

Informatique - CN2

Intégration numérique

D.Malka

MPSI 2020-2021

Sommaire

- 1 Position du problème
- 2 Méthode des rectangles
- 3 Méthode des trapèzes
- 4 Fonction prédéfinie du module `scipy.integrate`

Sommaire

- 1 Position du problème
- 2 Méthode des rectangles
- 3 Méthode des trapèzes
- 4 Fonction prédéfinie du module `scipy.integrate`

Intégration d'une fonction sur un intervalle

Le but de ce chapitre est d'aborder le calcul numérique de l'intégrale d'une fonction f sur un domaine fini délimité par ses bornes a et b :

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

Pourquoi une intégration numérique ?

- ▶ calcul analytique fastidieux
- ▶ expression exacte coûteuse
- ▶ l'intégrale n'admet pas d'expression analytique

Exemple d'intégrale non exprimable analytiquement

$$\text{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du$$

Principe d'une intégration numérique

Il existe différentes méthodes d'intégration numériques (Newton-Cotes, Legendre-Gauss, Monte-Carlo. . .). Nous nous intéressons à deux avatars de la méthode de Newton-Cotes composite : la *méthode des rectangles* et la *méthode des trapèzes*.

Principe de la méthode de Newton-Cotes composite

Soit $J = \int_a^b f(x)dx$.

On calcule une valeur approchée \tilde{J} de J en remplaçant la fonction f par une approximation \tilde{f} puis on intègre cette fonction :

$$J \approx \tilde{J} = \int_a^b \tilde{f}(x)dx$$

Sommaire

- 1 Position du problème
- 2 Méthode des rectangles**
- 3 Méthode des trapèzes
- 4 Fonction prédéfinie du module `scipy.integrate`

Principe de la méthode des rectangles

On décompose l'intervalle $[a, b]$ en N intervalles I_k de longueur h : $[a, a+h]$, $[a+h, a+2h]$, \dots , $[a+kh, a+(k+1)h]$, \dots $[b-h, b]$.

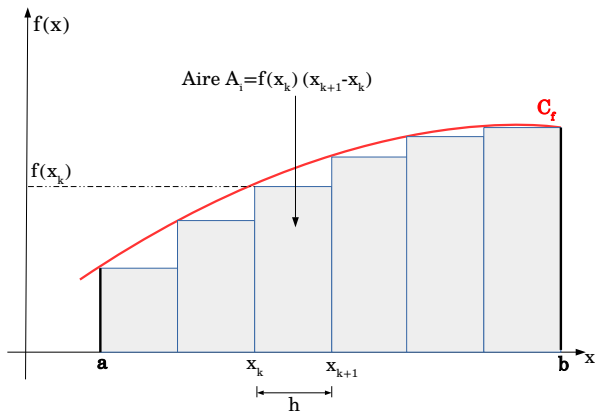
Évidemment : $Nh = b - a$ soit $h = \frac{b-a}{N}$.

h est appelé le pas d'intégration de l'algorithme.

Sur chaque intervalle $I_k = [a+kh, a+(k+1)h]$, on approche la fonction f par la fonction $\tilde{f}_k : x \rightarrow \tilde{f}(x_k)$ où $x_k = a+kh$.

Principe de la méthode des rectangles

Géométriquement :



On approche l'aire sous la courbe représentative de f par la somme des aires des rectangles de hauteur $f(x_k)$ et de largeur $(x_{k+1} - x_k)$.

Principe de la méthode des rectangles

Analytiquement, sur l'intervalle $[a, b]$, on obtient :

$$J = \int_a^b f(x) dx = \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} f(x) dx$$

$$\Rightarrow \tilde{J} = \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} \tilde{f}_k(x) dx$$

$$\Leftrightarrow \tilde{J} = \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} f(x_k) dx$$

$$\Leftrightarrow \tilde{J} = \sum_{k=0}^{N-1} f(x_k)(x_{k+1} - x_k)$$

avec $x_k = a + kh$ et $x_{k+1} = a + (k+1)h$:

$$\tilde{J} = \sum_{k=0}^{N-1} f(x_k)h \quad \text{avec} \quad x_k = a + kh$$

Implémentation en Python

```
1 def rect_integ(f, a, b, n):
2     """
3     Approximate the integral of function f over [a,b] using the
4     rectangle method.
5     h=(b-a)/n is the integration step.
6     f: function
7     a,b: float
8     n:int, number of calculation points
9     """
10    h=(b-a)/n
11    S=0
12    for k in range(n):
13        S=S+f(a+k*h)
14    J=S*h
15    return J
```

Essai

Essai sur $J = \int_0^1 4\sqrt{1-x^2}dx = \pi$.

CODE PYTHON

```
In [1]: rect_integ(lambda x:4*sqrt(1-x**2),0,1,10)# h=0.1
```

```
Out[1]: 3.30451832625
```

```
In [2]: rect_integ(lambda x:4*sqrt(1-x**2),0,1,10000)# h=0.0001
```

```
Out[2]: 3.14179147761
```

L'erreur $e = |\tilde{J} - J|$ semble diminuer avec la valeur du pas h .

Majoration de l'erreur

Erreur sur l'intégrale calculée

Soit $|e| = |\tilde{J} - J|$ l'erreur sur le calcul de l'intégrale J sur $[a, b]$.

$$e < \frac{1}{2} \text{Sup}_{[a,b]} |f'(\xi)| h$$

L'erreur commise au pire sur J est de l'ordre du pas d'intégration h .

La méthode des rectangles est une méthode d'ordre 1.

Majoration de l'erreur

Preuve - Erreur d'intégration sur $I_k = [x_k, x_{k+1}]$.

- ▶ Soit $e_k = \int_{x_k}^{x_{k+1}} f(x) dx - f(x_k)(x_{k+1} - x_k)$ l'erreur d'intégration sur $I_k = [x_k, x_{k+1}]$.
- ▶ En supposant $f \in C^1$ sur $[a, b]$ donc sur I_k , d'après le théorème des accroissements finis :

$\forall x \in I_k, \exists \xi \in I_k$ tel que $f(x) = f'(\xi)(x - x_k) + f(x_k)$ d'où :

- ▶ $e_k = \int_{x_k}^{x_{k+1}} f'(x)(x - x_k) dx$
- ▶ $\Rightarrow |e_k| = \left| \int_{x_k}^{x_{k+1}} f'(x)(x - x_k) dx \right|$
- ▶ $\Rightarrow |e_k| \leq \int_{x_k}^{x_{k+1}} |f'(x)(x - x_k)| dx$
- ▶ $\Rightarrow |e_k| \leq \text{Sup}_{I_k} |f'(x)| \int_{x_k}^{x_{k+1}} (x - x_k) dx$
- ▶ $\Leftrightarrow |e_k| \leq \text{Sup}_{I_k} |f'(x)| \frac{(x_{k+1} - x_k)^2}{2}$
- ▶ $\Leftrightarrow \boxed{|e_k| \leq \text{Sup}_{I_k} |f'(x)| \frac{h^2}{2}}$

Majoration de l'erreur

Preuve - Erreur d'intégration sur $I = [a, b]$.

▶ Soit $|e| = |\tilde{J} - J|$ l'erreur d'intégration sur $I_k = [a, b]$.

▶ $|e| = |\sum_{k=0}^{N-1} e_k|$

▶ $\Rightarrow |e| \leq \sum_{k=0}^{N-1} |e_k|$

▶ Or, on a montré que : $|e_k| \leq \text{Sup}_{I_k} |f'(\xi)| \frac{h^2}{2}$ donc :

▶ $|e| \leq \sum_{k=0}^{N-1} \text{Sup}_{I_k} |f'(\xi)| \frac{h^2}{2}$

▶ $\Rightarrow |e| \leq \sum_{k=0}^{N-1} \text{Sup}_{[a,b]} |f'(\xi)| \frac{h^2}{2}$

▶ $|e| \leq N \text{Sup}_{[a,b]} |f'(\xi)| \frac{h^2}{2}$

▶ Or $N = \frac{(b-a)}{h}$, d'où :

▶ $|e| \leq \text{Sup}_{[a,b]} |f'(\xi)| \frac{(b-a)}{2} h$

Au pire, l'erreur commise sur \tilde{J} est de l'ordre du pas d'intégration h .

Complexité (vs efficacité)

```
1 def rect_integ(f, a, b, n):
2     """
3     Approximate the integral of function f over [a,b] using the
4     rectangle method.
5     h=(b-a)/n is the integration step.
6     f: function
7     a,b: float
8     n:int, number of calculation points
9     """
10    h=(b-a)/n
11    S=0
12    for k in range(n):
13        S=S+f(a+k*h)
14    J=S*h
15    return J
```

Unité de mesure : appel de la fonction f .

Complexité linéaire avec n : $O(n) = O\left(\frac{1}{h}\right)$.

Sommaire

- 1 Position du problème
- 2 Méthode des rectangles
- 3 Méthode des trapèzes**
- 4 Fonction prédéfinie du module `scipy.integrate`

Principe de la méthode des trapèzes

On procède de manière analogue à la méthode des rectangles.

On décompose l'intervalle $[a, b]$ en N intervalles I_k de longueur h : $[a, a+h]$, $[a+h, a+2h]$, \dots , $[a+kh, a+(k+1)h]$, \dots $[b-h, b]$.

Évidemment : $Nh = b - a$ soit $h = \frac{b-a}{N}$.

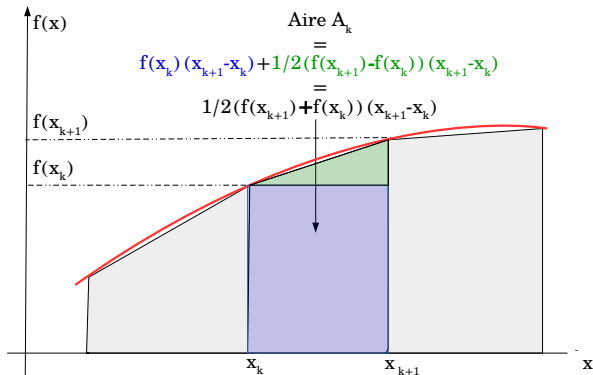
h est appelé le pas d'intégration de l'algorithme.

C'est le choix de la fonction \tilde{f}_k approchant f sur $I_k = [a+kh, a+(k+1)h]$ qui est différent :

$$\tilde{f}_k : x \rightarrow \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} (x - x_k) + f(x_k) \quad \text{avec} \quad x_k = a + kh$$

Principe de la méthode des trapèzes

Géométriquement :



On approche l'aire sous la courbe représentative de f par la somme des aires des trapèzes de largeur $(x_{k+1} - x_k)$.

Principe de la méthode des trapèzes

Analytiquement, sur l'intervalle $[a, b]$, on obtient :

$$J = \int_a^b f(x)dx = \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} f(x)dx$$
$$\Rightarrow \tilde{J} = \sum_{k=0}^{N-1} \int_{x_k}^{x_{k+1}} \tilde{f}_k(x)dx$$

En intégrant $\tilde{f}_k(x)$ ou plus simplement en évaluant géométriquement l'aire sous le segment de droite, on obtient :

$$\Leftrightarrow \tilde{J} = \sum_{k=0}^{N-1} \frac{1}{2} (f(x_{k+1}) + f(x_k))(x_{k+1} - x_k)$$

avec $x_k = a + kh$ et $x_{k+1} = a + (k+1)h$:

$$\tilde{J} = \frac{1}{2} \sum_{k=0}^{N-1} (f(a + (k+1)h) + f(a + kh))h$$

Implémentation en Python

```
1 def trap_integ(f, a, b, n):
2     """
3     Approximate the integral of function f over [a,b] using the
4     trapezoidal method.
5     h=(b-a)/n is the integration step.
6     f: function
7     a,b: float
8     n:int, number of calculation points
9     """
10    h=(b-a)/n
11    S=0
12    for k in range(n):
13        S=S+(f(a+(k+1)*h)+f(a+k*h))
14    J=1/2*S*h
15    return J
```

Essai

Essai sur $J = \int_0^1 4\sqrt{1-x^2}dx = \pi$.

CODE PYTHON

```
In [3]: trap_integ(lambda x:4*sqrt(1-x**2),0,1,10)# h=0.1
```

```
Out [3]: 3.14057361483
```

```
In [4]: trap_integ(lambda x:4*sqrt(1-x**2),0,1,10000)# h=0.0001
```

```
Out [4]: 3.14159147761
```

L'erreur $e = |\tilde{J} - J|$ semble diminuer avec la valeur du pas h .

Majoration de l'erreur

Erreur sur l'intégrale calculée

Soit $e = |\tilde{J} - J|$ l'erreur sur le calcul de l'intégrale J sur $[a, b]$.

$$e \leq \frac{1}{12} \text{Sup}_{[a,b]} |f''(\xi)| h^2$$

L'erreur commise au pire sur l'évaluation de J est de l'ordre du carré du pas d'intégration h .

La méthode des trapèzes est une méthode d'ordre 2.

Majoration de l'erreur

Théorème de Taylor-Lagrange

Soit f une fonction définie sur un intervalle $[u, v]$ de \mathbb{R} telle que :

- ▶ \forall entier naturel $k \leq n$, $f^{(k)}$ existe et est continue sur $[u, v]$;
- ▶ \forall entier naturel $k \leq n$, $f^{(k)}$ existe sur $]u, v[$.

Alors, $\exists \xi \in [u, v]$ tel que :

$$f(v) = f(u) + f'(u)(v - u) + \frac{1}{2}f''(u)(v - u)^2 + \cdots + \frac{1}{n!}f^{(n)}(u)(v - u)^n + \frac{f^{(n+1)}(\xi)}{(n+1)!}(v - u)^{n+1}$$

Majoration de l'erreur

Preuve - Erreur d'intégration sur $I_k = [x_k, x_{k+1}]$.

- ▶ Soit $e_k = \int_{x_k}^{x_{k+1}} \frac{1}{2}(f(x_{k+1}) - f(x_k))(x_{k+1} - x_k) - f(x)dx$ l'erreur d'intégration sur $I_k = [x_k, x_{k+1}]$.
- ▶ On suppose $f \in C^2$ sur $[a, b]$ donc sur I_k quelque soit k .
- ▶ En utilisant le développement de Taylor-Lagrange à l'ordre 2 sur $[x_k, x]$, $x \leq x_{k+1}$:

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(\xi)(x - x_k)^2$$

- ▶ Après intégration et majoration, il vient :

$$|e_k| \leq \frac{1}{12} \text{Sup}_{I_k} |f''(\xi)| h^3$$

- ▶ Puis, on applique l'inégalité triangulaire à $e = \sum_{k=0}^{N-1} e_k$:

$$|e| \leq \sum_{k=0}^{N-1} |e_k|$$

- ▶ Ce qui donne : $|e| \leq \frac{1}{12} \text{Sup}_{[a,b]} |f''(\xi)| h^2$

Complexité (vs efficacité)

```
1 def trap_integ(f,a,b,n):
2     """
3     Approximate the integral of function f over [a,b] using the
4     trapezoidal method.
5     h=(b-a)/n is the integration step.
6     f: function
7     a,b: float
8     n:int, number of calculation points
9     """
10    h=(b-a)/n
11    S=0
12    for k in range(n):
13        S=S+(f(a+(k+1)*h)+f(a+k*h))
14    J=1/2*S*h
15    return J
```

Unité de mesure : appel de la fonction f .

Complexité linéaire avec n : $O(n) = O\left(\frac{1}{h}\right)$.

Méthode des trapèzes vs méthode des rectangles

Essai sur $J = \int_0^1 4\sqrt{1-x^2}dx = \pi$.

CODE PYTHON

In [5]: `rect_integ(lambda x:4*sqrt(1-x**2),0,1,10)# h=0.1`

Out [5]: 3.30451832625

In [6]: `trap_integ(lambda x:4*sqrt(1-x**2),0,1,10)# h=0.1`

Out [6]: 3.14057361483

Méthode	Complexité	Erreur
Rectangles	$O\left(\frac{1}{h}\right)$	$\sim h$
Trapèzes	$O\left(\frac{1}{h}\right)$	$\sim h^2$

A complexité égale, la méthode des trapèzes fournit une approximation de l'intégrale nettement plus précise. En pratique, on préfère utiliser une méthode d'approximation d'ordre supérieur plutôt que diminuer le pas.

Sommaire

- 1 Position du problème
- 2 Méthode des rectangles
- 3 Méthode des trapèzes
- 4 Fonction prédéfinie du module `scipy.integrate`**

Fonctions prédéfinies du module `scipy.integrate`

Fonction `quad`. Utilise un algorithme à pas adaptatif écrit en langage FORTRAN.

Essai sur $J = \int_0^1 4\sqrt{1-x^2}dx = \pi$.

 CODE PYTHON

```
In [7]: trap_integ(lambda x:4*sqrt(1-x**2),0,1,1e8)# h=1e-8
Out[7]: 3.14159265355
```

```
In [8]: scipy.integrate.quad(lambda x:4*sqrt(1-x**2),0,1)# h variable
Out[8]: (3.1415926535897922, 3.533564552071766e-10)
```

On constate aussi que `quad` est nettement plus rapide que `trap_integ`!

En pratique, on utilisera les fonctions prédéfinies du module `scipy.integrate` plutôt que des fonctions implémentées par nos soins.