Quelques méthodes d'intégration numérique

Marc Lorenzi - 15 avril 2018

In [1]:
```
from math import *
import matplotlib.pyplot as plt
```

Le but de ce notebook est de comparer quelques méthodes d'intégration numérique. Nous nous intéresserons à trois métodes : les méthodes des rectangles et des trapèzes vues en cours, et la méthode de Gauss. Pas de théorie ici, juste des illustrations de la théorie ou des conjectures pour la méthode de Gauss.

1. La méthode des rectangles

Soit $f : [a, b] \to \mathbb{R}$. Pour tout $n \geq 1$, soit $S_n = \frac{b-a}{n} \sum_{k=0}^{n-1} f(a + k\frac{b-a}{n})$. Si f est continue alors $S_n \to \int_a^b f$ lorsque n tend vers l'infini. La raison essentielle à cela est que toute fonction continue peut être approchée de façon "convenable" par des fonctions en escalier. L'aire sous la courbe peut donc être approchée par des sommes d'aires de rectangles.

In [2]:
```
def rectangles(f, a, b, n):
    d = (b - a) / n
    s = 0
    for k in range(n):
        s += f(a + k * d)
    return s * d
```

In [3]:
```
rectangles(exp, 0, 1, 1000)
```

Out[3]:
```
1.7174228307349666
```

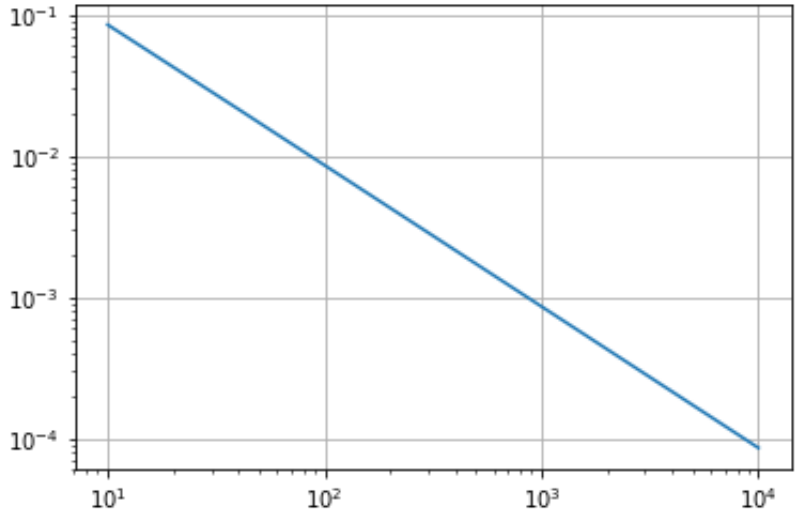
Si f est lipschitzienne, la théorie nous dit que l'erreur commise en approchant l'intégrale de f par S_n est un $O(\frac{1}{n})$. Un petit tracé de l'erreur avec des axes logarithmiques permet d'illustrer cela. En abscisse, le nombre de points, n. En ordonnée, l'erreur commise. On choisit d'intégrer l'exponentielle entre 0 et 1, mais rien ne vous empêche de tenter autre chose.

In [4]:
```
def erreur(methode, f, a, b, n, vraie_valeur):
    return abs(methode(f, a, b, n) - vraie_valeur)
```

In [5]:
```
erreur(rectangles, exp, 0, 1, 1000, exp(1) - 1)
```

Out[5]:
```
0.0008589977240784918
```

In [6]:
```
ks = [10 * k for k in range(1, 1001)]
srect = [erreur(rectangles, exp, 0, 1, k, exp(1) - 1) for k in ks]
plt.loglog(ks, srect)
plt.grid()
plt.show()
```



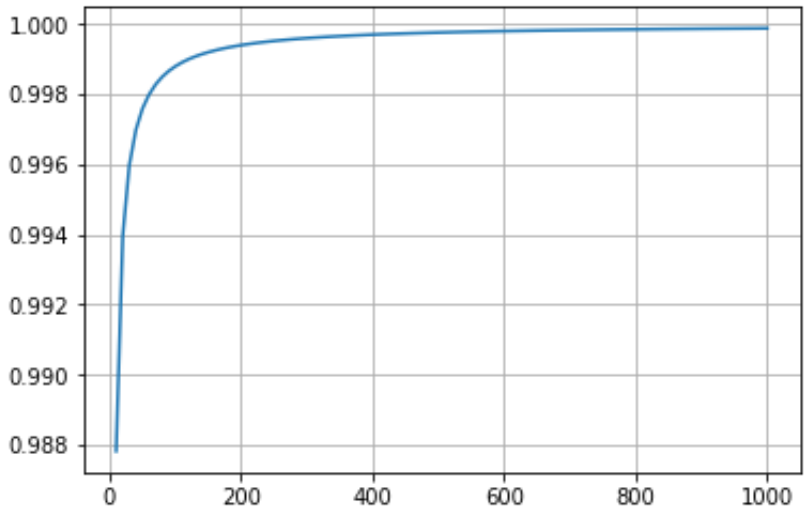
Retrouve-t-on bien cette fameuse erreur en $O(\frac{1}{n})$? Il semble que oui mais il y a mieux à tracer. Appelons E_n l'erreur commise lorsqu'on intègre avec n points. Imaginons que $E_n \sim \frac{c}{n^\alpha}$ où $c > 0$ et α est le nombre qui nous intéresse. Nous soupçonnons que $\alpha = 1$. On a $\frac{E_n}{E_{2n}} \sim 2^\alpha$ et donc tend vers 2^α lorsque n tend vers l'infini. Traçons donc $\lg \frac{E_n}{E_{2n}}$ où $\lg$ désigne le logarithme en base 2.

In [7]:
```
def estimation_erreur(methode, f, a, b, n, vraie_valeur):
    e1 = erreur(methode, f, a, b, n, vraie_valeur)
    e2 = erreur(methode, f, a, b, 2 * n, vraie_valeur)
    return log(e1 / e2) / log(2)
```

In [8]:
```
estimation_erreur(rectangles, exp, 0, 1, 1000, exp(1) - 1)
```

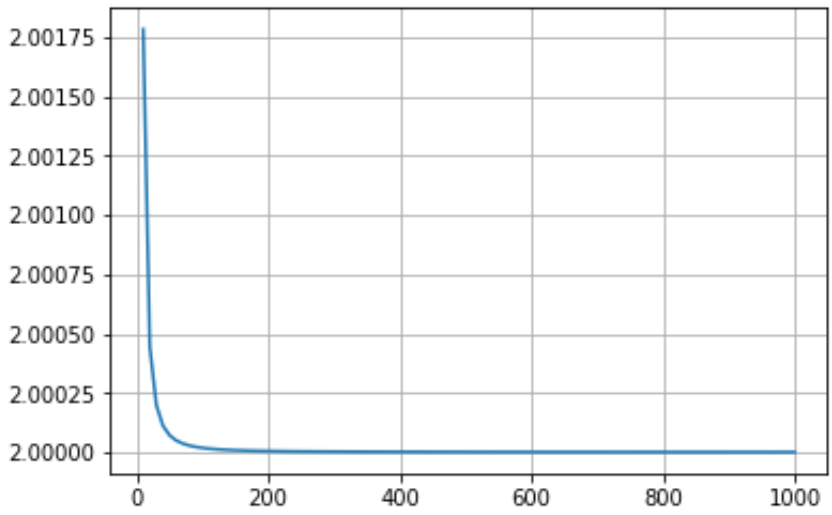
Out[8]:
```
0.9998797603856362
```

In [9]:
```
ks = [10 * k for k in range(1, 101)]
srect = [estimation_erreur(rectangles, exp, 0, 1, k, exp(1) - 1) for k
plt.plot(ks, srect)
plt.grid()
plt.show()
```



On retrouve bien $\alpha = 1$. Existe-t-il des fonctions pour lesquelles l'erreur commise est meilleure que prévu ? Eh bien oui. Prenons la fonction sinus sur $[0, \pi]$.

In [10]:
```
ks = [10 * k for k in range(1, 101)]
srect = [estimation_erreur(rectangles, sin, 0, pi, k, 2) for k in ks]
plt.plot(ks, srect)
plt.grid()
plt.show()
```



On voit que l'erreur est un $O(\frac{1}{n^2})$. Il y a des raisons subtiles à cela, que nous ne détaillerons pas ici. Mais existe-t-il une méthode qui donne à tous les coups (pour des fonctions suffisamment régulières) une erreur en $O(\frac{1}{n^2})$? Oui, par exemple la méthode des trapèzes.

2. La méthode des trapèzes

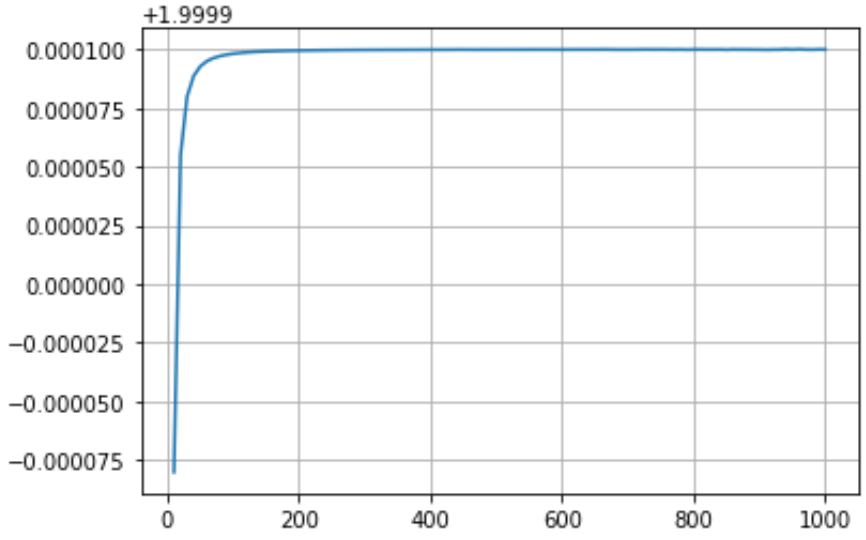
Toute fonction suffisamment régulière peut être approchée de façon "très convenable" par des fonctions affines par morceaux. Il s'ensuit que l'aire sous la courbe peut être approchée par des sommes d'aires de trapèzes. Le cours montre que ces sommes sont à peu près celles de la méthode des rectangles, à un terme près qui est $\frac{1}{2n}(b-a)(f(b)-f(a))$.

In [11]:
```
def trapezes(f, a, b, n):
    r = rectangles(f, a, b, n)
    return r + (b - a) * (f(b) - f(a)) / (2 * n)
```

In [12]:
```
trapezes(exp, 0, 1, 1000)
```

Out[12]:
```
1.7182819716491962
```

In [13]:
```
ks = [10 * k for k in range(1, 101)]
srect = [estimation_erreur(trapezes, exp, 0, 1, k, exp(1) - 1) for k i
plt.plot(ks, srect)
plt.grid()
plt.show()
```



Pour une fonction suffisamment régulière ($\mathcal{C}^2$ par exemple), l'erreur commise est cette fois-ci un $O(\frac{1}{n^2})$.

3. Méthode de Gauss

Peut-on faire mieux que les trapèzes ? Oui. Nous nous intéressons ici à une méthode due à Gauss. Pour être exacts, il s'agit de toute une famille de méthodes dont nous n'examinerons qu'un cas particulier. L'idée est d'approximer l'intégrale $\int_a^b f$ par des sommes $\sum_{k=0}^{n-1} a_k f(x_k)$ où les x_k sont choisis très judicieusement dans le segment $[a, b]$.

3.1 Un cas particulier

Soit $f : [a, b] \to \mathbb{R}$ une fonction "régulière". Posons $m = \frac{a+b}{2}$, $d = \frac{b-a}{2}$ et $r = \sqrt{\frac{3}{5}}$. Considérons $S = \frac{5}{9}f(m - rd) + \frac{8}{9}f(m) + \frac{5}{9}f(m + rd)$. On peut montrer que S est une bonne approximation (en un sens que nous ne préciserons pas ici) de $\int_a^b f$.

In [14]:
```
def gauss0(f, a, b):
    r = sqrt(3. / 5.)
    m = (a + b) / 2.
    d = (b - a) / 2.
    return d * (5 * f(m - d * r) + 8 * f(m) + 5 * f(m + d * r)) / 9
```

In [15]:
```
gauss0(exp, 0, 1)
```

Out[15]:
```
1.718281004372522
```

In [16]:
```
gauss0(exp, 0, 1) - exp(1) + 1
```

Out[16]:
```
-8.240865230213501e-07
```

Une erreur de 10^{-6}, alors qu'on a additionné 3 termes. C'est prometteur.

3.2 Cas général

La formule générale d'intégration de Gauss consiste à découper le segment $[a, b]$ en n segments de tailles égales, et à appliquer le cas particulier sur chacun des segments.

In [17]:
```
def gauss(f, a, b, n):
    s = 0
    d = (b - a) / n
    for k in range(n):
        s = s + gauss0(f, a + k * d, a + (k + 1) * d)
    return s
```

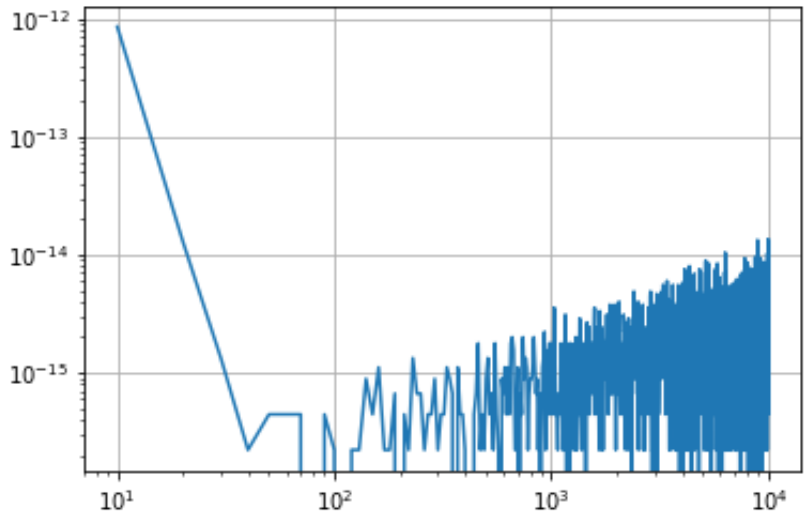
L'approximation obtenue est excellente.

In [18]:
```
erreur(gauss, exp, 0, 1, 100, exp(1) - 1)
```

Out[18]:
```
2.220446049250313e-16
```

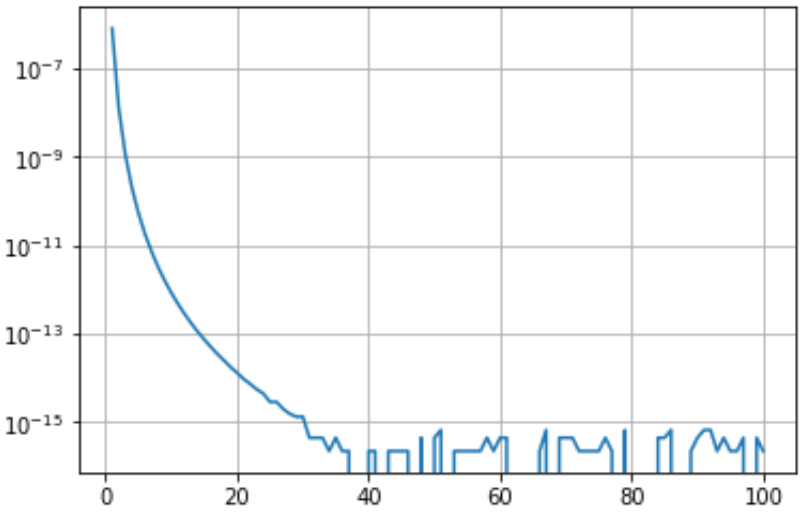
On obtient une erreur de 10^{-16} alors qu'avec la méthode des rectangles l'erreur était de $\frac{1}{100}$ et avec la métode des trapèzes de 10^{-4}. Comme précédemment, traçons l'erreur commise en fonction de n.

In [19]:
```
ks = [10 * k for k in range(1, 1001)]
sgauss = [erreur(gauss, exp, 0, 1, k, exp(1) - 1) for k in ks]
plt.loglog(ks,sgauss)
plt.grid()
plt.show()
```



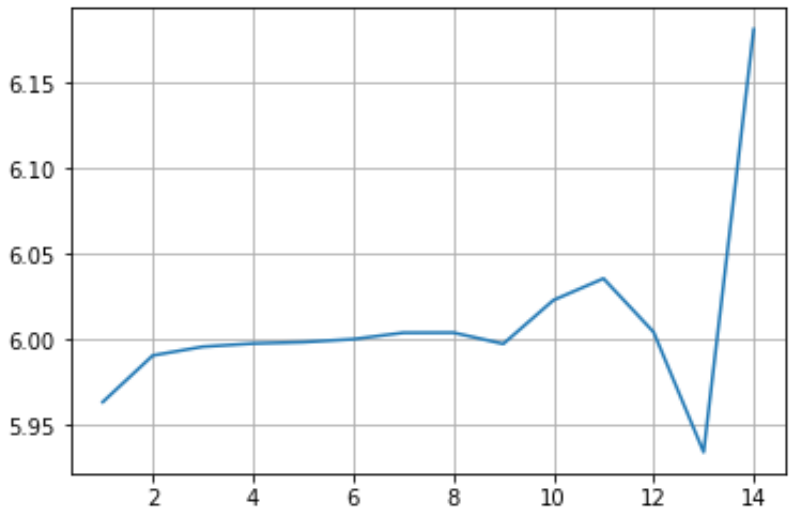
Ah mais c'est quoi ça ? Eh bien voilà ce qui arrive quand on utilise des outils trop puissants ! En fait, l'erreur commise par la méthode de Gauss devient rapidement inférieure aux erreurs d'arrondis. Ce que nous traçons pour les "grandes" valeurs de n ce n'est pas l'erreur commise, c'est du bruit. Au vu de la courbe ci-dessus, pour l'exemple de l'exponentielle entre 0 et 1, il est inutile de prendre une valeur de n supérieure à ... combien au juste ? Retraçons le graphe avec une échelle semi-logarithmique et des valeurs de k inférieures à 100.

In [20]:
```
ks = [k for k in range(1, 101)]
sgauss = [erreur(gauss, exp, 0, 1, k, exp(1) - 1) for k in ks]
plt.semilogy(ks,sgauss)
plt.grid()
plt.show()
```



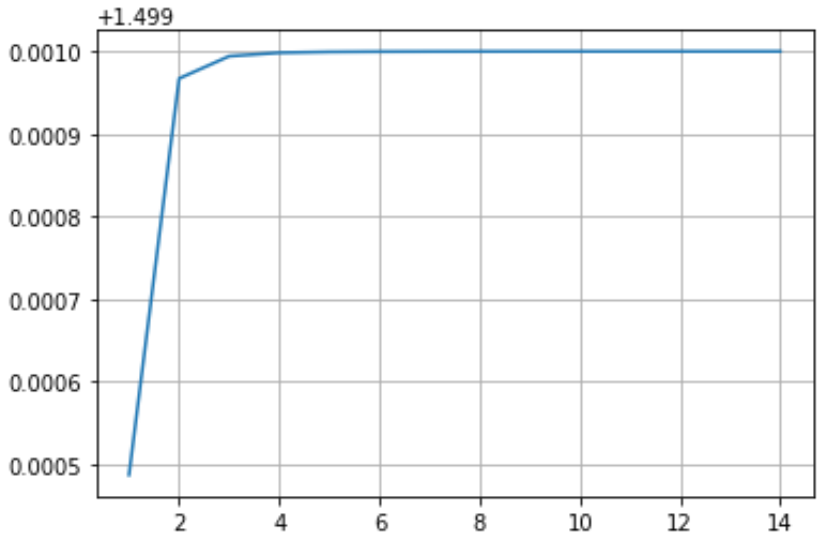
Inutile de dépasser 40 points pour intégrer. Mais quel est l'ordre de grandeur de l'erreur commise par la méthode de Gauss ?

In [21]:
```
ks = [k for k in range(1, 15)]
srect = [estimation_erreur(gauss, exp, 0, 1, k, exp(1) - 1) for k in ks
plt.plot(ks, srect)
plt.grid()
plt.show()
```



$\alpha = 6$ semble prometteur. Tout ceci reste bien entendu à prouver, et il s'avère que c'est bien la bonne valeur. Pour terminer, il faut noter que la régularité de la fonction à intégrer est importante pour que l'erreur soit petite. Essayons avec la fonction racine carrée sur 0, 1. On a $\int_0^1 \sqrt{x}\,dx = \frac{2}{3}$.

In [22]:
```
ks = [k for k in range(1, 15)]
srect = [estimation_erreur(gauss, sqrt, 0, 1, k, 2./3.) for k in ks]
plt.plot(ks, srect)
plt.grid()
plt.show()
```



On voit qu'ici l'erreur commise est un $O(1/n^{\frac{3}{2}})$, bien loin du $O(\frac{1}{n^6})$ espéré.

J'espère que vous avez compris le principe : ex-pé-ri-men-tez, c'est le grand intérêt des notebooks interactifs. Allez, une dernière question : quel est l'ordre de grandeur de l'erreur commise lorsqu'on intègre $\sqrt{x}$ avec la méthode des rectangles ? Des trapèzes ? Lorsqu'on intègre $\sin x$ avec la méthode de Gauss ? À vous de poursuivre ...