Intégration numérique

TD 12

Compétences

— Méthodes des rectangles et des trapèzes pour le calcul approché d'une intégrale sur un segment
— Concevoir un algorithme répondant à un problème précisement posé

Pré-conditions

L'objectif de ce TD est de calculer une valeur *approchée* d'une intégrale définie comme

$$I = \int_a^b f(x)dx$$

où f est une fonction réelle de variable réelle définie et continue sur un intervalle fermé borné $[a,b]$
Il n'est pas toujours aisé de mener à bien ce genre de calcul :

- il est possible que la primitive de la fonction f s'avère difficile à déterminer à partir des fonctions usuelles,
- ou bien que la fonction f ne soit connue qu'en un nombre fini de points comme c'est souvent le cas en sciences expérimentales.

On appelle *formule de quadrature* ou *formule d'intégration numérique* toute formule permettant de calculer une approximation de $I(f)$.

Les formules de Newton-Cotes

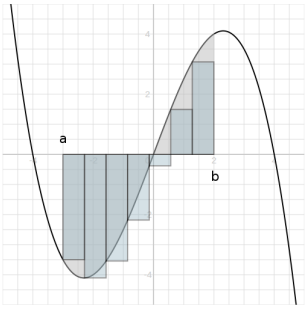
Les formules de Newton-Cotes sont des formules de quadrature de type interpolation, faisant intervenir une subdivision régulière de l'intervalle $[a,b]$. Elles recouvrent tous les cas habituellement utilisés en intégration approchée (rectangle, trapèze...) Elles sont dites de type fermé, car les extrémités de l'intervalle sont dans la subdivision.

Dans les méthodes d'intégration, l'intégrale d'une fonction continue sur un intervalle borné [a, b] est remplacée par une somme finie. Le choix de la subdivision de l'intervalle d'intégration et celui des coefficients qui interviennent dans la somme approchant l'intégrale sont des critères essentiels pour minimiser l'erreur.

Rectangle à gauche

- En décomposant l'intervalle d'intégration $[a,b]$ en m sous-intervalles de largeur $H = \frac{b-a}{m}$ avec $m \geq 1$
- En introduisant les nœuds de quadrature $x_k = a + kH$ pour $k = 0, 1, ..., m-1$ on obtient la formule du rectangle à gauche

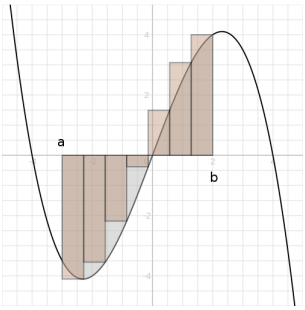
$$I_m(f) = H \sum_{k=0}^{m-1} f(a+kH)$$



Rectangle à droite

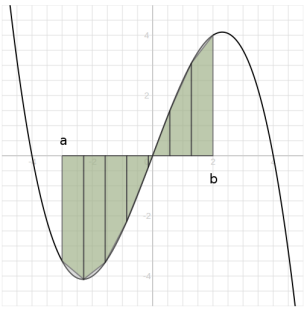
- En décomposant l'intervalle d'intégration $[a,b]$ en m sous-intervalles de largeur $H = \frac{b-a}{m}$ avec $m \geq 1$
- En introduisant les nœuds de quadrature $x_k = a + (k+1)H$ pour $k = 0, 1, ..., m-1$ on obtient la formule du rectangle à droite

$$I_m(f) = H \sum_{k=0}^{m-1} f(a+(k+1)H)$$



Méthode des trapèzes

- En décomposant l'intervalle d'intégration $[a,b]$ en m sous-intervalles de largeur $H=\frac{b-a}{m}$ avec $m \geq 1$
- En introduisant les nœuds de quadrature $x_k = a + kH$ pour $k = 0, 1, ..., m-1$ on obtient la formule des trapèzes



$$I_m(f) = H\left(\frac{1}{2}(f(a)+f(b)) + \sum_{k=1}^{m-1} f(a+kH)\right)$$

Erreur de quadrature :

$$E_m(f) = \frac{b-a}{12} H^2 |f''(\epsilon)| \quad \epsilon \in]a,b[$$

Implémentation et utilisation de la méthode des trapèzes

À partir d'une fonction connue

1. Implémenter une fonction dont le prototype est `trapeze_f(f, a, b, m)` et qui retourne un tableau numpy unidimensionnel constitué des m valeurs. *Dans un premier temps on ne cherchera pas à vectoriser le code.*
2. Tester votre fonction avec $I = \int_0^\pi sin(x)dx$, on prendra 8 puis 16 intervalles. La valeur exacte est bien évidemment $I = 2$.
3. Estimer l'erreur de quadrature commise et en déduire un encadrement de $I = \int_0^\pi sin(x)dx$ dans les deux cas.
4. Combien d'intervalles seraient nécessaires pour avoir une précision de l'ordre de 10^{-6} ?

À partir d'un ensemble de valeurs discrètes

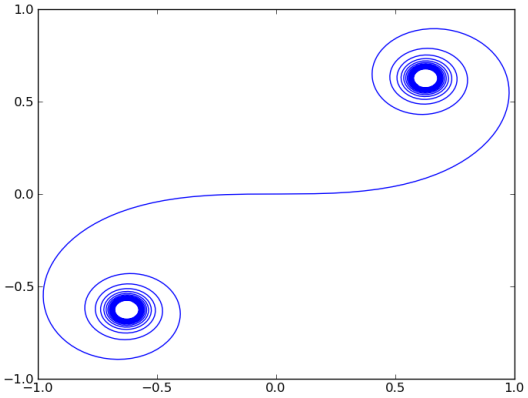
1. Implémenter une fonction dont le prototype est `trapeze_d(pts)` (avec `pts` un tableau de type `ndarray`) et qui retourne la valeur approchée de $\int_a^b f(x)dx$, f étant connue qu'à partir d'une liste de valeurs expérimentales.
2. Estimer $I = \int_0^{5/2} f(x)dx$ à partir des données suivantes

x	0	$\frac{1}{2}$	1	$\frac{3}{2}$	2	$\frac{5}{2}$
f(x)	$\frac{3}{2}$	2	2	1.6364	1.2500	0.9565

en utilisant la fonction `trapeze_d` $I \simeq 4.057325$

Représenter la spirale de Cornu

Virage en douceur La clothoïde est une spirale dont la courbure est en chaque point proportionnelle à la longueur parcourue. Une portion de clothoïde permet donc de relier en douceur une ligne droite (de coubure nulle) à un arc de cercle (de courbure constante, non nulle). C'est donc la trajectoire la plus *confortable*, celle que l'on adopte pour le tracé des autoroutes et des voies ferrées. La clothoïde est aussi utilisée pour former les loopings des manèges à sensations.
La clothoïde aurait été étudiée dès le dix-huitième siècle par Jacques Bernoulli (1654-1705).



1. Représenter cette spirale définie par :

$$x(t) = \int_0^t cos(u^2)du \quad \text{et} \quad y(t) = \int_0^t sin(u^2)du$$

la borne t variant dans l'intervalle $[-10, 10]$ avec un pas de 0.01
2. Que peut-on dire de la complexité en temps ?
3. Essayer de vectoriser la fonction `trapeze_f` pour améliorer les performances. On pourra utiliser `np.sum(y)` qui renvoie la somme des éléments de `y <ndarray>`.

Intégration numérique avec `scipy.integrate.quad`

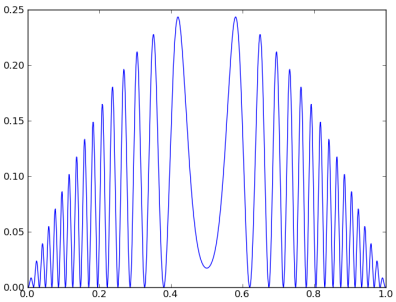
1. Calculer $I = \int_0^1 x(1-x)sin^2(200x(1-x))$ avec la méthode des trapèzes.
2. D'après la doc Python :

Examples

Calculate $\int_0^4 x^2 dx$ and compare with an analytic result

```
>>> from scipy import integrate
>>> x2 = lambda x: x**2
>>> integrate.quad(x2, 0, 4)
(21.333333333333332, 2.3684757858670003e-13)
>>> print(4**3 / 3.)  # analytical result
21.3333333333
```

Calculer la valeur de I avec la fonction quad.



3. Quelle doit-être la valeur du paramètre m de la fonction `trapeze` pour obtenir une précision similaire ?

Une méthode basée sur une technique probabiliste : Monte-Carlo

D'après Wikipédia

Le terme méthode de Monte-Carlo, désigne toute méthode visant à calculer une valeur numérique en utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes. Le nom de cette méthode, qui fait allusion aux jeux de hasard pratiqués à Monte-Carlo, a été inventé en 1947 par Nicholas Metropolis, et publié pour la première fois en 1949 dans un article coécrit avec Stanislaw Ulam.
Les méthodes de Monte-Carlo sont particulièrement utilisées pour calculer des intégrales en dimensions plus grandes que 1 (en particulier, pour calculer des surfaces et des volumes). Elles sont également couramment utilisées en physique des particules, où des simulations probabilistes permettent d'estimer la forme d'un signal ou la sensibilité d'un détecteur...

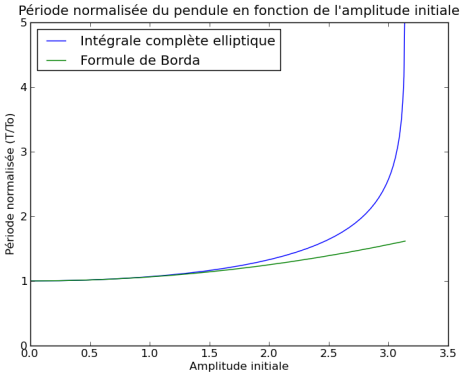
Le principe de l'algorithme est relativement simple :

- On tire un point x_1 au hasard dans l'intervalle $[a, b]$
- On calcule $f(x_1)$
- Puis on recommence avec $x_2, x_2, \ldots, x_m$ et on calcule $f(x_2), f(x_3), \ldots, f(x_m)$
- L'aire de $I = \int_a^b f(x)dx$ est alors donnée par

$$I \simeq \frac{(b-a)f(x_1) + (b-a)f(x_2) + \cdots + (b-a)f(x_m)}{m} \simeq \frac{(b-a)}{m} \sum_{k=1}^{m} f(x_k)$$

Implémenter la fonction `monteCarlo(f, a, b, m)` et comparer avec les méthodes d'intégration numériques précédentes.

Période d'oscillation du pendule



Vous avez vu en cours de physique que la variation de la période réduite $T^* = T/T_0$ en fonction de θ_0 peut être calculée par :

$$\frac{T}{T_0} = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} \frac{d\phi}{\sqrt{1-k^2 sin^2(\phi)}} \quad \text{avec} \quad k = sin(\theta_0/2) \quad \text{et} \quad sin\phi = \frac{1}{k} sin(\theta/2)$$

Tracer la courbe de $T^* = f(\theta_0)$

On termine avec `scipy.special.ellipk`

Répondre à la question précédente à l'aide de la doc *scipy*

scipy.special.ellipk

scipy.special.ellipk(*m*)

Computes the complete elliptic integral of the first kind.

This function is defined as

$$K(m) = \int_0^{\pi/2} [1 - m\sin(t)^2]^{-1/2} dt$$

Parameters : **m** : *array_like*
The parameter of the elliptic integral.

Returns : **K** : *array_like*
Value of the elliptic integral.

Notes

For more precision around point m = 1, use ellipkm1.

Attention sous Scipy $K(k)$ est définie comme :

$$ellipk(m) = \int_0^{\frac{\pi}{2}} \frac{d\phi}{\sqrt{1 - m\ sin^2(\phi)}}$$

Autrement dit $T^* = \frac{2}{\pi} * ellipk(k^2)$ et $k = sin(\frac{k}{2})$