TP n° 11 : Calcul approché d'intégrales - Correction

1. $x_0 = a$ et $x_n = b$.
   Pour $i \in \{0\,;n-1\}$, l'intervalle $[x_i\,;x_{i+1}]$ a pour longueur $x_{i+1} - x_i = \dfrac{b-a}{n}$.
2. Sur chaque intervalle $[x_i\,;x_{i+1}]$, on approxime l'aire sous la courbe par celle du rectangle dont la hauteur correspond à la valeur de f à *gauche* de cet intervalle, *i.e.* par $f(x_i)$.
3. L'aire d'un rectangle est donnée par `base*hauteur` donc ici, elle vaut $(x_{i+1} - x_i)f(x_i) = \dfrac{b-a}{n}f(x_i)$.
4. La somme des aires de tous les rectangles est $\displaystyle\sum_{i=0}^{n-1} \frac{b-a}{n} f(x_i) = \frac{b-a}{n} \sum_{i=0}^{n-1} f(x_i)$.
5.
```
def RectanglesGauche(f, a, b, n):
    I = 0
    x = a
    h = (b - a)/n    #Largeur de chaque rectangle
    while x < b:
        I = I + f(x)*h    #On ajoute l'aire du rectangle dont le point
                          #en bas à gauche a pour abscisse x
        x = x + h    #On avance sur les abscisses pour aller au
                     #rectangle suivant
    return I
```
6.
```
>>> G10 = RectanglesGauche(sin,0,pi,10)
>>> G10
1.9835235375094544
>>> G100 = RectanglesGauche(sin,0,pi,100)
>>> G100
1.9998355038874456
```
7. $$I = \int_0^\pi \sin(x)\,\mathrm{d}x = [-\cos x]_0^\pi = -\cos(\pi) + \cos(0) = 2.$$
   D'où :
```
>>> abs(2-G10)
0.016476462490545574
>>> abs(2-G100)
0.0001644961125544242
```
8.
```
def g(x):
    return exp(-x**2)
```
9.
```
>>> RectanglesGauche(g,0,10,100)
0.9362269254527581
>>> RectanglesGauche(g,0,10,1000)
0.891226925452757
>>> J = sqrt(pi)/2
```

```
>>> abs(J-RectanglesGauche(g,0,10,100))
0.050000000000000155
>>> abs(J-RectanglesGauche(g,0,10,1000))
0.004999999999999005
```

10.

```
def RectanglesDroite(f, a, b, n):
    I = 0
    x = a
    h = (b - a)/n       #Largeur de chaque rectangle
    while x <= b:
        x = x + h
        I = I + f(x)*h      #On ajoute l'aire du rectangle dont le
                            #point en bas à gauche a pour abscisse x
    return I
```

11.

```
>>> D10 = RectanglesDroite(sin,0,pi,10)
>>> D10
1.8864429855731812
>>> D100 = RectanglesDroite(sin,0,pi,100)
>>> D100
1.9988487057878104
```

12.

```
>>> abs(2-D10)
0.11355701442681876
>>> abs(2-D100)
0.0011512942121896241
```

13.

```
>>> RectanglesDroite(g,0,10,100)
0.836226925452758
>>> RectanglesDroite(g,0,10,1000)
0.881226925452757
>>> J = sqrt(pi)/2
>>> abs(J-RectanglesDroite(g,0,10,100))
0.04999999999999993
>>> abs(J-RectanglesDroite(g,0,10,1000))
0.005000000000000893
```

14. Le nom de la méthode vient du fait que sur chaque intervalle $[x_i\,;x_{i+1}]$, on approxime l'intégrale de la fonction par l'aire d'un trapèze.

15. La hauteur est $h = x_{i+1} - x_i = \dfrac{b-a}{n}$, les deux bases ont pour longueur $b = f(x_i)$ et $B = f(x_{i+1})$ donc l'aire dudit trapèze est
$$h\frac{b+B}{2} = \frac{b-a}{n} \times \frac{f(x_i)+f(x_{i+1})}{2}.$$

16.

```
def Trapezes(f, a, b, n):
    I = 0
    x = a
    h = (b - a)/n     #Hauteur de chaque trapèze
    while x < b:
```

```
        I = I + h*(f(x)+f(x+h))/2     #On ajoute l'aire du trapèze
                                      #dont le point en bas à
                                      #gauche a pour abscisse x
        x = x + h
    return I
```

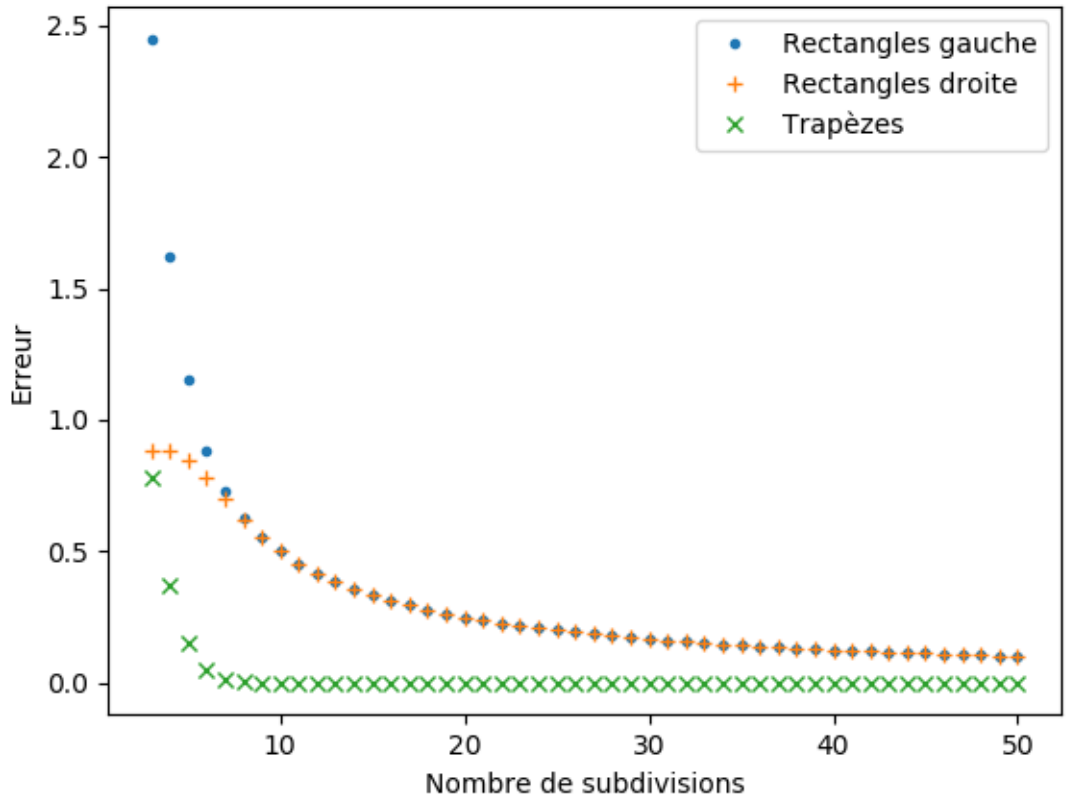
17.

```
>>> T10 = Trapezes(sin,0,pi,10)
>>> abs(2-T10)
0.01647646249054535
>>> T100 = Trapezes(sin,0,pi,100)
>>> abs(2-T100)
0.0006578951623725793
```

```
>>> J = sqrt(pi)/2
>>> abs(J-Trapezes(g,0,10,100))
1.1102230246251565e-16
```

Ces approximations semblent meilleurs que celles obtenues avec les méthodes des rectangles.

Remarque. On a tracé ci-dessous l'erreur commise par chacun des trois méthodes pour un nombre de rectangles compris entre 3 et 50. Il semble que les deux méthodes des rectangles ont des niveaux de précision comparable et que celle des trapèzes soit meilleure.



18.

```
>>> RectAdapt(sin, 0, pi, 10**(-3))
(1.9997429724458322, 80)
```

On remarque que la première valeur de ce couple est très proche de la valeur de l'intégrale correspondante.

19. Cette fonction renvoie un couple composé d'un flottant et d'un entier. Le premier est une valeur approchée de $\int_a^b f(t)\,\mathrm{d}t$ à `eps` près via la méthode des rectangles à gauche et le second est le nombre

de subdivisons utilisé.

20. Le produit $(b-a)M$ est l'aire du rectangle allant de a à b en abscisses et de 0 à M en ordonnées.
Le quotient $\dfrac{P}{n}$ est la proportion des points tirés au hasard qui se trouvent sous la courbe.
Comme $\displaystyle\int_a^b f(t)\,\mathrm{d}t$ est l'aire située sous la courbe, le quotient $\dfrac{1}{(b-a)M}\displaystyle\int_a^b f(t)\,\mathrm{d}t$ est la proportion de l'aire du rectangle située sous la courbe, elle doit donc être proche de la proportion de points se trouvant sous la courbe car ceux-ci sont tirés aléatoirement et de façon uniforme.

21.

```
from random import uniform
def MonteCarlo(f, a, b, M, n):
    P = 0
    for i in range(n):
        x = uniform(a, b)
        y = uniform(0, M)
        if y<=f(x):
            P = P+1
    return (b-a)*M*P/n
```

22. On prend $M = 1$ car $\forall x \in [0\,;\pi]$, $0 \leqslant \sin x \leqslant 1$. On obtient :

```
>>> MonteCarlo(sin,0,pi,1,100)
1.9163715186897738
>>> MonteCarlo(sin,0,pi,1,1000)
1.9666370011472105
>>> MonteCarlo(sin,0,pi,1,1000)
2.0326104468725963
>>> MonteCarlo(sin,0,pi,1,1000)
1.9980529276831085
```

On obtient bien des valeurs proches de 2, la valeur exacte de l'intégrale que l'on a approximée. De plus, on remarque que plusieurs appels identique de cette fonction peuvent fournir des résultats différents, cela est dû à l'aléatoire des tirages. On parle d'algorithme probabiliste (ou randomisé).