Quadratures

Sébastien Joannès

October 1, 2015

Contents

1 Qu'est-ce qu'une quadrature ?

1.1 Introduction

En mathématiques, le terme **quadrature** est une opération géométrique visant à rechercher et construire un carré d'aire égale à une surface donnée. La quadrature la plus célèbre est probablement la **quadrature du cercle**, problème vieux de 2000 ans tout simplement impossible à réaliser à la règle et au compas.
Depuis le XVIIe siècle, le terme **quadrature** est associé au calcul d'aires et au calcul intégral.

Soit f une fonction dont on ne connaît les valeurs qu'en un nombre fini de points (mesures) ou que la primitive ne peut se calculer analytiquement. La **quadrature** vise à approcher la quantité I suivante:

$$I = \int_a^b f(x)\,\mathrm{d}x \tag{1}$$

Il s'agit donc de déterminer l'aire de la surface délimitée par l'axe (Ox), les droites d'équation $x = a$ et $x = b$ et la courbe d'équation $y = f(x)$.

1.2 Les principales méthodes de quadrature

Les principales méthodes de quadrature repose sur l'interpolation par morceaux de la fonction f.

- $[a, b]$ est découpé en sous-intervalles $[x_i, x_{i+1}]$ ou $[x_{i-1}, x_i, x_{i+1}]$ pour lesquels nous connaissons $y_{i-1} = f(x_{i-1})$, $y_i = f(x_i)$ et $y_{i+1} = f(x_{i+1})$.
- Une interpolation de la fonction est réalisée sur chaque intervalle.
- C'est l'intégrale du polynôme d'interpolation qui est évaluée sur chaque sous-intervalle.

Les différences entre les méthodes de quadrature proviennent du nombre de points d'interpolation pour chaque sous-intervalle:

- Méthode à 1 point : méthode des rectangles
- Méthode à 2 points : méthode des trapèzes
- Méthode à 3 points : méthode de Simpson
- Méthode à n points : méthode de Newton-Cotes

2 Erreur d'interpolation et précision de la quadrature

Soit f une fonction de classe C^{n+1} sur l'intervalle $[a, b]$. Il est possible de démontrer que:
$\forall x \in [a, b]$, il existe $\eta \in [a, b]$ tel que

$$f(x) - p(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_n)}{(n+1)!} f^{(n+1)}(\eta) \tag{2}$$

En particulier, le point η étant inconnu, le corollaire donnant une majoration de l'erreur d'interpolation est plus utile:

$$|f(x) - p(x)| \leq \frac{|(x - x_0)(x - x_1)\dots(x - x_n)|}{(n+1)!} \sup_{x \in [a,b]} |f^{(n+1)}(x)| \tag{3}$$

3 Méthodes de quadrature simple

```
>>> import numpy as np
... import scipy as sp
... import matplotlib as mpl
... import matplotlib.pyplot as plt
... %matplotlib inline
```

Le module **integrate** de Scipy contient de nombreuses possibilités de quadrature.
Découvrez ces nombreuses fonctionnalités via l'aide du module integrate.

```
>>> from scipy import integrate
... #help(integrate)
```

3.1 Méthode des rectangles

Soit $x_0, x_1, \ldots, x_n$, $n+1$ points régulièrement espacés, i.e. $x_i = a + i\,(b-a)\,/n$.
Sur chaque intervalle $[x_i, x_{i+1}]$, la fonction f est approchée par la constante $f(x_i)$ et l'approximation de I par la méthode des rectangles et donnée par:

$$I_R = \frac{b-a}{n} \sum_{i=0}^{n-1} f\,(x_i) \tag{4}$$

3.2 Méthode des trapèzes

Soit $x_0, x_1, \ldots, x_n$, $n+1$ points régulièrement espacés, i.e. $x_i = a + i\,(b-a)\,/n$.
Sur chaque intervalle $[x_i, x_{i+1}]$, la fonction f est approchée par la fonction affine coïncidant en x_i et x_{i+1}. L'approximation de I par la méthode des rectangles et donnée par:

$$I_T = \frac{b-a}{2n} \left[f\,(x_0) + 2 \sum_{i=1}^{n-1} f\,(x_i) + f\,(x_n) \right] \tag{5}$$

```
>>> x = np.linspace(-1,1,10)
... y = x**2
... print integrate.trapz(y, x)

0.683127572016
```

3.3 Méthode de Simpson

Pour la méthode de Simpson, la fonction f est approximée sur chaque intervalle par une parabole.

```
>>> x = np.linspace(-1,1,10)
... y = x**2
... print integrate.simps(y, x)

0.66849565615
```

3.4 Méthode de Newton-Cotes

La méthode de Newton-Cotes est une généralisation des méthodes précédentes au degré n. Il s'agit de déterminer p le polynôme d'interpolation de f sur un sous-intervalle Ω_i de $[a, b]$ puis de calculer l'intégrale de p.

$$p\,(x) = \sum_{i=0}^{n} f\,(x_i)\,L_i\,(x) \tag{6}$$

Sur Ω_i, on a:

$$I_{\Omega_i} = \int_{\Omega_i} f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} f(x_i) \int_{\Omega_i} L_i(x)\,\mathrm{d}x = \sum_{i=0}^{n} \omega_i f(x_i) \tag{7}$$

ω_i correspond au **poid** (pondération) associé à la valeur $f(x_i)$ et est différent d'une méthode à l'autre.

4 Méthode de Gauss

Dans certaines circonstances (degré d'interpolation élevé), la méthode de Newton-Cotes peut ne pas converger. La méthode de Gauss permet de choisir de manière optimale les poids ω_i ainsi que les points x_i afin d'éviter tout phénomène de Runge.
On peut en particulier distinguer les méthodes de:

- Gauss-Legendre
- Gauss-Chebyshev
- Gauss-Lobato
- Gauss-Laguerre
- Gauss-Radau

Ces différentes configurations sont très utilisés pour la méthode des éléments finis.

Pour la configuration de Gauss-Legendre, les points ainsi que les poids sont donnés par:

```
>>> print np.polynomial.legendre.leggauss(1)
... print np.polynomial.legendre.leggauss(2)
... print np.polynomial.legendre.leggauss(3)
```

```
(array([ 0.]), array([ 2.]))
(array([-0.57735027,  0.57735027]), array([ 1.,  1.]))
(array([-0.77459667,  0.        ,  0.77459667]), array([ 0.55555556,  0.88888889,  0.55555556]))
```

Nous obtenons de même les points et les poids de la configuration de Gauss-Chebyshev:

```
>>> print np.polynomial.chebyshev.chebgauss(1)
... print np.polynomial.chebyshev.chebgauss(2)
... print np.polynomial.chebyshev.chebgauss(3)
```

```
(array([  6.12323400e-17]), array([ 3.14159265]))
(array([ 0.70710678, -0.70710678]), array([ 1.57079633,  1.57079633]))
(array([  8.66025404e-01,   6.12323400e-17,  -8.66025404e-01]), array([ 1.04719755,  1.04719755,  1.
```

5 Méthode de Monte-Carlo (exemple de l'estimation de pi)