750	+1.734375	+1.726562	-0.000743	a=c
7	+1.726562	+1.734375	+1.730469	+0.009330	b=c
8	+1.726562	+1.730469	+1.728516	+0.004294	b=c
9	+1.726562	+1.728516	+1.727539	+0.001776	b=c
10	+1.726562	+1.727539	+1.727051	+0.000517	b=c
11	+1.726562	+1.727051	+1.726807	-0.000113	a=c
12	+1.726807	+1.727051	+1.726929	+0.000202	b=c
13	+1.726807	+1.726929	+1.726868	+0.000045	b=c
14	+1.726807	+1.726868	+1.726837	-0.000034	a=c
15	+1.726837	+1.726868	+1.726852	+0.000005	b=c
16	+1.726837	+1.726852	+1.726845	-0.000015	a=c
17	+1.726845	+1.726852	+1.726849	-0.000005	a=c
18	+1.726849	+1.726852	+1.726851	+0.000000	b=c
19	+1.726849	+1.726851	+1.726850	-0.000002	a=c

[28]:
1.7268500328063965

1.8 Exercice 16

Résolution de exp(-x)+ln(x)=0 : x>0

[29]:
```
x=Symbol('x')
#f = lambda x : np.exp(-x)
t = np.linspace(0.01,10, 120)
plt.plot(t, np.exp(-t)+np.log(t), linewidth=2)
####
plt.title('exercice 16')
plt.grid()
#plt.axis('equal')
#plt.legend()
####
plt.savefig('courbeexo16a.pdf')
plt.show()
```



[30]:
```
dicho(lambda x : exp(-x)+log(x),0.1,2.0,20)
```

i	a	b	c=(a+b)/2	f(c)	decision
0	+0.100000	+2.000000	+1.050000	+0.398728	b=c
1	+0.100000	+1.050000	+0.575000	+0.009320	b=c
2	+0.100000	+0.575000	+0.337500	-0.372638	a=c
3	+0.337500	+0.575000	+0.456250	-0.151059	a=c
4	+0.456250	+0.575000	+0.515625	-0.065248	a=c
5	+0.515625	+0.575000	+0.545313	-0.026736	a=c

```
 6  +0.545313  +0.575000  +0.560156  -0.008420     a=c
 7  +0.560156  +0.575000  +0.567578  +0.000520     b=c
 8  +0.560156  +0.567578  +0.563867  -0.003932     a=c
 9  +0.563867  +0.567578  +0.565723  -0.001702     a=c
10  +0.565723  +0.567578  +0.566650  -0.000590     a=c
11  +0.566650  +0.567578  +0.567114  -0.000035     a=c
12  +0.567114  +0.567578  +0.567346  +0.000243     b=c
13  +0.567114  +0.567346  +0.567230  +0.000104     b=c
14  +0.567114  +0.567230  +0.567172  +0.000035     b=c
15  +0.567114  +0.567172  +0.567143  -0.000000     a=c
16  +0.567143  +0.567172  +0.567158  +0.000017     b=c
17  +0.567143  +0.567158  +0.567150  +0.000009     b=c
18  +0.567143  +0.567150  +0.567147  +0.000004     b=c
19  +0.567143  +0.567147  +0.567145  +0.000002     b=c
```

[30]:
```
0.5671441555023193
```

[31]:
```
def newton(f,df,u0,nbpas,format="%+.15g",formatN="%17s",formatI="%17i") :
    '''Propose la valeur approchée d'une éventuelle solution
    de f(x) == 0 pour x proche de u0
    avec f et sa dérivée df toutes deux connues'''
    i=0
    u=u0

    print(((formatN+"  ")*5)%('i','u','f(u)','df(u)','prochain u'))
    for i in range(nbpas) :
        usuivant = u-f(u)/df(u)
        #print(i,u,f(u),df(u),usuivant)
        print((formatI+"  "+format+"  "+format+"  "+format+"  "+format)%(i,u,f(u),df(u),usuivant))
        u=usuivant
    return usuivant
######
x = Symbol('x')
u = newton(lambda x : exp(-x)+log(x), lambda x : -exp(-x)+1/x,0.4,10)
print(u)
fdeu=exp(-u)+log(u)
print(fdeu)
print(log(abs(fdeu))/log(2.0))
```

```
                i                   u                f(u)                df(u)
prochain u
                0  +0.4  -0.245970685838516  +1.82967995396436
+0.534433721758591
                1  +0.534433721758591  -0.0405465208155991  +1.28513837637463
+0.565984036931639
                2  +0.565984036931639  -0.00138827000553066  +1.19903314838377
+0.567141861473356
```

```
                3  +0.567141861473356  -1.70912422636782e-06  +1.19608317603973
+0.567143290407614
                4  +0.567143290407614  -2.59503529775884e-12  +1.19607954394763
+0.567143290409784
                5  +0.567143290409784  -1.11022302462516e-16  +1.19607954394211
+0.567143290409784
                6  +0.567143290409784  +1.11022302462516e-16  +1.19607954394211
+0.567143290409784
                7  +0.567143290409784  -1.11022302462516e-16  +1.19607954394211
+0.567143290409784
                8  +0.567143290409784  +1.11022302462516e-16  +1.19607954394211
+0.567143290409784
                9  +0.567143290409784  -1.11022302462516e-16  +1.19607954394211
+0.567143290409784
0.567143290409784
1.11022302462516e-16
-53.0000000000000
```

[32]:
```
# Pour tracer les courbes
import numpy as np
import matplotlib.pyplot as plt
# Pour les calculs formels
from sympy import *
#  pour automatiquement afficher les résultats avec jsmath
init_printing()
```

[ ]:

[ ]:

[ ]:

[ ]: