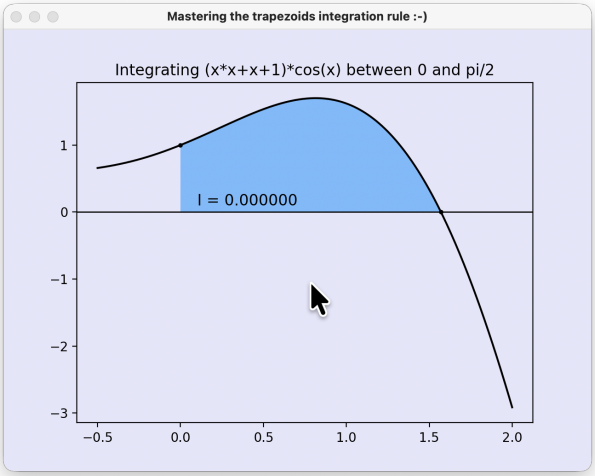
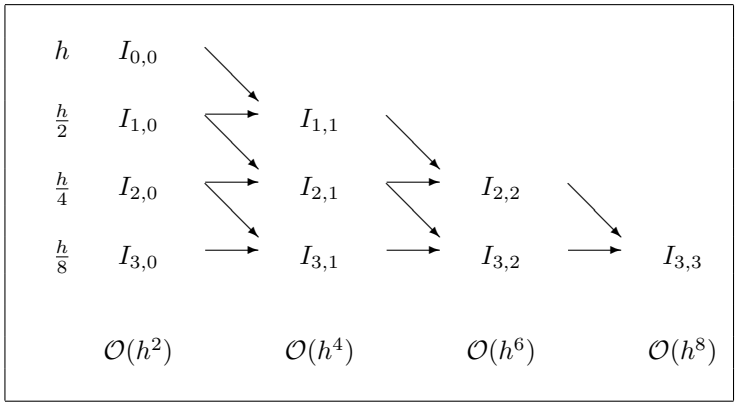
Python for dummies : problème 6
Des trapèzes, des trapèzes...
Toujours de trapèzes...
Des trapèzes récursifs ?
Encore de trapèzes !



Le devoir de la semaine passée était très simple, il s'agissait d'implémenter la règle composite d'intégration des trapèzes. Nous allons maintenant continuer notre travail en implémentant la méthode de Romberg qui consiste à effectuer l'extrapolation de Richardson pour la méthode composite des trapèzes.

$$I_{i,j} = \frac{\left(2^{2j} I_{i,j-1} - I_{i-1,j-1}\right)}{\left(2^{2j} - 1\right)}$$

Les termes $I_{0,0}$, $I_{1,0}$, $I_{2,0}$, $I_{3,0}$ et $I_{i,0}$ sont obtenus en appliquant successivement la méthode des trapèzes, avec n, $2n$, $4n$, $8n$ et $2^i n$ sous-intervalles.



Plus précisément, on vous demande de :

1. Ecrire une fonction `I,n,errorEst = romberg(f,a,b,n,nmax,tol)` qui calcule l'intégrale de la fonction `f` entre les abscisses `a` et `b` en utilisant la méthode de Romberg avec $I_{0,0}$ obtenu avec une intégration avec n trapèzes. On ajoutera successivement des lignes dans le tableau des valeurs de la méthode de Romberg jusqu'à obtenir une précision inférieure à une tolérance ϵ qui sera toujours un nombre réel strictement positif défini par l'argument `tol`. Pour estimer l'erreur d'intégration, on comparera la valeur obtenue $I_{i,i}$ avec celle de $I_{i,i-1}$ et on arrêtera le calcul lorsque

   $$|I_{i,i} - I_{i,i-1}| < \epsilon$$

   .

   A nouveau, afin d'éviter de se perdre dans les affres d'un calcul sans fin, on arrêtera le processus, lorsque le nombre d'intervalles à considérer est supérieur à un valeur `nmax`. Il faut donc bien arrêter le calcul avant d'effectuer une intégration avec un nombre d'intervalles supérieur à `nmax`. A titre d'exemple, si on ne peut pas atteindre la précision requise pour `n=1,nmax=100`, le programme devra

renvoyer la valeur obtenue avec $n = 64$. Votre fonction retournera -à nouveau- trois données : la valeur `I` de l'intégrale calculée, le nombre `n` d'intervalles utilisés pour obtenir l'erreur requise, l'estimation de l'erreur `errorEst`.

2. Le devoir est simple, mais faut quand-même essayer de réaliser une implémentation la plus efficace possible en évitant les calculs inutiles. Donc, choisir une implémentation récursive est une option possible mais c'est pas très efficace et donc cela ne vaut permettra pas de récolter la totalité des points pour ce devoir :-)

3. Ah oui : il faut évidemment ne pas faire trop d'appel à la fonction `f` lorsque vous effectuez les intégrations successives par la méthode composite des trapèzes. Dans cette optique, il est conseillé, proposé, permis et même suggéré de vous inspirer fortement de la solution de l'enseignant du devoir précédent `trapezes.py` pour effectuer une implémentation efficace de ce devoir ! Cela signifie évidemment que ce genre d'inspiration ne sera pas considéré exceptionnellement comme du plagiat. Heureusement pour Nathan et Ange, il est possible de mettre une option dans le logiciel anti-plagiat pour accepter ce genre d'exception dans l'analyse de vos soumissions.

4. Comme d'habitude, pour tester votre programme, on vous a fourni un tout petit programme simple `rombergTest.py` pour intégrer la fonction $(x^2 + x - 1)\cos(x)$ entre 0 et $\pi/2$.
   Notez que l'ensemble des valeurs que vous devriez obtenir dans le tableau de Romberg se trouve dans les notes de cours et les transparents du professeur !
   C'est difficile de ne pas être plus gentil :-)

```
from matplotlib import pyplot as plt
from numpy import *
from romberg import romberg

def u(x):
  return (x*x+x+1)*cos(x)

a = 0
b = pi/2
n = 1

I,n,errorEst = romberg(u,a,b,n,100,1e-8)
errorExact = abs(2.03819742706724 - I)
print("\n ======= Integral of (x*x+x+1)*cos(x) between 0 and pi/2 = %21.14e " % I)
print("  True error = %14.7e" % errorExact)
print("  Est. error = %14.7e" % errorEst)
print("  Number of intervals = %d" % n)

plt.figure("Mastering the trapezoids integration rule :-)")
x = array([a,b])
plt.plot(x,u(x),'.k',markersize=5)
x = linspace(a,b,200)
plt.fill(append(x,[0]),append(u(x),[0]),'xkcd:sky blue')
x = linspace(-0.5,2.0,300)
plt.plot(x,u(x),'-k')
plt.title('Integrating (x*x+x+1)*cos(x) between 0 and pi/2')
plt.gca().axhline(y=0,color='k',linewidth=1.0)
plt.text(0.1,0.1,"I = %8.6f" % I,fontsize=12)
plt.show()
```

5. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera soumise sur `zouLab` sans y adjoindre le programme de test fourni ! Cette fonction devra être soumise via le web : ce travail est individuel et sera évalué.