L'intégration numérique

Ahmed Ammar (ahmed.ammar@fst.utm.tn)

Institut Préparatoire aux Études Scientifiques et Techniques, Université de Carthage.

Jan 4, 2020

Contents

1 Introduction

L'intégration numérique est un chapitre important de l'analyse numérique et un outil indispensable en physique numérique. On intègre numériquement dans deux cas principaux:

- on ne peut pas intégrer analytiquement,
- l'intégrande est fourni non pas sous la forme d'une fonction mais de tableaux de mesures, cas d'ailleurs le plus fréquent dans la vraie vie.

Les méthodes numériques d'intégration d'une fonction sont nombreuses et les techniques très diverses. Des très simples, comme la méthode des rectangles aux très complexes comme certaines variétés de la méthode de Monte-Carlo.

2 Idées de base de l'intégration numérique

Nous considérons l'intégrale

$$\int_a^b f(x)dx \tag{1}$$

La plupart des méthodes numériques de calcul de cette intégrale divisent l'intégrale d'origine en une somme de plusieurs intégrales, chacune couvrant une partie plus petite de l'intervalle d'intégration d'origine $[a, b]$. Cette réécriture de l'intégrale est basée sur une sélection de points d'intégration x_i, $i = 0, 1, \ldots, n$ qui sont répartis sur l'intervalle $[a, b]$. Les points d'intégration peuvent ou non être répartis uniformément. Une distribution uniforme simplifie les expressions et est souvent suffisante, nous nous limiterons donc principalement à ce choix. Les points d'intégration sont ensuite calculés comme:

$$x_i = a + ih, \quad i = 0, 1, \ldots, n \tag{2}$$

où

$$h = \frac{b-a}{n} \tag{3}$$

Compte tenu des points d'intégration, l'intégrale d'origine est réécrite sous la forme d'une somme d'intégrales, chaque intégrale étant calculée sur le sous-intervalle entre deux points d'intégration consécutifs. L'intégrale dans (1) est donc exprimée comme:

$$\int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx \tag{4}$$

Notez que $x_0 = a$ et $x_n = b$.

En partant de (4), les différentes méthodes d'intégration différeront dans la façon dont elles approchent chaque intégrale du côté droit. L'idée fondamentale est que chaque terme est une intégrale sur un petit intervalle $[x_i, x_{i+1}]$, et sur ce petit intervalle, il est logique d'approximer f par une forme simple, disons une constante, une ligne droite ou une parabole , que nous pouvons facilement intégrer à la main. Les détails deviendront clairs dans les exemples à venir.

2.1 Exemple de calcul

Pour comprendre et comparer les méthodes d'intégration numérique, il est avantageux d'utiliser une intégrale spécifique pour les calculs et les illustrations graphiques. En particulier, nous voulons utiliser une intégrale que nous pouvons calculer à la main de sorte que la précision des méthodes d'approximation puisse être facilement évaluée. Notre intégrale spécifique est tirée de la physique de base.

Supposons que vous accélérez votre voiture du repos et demandez-vous jusqu'où vous allez en T secondes. La distance est donnée par l'intégrale $\int_0^T v(t)dt$, où $v(t)$ est la vitesse en fonction du temps. Une fonction de vitesse en augmentation rapide pourrait être:

$$v\left(t\right) = 3t^2 e^{t^3} \tag{5}$$

La distance après une seconde est

$$\int_0^1 v(t)dt \tag{6}$$

qui est l'intégrale que nous cherchons à calculer par des méthodes numériques. Heureusement, l'expression choisie de la vitesse a une forme qui permet de calculer facilement l'anti-dérivé comme

$$V(t) = e^{t^3} - 1 \tag{7}$$

Nous pouvons donc calculer la valeur exacte de l'intégrale comme $V(1) - V(0) \approx 1.718$ (arrondi à 3 décimales pour plus de commodité).

3 La règle trapézoïdale composite

L'intégrale $\int_a^b f(x)dx$ peut être interprété comme l'aire entre l'axe des x et le graphique $y = f(x)$ de fonction à intégrer. La figure 1 illustre cette zone de choix (6). Le calcul de l'intégrale $\int_0^1 f(t)dt$ revient à calculer l'aire de la zone hachurée.

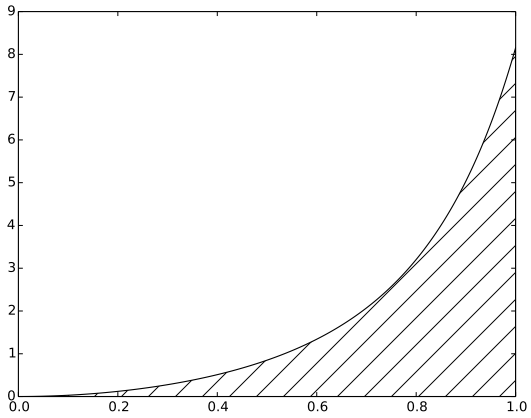

Figure 1: L'intégrale de $v(t)$ interprétée comme l'aire sous le graphique de v.

Si nous remplaçons le vrai graphique de la figure 1 par un ensemble de segments de ligne droite, nous pouvons voir la zone plutôt comme composée de

trapèzes, dont les zones sont faciles à calculer. Ceci est illustré sur la figure 2, où 4 segments de ligne droite donnent naissance à 4 trapèzes, couvrant les intervalles de temps $[0, 0.2)$, $[0.2, 0.6)$, $[0.6, 0.8)$ et $[0.8, 1.0]$. Notez que nous en avons profité pour démontrer les calculs avec des intervalles de temps de tailles différentes.

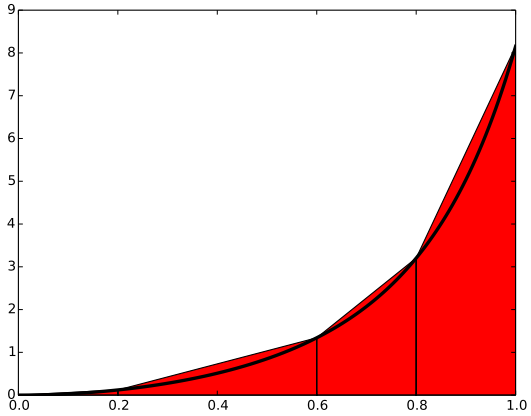

Figure 2: Calculer approximativement l'intégrale d'une fonction comme la somme des aires des trapèzes.

Les aires des 4 trapèzes représentés sur la figure 2 constituent maintenant notre approximation de l'intégrale (6):

$$\int_0^1 v(t)dt \approx h_1(\frac{v(0)+v(0.2)}{2}) + h_2(\frac{v(0.2)+v(0.6)}{2}) + h_3(\frac{v(0.6)+v(0.8)}{2}) + h_4(\frac{v(0.8)+v(1.0)}{2}) \tag{8}$$

où

$$h_1 = (0.2 - 0.0) \tag{9}$$
$$h_2 = (0.6 - 0.2) \tag{10}$$
$$h_3 = (0.8 - 0.6) \tag{11}$$
$$h_4 = (1.0 - 0.8) \tag{12}$$

Avec $v(t) = 3t^2e^{t^3}$, chaque terme dans (8) est facilement calculé et notre calcul approximatif donne

$$\int_0^1 v(t)dt \approx 1.895 \tag{13}$$

Par rapport à la vraie réponse de 1.718, cela est d'environ 10%. Cependant, notez que nous avons utilisé seulement 4 trapèzes pour approximer la zone. Avec

plus de trapèzes, l'approximation serait devenue meilleure, puisque les segments de droite du côté trapézoïdal supérieur suivraient alors le graphique de plus près. Faire un autre calcul avec plus de trapèzes n'est pas trop tentant pour un humain paresseux, mais c'est un travail parfait pour un ordinateur! Dérivons donc les expressions d'approximation de l'intégrale par un nombre arbitraire de trapèzes.

3.1 La formule générale

Pour une fonction donnée $f(x)$, nous voulons approximer l'intégrale $\int_a^b f(x)dx$ par n trapézoïdes (de largeur égale). Nous commençons par (3.5) et approchons chaque intégrale du côté droit avec un seul trapèze. En détail,

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx, \\ &\approx h\frac{f(x_0)+f(x_1)}{2} + h\frac{f(x_1)+f(x_2)}{2} + \ldots + \\ &\quad h\frac{f(x_{n-1})+f(x_n)}{2}\end{aligned} \tag{14}$$

En simplifiant le côté droit de (3.15), nous obtenons

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \ldots + 2f(x_{n-1}) + f(x_n)\right] \tag{15}$$

qui est écrit de façon plus compacte comme

$$\int_a^b f(x)\,dx \approx h\left[\frac{1}{2}f(x_0) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(x_n)\right] \tag{16}$$

Règles d'intégration composites
Le mot composite est souvent utilisé lorsqu'une méthode d'intégration numérique est appliquée avec plus d'un sous-intervalle. à vrai dire alors, écrire, par exemple, "la méthode trapézoïdale", devrait impliquer l'utilisation d'un seul trapèze, tandis que "la méthode trapézoïdale composite" est le nom le plus correct lorsque plusieurs trapèzes sont utilisés. Cependant, cette convention de dénomination n'est pas toujours suivie, donc dire que "la méthode trapézoïdale" peut pointer vers un seul trapèze ainsi que la règle composite avec de nombreux trapèzes.

3.2 Implémentation

Implémentation spécifique ou générale? Supposons que notre objectif principal était de calculer l'intégrale spécifique $\int_0^1 v(t)dt$ avec $v(t) = 3t^2e^{t^3}$. D'abord, nous avons joué avec un simple calcul de main pour voir de quoi il

s'agissait, avant de développer (comme c'est souvent le cas en mathématiques) une formule générale (16) pour l'intégrale générale ou «abstraite» $\int_a^b f(x)dx$. Pour résoudre notre problème spécifique $\int_0^1 v(t)dt$, nous devons ensuite appliquer la formule générale (16) aux données données (fonction et limites intégrales) dans notre problème. Bien que simples en principe, les étapes pratiques sont déroutantes pour beaucoup car la notation dans le problème abstrait de (16) diffère de la notation dans notre problème spécial. Clairement, les f, x et h dans (16) correspondent à v, t et peut-être Δt pour la largeur trapézoïdale dans notre problème spécial.

Le dilemme du programmeur

1. Faut-il écrire un programme spécial pour l'intégrale spéciale, en utilisant les idées de la règle générale (16), mais en remplaçant f par v, x par t et h par Δt?

2. Faut-il implémenter la méthode générale (16) telle qu'elle se présente dans une fonction générale `trapezoid(f, a, b, n)` et résoudre le problème spécifique en question par un appel spécialisé à cette fonction?

L'alternative 2 est toujours le meilleur choix!

La première alternative dans l'encadré ci-dessus semble moins abstraite et donc plus attrayante pour beaucoup. Néanmoins, comme nous l'espérons, cela sera évident à partir des exemples, la deuxième alternative est en fait la plus simple et la plus fiable d'un point de vue mathématique et de programmation. Ces auteurs affirmeront que la deuxième alternative est l'essence même du pouvoir des mathématiques, tandis que la première alternative est la source de beaucoup de confusion sur les mathématiques!

Implémentation avec fonctions. Pour l'intégrale $\int_a^b f(x)dx$ calculée par la formule (16), nous voulons que le trapèze de la fonction Python correspondante prenne tout f, a, b et n en entrée et renvoie l'approximation à l'intégrale.

Nous écrivons une fonction Python `trapezoidal` dans un fichier "trapezoidal.py":"" aussi proche que possible de la formule (16), en nous assurant que les noms de variables correspondent à la notation mathématique:

```
def trapezoidal(f, a, b, n):
    h = float(b-a)/n
    result = 0.5*f(a) + 0.5*f(b)
    for i in range(1, n):
        result += f(a + i*h)
    result *= h
    return result
```

Résoudre notre problème spécifique en une session. Le simple fait d'avoir la fonction `trapezoidal` comme seul contenu d'un fichier trapezoidal.py fait automatiquement de ce fichier un module que nous pouvons importer et tester dans une session interactive:

```
>>> from trapezoidal import trapezoidal
>>> from math import exp
>>> v = lambda t: 3*(t**2)*exp(t**3)
>>> n = 4
>>> numerical = trapezoidal(v, 0, 1, n)
>>> numerical
1.9227167504675762
```

Calculons l'expression exacte et l'erreur dans l'approximation:

```
>>> V = lambda t: exp(t**3)
>>> exact = V(1) - V(0)
>>> exact - numerical
-0.20443492200853108
```

Cette erreur est-elle convaincante? On peut essayer un n plus grand:

```
>>> numerical = trapezoidal(v, 0, 1, n=400)
>>> exact - numerical
-2.1236490512777095e-05
```

Heureusement, beaucoup plus de trapèzes donnent une erreur beaucoup plus petite.

Résoudre notre problème spécifique dans un programme. Au lieu de calculer notre problème spécial dans une session interactive, nous pouvons le faire dans un programme. Comme toujours, un morceau de code faisant une chose particulière est mieux isolé en tant que fonction même si nous ne voyons aucune raison future d'appeler la fonction plusieurs fois et même si nous n'avons pas besoin d'arguments pour paramétrer ce qui se passe à l'intérieur de la fonction. Dans le cas présent, nous mettons simplement les instructions que nous aurions autrement mises dans un programme principal, à l'intérieur d'une fonction:

```
def application():
    from math import exp
    v = lambda t: 3*(t**2)*exp(t**3)
    n = input('n: ')
    numerical = trapezoidal(v, 0, 1, n)

    # Compare with exact result
    V = lambda t: exp(t**3)
    exact = V(1) - V(0)
    error = exact - numerical
    print('n=%d: %.16f, error: %g' % (n, numerical, error))
```

Maintenant, nous calculons notre problème spécial en appelant `application()` comme la seule instruction du programme principal. Les fonctions `trapezoidal`

et `application` résident toutes deux dans le fichier `trapezoidal.py`, qui peut être exécuté en tant que

```
Terminal> python trapezoidal.py
n: 4
n=4: 1.9227167504675762, error: -0.204435
```

Faire un module. Lorsque nous avons les différentes parties de notre programme comme une collection de fonctions, il est très simple de créer un *module* qui peut être importé dans d'autres programmes. Ce fait, avoir notre code comme module, signifie que la fonction `trapezoidal` peut facilement être réutilisée par d'autres programmes pour résoudre d'autres problèmes. Les exigences d'un module sont simples: mettez tout à l'intérieur des fonctions et laissez les appels de fonction dans le programme principal être dans le soi-disant *bloc de test*:

```
if __name__ == '__main__':
    application()
```

Le test `if` est vrai si le fichier de module, `trapezoidal.py`, est exécuté en tant que programme et faux si le module est importé dans un autre programme. Par conséquent, lorsque nous effectuons une importation: `from trapezoidal import trapezoidal` dans un fichier, le test échoue et `application()` n'est pas appelée, c'est-à-dire que notre problème spécial n'est pas résolu et n'imprime rien à l'écran. D'un autre côté, si nous exécutons `trapezoidal.py` dans la fenêtre du terminal, la condition de test est positive, `application()` est appelée et nous obtenons une sortie dans la fenêtre:

```
Terminal> python trapezoidal.py
n: 400
n=400: 1.7183030649495579, error: -2.12365e-05
```

Lorsque vous regardez maintenant les deux solutions, le programme plat spécial et le programme basé sur les fonctions avec la fonction polyvalente `trapezoidal`, vous vous rendez compte que *la mise en œuvre d'un algorithme mathématique général dans une fonction générale* nécessite une réflexion un peu plus abstraite, mais le code résultant peut être utilisé encore et encore. Essentiellement, si vous appliquez le style plat spécial, vous devez retester l'implémentation de l'algorithme après chaque changement de programme.

4 La méthode du point milieu composite

4.1 L'idée

Plutôt que d'approximer l'aire sous une courbe par des trapèzes, nous pouvons utiliser des rectangles simples. Il peut sembler moins précis d'utiliser des lignes horizontales et non des lignes obliques suivant la fonction à intégrer, mais une

méthode d'intégration basée sur des rectangles (la méthode du point milieu) est en fait légèrement plus précise que celle basée sur des trapèzes!

Dans la méthode du milieu, nous construisons un rectangle pour chaque sous-intervalle où la hauteur est égale à f au milieu du sous-intervalle. Faisons-le pour quatre rectangles, en utilisant les mêmes sous-intervalles que nous avions pour les calculs manuels avec la méthode trapézoïdale: $[0, 0.2)$, $[0.2, 0.6)$, $[0.6, 0.8)$ et $[0.8, 1.0]$. On a

$$\int_0^1 f(t)dt \approx h_1 f\left(\frac{0+0.2}{2}\right) + h_2 f\left(\frac{0.2+0.6}{2}\right) + h_3 f\left(\frac{0.6+0.8}{2}\right) + h_4 f\left(\frac{0.8+1.0}{2}\right) \tag{17}$$

où h_1, h_2, h_3 et h_4 sont les largeurs des sous-intervalles, utilisées précédemment avec la méthode du trapèze et définies dans (9)-(12).

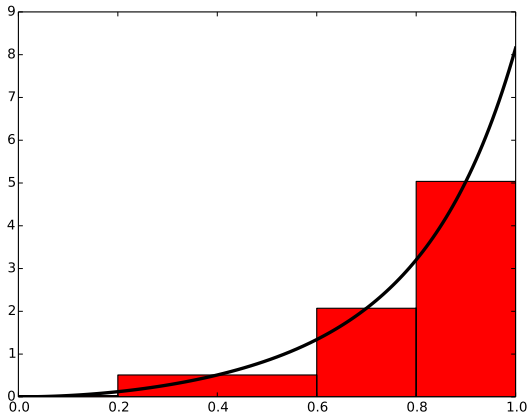

Figure 3: Calcul approximatif de l'intégrale d'une fonction comme la somme des aires des rectangles.

Avec $f(t) = 3t^2e^{t^3}$, l'approximation devient 1.632. Comparé à la vraie réponse (1.718), c'est environ 5% trop petit, mais c'est mieux que ce que nous avons obtenu avec la méthode trapézoïdale (10%) avec les mêmes sous-intervalles. Plus de rectangles donnent une meilleure approximation.

4.2 La formule générale

Dérivons une formule pour la méthode du milieu basée sur n rectangles d'égale largeur:

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx, \\ &\approx hf\left(\frac{x_0+x_1}{2}\right) + hf\left(\frac{x_1+x_2}{2}\right) + \ldots + hf\left(\frac{x_{n-1}+x_n}{2}\right) \quad (18) \\ &\approx h\left(f\left(\frac{x_0+x_1}{2}\right) + f\left(\frac{x_1+x_2}{2}\right) + \ldots + f\left(\frac{x_{n-1}+x_n}{2}\right)\right) \quad (19)\end{aligned}$$

Cette somme peut être écrite de façon plus compacte comme

$$\int_a^b f(x)dx \approx h\sum_{i=0}^{n-1} f(x_i) \quad (20)$$

où $x_i = \left(a + \frac{h}{2}\right) + ih$.

4.3 Implémentation

Nous suivons les conseils et les enseignements tirés de la mise en œuvre de la méthode trapézoïdale et réalisons une fonction `midpoint(f, a, b, n)` (dans un fichier `midpoint.py`) pour mettre en œuvre la formule générale (20):

```
def midpoint(f, a, b, n):
    h = float(b-a)/n
    result = 0
    for i in range(n):
        result += f((a + h/2.0) + i*h)
    result *= h
    return result
```

Nous pouvons tester la fonction comme nous l'avons expliqué pour la méthode du trapèze similaire. L'erreur dans notre problème particulier $\int_0^1 3t^2e^{t^3}\,dt$ avec quatre intervalles est maintenant d'environ 0.1 contrairement à 0.2 pour la règle du trapèze. Les différences sont rarement d'une importance pratique, et sur un ordinateur portable, nous pouvons facilement utiliser $n = 10^6$ et obtenir la réponse avec une erreur d'environ 10^{-12} en quelques secondes.

4.4 Comparaison des méthodes du trapèze et du point milieu

L'exemple suivant montre la facilité avec laquelle nous pouvons combiner les fonctions `trapezoidal` et `midpoint` pour comparer les deux méthodes dans le fichier `compare_integration_methods.py`:

```
from trapezoidal import trapezoidal
from midpoint import midpoint
from math import exp

g = lambda y: exp(-y**2)
a = 0
b = 2
print('    n        midpoint          trapezoidal')
for i in range(1, 21):
    n = 2**i
    m = midpoint(g, a, b, n)
    t = trapezoidal(g, a, b, n)
    print('%7d %.16f %.16f' % (n, m, t))
```

Notez les efforts mis en forme agréable - la sortie devient

```
n        midpoint          trapezoidal
      2 0.8842000076332692 0.8770372606158094
      4 0.8827889485397279 0.8806186341245393
      8 0.8822686991994210 0.8817037913321336
     16 0.8821288703366458 0.8819862452657772
     32 0.8820933014203766 0.8820575578012112
     64 0.8820843709743319 0.8820754296107942
    128 0.8820821359746071 0.8820799002925637
    256 0.8820815770754198 0.8820810181335849
    512 0.8820814373412922 0.8820812976045025
   1024 0.8820814024071774 0.8820813674728968
   2048 0.8820813936736116 0.8820813849400392
   4096 0.8820813914902204 0.8820813893068272
   8192 0.8820813909443684 0.8820813903985197
  16384 0.8820813908079066 0.8820813906714446
  32768 0.8820813907737911 0.8820813907396778
 131072 0.8820813907631487 0.8820813907610036
 262144 0.8820813907625702 0.8820813907620528
 524288 0.8820813907624605 0.8820813907623183
1048576 0.8820813907624268 0.8820813907623890
```

Une inspection visuelle des chiffres montre à quelle vitesse les chiffres se stabilisent dans les deux méthodes. Il semble que 13 chiffres se soient stabilisés dans les deux dernières lignes.

Remarque
Les méthodes trapézoïdale et médiane ne sont que deux exemples dans une jungle de règles d'intégration numérique. D'autres méthodes célèbres sont la règle de Simpson et la quadrature de Gauss. Ils fonctionnent tous de la même manière:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} w_i f(x_i)$$

Autrement dit, l'intégrale est approximée par une somme d'évaluations de fonctions, où chaque évaluation $f(x_i)$ reçoit un poids w_i. Les différentes méthodes diffèrent par la façon dont elles construisent les points d'évaluation

x_i et les poids w_i. Nous avons utilisé des points x_i également espacés, mais une précision plus élevée peut être obtenue en optimisant l'emplacement de x_i.

5 Intégrales doubles et triples

5.1 La règle du point milieu pour une double intégrale

Étant donné une intégrale double sur un domaine rectangulaire $[a, b] \times [c, d]$,

$$\int_a^b \int_c^d f(x, y) dy dx$$

comment approcher cette intégrale par des méthodes numériques?

Dérivation via des intégrales unidimensionnelles. Puisque nous savons comment traiter les intégrales dans une variable, une approche fructueuse consiste à considérer l'intégrale double comme deux intégrales, chacune dans une variable, qui peut être approximée numériquement par les formules unidimensionnelles précédentes. À cette fin, nous introduisons une fonction d'aide $g(x)$ et écrivons

$$\int_a^b \int_c^d f(x, y) dy dx = \int_a^b g(x) dx, \quad g(x) = \int_c^d f(x, y) dy$$

Chacune des intégrales

$$\int_a^b g(x) dx, \quad g(x) = \int_c^d f(x, y) dy$$

peut être discrétisé par n'importe quelle règle d'intégration numérique pour une intégrale dans une variable. Utilisons la méthode du point médian (22) et commençons par $g(x) = \int_c^d f(x, y) dy$. Nous introduisons n_y intervalles sur $[c, d]$ de longueur h_y. La règle médiane pour cette intégrale devient alors

$$g(x) = \int_c^d f(x, y) dy \approx h_y \sum_{j=0}^{n_y - 1} f(x, y_j), \quad y_j = c + \frac{1}{2} h_y + j h_y$$

L'expression semble quelque peu différente de (22), mais c'est à cause de la notation: puisque nous nous intégrons dans la direction y et que nous devrons travailler avec x et y comme coordonnées, nous devons utiliser n_y pour n, h_y pour h et le compteur i est plus naturellement appelé j lors de l'intégration dans y. Les intégrales dans la direction x utiliseront h_x et n_x pour h et n, et i comme compteur.

L'intégrale double est $\int_a^b g(x)dx$, qui peut être approximée par la méthode du point médian:

$$\int_a^b g(x)dx \approx h_x \sum_{i=0}^{n_x-1} g(x_i), \quad x_i = a + \frac{1}{2}h_x + ih_x$$

En rassemblant les formules, nous arrivons à la méthode du point milieu composite pour une double intégrale:

$$\begin{aligned}\int_a^b \int_c^d f(x,y)dydx &\approx h_x \sum_{i=0}^{n_x-1} h_y \sum_{j=0}^{n_y-1} f(x_i, y_j) \\ &= h_x h_y \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} f(a + \frac{h_x}{2} + ih_x, c + \frac{h_y}{2} + jh_y) \qquad (21)\end{aligned}$$

La formule (21) peut également être dérivée directement dans le cas bidimensionnel en appliquant l'idée de la méthode du point milieu. Nous divisons le rectangle $[a,b] \times [c,d]$ en $nx \times ny$ cellules de taille égale. L'idée de la méthode du point milieu est d'approximer f par une constante sur chaque cellule et d'évaluer la constante au point médian. La cellule (i,j) occupe la zone

$$[a + ih_x, a + (i+1)h_x] \times [c + jh_y, c + (j+1)h_y],$$

et le milieu est (x_i, y_j) avec

$$x_i = a + ih_x + \frac{1}{2}h_x, \quad y_j = c + jh_y + \frac{1}{2}h_y$$

L'intégrale sur la cellule est donc $h_x h_y f(x_i, y_j)$, et l'intégrale double totale est la somme sur toutes les cellules, qui n'est rien d'autre que la formule (21).

Programmation d'une double somme. La formule (21) implique une double somme, qui est normalement implémentée sous la forme d'une boucle double. Une fonction Python implémentant (21) peut ressembler à

```
def midpoint_double1(f, a, b, c, d, nx, ny):
    hx = (b - a)/float(nx)
    hy = (d - c)/float(ny)
    I = 0
    for i in range(nx):
        for j in range(ny):
            xi = a + hx/2 + i*hx
            yj = c + hy/2 + j*hy
            I += hx*hy*f(xi, yj)
    return I
```

Si cette fonction est stockée dans un fichier de module `midpoint_double.py`, nous pouvons calculer une intégrale, par exemple, $\int_2^3 \int_0^2 (2x+y)dxdy = 9$ dans un shell interactif et démontrer que la fonction calcule le bon nombre:

```
>>> from midpoint_double import midpoint_double1
>>> def f(x, y):
...     return 2*x + y
...
>>> midpoint_double1(f, 0, 2, 2, 3, 5, 5)
9.0
```

Réutilisation du code pour les intégrales unidimensionnelles. Il est très naturel d'écrire une méthode de point médian bidimensionnelle comme nous l'avons fait dans la fonction `midpoint_double1` lorsque nous avons la formule (21). Cependant, nous pourrions également demander, tout comme nous l'avons fait en mathématiques, pouvons-nous réutiliser une implémentation bien testée pour les intégrales unidimensionnelles pour calculer les doubles intégrales? Autrement dit, pouvons-nous utiliser la fonction `midpoint`.

```
def midpoint(f, a, b, n):
    h = float(b-a)/n
    result = 0
    for i in range(n):
        result += f((a + h/2.0) + i*h)
    result *= h
    return result
```

de la section 4.3 "deux fois"? La réponse est oui, si nous pensons comme nous l'avons fait dans les mathématiques: calculer l'intégrale double comme règle du point milieu pour intégrer $g(x)$ et définir $g(x_i)$ en termes d'une règle médiane sur f dans la coordonnée y.

```
def midpoint_double2(f, a, b, c, d, nx, ny):
    def g(x):
        return midpoint(lambda y: f(x, y), c, d, ny)

    return midpoint(g, a, b, nx)
```

L'avantage important de cette implémentation est que nous réutilisons une fonction bien testée pour la règle du point milieu unidimensionnelle standard et que nous appliquons la règle unidimensionnelle exactement comme dans les mathématiques.

Vérification via les fonctions de test.