Chapitre 7
Premiers algorithmes numériques

Dans ce chapitre, nous allons rencontrer nos premiers algorithmes numériques, c'est à dire des algorithmes qui travaillent sur des nombres flottants et non plus sur des entiers. Il importe de se souvenir que contrairement aux calculs sur le type *int*, les calculs sur le type *float* comportent de nombreuses approximations : conversion entre nombres décimaux et nombres dyadiques, calculs approchés... avec pour conséquences les phénomènes d'absorbtion et de cancellation [1].

1. Égalité entre nombres flottants

De ceci il résulte que parler d'égalité entre nombres flottants n'a en général pas de sens car les nombres que l'on cherche à comparer sont souvent le résultat de plusieurs calculs, chacun d'eux étant potentiellement entachés de plusieurs approximations, sans même parler de l'approximation initiale issue de la conversion approximative entre nombres décimaux et nombres dyadiques.
Dès lors que les quantités manipulées sont des flottants, il est donc potentiellement très risqué d'assurer la terminaison d'un algorithme par un test d'égalité `x == y`. On préférera en général une condition traduisant peu ou prou le fait que les quantités x et y sont « proches », dans un sens qu'il conviendra de préciser.

1.1 Comment définir la « proximité » de deux nombres ?

Une première solution, simple, consiste à choisir une marge d'erreur absolue : étant donnée une valeur $\varepsilon > 0$ fixée arbitrairement (par exemple $\varepsilon = 10^{-12}$), sont considérés comme proches deux nombres x et y tels que $|x-y| \leqslant \varepsilon$.
Par souci de simplicité il pourra arriver qu'on fasse un tel choix, mais ce n'est en général pas une bonne solution, car une valeur choisie dans l'absolu parce qu'elle nous parait petite peut se révéler trop grande lorsque les nombres à comparer sont eux-même très petits (on conçoit bien que comparer la proximité de deux nombres de l'ordre de 10^{-18} avec $\varepsilon = 10^{-12}$ n'a pas grand sens) ou au contraire très grands.
Une solution plus raisonnable consiste à mesurer l'erreur relative : sont considérés comme égaux deux nombres flottants qui vérifient $|x-y| \leqslant \varepsilon|y|$, où ε est une valeur « petite » fixée arbitrairement. Mais cette situation présente aussi des inconvénients :

- cette relation n'est pas symétrique : on peut avoir $|x-y| \leqslant \varepsilon|y|$ et $|y-x| > \varepsilon|x|$;
- cette relation présente un défaut majeur lorsque x et y sont de signes opposés : si $y = -x$ cette relation ne sera jamais vérifiée dès lors que $\varepsilon < 2$, même lorsque x et y seront égaux aux plus petits des nombres flottants non nuls.

Bref, le problème de l'égalité de deux nombres flottants ne possède pas de réponse simple et absolue, et demande souvent un traitement au cas par cas. Une telle discussion est esquissée dans l'exemple qui suit, mais dans la suite de ce cours et pour des raisons de simplicité on se contentera le plus souvent d'une marge d'erreur relative voire absolue (mais en gardant à l'esprit le problème que cela pose).

Remarque. Il existe dans le module Numpy une fonction `isclose` dédiée à la comparaison de deux nombres flottants. Cette fonction possède deux valeurs `rtol` (par défaut égale à $\varepsilon_r = 10^{-5}$) et `atol` (par défaut égale à $\varepsilon_a = 10^{-8}$) et renvoie la valeur `True` dès lors que $|x-y| \leqslant \varepsilon_a + \varepsilon_r|y|$. On observera que cette fonction combine une tolérance absolue et une tolérance relative ; néanmoins elle n'est pas symétrique et dans de rares cas `isclose(x, y)` peut retourner `True` et `isclose(y, x)` retourner `False`. Depuis la version 3.5 de Python il existe dans le module math une fonction de même nom qui elle utilise la relation $|x-y| \leqslant \varepsilon_a + \varepsilon_r \max(|x|,|y|)$ (avec des valeurs par défaut $\varepsilon_r = 10^{-9}$ et $\varepsilon_a = 0$).

1. Se référer au chapitre 4.

1.2 Résolution d'une équation du second degré

Le problème de l'égalité à 0 peut aisément être mis en évidence en cherchant à rédiger une fonction résolvant une équation du second degré à coefficients réels de la forme : $ax^2 + bx + c = 0$. La figure 1 présente une solution naïve à ce problème.

```
from numpy import sqrt

def solve(a, b, c):
    delta = b * b - 4 * a * c
    if delta < 0:
        print("pas de solution")
    elif delta > 0:
        x, y = (-b-sqrt(delta))/2/a, (-b+sqrt(delta))/2/a
        print("deux racines simples {} et {}".format(x, y))
    else:
        x = -b/2/a
        print("une racine double {}".format(x))
```

Figure 1 – Résolution naïve d'une équation du second degré.

Testons-la avec les valeurs $a = 0,01$, $b = 0,2$ et $c = 1$ puis avec $a = 0,011025$, $b = 0,21$, et $c = 1$:

```
>>> solve(0.01, 0.2, 1)
deux racines simples -10.000000131708903 et -9.999999868291098
>>> solve(0.011025, 0.21, 1)
pas de solution
```

Or dans les deux cas le discriminant de l'équation est nul (vérifiez-le en faisant le calcul à la main). Cependant les erreurs de calculs inhérentes à la manipulation des nombres flottants conduisent dans le premier cas à obtenir un discriminant égal à $6,938893903907228 \cdot 10^{-18} = 2^{-57}$ et dans le second cas à $-6,938893903907228 \cdot 10^{-18} = -2^{57}$. Pour pallier à ce problème, on peut observer que lorsque le discriminant $\Delta = b^2 - 4ac$ est très petit devant b^2 les deux racines sont quasiment confondues. Une solution consiste donc à remplacer la condition $\Delta = 0$ par la condition $|\Delta| \ll b^2$, ce qui va se traduire concrètement par $|\Delta| \leqslant \varepsilon b^2$, la valeur de ε étant choisie petite. En choisissant par exemple $\varepsilon = 2^{-52}$ on obtient la version présentée figure 2.

```
def solve2(a, b, c, epsilon=2**(-52)):
    delta = b * b - 4 * a * c
    if delta < -epsilon*b**2:
        print("pas de solution")
    elif delta > epsilon*b**2:
        x, y = (-b-sqrt(delta))/2/a, (-b+sqrt(delta))/2/a
        print("deux racines simples {} et {}".format(x, y))
    else:
        x = -b/2/a
        print("une racine double {}".format(x))
```

Figure 2 – Résolution numérique d'une équation du second degré.

Testons-la avec le même jeu de valeurs :

```
>>> solve2(0.01, 0.2, 1)
une racine double -10.0
>>> solve2(0.011025, 0.21, 1)
une racine double -9.523809523809524
```

Remarque. La valeur $\varepsilon = 2^{-52} \approx 2,22 \cdot 10^{-16}$ n'est pas choisie au hasard, il s'agit de l'*epsilon machine*, c'est-à-dire la plus grande erreur relative que peut provoquer l'arrondissement arithmétique en virgule flottante. En d'autres termes, $1 + \varepsilon$ est la plus petite quantité strictement supérieure à 1 qui soit discernable de 1. En effet, la

représentation machine de $1+\varepsilon$ est :

$$1+\varepsilon = (1, \underbrace{0000\cdots0000}_{\text{51 fois 0}}1)_2 \times 2^0$$

(rappelons que les nombres flottants sont représentés avec une mantisse à 52 bits).
En d'autres termes, il n'y a aucun intérêt à prendre une erreur relative plus petite, comme on peut le constater :

```
>>> solve2(0.01, 0.2, 1, epsilon=2**(-53))
deux racines simples -10.000000131708903 et -9.999999868291098
>>> solve2(0.011025, 0.21, 1, epsilon=2**(-53))
pas de solution
```

et dans la pratique il sera prudent de choisir une valeur de ε plus importante sachant que les erreurs d'approximation peuvent se cumuler.

2. Recherche d'une racine par dichotomie

Le premier problème numérique auquel nous allons nous intéresser est celui de la recherche d'une valeur approchée de la racine d'une fonction. D'un point de vue mathématique, nous considérons une fonction continue $f : [a,b] \to \mathbb{R}$ telle que $f(a)$ et $f(b)$ soient de signes opposés, ce qui se traduit par la relation : $f(a)f(b) \leqslant 0$. Le théorème des valeurs intermédiaires assure alors qu'il existe un réel $c \in [a,b]$ tel que $f(c) = 0$ (illustration figure 3).

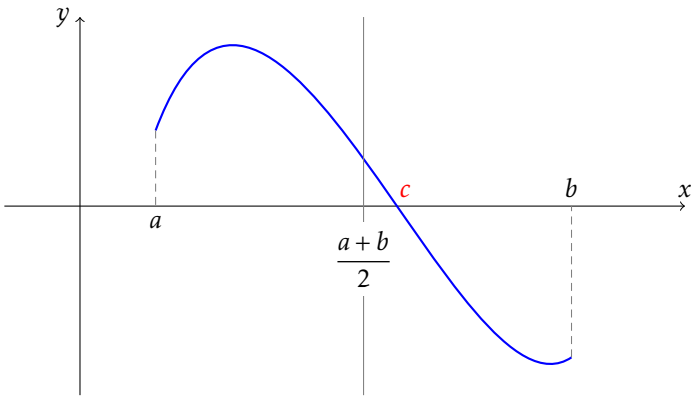


Figure 3 – Recherche d'une racine par dichotomie.

2.1 Description de la méthode

Le principe de la recherche dichotomique consiste à calculer $f\left(\frac{a+b}{2}\right)$. Suivant le signe de cette quantité l'une des deux relations est nécessairement vérifiée :

$$f(a)f\left(\frac{a+b}{2}\right) \leqslant 0 \qquad \text{ou} \qquad f\left(\frac{a+b}{2}\right)f(b) \leqslant 0.$$

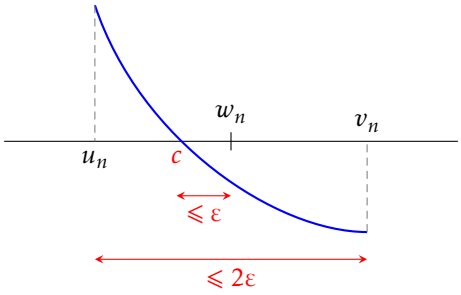
Le premier cas assure l'existence d'une racine dans l'intervalle $\left[a, \frac{a+b}{2}\right]$, le second dans l'intervalle $\left[\frac{a+b}{2}, b\right]$, ramenant dans chacun des deux cas la recherche à un intervalle d'amplitude moitié moins grande.
L'algorithme de recherche consiste donc à itérer deux suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ définies par les valeurs initiales $u_0 = a$ et $v_0 = b$ et la relation de récurrence :

$$(u_{n+1}, v_{n+1}) = \begin{cases} (u_n, m) & \text{si } f(u_n)f(m) \leqslant 0 \\ (m, v_n) & \text{sinon} \end{cases} \qquad \text{avec } m = \frac{u_n + v_n}{2}$$

Analyse de l'algorithme

La validité de cet algorithme est assurée par l'invariant : $\forall n \in \mathbb{N}$, $f(u_n)f(v_n) \leqslant 0$, qui assure pour tout $n \in \mathbb{N}$ l'existence d'une racine dans l'intervalle $[u_n, v_n]$.

Il reste à choisir la condition de terminaison de cet algorithme. Sachant que u_n est une approximation par défaut d'une racine de f et v_n une approximation par excès, on choisit une valeur $\varepsilon > 0$ et on retourne $w_n = \dfrac{u_n + v_n}{2}$ dès lors que $v_n - u_n \leqslant 2\varepsilon$. De la sorte, on est assuré qu'il existe une racine c de f vérifiant : $|w_n - c| \leqslant \varepsilon$.



Coût de l'algorithme

Il est facile de prouver que $v_n - u_n = \dfrac{b-a}{2^n}$ (l'intervalle de recherche est divisé par deux à chaque itération) ; ainsi un résultat est retourné dès lors que $\dfrac{b-a}{2^n} \leqslant 2\varepsilon$, c'est à dire : $n \geqslant \log_2\left(\dfrac{b-a}{\varepsilon}\right) - 1$.
En général, on choisit pour ε une puissance de 10 car si $\varepsilon = 10^{-p}$ cet algorithme assure $p-1$ chiffres significatifs après la virgule (en faisant bien entendu abstraction des approximations successives). Dans ce cas, l'algorithme se termine dès lors que $n \geqslant \log_2(b-a) + p\log_2(10) - 1 = \mathrm{O}(p)$; on peut donc affirmer qu'il s'agit d'un algorithme de coût linéaire vis-à-vis du nombre de décimales souhaitées après la virgule [2].

2.2 Mise en œuvre pratique

La recherche déchotomique nécessite quatre paramètres : la fonction f, les valeurs de a et b et la valeur de ε (qui par défaut sera égal à 10^{-12}).

```
def dicho(f, a, b, epsilon=1e-12):
    if f(a) * f(b) > 0:
        return None
    u, v = a, b
    while abs(v - u) > 2 * epsilon:
        w = (u + v) / 2
        if f(u) * f(w) <= 0:
            v = w
        else:
            u = w
    return (u + v) / 2
```

Utilisons cet algorithme pour chercher l'unique racine de la fonction sinus entre 3 et 4 :

```
>>> from numpy import sin
>>> dicho(sin, 3, 4)
3.141592653589214
```

Obtenons maintenant une approximation de $\sqrt{2}$:

```
>>> def f(x): return x * x - 2
...
>>> dicho(f, 1, 2)
1.4142135623724243
```

2. Nous étudierons bientôt un algorithme bien plus rapide mais moins robuste, l'algorithme de Newton-Raphson.

Dans ce dernier exemple, il a été nécessaire de définir la fonction $f : x \mapsto x^2 - 2$ avant d'appliquer l'algorithme. On aurait pu s'en passer en utilisant une *fonction anonyme* que l'on peut définir à l'aide du mot-clef `lambda` :

```
>>> dicho(lambda x: x * x - 2, 1, 2)
1.4142135623724243
```

2.3 Utilisation du module scipy

`scipy` est un ensemble de modules spécialement dédiés au calcul numérique ; il n'est donc pas étonnant d'y trouver une fonction permettant de réaliser la recherche dichotomique que nous venons d'étudier. Cette fonction se nomme `bisect` et se trouve dans le module `scipy.optimize`. Comme la majorité des autres fonctions de ce module ne nous intéressent pas pour l'instant, on se contente de charger en mémoire cette unique fonction :

```
>>> from scipy.optimize import bisect
```

Avant de l'utiliser, lisons la documentation qui lui est associée :

```
>>> help(bisect)
```

On trouvera figure 4 une version (un peu simplifiée) de ce qu'on obtient.

```
bisect(f, a, b, xtol=1e-12, maxiter=100)
    Find root of a function within an interval.

    Basic bisection routine to find a zero of the function f between the
    arguments a and b. f(a) and f(b) can not have the same signs.
    Slow but sure.

    Parameters
    ----------
    f : function
        Python function returning a number.  f must be continuous, and
        f(a) and f(b) must have opposite signs.
    a : number
        One end of the bracketing interval [a,b].
    b : number
        The other end of the bracketing interval [a,b].
    xtol : number, optional
        The routine converges when a root is known to lie within xtol of the
        value return. Should be >= 0.
    maxiter : number, optional
        if convergence is not achieved in maxiter iterations, and error is
        raised.  Must be >= 0.

    Returns
    -------
    x0 : float
        Zero of f between a and b.
```

Figure 4 – La page d'aide de la fonction `bisect`.

Comme on peut le constater, cette fonction agit peu ou prou de la même façon que la notre. Seul paramètre supplémentaire : un nombre d'itérations maximal au delà duquel une erreur est déclenchée. Ceci peut en effet être nécessaire si l'utilisateur de cette fonction choisit une valeur de ε trop faible relativement aux valeurs de a et b. Dans ce cas, et passé un certain rang, il se peut que les valeurs de u et v soient tellement proches que le calcul de $\dfrac{u+v}{2}$ retourne la valeur de u ou de v. Dans ce cas, l'algorithme ne se termine plus.

Il est facile de modifier notre algorithme pour calquer ce comportement :

```
def dicho(f, a, b, epsilon=1e-12, maxiter=100):
    if f(a) * f(b) > 0:
        return None
    n = 0
    u, v = a, b
    while abs(v - u) > 2 * epsilon:
        n += 1
        if n > maxiter:
            chn = 'Échec après {} itérations.'.format(maxiter)
            raise RuntimeError(chn)
        w = (u + v) / 2
        if f(u) * f(w) <= 0:
            v = w
        else:
            u = w
    return (u + v) / 2
```

L'instruction `raise` déclenche une exception (ici l'exception `RuntimeError`) qui met immédiatement fin à l'exécution de la fonction. Cette exception s'accompagne d'une chaîne de caractères qui s'affiche à la suite de l'exception :

```
>>> dicho(sin, 3, 4, maxiter=10)
RuntimeError: Échec après 10 itérations.
```

3. Valeur approchée d'une intégrale

La calcul d'une intégrale définie de la forme :

$$\mathrm{I}_{u,v}(f) = \int_u^v f(t)\,\mathrm{d}t$$

où f est une fonction continue sur le segment $[u,v]$ à valeurs dans $\mathbb{R}$, est un problème classique intervenant dans de nombreux domaines, qu'ils soient scientifiques ou non. Cette évaluation peut cependant s'avérer difficile voire impossible en pratique car il n'est pas toujours possible de déterminer une primitive de la fonction f, même en utilisant les techniques de changement de variable ou d'intégration par parties.
Nous allons nous intéresser à certaines méthodes de quadrature qui consistent à approcher la valeur de l'intégrale par une somme pondérée finie de valeurs de la fonction f en des points choisis ; en d'autres termes ces méthodes fournissent une approximation de $\mathrm{I}_{u,v}(f)$ par la quantité :

$$\mathrm{I}^p_{u,v}(f) = \sum_{i=0}^{p} \alpha_i f(x_i)$$

les coefficients $\alpha_0,\ldots,\alpha_p$ étant réels et dépendants de l'entier p et les points $x_0,\ldots,x_p$ appartenant à $[u,v]$.
Dans la formule ci-dessus les points x_i et les coefficients α_i sont respectivement appelés *nœuds* et *poids* de la formule de quadrature. L'*erreur de quadrature* est la quantité :

$$\mathrm{E}_{u,v}(f) = \mathrm{I}_{u,v}(f) - \mathrm{I}^p_{u,v}(f)$$

et on dira qu'une méthode de quadrature est *d'ordre k* quand l'erreur commise est nulle lorsque f est un polynôme de degré inférieur ou égal à k.

Méthodes de quadratures composites
Les méthodes de quadratures composites pour calculer une valeur approchée de l'intégrale :

$$\mathrm{I}(f) = \int_a^b f(t)\,\mathrm{d}t$$

consistent à subdiviser l'intervalle $[a,b]$ en n sous-intervalles $a = u_0 < u_1 < \cdots < u_n = b$ (en général avec un pas constant) et à appliquer une méthode de quadrature sur chacun des intervalles $[u_i, u_{i+1}]$. Dans ce cas, l'erreur

commise est égale à :

$$\mathscr{E}_n(f) = \sum_{i=0}^{n-1} \mathrm{E}_{u_i,u_{i+1}}(f)$$

On gardera en mémoire l'expression d'une subdivision de pas régulier : $u_k = a + k\dfrac{b-a}{n}$, $0 \leqslant k \leqslant n$.

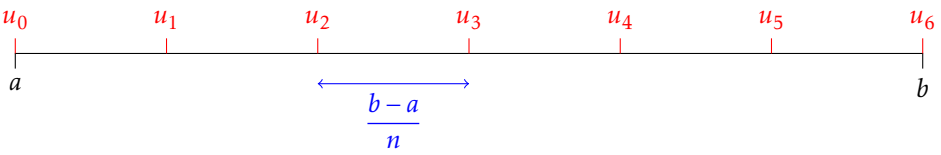


Figure 5 – Une subdivision de pas régulier avec $n = 6$.

Enfin, les différentes majorations des erreurs commises par les méthodes que nous allons étudier reposent sur le théorème de Rolle, dont on rappelle ici l'énoncé :

Théorème (Rolle). — *Si $g : [a,b] \to \mathbb{R}$ est continue sur le segment $[a,b]$, dérivable sur l'intervalle $]a,b[$ et si $g(a) = g(b)$ alors il existe $c \in]a,b[$ tel que $g'(c) = 0$.*

3.1 Méthode du rectangle

La méthode de quadrature du rectangle est la plus simple qui soit : elle consiste à approcher la fonction f par la valeur qu'elle prend en un point de l'intervalle $[u,v]$, en général une de ses extrémités. Si on choisit pour unique nœud $x_0 = u$ et pour poids $\alpha_0 = v - u$, ceci conduit à approcher $\mathrm{I}_{u,v}(f)$ par :

$$\mathrm{I}^0_{u,v}(f) = (v-u)f(u)$$

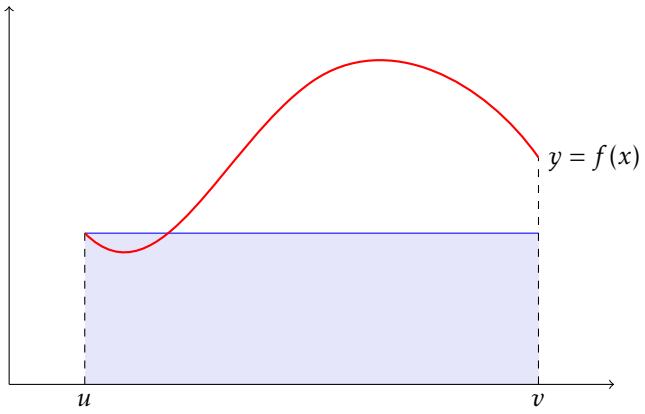


Figure 6 – La méthode du rectangle gauche. La valeur de $\mathrm{I}^0_{u,v}(f)$ correspond à l'aire colorée.

Théorème. — *La méthode du rectangle est d'ordre 0, et si f est de classe $\mathscr{C}^1$, l'erreur de quadrature vérifie :*

$$|\mathrm{E}_{u,v}(f)| \leqslant \mathrm{M}_1 \frac{(v-u)^2}{2}$$

où M_1 est un majorant de $|f'|$ sur l'intervalle $[u,v]$.

Preuve. Si f est un polynôme constant, on pose $f(x) = \lambda$ et dans ce cas :

$$\mathrm{I}_{u,v}(f) = \int_u^v \lambda \,\mathrm{d}t = \lambda(v-u) = (v-u)f(u) = \mathrm{I}^0_{u,v}(f),$$

ce qui montre que la méthode des rectangles est d'ordre 0.
Si f est une fonction de classe $\mathscr{C}^1$, l'erreur commise vaut :

$$\mathrm{E}_{u,v}(f) = \int_u^v f(t)\,\mathrm{d}t - (v-u)f(u) = \int_u^v \Big(f(t)-f(u)\Big)\,\mathrm{d}t \qquad \text{donc} \qquad |\mathrm{E}_{u,v}(f)| \leqslant \int_u^v |f(t)-f(u)|\,\mathrm{d}t.$$

Fixons $t \in]u,v]$ et considérons la fonction $g : x \mapsto f(x) - f(u) - \mathrm{K}(x-u)$, la valeur de K étant choisie de sorte que $g(t) = 0$. La fonction g est de classe $\mathscr{C}^1$ et vérifie $g(u) = g(t) = 0$ donc d'après le théorème de Rolle il existe $c \in]u,t[$ tel que $g'(c) = 0$, ce qui signifie que $\mathrm{K} = f'(c)$. On a donc $f(t) - f(u) = f'(c)(t-u)$, ce qui prouve la majoration : $|f(t)-f(u)| \leqslant \mathrm{M}_1(t-u)$. On en déduit :

$$|\mathrm{E}_{u,v}(f)| \leqslant \mathrm{M}_1 \int_u^v (t-u)\,\mathrm{d}t = \mathrm{M}_1 \frac{(v-u)^2}{2}.$$

□

• Méthode du rectangle composite

La méthode du rectangle composite consiste à considérer une subdivision $(u_0, u_1, \ldots, u_n)$ de $[a,b]$ de pas régulier et à utiliser la linéarité de l'intégrale :

$$\mathrm{I}(f) = \int_a^b f(t)\,\mathrm{d}t = \sum_{k=0}^{n-1} \int_{u_k}^{u_{k+1}} f(t)\,\mathrm{d}t$$

pour approcher chacune des intégrales $\mathrm{I}_{u_k,u_{k+1}}(f)$ par $(u_{k+1}-u_k)f(u_k) = \dfrac{b-a}{n} f\left(a + k\dfrac{b-a}{n}\right)$.
Autrement dit, cette méthode consiste à approcher $\mathrm{I}(f)$ par :

$$\boxed{\frac{b-a}{n} \sum_{k=0}^{n-1} f\left(a + k\frac{b-a}{n}\right).}$$

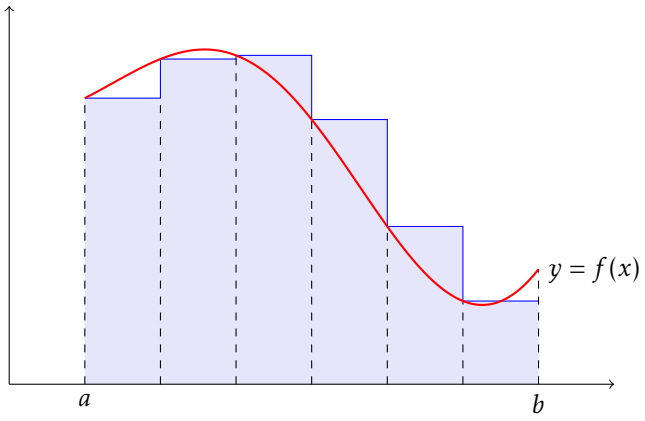


Figure 7 – Méthode du rectangle composite.

Théorème. — *Si f est de classe $\mathscr{C}^1$, l'erreur de la méthode du rectangle composite vérifie :*

$$|\mathscr{E}_n(f)| \leqslant \mathrm{M}_1 \frac{(b-a)^2}{2n}$$

où M_1 est un majorant de $|f'|$ sur l'intervalle $[a,b]$.

Preuve. On a $|\mathscr{E}_n(f)| \leqslant \displaystyle\sum_{k=0}^{n-1} |\mathrm{E}_{u_k,u_{k+1}}(f)|$ avec $|\mathrm{E}_{u_k,u_{k+1}}(f)| \leqslant \mathrm{M}_1 \dfrac{(b-a)^2}{2n^2}$ donc $|\mathscr{E}_n(f)| \leqslant \mathrm{M}_1 \dfrac{(b-a)^2}{2n}$. □

3.2 Méthode du point milieu

Una amélioration très simple de la méthode du rectangle consiste à approcher la fonction f par la valeur qu'elle prend au point milieu de l'intervalle $[u,v]$, autrement dit à choisir pour unique nœud $x_0 = \dfrac{u+v}{2} = w$ et pour poids $\alpha_0 = (v-u)$, ce qui conduit à approcher $\mathrm{I}_{u,v}(f)$ par :

$$\mathrm{I}^0_{u,v}(f) = (v-u)f(w)$$

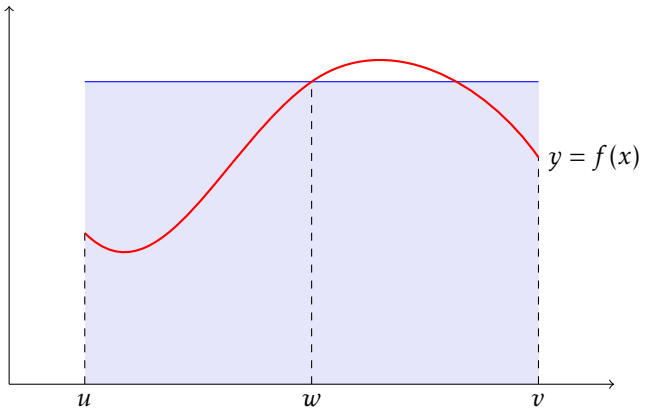


FIGURE 8 – La méthode du point milieu. La valeur de $\mathrm{I}^0_{u,v}(f)$ correspond à l'aire colorée.

THÉORÈME. — *La méthode du point milieu est une méthode d'ordre 1, et si f est de classe $\mathscr{C}^2$, l'erreur de quadrature vérifie :*

$$|\mathrm{E}_{u,v}(f)| \leqslant \mathrm{M}_2 \frac{(v-u)^3}{24}$$

où M_2 est un majorant de $|f''|$ sur l'intervalle $[u,v]$.

Preuve. Si f est un polynôme de degré inférieur ou égal à 1 on pose $f(x) = ax+b$ et on calcule :

$$\mathrm{I}_{u,v}(f) = \int_u^v (at+b)\,\mathrm{d}t = \frac{a}{2}(v^2-u^2) + b(v-u) = (v-u)\left(a\frac{u+v}{2}+b\right) = \mathrm{I}^0_{u,v}(f).$$

Si f est une fonction de classe $\mathscr{C}^2$, l'erreur de quadrature est égale à :

$$\mathrm{E}_{u,v}(f) = \int_u^v f(t)\,\mathrm{d}t - (v-u)f(w).$$

L'astuce consiste ici à écrire $(v-u)f(w)$ sous la forme $\displaystyle\int_u^v \Big(f(w)+(t-w)f'(w)\Big)\mathrm{d}t$ en jouant sur le fait que $\displaystyle\int_u^v (t-w)\,\mathrm{d}t = 0$. On obtient ainsi :

$$\mathrm{E}_{u,v}(f) = \int_u^v \Big(f(t) - f(w) - (t-w)f'(w)\Big)\mathrm{d}t.$$

Fixons alors $t \neq w$ et considérons la fonction $g : x \mapsto f(x) - f(w) - (t-w)f'(w) - \mathrm{K}\dfrac{(x-w)^2}{2}$, la valeur de K étant choisie de sorte que $g(t) = 0$.
On a $g(w) = g(t) = 0$ donc d'après le théorème de ROLLE il existe $c_1 \in]w,t[$ tel que $g'(c_1) = 0$. Mais $g'(x) = f'(x) - f'(w) - \mathrm{K}(x-w)$ donc $g'(w) = 0$; on peut donc de nouveau appliquer le théorème de ROLLE entre c_1 et w et affirmer l'existence d'un réel $c_2 \in]w,c_1[$ tel que $g''(c_2) = 0$, égalité qui s'écrit : $f''(c_2) - \mathrm{K} = 0$.
Ainsi, l'égalité $g(t) = 0$ peut aussi s'écrire $f(t) - f(w) - (t-w)f'(w) = f''(c_2)\dfrac{(t-w)^2}{2}$ ce qui implique :

$$|\mathrm{E}_{u,v}(f)| \leqslant \mathrm{M}_2 \int_u^v \frac{(t-w)^2}{2}\,\mathrm{d}t = \mathrm{M}_2 \frac{(v-u)^3}{24}.$$

□

- **Méthode du point milieu composite**

Elle consiste à appliquer la méthode du point milieu à une subdivision de pas régulier du segment $[a,b]$, ce qui revient à approcher $\mathrm{I}(f)=\displaystyle\int_a^b f(t)\,\mathrm{d}t$ par :

$$\boxed{\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a+\left(k+\frac{1}{2}\right)\frac{b-a}{n}\right).}$$

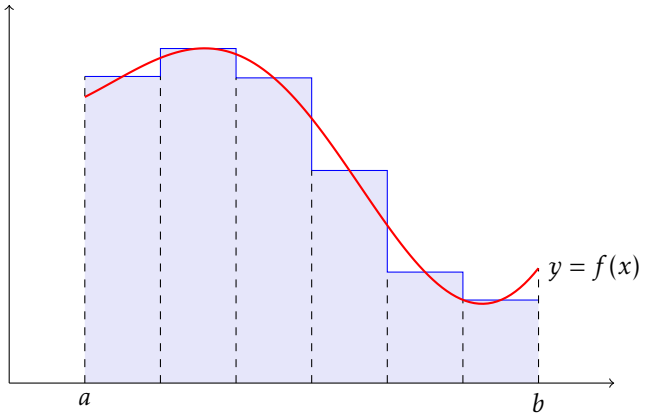


FIGURE 9 – Méthode du point milieu composite.

THÉORÈME. — *Si f est de classe $\mathscr{C}^2$, l'erreur de la méthode du point milieu composite vérifie :*

$$|\mathscr{E}_n(f)|\leqslant \mathrm{M}_2\frac{(b-a)^3}{24n^2}$$

où M_2 est un majorant de $|f''|$ sur $[a,b]$.

Preuve. On a $|\mathscr{E}_n(f)|\leqslant\displaystyle\sum_{k=0}^{n-1}|\mathrm{E}_{u_k,u_{k+1}}(f)|$ avec $|\mathrm{E}_{u_k,u_{k+1}}(f)|\leqslant \mathrm{M}_2\frac{(b-a)^3}{24n^3}$ donc $|\mathscr{E}_n(f)|\leqslant \mathrm{M}_2\frac{(b-a)^3}{24n^2}$. □

3.3 Méthode du trapèze

La méthode du trapèze consiste à approcher sur $[u,v]$ la fonction f par la fonction affine $\tilde{f}$ joignant les points $(u,f(u))$ et $(v,f(v))$. Il est facile d'obtenir l'expression de $\tilde{f}$ sur $[u,v]$:

$$\tilde{f}(x)=f(u)+(x-u)\frac{f(v)-f(u)}{v-u}=\frac{v-x}{v-u}f(u)+\frac{x-u}{v-u}f(v)$$

ce qui conduit à approcher $\mathrm{I}_{u,v}(f)$ par $\mathrm{I}^1_{u,v}(f)=f(u)\displaystyle\int_u^v\frac{v-x}{v-u}\,\mathrm{d}x+f(v)\int_u^v\frac{x-u}{v-u}\,\mathrm{d}x=(v-u)\frac{f(u)+f(v)}{2}$.
Il s'agit donc d'une méthode de quadrature à deux nœuds $x_0=u$ et $x_1=v$ avec les poids $\alpha_0=\alpha_1=\dfrac{v-u}{2}$.

THÉORÈME. — *La méthode du trapèze est une méthode d'ordre 1, et si f est de classe $\mathscr{C}^2$, l'erreur de quadrature vérifie :*

$$|\mathrm{E}_{u,v}(f)|\leqslant \mathrm{M}_2\frac{(v-u)^3}{12}$$

où M_2 est un majorant de $|f''|$ sur l'intervalle $[u,v]$.

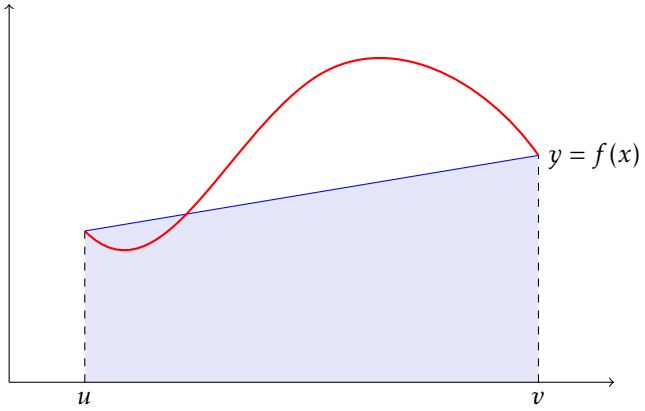


FIGURE 10 – La méthode du trapèze. La valeur de $\mathrm{I}^1_{u,v}(f)$ correspond à l'aire colorée.

Preuve. Si f est un polynôme de degré inférieur ou égal à 1 alors $\tilde{f} = f$ donc $\mathrm{I}_{u,v}(f) = \mathrm{I}^1_{u,v}(f)$; la méthode est bien d'ordre 1.
Si f est de classe $\mathscr{C}^2$, l'erreur de quadrature est égale à :

$$\mathrm{E}_{u,v}(f) = \int_u^v (f(t) - \tilde{f}(t))\,\mathrm{d}t.$$

Fixons $t \in]u,v[$ et considérons la fonction $g : x \mapsto f(x) - \tilde{f}(x) - \mathrm{K}\dfrac{(x-v)(x-u)}{2}$, la constante K étant choisie de sorte que $g(t) = 0$.
On a $g(u) = g(t) = g(v) = 0$ donc d'après le théorème de ROLLE il existe $c_1 \in]u,t[$ et $c_2 \in]t,v[$ tels que $g'(c_1) = g'(c_2) = 0$. Toujours d'après le théorème de ROLLE, il existe $c_3 \in]c_1,c_2[$ tel que $g''(c_3) = 0$. Mais $g''(x) = f''(x) - \mathrm{K}$ donc $\mathrm{K} = f''(c_3)$.
Ainsi, $g(t) = 0 \iff f(t) - \tilde{f}(t) = f''(c_3)\dfrac{(t-v)(t-u)}{2}$ ce qui implique :

$$|\mathrm{E}_{u,v}(f)| \leqslant \mathrm{M}_2 \int_u^v \frac{(v-t)(t-u)}{2}\,\mathrm{d}t = \mathrm{M}_2 \frac{(v-u)^3}{12}.$$

□

• Méthode du trapèze composite

Elle consiste à appliquer la méthode du trapèze à une subdivision de pas régulier du segment $[a,b]$, ce qui revient à approcher $\mathrm{I}(f) = \displaystyle\int_a^b f(t)\,\mathrm{d}t$ par :

$$\frac{b-a}{n}\sum_{k=0}^{n-1}\frac{f(u_k)+f(u_{k+1})}{2} = \frac{b-a}{2n}\left(\sum_{k=0}^{n-1} f(u_k) + \sum_{k=1}^{n} f(u_k)\right) = \frac{b-a}{n}\left(\sum_{k=0}^{n-1} f(u_k) + \frac{f(b)-f(a)}{2}\right)$$

$$= \boxed{\frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a + k\frac{b-a}{n}\right) + \frac{b-a}{n} \times \frac{f(b)-f(a)}{2}.}$$

Autrement dit, la méthode du trapèze apparaît comme une méthode du rectangle à qui on a ajouté un terme correctif $\dfrac{b-a}{n} \times \dfrac{f(b)-f(a)}{2}$. Elle n'en reste pas moins moins intéressante que la méthode du point milieu, puisque l'erreur de la méthode du trapèze composite vérifie :

THÉORÈME. — *Si f est de classe $\mathscr{C}^2$, l'erreur de la méthode du trapèze composite vérifie :*

$$|\mathscr{E}_n(f)| \leqslant \mathrm{M}_2 \frac{(b-a)^3}{12n^2}$$

où M_2 est un majorant de $|f''|$ sur $[a,b]$.

3.4 La méthode de Simpson

Les méthode du rectangle et du point milieu consistent à approcher f sur $[u,v]$ par un polynôme constant ; la méthode du trapèze à approcher f par un polynôme de degré inférieur ou égal à 1. Il est logique de poursuivre cette démarche en approchant maintenant f par un polynôme de degré inférieur ou égal à 2. La théorie de l'*interpolation de* Lagrange que vous étudierez en cours de mathématique prouve l'existence d'une unique fonction polynomiale $\tilde{f}$ de degré inférieur ou égal à 2 vérifiant :

$$\tilde{f}(u)=f(u),\quad \tilde{f}(v)=f(v),\quad \tilde{f}(w)=f(w)\qquad \text{avec}\qquad w=\frac{u+v}{2}.$$

La méthode de quadrature qui en résulte porte le nom de méthode de Simpson ; il s'agit d'une méthode à trois nœuds u, v et w qui consiste à approcher $\mathrm{I}_{u,v}(f)$ par $\displaystyle\int_u^v \tilde{f}(t)\,\mathrm{d}t=\alpha_0 f(u)+\alpha_1 f(v)+\alpha_2 f(w)$.

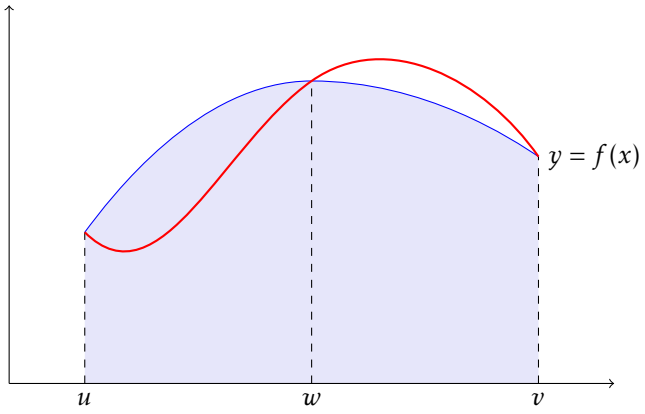


Figure 11 – La méthode de Simpson. La valeur de $\mathrm{I}^2_{u,v}(f)$ correspond à l'aire colorée.

Par construction $\tilde{f}=f$ lorsque f est une fonction polynomiale de degré inférieur ou égal à 2 ; en appliquant la formule de quadrature aux fonctions polynomiales $t\mapsto 1$, $t\mapsto t-u$, $t\mapsto (t-u)(v-t)$ on obtient un système linéaire dont la résolution nous fournit les valeurs des poids associés à cette méthode :

$$\begin{cases} v-u=\alpha_0+\alpha_1+\alpha_2\\ \dfrac{(v-u)^2}{2}=\alpha_1(v-u)+\alpha_2\dfrac{(v-u)}{2}\\ \dfrac{(v-u)^3}{6}=\alpha_2\left(\dfrac{v-u}{2}\right)^2 \end{cases}\iff \begin{cases} \alpha_0=\alpha_1=\dfrac{1}{6}(v-u)\\ \alpha_2=\dfrac{4}{6}(v-u) \end{cases}$$

Ainsi, la méthode de Simpson consiste à approcher $\mathrm{I}_{u,v}(f)$ par :

$$\mathrm{I}^2_{u,v}(f)=\frac{1}{6}(v-u)f(u)+\frac{4}{6}(v-u)f(w)+\frac{1}{6}(v-u)f(v).$$

Théorème. — *La méthode de* Simpson *est une méthode d'ordre 3, et si f est de classe $\mathscr{C}^4$, l'erreur de quadrature vérifie :*

$$|\mathrm{E}_{u,v}(f)|\leqslant \mathrm{M}_4\frac{(v-u)^5}{2880}$$

où M_4 est un majorant de $|f^{(4)}|$ sur l'intervalle $[u,v]$.

Preuve. À première vue, puisque f est approché par un polynôme de Lagrange de degré 2 on devrait plutôt s'attendre à ce que la méthode soit de degré 2. Néanmoins, il se trouve que la formule reste une égalité pour tout polynôme de degré inférieur ou égal à 3. Pour le prouver, il suffit par linéarité de le vérifier pour un seul polynôme de degré 3, par exemple $f(x)=(x-u)^3$:

$$\mathrm{I}_{u,v}(f)=\int_u^v (t-u)^3\,\mathrm{d}t=\frac{1}{4}(v-u)^4$$

$$\text{et}\quad \mathrm{I}^2_{u,v}(f)=\frac{v-u}{6}f(u)+\frac{4(v-u)}{6}f(w)+\frac{v-u}{6}f(v)=0+\frac{1}{12}(v-u)^4+\frac{1}{6}(v-u)^4=\frac{1}{4}(v-u)^4.$$

Si f est de classe $\mathscr{C}^4$, l'erreur de quadrature est égale à :

$$\mathrm{E}_{u,v}(f) = \int_u^v (f(t) - \tilde{f}(t))\,\mathrm{d}t.$$

Tout comme pour la méthode du point milieu, l'astuce consiste ici à considérer non pas la fonction polynomiale $\tilde{f}$ mais une fonction polynomiale p vérifiant : $p(u) = f(u)$, $p(v) = f(v)$, $p(w) = f(w)$ et $p'(w) = f'(w)$. Puisque nous venons de constater que la méthode est de degré 3 l'erreur de quadrature s'écrit encore :

$$\mathrm{E}_{u,v}(f) = \int_u^v (f(t) - p(t))\,\mathrm{d}t.$$

Fixons $t \in]u,v[$ et considérons la fonction $g : x \mapsto f(x) - p(x) - \mathrm{K}\dfrac{(x-v)(x-u)(x-w)^2}{24}$, la constante K étant choisie de sorte que $g(t) = 0$.
On a $g(u) = g(v) = g(w) = g(t) = 0$ donc d'après le théorème de Rolle la fonction g' possède trois racines entre ces différentes valeurs. De plus, il se trouve que w est encore racine de g' ; la fonction g' possède donc au moins quatre racines, et par application successives du théorème de Rolle on en déduit que g'' possède au moins trois racines, $g^{(3)}$ au moins deux racines et $g^{(4)}$ au moins une racine $c \in]u,v[$.
Sachant que $g^{(4)}(c) = f^{(4)}(c) - \mathrm{K}$ on en déduit que $\mathrm{K} = f^{(4)}(c)$ et que l'égalité $g(t) = 0$ peut s'écrire : $f(t) - p(t) = f^{(4)}(c)\dfrac{(t-v)(t-u)(t-w)^2}{24}$. Ainsi, l'erreur de quadrature vérifie :

$$|\mathrm{E}_{u,v}(f) \leqslant \mathrm{M}_4 \int_u^v \frac{(v-t)(t-u)(t-w)^2}{24}\,\mathrm{d}t = \mathrm{M}_4 \frac{(v-u)^5}{2880}.$$

□

• Méthode de Simpson composite

Elle consiste à appliquer la méthode de Simpson à une subdivision de pas régulier du segment $[a,b]$, ce qui revient à approcher $\mathrm{I}(f) = \displaystyle\int_a^b f(t)\,\mathrm{d}t$ par :

$$\boxed{\frac{b-a}{n}\sum_{k=0}^{n-1}\left(\frac{1}{6}f(u_k) + \frac{4}{6}f\left(\frac{u_k+u_{k+1}}{2}\right) + \frac{1}{6}f(u_{k+1})\right).}$$

Théorème. — *Si f est de classe $\mathscr{C}^4$, l'erreur de la méthode de* Simpson *composite vérifie :*

$$|\mathscr{E}_n(f)| \leqslant \mathrm{M}_4 \frac{(b-a)^5}{2280n^4}$$

où M_4 est un majorant de $|f^{(4)}|$ sur $[a,b]$.

3.5 Méthodes de Newton-Côtes

Les méthodes que nous venons d'étudier sont des cas particuliers des méthodes de quadrature de Newton-Côtes basées sur l'interpolation de Lagrange à nœuds équirépartis dans l'intervalle $[u,v]$. On distingue :

- les formules fermées pour lesquelles les extrémités de l'intervalle $[u,v]$ font partie des nœuds, et parmi lesquelles se trouvent les méthodes du trapèze et de Simpson ;
- les formules ouvertes pour lesquelles les extrémités de l'intervalle ne font pas partie des nœuds. C'est le cas de la méthode du point milieu.

On peut montrer que ces méthodes à $p+1$ nœuds sont :

- d'ordre p lorsque p est impair (c'est le cas de la méthode des trapèzes) ;
- d'ordre $p+1$ lorsque p est pair (c'est le cas des méthodes du point milieu et de Simpson)

ce qui explique pourquoi la méthode des trapèzes n'est pas meilleure que la méthode du point milieu.

Pour $p = 4$ on obtient la méthode de Villarceau ; pour $p = 6$ la méthode de Hardy. Néanmoins, bien que théoriquement plus efficaces que la méthode de Simpson, ces méthodes ne sont que peu ou pas utilisées car plus le nombre de points d'interpolation augmente, plus les calculs à effectuer pour établir la formule sont nombreux ; or qui dit calcul sur les nombres flottants dit erreurs d'approximations. Autrement dit, le gain théorique que l'on pourrait espérer obtenir en appliquant ces méthodes se trouve contrebalancé par la succession des erreurs d'approximation sur les nombres flottants. Ainsi, parmi toutes ces méthodes bases sur l'interpolation de Lagrange la méthode de Simpson apparaît comme un bon compromis entre efficacité théorique et pratique.

3.6 Méthodes de Gauss

Les méthodes de Gauss utilisent une subdivision particulière de l'intervalle $[u, v]$ où les points x_i sont racines d'une certaine famille de polynômes et ne sont pas régulièrement espacés, contrairement aux méthodes précédentes. Les méthodes de Gauss sont les méthodes les plus répandues et les plus précises, car ces méthodes sont d'ordre $2p+1$, contre p ou $p+1$ pour les méthodes de Newton-Côtes. Nous n'en dirons pas plus car ces méthodes sont hors programme.

• La fonction quad du module scipy.integrate

Le module `scipy.integrate` contient une fonction nommée `quad` qui permet le calcul approché d'une intégrale. Cette fonction, très efficace, utilise différentes routines (la plupart sont basées sur des méthodes de Gauss) de manière à obtenir la meilleure approximation possible. Dans son utilisation la plus simple, cette fonction prend trois paramètres : f, a et b et renvoie un tuple (I, e) où I est une valeur approchée de l'intégrale $\mathrm{I}(f) = \int_a^b f(t)\,\mathrm{d}t$ et e une estimation de l'erreur $|\mathrm{I}(f) - \mathrm{I}|$.
On notera que les valeurs a et b peuvent éventuellement être égales à $\pm\infty$.

```
>>> import numpy as np
>>> from scipy.integrate import quad

>>> quad(lambda x: np.sin(x), 0, np.pi)
(2.0, 2.220446049250313e-14)

>>> quad(lambda x: np.exp(-x*x), -np.inf, np.inf)
(1.7724538509055159, 1.4202636780944923e-08)
```

Le premier exemple calcule une valeur approchée de l'intégrale $\int_0^\pi \sin(t)\,\mathrm{d}t = 2$; le second une valeur approchée de l'intégrale de Gauss $\int_{-\infty}^{+\infty} \mathrm{e}^{-t^2}\,\mathrm{d}t = \sqrt{\pi}$.

```
>>> np.sqrt(np.pi)
1.7724538509055159
```