L1-S2 : UE Simulations numériques

SEANCE 8

Intégrales

6 avril 2022

Calculs d'intégrales

On souhaite calculer la valeur de l'intégrale d'une fonction f entre deux points connus, a et b : $\int_a^b f(x)\mathrm{d}x$

- Si la primitive F de f est connue analytiquement, l'intégrale est simplement : $\int_a^b f(x)\mathrm{d}x = F(b) - F(a)$
- Si la fonction f est échantillonnée en un nombre fini de points, $f(a),\ldots,f(x_i),\ldots,f(b)$, il est également possible **d'évaluer numériquement l'intégrale en évaluant l'aire sous la courbe $f(x)$ comprise entre les abscisses $x_i = a$ et $x_f = b$.**

Cette seconde approche numérique est très utile (en particulier lorsque la primitive de la fonction f n'est pas connue analytiquement).

Enfin, la comparaison entre le calcul analytique et numérique est essentielle car elle permet de vérifier la validité des algorithmes et d'évaluer la précision des calculs numériques.

Calculs d'intégrales

Losque, la fonction f est échantillonnée en un nombre fini de points, différentes méthodes d'intégration numérique peuvent être utilisées :

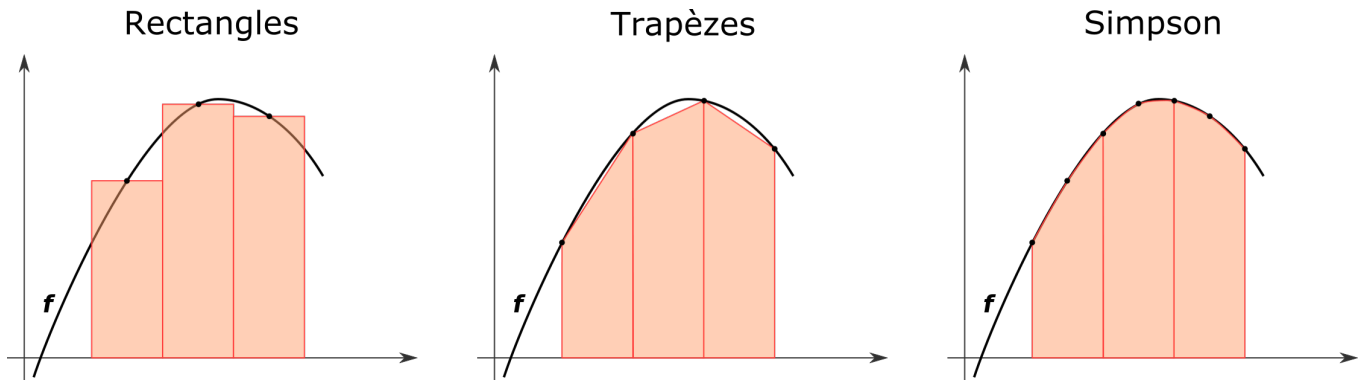
- intégration à partir d'une interpolation polynomiale de f entre points régulièrement espacés $\rightarrow$**formules de Newton - Cotes** (*simple et robuste - standard pour une fonction facile à évaluer*)
- "optimisation" des points où sont évalués la fonction $\rightarrow$ **quadrature Gaussienne** (*plus de liberté - plus difficile d'estimer l'erreur commise*)
- échantillonage aléatoire de la fonction $\rightarrow$ **calcul "Monte-Carlo"** (*particulièrement utile à N-dimensions*)

Cette séance : formules de Newton - Cotes + estimation de la précision.

Formules de Newton - Cotes

$f(x)$ est échantillonnée entre a et b. On souhaite estimer $I = \int_a^b f(x)\mathrm{d}x$.

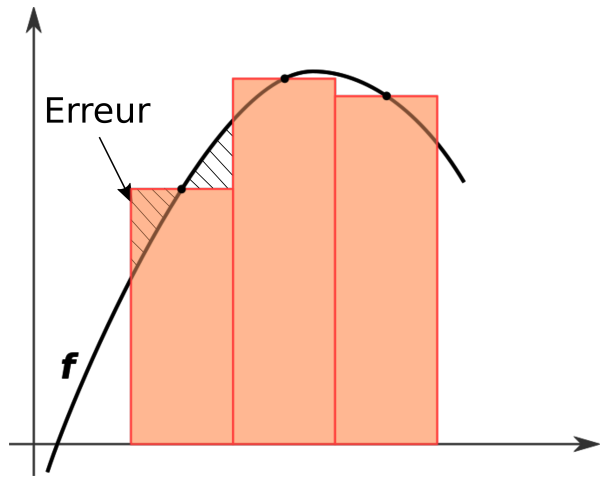
- ordre 0 : on approxime f par une fonction en escalier évaluée en un point par sous intervalle $\rightarrow$ **somme de Riemann**
- ordre 1 : on approxime f par une fonction affine évaluée en deux points par sous intervalle $\rightarrow$ **méthode des trapèzes**
- ordre 2 : on approxime f par une parabole évaluée en trois points par sous intervalle $\rightarrow$ **méthode de Simpson**



Ces méthodes ne sont rigoureusement exactes que pour les polynômes d'ordre associé $\rightarrow$ **erreur commise ?**

Formules de Newton - Cotes

Quelle est l'erreur commise sur I par la méthodes des rectangles ?



On considère d'abord **deux points** $[a, b]$ espacés par un pas $h = b - a$. Pour deux points seulement, la formule de Riemann donne :

$$\int_a^b f(x)\mathrm{d}x \simeq h \times f(\xi) = I_f$$

ξ est un point d'abscisse quelconque entre les deux bornes a et b qui définit la hauteur $f(\xi)$ du rectangle de largeur h.

Formules de Newton - Cotes

La position de ξ est cruciale ! Il suffit d'un D.L. pour le voir :

$$f(x) = f(\xi) + f'(\xi) \times (x - \xi) + f''(\xi) \times \frac{(x-\xi)^2}{2!} + \dots$$

Si on intègre ce DL sur la variable x dans l'intervalle $[a, b]$, l'erreur est

$$\int_a^b f(x)\mathrm{d}x - I_f = \frac{f'(\xi)}{2}\left[(x-\xi)^2\right]_a^b + \frac{f''(\xi)}{3 \times 2!}\left[(x-\xi)^3\right]_a^b + \dots$$

- soit pour $\xi = a$ ou $\xi = b$ une erreur $\int_a^b f(x)\mathrm{d}x - I_f = \frac{h^2}{2} f'(\xi) + \dots$
- soit pour $\xi = \frac{a+b}{2}$ une erreur $\int_a^b f(x)\mathrm{d}x - I_f = \frac{h^3}{24} f''(\xi) + \dots$

en n'oubliant pas que h est censé être petit.

Formules de Newton - Cotes

Généralisons à n points x_i séparés d'un pas $h = \frac{b-a}{n}$, et calculons l'erreur commise sur I.

- Si on fait la **somme à gauche** $\xi_i = x_i = a + i \times h$:

$$I = \int_a^b f(x)\mathrm{d}x = \sum_{i=0}^{n-1} hf(x_i) + n \times O(h^2 f') = \sum_{i=0}^{n-1} hf(x_i) + O\left(\frac{(b-a)^2 f'}{n}\right)$$

  l'erreur est en $O(1/n)$.

- Si on fait la **somme à droite** $\xi_i = x_{i+1} = a + (i+1) \times h$:

$$I = \int_a^b f(x)\mathrm{d}x = \sum_{i=1}^{n} hf(x_i) + n \times O(h^2 f') = \sum_{i=1}^{n} hf(x_i) + O\left(\frac{(b-a)^2 f'}{n}\right)$$

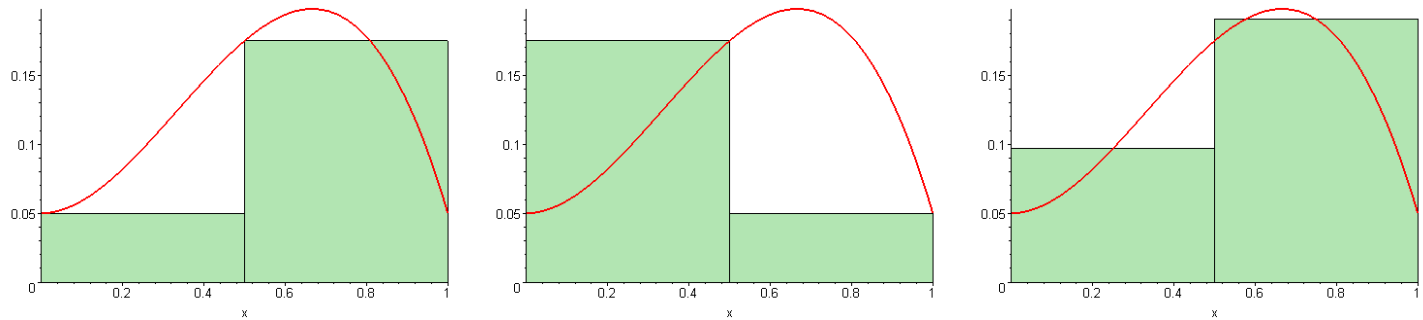
  l'erreur est en $O(1/n)$.

- Si on fait la **somme au milieu** $\xi_i = (x_i + x_{i+1})/2 = a + (i + \frac{1}{2}) \times h$:

$$I = \int_a^b f(x)\mathrm{d}x = \sum_{i=0}^{n-1} hf(\xi_i) + n \times O(h^3 f'') = \sum_{i=0}^{n-1} hf(\xi_i) + O\left(\frac{(b-a)^3 f''}{n^2}\right)$$

  l'erreur est en $O(1/n^2)$.

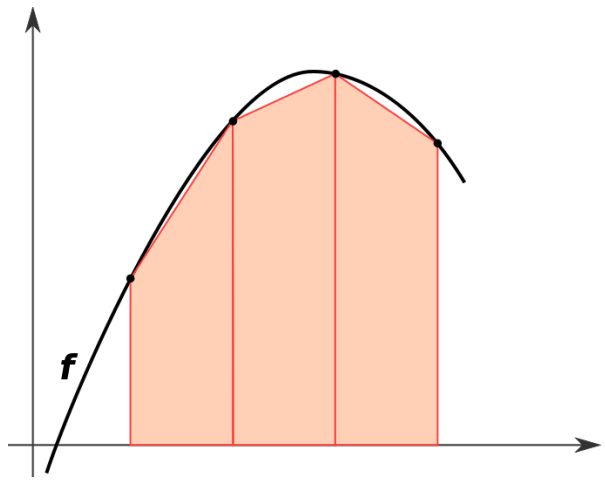
Formules de Newton - Cotes



Dans l'ordre : somme à gauche, somme à droite, somme au milieu.
Source : `https://fr.wikipedia.org/wiki/Somme_de_Riemann`.

Newton - Cotes - Ordre 1 : méthode des trapèzes

La **méthode des trapèzes** consiste à approximer la fonction par des **segments de droite sur des sous-intervalles**

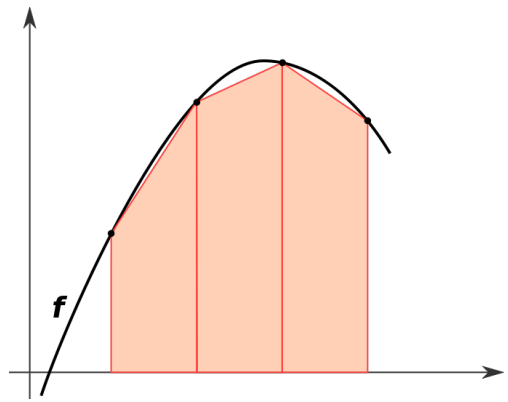


Pour **deux points**, à l'aide de la formule de l'aire du trapèze, l'intégrale s'écrit :

$$I = \int_a^b f(x)\mathrm{d}x = h \times \frac{f(a) + f(b)}{2} + O(h^3 f'')$$

Notez que l'erreur est du même ordre que celle obtenue pour la somme de Riemann au milieu.

Newton - Cotes - Ordre 1 : méthode des trapèzes



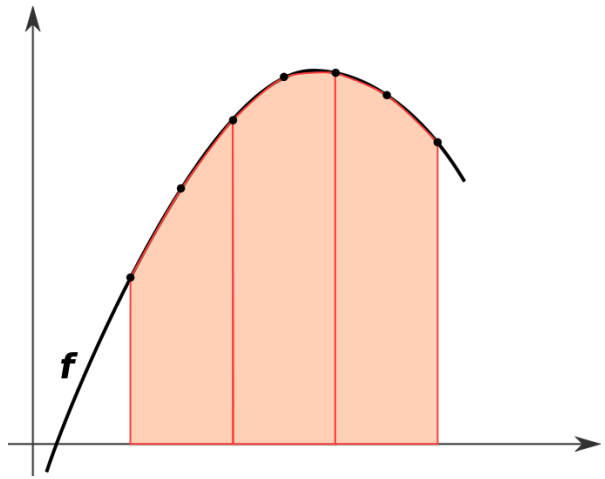
En généralisant à n points, pour $x_i = a + i \times h$, on obtient :

$$I = \int_a^b f(x)\mathrm{d}x = h \times \left(\frac{f_0}{2} + f_1 + f_2 + \ldots + f_{n-1} + \frac{f_n}{2} \right) + O\left(\frac{(b-a)^3 f''}{n^2} \right)$$

en notant $f_i = f(x_i)$.

Newton - Cotes - Ordre 2 : méthode de Simpson

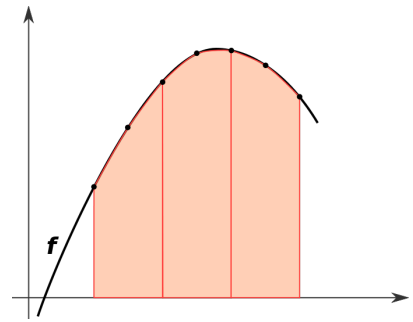
La **méthode de Simpson** consiste à approximer la fonction f par des **morceaux de paraboles** (avec un nombre impair de points !)



Pour trois points d'abscisse a, $m = (a+b)/2$ et b, le polynôme approximant est :

$$P(x) = f(a)\frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m)\frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b)\frac{(x-a)(x-m)}{(b-a)(b-m)}$$

Newton - Cotes - Ordre 2 : méthode de Simpson



Pour trois points seulement, l'intégrale s'écrit :

$$I = \int_a^b f(x)\mathrm{d}x = \frac{b-a}{2} \times \left(\frac{1}{3} f(a) + \frac{4}{3} f\left(\frac{a+b}{2} \right) + \frac{1}{3} f(b) \right) + O(h^5 f^{(4)})$$

$$\simeq \int_a^b P(x)\mathrm{d}x$$

En généralisant à $n+1$ points, pour $x_i = a + i \times h$, on obtient :

$$I = \int_a^b f(x)\mathrm{d}x = h \times \left(\frac{1}{3} f_0 + \frac{4}{3} f_1 + \frac{2}{3} f_2 + \frac{4}{3} f_3 + \ldots + \frac{2}{3} f_{n-3} + \frac{4}{3} f_{n-2} + \frac{1}{3} f_{n-1} \right)$$

$$+ O\left(\frac{(b-a)^5 f^{(4)}}{n^4} \right)$$

Dans la pratique : module integrate

Dans python, on peut utiliser le module `integrate` de `scipy` pour calculer une intégrale à partir d'un numpy array.

In [1]:
```
1 from scipy import integrate
2 help(integrate)
```

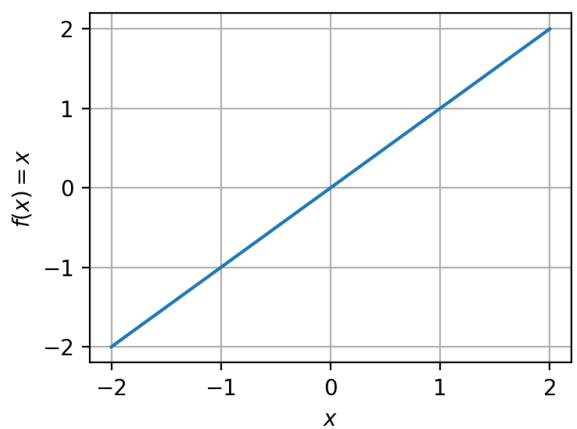
Out[1]:
```
Methods for Integrating Functions given fixed
    samples.

   trapz          -- Use trapezoidal rule to
    compute integral
                      from samples.
   cumtrapz       -- Use trapezoidal rule to
    cumulatively
                      compute integral.
   simps          -- Use Simpson's rule to
    compute integral
                      from samples.
   romb           -- Use Romberg Integration to
    compute integral
                      from (2**k + 1) evenly-
    spaced samples.
```

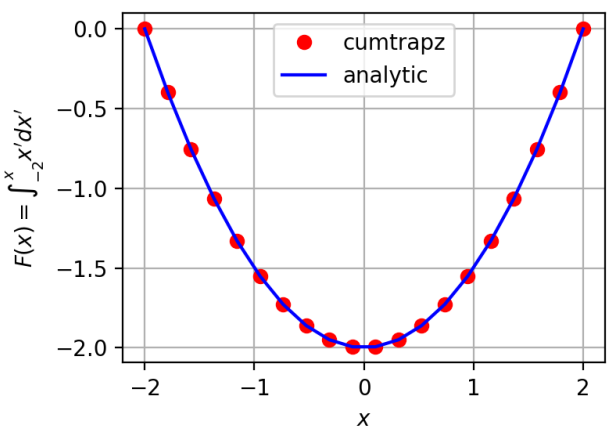
Dans la pratique : module integrate

Par exemple, la fonction `cumtrapz` de `integrate` renvoie l'ensemble des intégrales obtenues par la méthode des trapèzes de x_0 à x_i, soit une approximation numérique de la primitive :

In [2]:

```
from scipy import integrate
import numpy as np
import matplotlib . pyplot as plt
x = np.linspace(-2 , 2, 20)
y = x
y_int = integrate.cumtrapz(y , x, initial=0)
z = x*x/2 - 2
plt.plot(x, y_int,'ro', label=' cumtrapz')
plt.plot(x, z, label='analytic')
plt.show()
```





Dans la pratique : fonction quad

Un outil général d'intégration des fonctions 1D existe dans le module `scipy.integrate` : quad (pour quadrature). Il prend comme argument : une fonction et ses bornes puis renvoie l'intégrale et la précision absolue sur cette intégrale.

In [3]:
```
from scipy.integrate import quad

#fonction x^2
def f(x):
  return x*x

# Corps du programme
a = 0 # borne inf de l'integration
b = 1 # borne sup de l'integration
print(quad(f,a,b))
```

Out[3]:
```
(0.3333333333333333, 3.700743415417189e-15)
```

Dans la pratique : fonction quad

La fonction quad fait appel sans le dire à des méthodes identiques ou similaires à celles développées dans le cours.

Astuce : pour les fonctions python extrêmement simples (ou à plusieurs paramètres), on peut utiliser la syntaxe `lambda` de python. Le code suivant est parfaitement équivalent au précédent :

In [4]:
```
from scipy.integrate import quad

#Corps du programme
a = 0 #borne inf de l'integration
b = 1 #borne sup de l'integration
f = lambda x: x*x #fonction x^2
print(quad(f,a,b))
```

Out[4]:
```
(0.3333333333333333, 3.700743415417189e-15)
```

Dans la pratique : intégrales “impropres” convergentes

Que faire quand, bien que l'intégrale converge :

- la fonction ne peut pas être évaluée en un de ces points ($\sin(x)/x$ en $x = 0$)
- la fonction diverge en une de ces bornes ($1/\sqrt{x}$ en $x = 0$)
- l'intégrale court jusqu'à une borne infinie ($\int_0^\infty \mathrm{e}^{-x}\mathrm{d}x$)

Pour les deux premiers points, en particulier le premier, un moyen de s'en sortir consiste à utiliser une **intégration dite ouverte** (par opposition à fermée...), i.e. plutôt que d'effectuer l'intégrale entre a et b inclus, on l'effectue **entre** $a + h$ **et** $b - h$ **inclus.**

Dans la pratique : intégrales “impropres” convergentes

Lorsqu'une borne est infinie, si la fonction chute suffisamment rapidement, on peut se contenter d'imposer une **borne supérieure “grande”** (à déterminer au cas par cas avec une étude de convergence).

On peut aussi recourir à un **changement de variable**, par exemple si $b \to \infty$ et $a > 0$:

$$\int_a^b f(x)dx = \int_{1/b}^{1/a} f\left(\frac{1}{t}\right)\frac{\mathrm{d}t}{t^2}$$

ou encore :

$$\int_a^b f(x)dx = \int_{\exp(-b)}^{\exp(-a)} f(-\ln t)\frac{\mathrm{d}t}{t}$$

qui sont très aisés à mettre en place par le développeur...

A retenir

La précision d'une intégrale dépend de la **méthode utilisée** :

- Riemann (milieu) : $O\left(\frac{(b-a)^3 f''}{n^2}\right)$
- trapèzes : $O\left(\frac{(b-a)^3 f''}{n^2}\right)$
- Simpson : $O\left(\frac{(b-a)^5 f^{(4)}}{n^4}\right)$

et du **pas d'intégration** h.

Les fonctions de la librairie `scipy` sont confortables pour les développeurs mais il ne faut jamais oublier qu'elles fournissent un résultat avec une certaine **précision** qui dépend entre autre de **l'échantillonnage** de la fonction.

À vos TPs !

1. Ouvrir un terminal :
   - soit sur https ://jupyterhub.ijclab.in2p3.fr
   - soit sur un ordinateur du 336
2. Télécharger la séance d'aujourd'hui :

```
methnum fetch L1/Seance8 TONGROUPE
```

   en remplaçant TONGROUPE par ton numéro de groupe.
3. Sur un ordinateur du bâtiment 336 uniquement, lancer le jupyter :

```
methnum jupyter notebook
```

4. Pour soumettre la séance, dans le terminal taper :

```
methnum submit L1/Seance8 TONGROUPE
```

À vos TPs !

Rappel : votre gitlab personnel sert de sauvegarde pour passer vos documents d'une plateforme à l'autre via les commandes methnum/fetch.