Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université D'Adrar
Faculté des Sciences et de la Technologie
Département des Mathématiques et Informatique

MÉMOIRE

Pour l'obtention du diplôme de

MASTER

En Mathématiques

Spécialité :

Analyse Fonctionnelle et Applications

Présenté par

ABEKHTI Amal

Thème

Calcul d'Intégral par la Méthode de Monte-Carlo

Soutenu publiquement le 01 /07 /2019
devant le jury composé de :

M. OUAHAB Abdelghani	Professeure	Université d'Adrar	Président
M. GASMI Laid	Maître de conférence B	Université d'Adrar	Rapporteur
M. BOUAZIZ Saïd	Maître assistant A	Université d'Adrar	Examinateur

2018–2019

Remerciement

Je remercie Dieu de m'avoir aidé à accomplir ce travail, puis je veux ex
de soutiens moraux.

J'adre Gasmi Laid qui
a pro
début à la fin de ce travail.

Cous me Ouahab Abdelghani,
qui m'a fait l'honneur de pré
aussi à M. Bouaziz Saïd d'avoir acce
d'examiner mon travail.

Je remercie aussi le
enseigné au long de ma vie scolaire.

J'adre
J'adre
apporté leur aide et qui ont contribué de près
de ce mémoire.

Dédicace

Ce mode

À me

leurs amours, leurs soutiens et leurs encouragement

À me

À tous membre

nom petit et grand. À le

À toute

tous le

Abekhti amal.

Table des matières

Table des figures

Liste des tableaux

Liste des symboles

$\mathbb{R}$:*L'ensemble des nombres réel.*

$\mathbb{N}$:*L'ensemble des nombres entier.*

$I_n(f)$:*La valeur approchée de l'intégrale de la fonction* f.

I_r :*La valeur approchée de l'intégrale par la méthode du Rectangles.*

I_M :*La valeur approchée de l'intégrale par la méthode du Point milieu.*

I_T :*La valeur approchée de l'intégrale par la méthode du Trapèzes.*

I_S :*La valeur approchée de l'intégrale par la méthode du Simpson.*

$\mathbf{R}_n(f)$: *L'erreur de quadrature numérique de la fonction* f.

$\mathbf{R}_r$: *L'erreur de quadrature numérique de la méthode des Rectangles.*

$\mathbf{R}_M$: *L'erreur de quadrature numérique de la méthode de Point milieu.*

$\mathbf{R}_T$: *L'erreur de quadrature numérique de la méthode des Trapèzes.*

$\mathbf{R}_S$: *L'erreur de quadrature numérique de la méthode de Simpson.*

$\mathcal{C}^{n+1}$: *L'ensemble des fonction* $(n+1)$ *fois continument différentiable .*

σ : *L'écart type.*

$Var(X)$: *La variance de la variable aléatoire* X.

$E(X)$: *L'espérance de la variable aléatoire* X.

$\mathcal{U}$: *Loi uniforme.*

$\mathcal{N}$: *Loi gaussienne.*

$Cov(X,Y)$: *La covariance de* X *et* Y .

i.i.d : *Indépendantes identiquement distribuées.*

Introduction Générale

La méthode de Monte-Carlo (MC) a apparu dans la seconde guerre mondiale, particulièrement dans le cadre du projet américain " Manhattan " portant sur le développement de l'arme nucléaire. Cette époque correspond également à la construction des premiers " ordinateurs ". Il est difficile de savoir exactement qui parmi les chercheurs Von Neumann, Ulam, Fermi et Metropolis, leur a donné le nom de Monte-Carlo qui fait référence aux jeux de hasard pratiqués dans la principauté de Monaco, a été inventé en 1947 par Nicholas Metropolis, et publié pour la première fois en 1949 dans un article coécrit avec Stanislaw Ulam [2]. Le terme de Monte-Carlo désigne une famille de méthodes algorithmique. D'un point de vue mathématique un méthode de Monte-Carlo peut servir au calcule d'intégrale(que l'on souhaite explique dans ce mémoire) et à la résolution d'équation aux dérivées partielles, de système linéaire et de problème d'optimisation. Pratiquement ces technique sont couramment utilisées dans divers domaines (physique, ingénierie,...).

Dans ce mémoire, nous avons présenté la méthode de Monte-Carlo du point de vue de calcule d'intégrale. En pratique pour appliquer la méthode, il faut modéliser le problème considéré sous forme de l'intégrale d'une fonction en utilisant les outils du calcul des probabilités puis de la simuler numériquement en l'approchant par la moyenne de la fonction en des points pseudo-aléatoires. Ces points sont engendrés par ordinateur et ont des propriétés semblables à celles des points aléatoires.

À partir de l'estimateur de Monte-Carlo du paramètre étudié, il est possible de construire un intervalle de confiance qui mesure la précision de la méthode par conséquent, pour obtenir une bonne précision, il est nécessaire d'effectuer un grand nombre de simulations [18]. La taille de cet intervalle est de l'ordre de $n^{-1/2}$, pour n points de simulation, et elle est indépendante de la dimension de l'espace, ce qui les rend avantageuses pour des problèmes de dimension élevée, de plus sa vitesse de convergence est en $O\left(\frac{1}{\sqrt{n}}\right)$, n étant le nombre de points de la suite utilisée. Cependant, cette méthode a plusieurs inconvénients ; en particulier, elle ne permet pas d'exploiter les éventuelles propriétés de régularité de la fonction à intégrer et leur taux de convergence est faible, puisque, pour réduire l'erreur d'un facteur de 100, il faut multiplier n par 10^4.

L'objectif est de défini une méthode qui converge plus rapidement que les méthodes d'analyse numérique classiques, la méthode que nous avons traité ici est la méthode de Monte-Carlo, qui a comme avantage attrayant d'exiger moins de temps de calcul, et une convergence plus rapide.

Ce mémoire est constitué d'une introduction générale, des trois chapitres, d'une conclusion et perspectives, d'une bibliographie et d'une annexe.

Le premier chapitre est un rappel des formules de quelques méthodes numérique

(Méthode des Rectangles, Méthode des Trapèzes,Méthode de Point milieu et Méthode de Simpson) ainsi que les proposition fondamentales de l'erreur numérique avec des corollaire et des remarques.

Le deuxième chapitre explique la méthode de Monte-Carlo pour calculer d'intégral par l'espérance dans le cas simple (unidimensionnelle) et compliquée (multidimensionnelles), de plus pour estimer l'erreur numérique de cette méthode on étudie la convergence et la vitesse de convergence avec les deux théorèmes fondamentales ; Théorème de loi forte des grands nombres et Théorème de limite centrale, ainsi on estimer de la variance et on définie la notion " intervalle de confiance ". À la fin de ce chapitre on donne quelques remarques fondamentales.

Dans **le troisième chapitre**, nous allons inclure quelques applications numériques pour calculer l'intégration dans la méthode de Monte-Carlo de plus, on fait des comparaison entre les méthodes de chapitre 1 et la méthode de Monte-Carlo.

On utilisant le logiciel **R.3.5.1** pour l'installer (voir `https://cran.r-project.org/bin/windows/base/old/3.5.1/`) et le logiciel **Anaconda 3.6** pour l'installer (voir `https://www.anaconda.com/distribution/#windows`).

Finalement, dans la conclusion, nous résumons les résultats obtenus et nous donnons des pistes de recherche pour des études futures.

Chapitre 1

Méthodes de résolution d'un intégral

En analyse numérique, il existe toute une famille d'algorithmes permettant d'approcher la valeur numérique d'une intégrale. Toutes consistent à approcher l'intégrale $I = \int f(x)dx$ par une formule dite de quadrature, du type

$$I(f) = \sum_{i=0}^{p} \omega_i f(x_i)$$

Dans le chapitre suivante, on présente quelques méthodes d'intégration numérique par morceaux, on va étudier des propositions sur la valeur approchée et l'erreur d'intégration en utilise les polynôme d'interpolation.

1.1 Position du problème

Nous allons maintenant nous intéressons à l'obtention d'une valeur numérique approchée de l'intégrale définie :

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

où $[a, b]$, est un intervalle de $\mathbb{R}$ et g une fonction réelle intégrable sur l'intervalle $[a, b]$. Supposons que $\forall x \in [a, b]$

$$g(x) = f(x)\omega(x)$$

avec

$$\omega(x) > 0, \quad \forall x \in]a, b[, \quad \text{et} \quad \int_a^b \omega(x)dx < +\infty$$

où f une fonction réelle, $a < b$, ω est une fonction donnée, dite "*fonction poids*", ne s'annulant pas sur l'intervalle $]a, b[$ et intégrable sur $[a, b]$ telle que l'intégrale $I(g)$ existe Donc

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

L'idée de base des méthodes numériques pour résoudre ce problème (qu'on appelle méthodes de quadrature) est de remplacer f que l'on ne sait pas intégrer par son polynôme d'interpolation. On a alors des méthodes de quadrature de type interpolation. Le problème

de l'intégration numérique (ou dite de quadrature) peut se présenter de deux façons différent :
- Une fonction $f(x)$ est connue par quelques-uns de ses points de collocation $\{x_i, f(x_i)\}_{i=0}^n$ (qui sont régulièrement espacés au non). Comment fait-on pour estimer la valeur de l'intégrale

$$\int_{x_0}^{x_n} f(x)dx$$

alors que l'expression analytique de $f(x)$ n'est pas connue ? ou encore, on cherche la valeur de l'intégrale définie $I(f) = \int_a^b f(x)dx$ lorsque l'expression analytique de l'intégrand $f(x)$ est connue, mais non sa primitive.
- On connaît des formules qui permettent de calculer l'intégrale de certaines fonctions, mais, il y a relativement peu de fonctions dont on sait calculer l'intégrale, il suffit même de changer légèrement l'expression d'une fonction pour passer d'une fonction que l'on sait intégrer à une fonction qu'on ne sait pas intégrer.
Ce sont quelques exemples concrets.

Exemple 1.1

1. Comment calculer

$$I_1 = \int_a^b e^{-x^2} dx$$

ou la fonction de répartition de la loi normale $\mathcal{N}(0,1)$

$$I_2 = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$$

qui n'ont pas de primitive explicite ?
2. On sait que

$$\int_a^b \sin x dx = \cos a - \cos b$$

mais on ne sait pas calculer

$$\int_a^b \frac{\sin x}{x} dx \quad ou \quad \int_a^b \sin(x^2) dx$$

or la fonction $x \longmapsto \dfrac{\sin x}{x}$ *est intégrable sur tout intervalle* $[a, b]$ *puisqu'elle est continue sur* $\mathbb{R}$. *Il en est de même pour la fonction* $x \longmapsto sin(x^2)$.

Dans ce cas nous intéressons à utiliser les méthodes de type interpolation. Pour ces méthodes, il y a trois étapes :
- Établi des "formules de quadrature simple" par interpolation.
- Décomposer l'intervalle en intervalles élémentaires.
- En déduire des "formules composites" ; nous interpolerons la fonction $f(x)$ ou ses points de collocation avec sa polynôme P_n d'interpolation, puis nous intégrerons explicitement ce polynôme.

Les avantages des méthodes de type interpolation :

i) Les polynômes sont faciles à intégrer.

ii) Cette méthode est utilisable même si on ne connait que des valeurs de f puis qu'on peut alors construire le polynôme P_n d'interpolation de f sur ces valeurs.

Nous supposons dans ce qui suit que la distance entre points de collocation adjacents est constante et vaut h (le pas).
On a quelque définition importante :

Définition 1.1

i) On appelle base de Lagrange associée au support $\{x_0, ..., x_n\}$ la famille des fonction polynôme $(L_i^{(n)})_{0 \leq iLeqn}$ (que nous noterons simplement L_i s'il n'ya pas de confusion) définie par

$$L_i(x) = \prod_{j=0, j\neq i}^{n} \frac{(x - x_i)}{(x_i - x_j)}, \quad \forall i \in \{0, .., n\}$$

Les polynôme $L_i(x)$ sont appelés polynome de Lagrange.

ii) Soit f un fonction définie sur un intervalle $V = [a, b]$ de $\mathbb{R}$ et $x_0, ..., x_n$ des point distincts de V. On appellera erreur d'interpolation d'ordre n, au point $x \in [a, b]$ dans l'interpolation polynômiale de f par le polynôme $P_n(x)$ le réel $E_n(x)$ définie par :

$$E_n(x) = f(x) - P_n(x).$$

iii) On appelle formule de quadrature à $(n + 1)$ points, la somme finie

$$I_n(f) = \sum_{i=0}^{n} f(x_i) A_i^{(n)}, \tag{1.1}$$

où les A_i^n telle que

$$A_i^{(n)} = \int_a^b L_i(x) w(x) dx \tag{1.2}$$

ne dépendent pas de f et $x_i \in [a, b]$.

iv) L'erreur de quadrature numérique est donnée par

$$R_n(f) := I(f) - \int_a^b P_n(x) w(x) dx = \int_a^b E_n(x) w(x) dx.$$

v) Une méthode d'intégration approchée est dit d'ordre n si $I(f) = I_n(f)$ pour tout $f \in \mathbb{R}_n[x]$ et si $I(f) \neq I_n(f)$ pour au moins un $f \in \mathbb{R}_{n+1}[x]$.

vi) Une formule de quadrature est dit exacte sur un ensemble V si : $\forall f \in V,\ R_n(f) = 0$.

vii) Le n-ième différence divisée de la fonction f sont définies à partir des $(n - 1)$-ième différence divisée comme suit

$$f[x_0, x_1, ..., x_n] = \frac{f[x_1, ..., x_n] - f[x_0, x_1, ..., x_{n-1}]}{x_n - x_0}$$

Lemme 1.1

On a

$$\frac{d}{dx} f[x, x_0, ..., x_n] = \frac{f^{(n+2)}(\xi)}{(n + 2)!} \quad , \xi \in [a, b]$$

On rappelle le théorème suivant dont nous ferons un usage constant et qui est un outil essentiel dans les méthodes d'approximation des intégrales et pour le démontrer il faut présenter ces deux théorème.

Théorème 1.1 (Théorème des valeurs extrêmes[12])
Soit f une fonction définie sur un intervalle réel $[a, b]$ à valeurs réelles. Si f est continue alors, la fonction f est bornée et atteint ses bornes, autrement dit il existe deux réels c et d de l'intervalle $[a, b]$ tels que pour tout x de $[a, b]$, les inégalités suivantes soient vérifiées :

$$f(c) \leq f(x) \leq f(d).$$

Théorème 1.2 (Théorème des valeurs intermédiaires[12])
Pour toute fonction f définie et continue sur un intervalle J et à valeurs réelles, l'image $f(J)$ est un intervalle.
Autrement dit ; pour toute application continue $f : [a, b] \to \mathbb{R}$ et tout réel u compris entre $f(a)$ et $f(b)$, il existe au moins un réel c compris entre a et b tel que $f(c) = u$.

Théorème 1.3 (Second théorème de la moyenne[6])
Soient f et g deux fonctions à valeurs réelles définies et continues sur un intervalle $[a, b]$. On suppose que g ne change pas de signe dans $[a, b]$. Alors, il existe un point ξ dans l'intervalle $[a, b]$ où on a l'égalité

$$\int_a^b f(x)g(x)dx = f(\xi)\int_a^b g(x)dx.$$

Démonstration.
On peut supposer que la fonction g est à valeurs positives ou nulles (quitte à la remplacer par $-g$ si nécessaire).
Cela exclut en particulier que g soit constamment nulle ou que f soit constante.
D'après le théorème (1.1) et le théorème (1.2), l'image par f du segment $[a, b]$ est un segment $[m, M]$ avec $m < M$, et l'image de l'intervalle ouvert $]a, b[$ est un intervalle inclus dans ce segment et qui n'en diffère que par au plus deux points, donc

$$]m, M[\subset f(]a, b[).$$

Puisque g est continue, positive et non constamment nulle, son intégrale sur $[a, b]$ est strictement positive. Pour prouver que

$$\frac{\int_a^b f(x)g(x)\ \mathrm{d}x}{\int_a^b g(x)\ \mathrm{d}x} \in f(]a, b[),$$

il suffit donc de vérifier que

$$m\int_a^b g(x)\ \mathrm{d}x < \int_a^b f(x)g(x)\ \mathrm{d}x < M\int_a^b g(x)\ \mathrm{d}x.$$

Montrons par exemple la première inégalité stricte (le raisonnement pour la seconde est analogue). La fonction $(f - m)g$ étant continue, positive et non constamment nulle, l'application

$$y \mapsto \int_a^y (f(x) - m)g(x)\ \mathrm{d}x$$

est croissante et non constante, si bien que sa valeur en b est strictement supérieure à celle en a i.e.

$$\int_a^y (f(x) - m)g(x)\ \mathrm{d}x < \int_a^b (f(x) - m)g(x)\ \mathrm{d}x$$

Ainsi,

$$m \int_a^b g(x)\ \mathrm{d}x < \int_a^b f(x)g(x)\ \mathrm{d}x,$$

ce qui termine la démonstration. □

1.2 Cadre d'étude

Soit $a, b \in \mathbb{R}, f : [a, b] \longrightarrow \mathbb{R}$ une fonction continue [17], nous intéressons au calcul de l'intégrale

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

Soit $a \leq x_0 < x_1 < ... < x_n \leq b$, $(n+1)$ points distincts pris dans l'intervalle $[a, b]$ et soit P_n l'interpolant de Lagrange de f aux points $(x_i)_{i=0}^n$.
On a donc

$$P_n(x) = \sum_{i=0}^{n} f(x_i)L_i(x), \quad x \in [a, b]$$

où

$$L_i(x) := \frac{(x - x_0)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

Notons

$$w_i := \int_a^b L_i(x)dx.$$

Le calcul des w_i est particulièrement aisé puisqu'il s'agit d'intégrales de polynômes. On peut ainsi "approcher" l'intégrale $\int_a^b f(x)dx$ par

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

est une formule d'intégration numérique pour f sur l'intervalle $[a, b]$.

Remarque 1.1

Si f est un polynôme de degré n, alors $I(f) = \int_a^b f(x)dx$. Nous nous intéresserons donc aux formules d'intégration numérique qui sont exactes pour des polynômes.

Soit $a = x_0 < x_1 < ... < x_{n-1} < x_n = b$, et soit P_n le polynôme d'interpolation de f en ces points. D'après le théorème sur l'obtention du l'erreur d'interpolation de Lagrange[5], on a

$$f(x) = P_n(x) + E_n(x),$$

où $E_n(x)$ est l'erreur d'interpolation en x. Si on intégrant sur $[a, b]$, on obtient :

$$\int_a^b f(x)w(x)dx = \int_a^b P_n(x)w(x)dx + \int_a^b E_n(x)w(x)dx.$$

Par conséquent,

$$I(f) = I_n(f) + R_n(f) \tag{1.3}$$

où $I_n(f)$ définie par (1.1), et

$$R_n(f) = \int_a^b E_n(x)w(x)dx. \tag{1.4}$$

Utilisons maintenant la formule d'interpolation de Lagrange, il vient :

$$\begin{aligned}\int_a^b f(x)w(x)dx &= \int_a^b \sum_{i=0}^{n} f(x_i)w(x)L_i dx + \int_a^b \prod_{i=0}^{n}(x-x_i)w(x)\frac{f^{(n+1)}(\xi(x))}{(n+1)!}dx \\ &= \sum_{i=0}^{n} A_i^n f(x_i) + \frac{1}{(n+1)!}\int_a^b \prod_{i=0}^{n}(x-x_i)w(x)f^{(n+1)}(\xi(x))dx.\end{aligned}$$

La formule de quadrature s'écrit :

$$\int_a^b f(x)w(x)dx \simeq \sum_{i=0}^{n} A_i^n f(x_i) + R_n(f)$$

où l'erreur est donnée par

$$R_n(f) = \frac{1}{(n+1)!}\int_a^b \prod_{i=0}^{n}(x-x_i)w(x)f^{(n+1)}(\xi(x))dx.$$

1.3 Méthodes d'intégration numérique par morceaux

Pour augmenter la précision, nous allons construire une subdivision de l'intervalle $[a,b]$ $a = x_0 < x_1 < ... < x_{k-1} < x_k = b$, en posant $h = \max_{0\le i\le k-1}(x_{i+1} - x_i)$. La précision de la méthode d'intégration numérique ainsi construite peut être déterminée en fonction de h. Dans tout ce qui suite nous supposons que

$$\forall x \in [a,b], \qquad w(x) = 1$$

Fixons à présenter les notations. D'après (1.3)

$$I(f) = \int_a^b f(x)dx = \int_a^b P_n(x)dx + \int_a^b f[x_0,...,x_n,x]\prod_{i=0}^{n}(x-x_i)dx$$

Avec la valeur approchée de $I(f)$ est

$$I_n(f) = \int_a^b P_n(x)dx$$

et l'erreur d'interpolation

$$R_n(f) = \int_a^b f[x_0,...,x_n,x]\prod_{i=0}^{n}(x-x_i)dx$$

Nous alors étudions trois cas particuliers de la formule (1.3) correspondant à $n = 0, 1, 2$.

1.3.1 Méthodes d'ordres plus bas

L'interpolation linéaire quadrature du rectangle et du point milieu

Interpolation par une fonction P_n***, de degré*** $n = 0$
On a le résultat suivant [6] :

Proposition 1

Dans le cas degré $n = 0$*, le support d'interpolation est réduit à* $\{x_0\}$*. Si* f *est de classe* $\mathcal{C}^1$ *sur* $[a, b]$*, la valeur approchée est*

$$I_0(f) = (b - a)f(x_0)$$

et l'erreur d'intégration est

$$R_0(f) = \int_a^b f[x_0, x](x - x_0)dx$$

Démonstration.
Cette méthode utilise le polynôme de degré le plus bas, à savoir le polynôme constant :

$$P_0(x) := f[x_0] = f(x_0)$$

Posons $f_0 = f(x_0)$,alors l'intégrale approchée $I_0(f) = \int_a^b P_0(x)dx$ se calcule alors trivialement et donne :

$$I_0(f) = (b - a)f_0$$

Il s'agit de l'aire du rectangle.

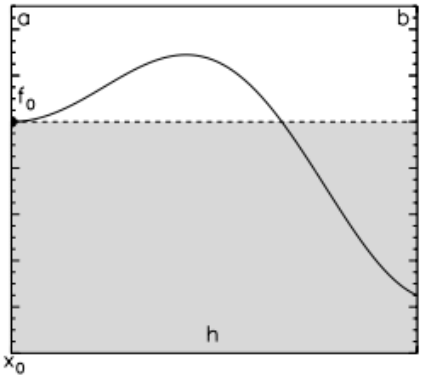


Fig. 1.1 – Formules des rectangles (n=0)

Comme l'application f est continue, le caractère garantit la continuité de l'application $x \longmapsto f[x_0, x]$ et donc l'existence de

$$R_0(f) = \int_a^b f[x_0, x](x - x_0)dx$$

□

En choisissant comme extrémité de l'intervalle $[a, b]$, on obtient le résultat suivant :

Corollaire 1 (Méthode du Rectangles)

Si f est de classe $\mathcal{C}^1([a,b])$ alors, pour $x_0 = a$ la valeur approchée vaut

$$I_r = (b-a)f(a).$$

Cette formule est d'ordre 0. *L'erreur d'intégration vaut*

$$R_r = \frac{(b-a)^2}{2} f^{'}(\xi), \quad \xi \in]a,b[,$$

et, pour $x_0 = b$,

$$I_r = (b-a)f(b)$$

et l'erreur d'intégration vaut

$$R_r = \frac{(b-a)^2}{2} f^{'}(\xi), \quad \xi \in]a,b[,$$

Démonstration.
- Calcul de I_r. Traitons le cas $x_0 = a$ (le cas $x_0 = b$ étant similaire). L'expression de la valeur de I_r provient de la proposition (1) le cas $x_0 = a$ on obtient

$$I_r = I_0(f) = \int_a^b P_0(x)dx = \int_a^b f[a]dx = (b-a)f(a).$$

- Pour calculer l'erreur, on peut utiliser le théorème des accroissements finis [11] :

$$\forall x \in [a,b], \exists\, \xi \in [a,b] \text{ tel que } f(x) = f(a) + (x-a)f^{'}(\xi)$$

En remplaçant dans l'expression de l'intégrale et de l'erreur, on trouve :

$$\begin{aligned} R_r &= \int_a^b f(x)dx - I_0(f) \\ &= \int_a^b (f(x) - P_0(x))dx \\ &= \int_a^b (f(x) - f(a))dx \\ &= \int_a^b (x-a)f^{'}(\xi)dx \\ &= f^{'}(\xi)\int_0^{b-a} x dx \\ &= \frac{(b-a)^2}{2} f^{'}(\xi) \end{aligned}$$

qui est le résultat annoncé. □

L'erreur R_r n'est pas connue car la valeur de $\xi \in]a,b[$ reste indéterminée. Cependant, on peut la majorer par la plus grande valeur de la dérivée sur le domaine considéré, alors pour $h = b - a$:

$$|R_r| \leq \frac{h^2}{2} \sup_{[a,b]}(|f^{'}|)$$

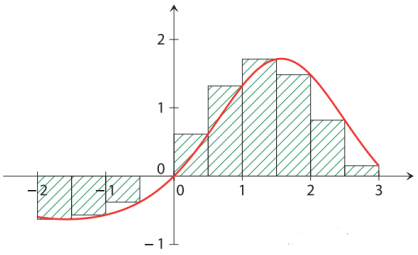


Fig. 1.2 – Méthode des rectangles à droite

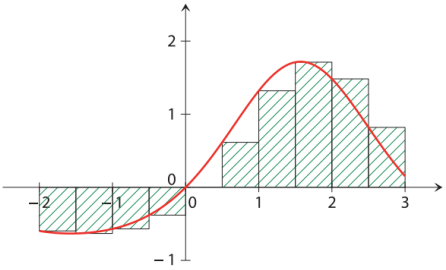


Fig. 1.3 – Méthode des rectangles à gauche

Remarque 1.2

- Cette méthode d'intégration est exacte pour toutes les fonctions f constantes (dans ce cas $R_r = 0$ puisque qu'elles vérifient $f' = 0$). Dans le cas plus général cette méthode est d'autant plus précise que les variations de f sont faibles (f' petit).
- Plus le domaine $[a, b]$ est petit, plus l'erreur est faible. Cette erreur décroit en h^2.

En choisissant pour $x_0 = \dfrac{a+b}{2} = m$, le milieu de l'intervalle $[a, b]$ avec un pas $h = \dfrac{b-a}{2}$, on obtient le résultat suivant :

Corollaire 2 (Méthode du Point milieu)

Si f est de classe $\mathcal{C}^2([a.b])$, alors la valeur approchée vaut

$$I_M = (b-a)f\left(\frac{a+b}{2}\right)$$

Cette méthode est d'aborde 1 *et l'erreur d'intégration vaut*

$$R_M = \frac{(b-a)^3}{24} f''(\xi), \quad \xi \in]a, b[$$

Démonstration.
- Calcul de I_M ; on applique encore la proposition (1) ; d'aù l'expression de I_M.

- La méthode du point milieu est d'ordre 1 ; en effet

$$\begin{aligned} I_r(x) &= \int_a^b x dx \\ &= \frac{b^2 - a^2}{2} \\ &= (b-a)\frac{b+a}{2} = I_0(x) \end{aligned}$$

par contre, on n'a pas $I_r(x^2) = I_0(x^2)$.
Pour calculer l'erreur, on peut utiliser le théorème des accroissements finis[11] au deuxième ordre, $\forall x \in [a,b], \exists\, \xi \in [a,b]$ tel que :

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

En remplaçant dans l'expression de l'intégrale et d'erreur, on trouve :

$$\begin{aligned} R_M &= \int_a^b f(x)dx - I_0(f) \\ &= \int_a^b (f(x) - P_0(x))dx \\ &= \int_a^b (f(x) - f(\frac{a+b}{2}))dx \\ &= \int_a^b \left[(x - \frac{a+b}{2})f'(\frac{a+b}{2}) + (x - \frac{a+b}{2})^2 \frac{f''(\xi)}{2} \right] dx \\ &= f'(\frac{a+b}{2}) \int_{-\frac{b-a}{2}}^{\frac{b-a}{2}} x dx + \frac{f''(\xi)}{2} \int_{-\frac{b-a}{2}}^{\frac{b-a}{2}} x^2 dx \\ &= \frac{f''(\xi)}{3} (\frac{b-a}{2})^3 \end{aligned}$$

□

L'erreur R_M n'est pas connue car la valeur de $\xi \in [a,b]$ reste indéterminée. Cependant, on peut la majorer par la plus grande valeur de la dérivée seconde sur le domaine considéré, alors pour $h = b - a$:

$$|R_M| \leq \frac{h^3}{24} \sup_{[a,b]}(|f''|)$$

Remarque 1.3

-Cette intégrale numérique nécessite une unique évaluation de la fonction f (en $x_0 = \frac{a+b}{2}$) et correspond donc aussi à ce qu'on peut faire de plus rapide.
-Du fait des symétries, cette méthode d'intégration est exacte pour les fonctions f constante, mais aussi pour les fonctions affines (dans ce cas $R_M = 0$ puis qu'elles vérifient $f'' = 0$), dans le cas plus général,cette méthode est d'autant plus précise que les variations de f sont faibles (f'' petit).
-Plus le domaine $[a,b]$ est petit, plus l'erreur est faible. Cette erreur décroit en $\frac{h}{3}$, c'est-à-dire plus vite que l'erreur de la méthode précédente. Ainsi, pour des domaines $[a,b]$ suffisamment petits, la méthode du point milieu est toujours plus précise que la méthode précédente.

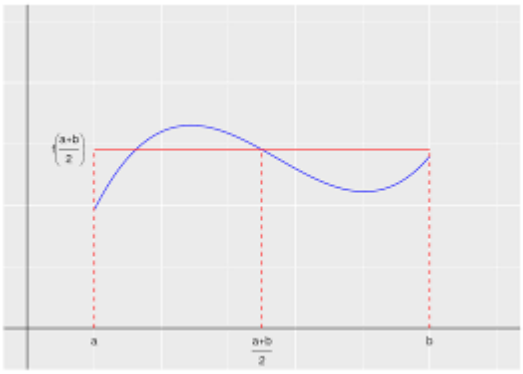


Fig. 1.4 – Méthode du point milieu

L'interpolation linéaire : quadrature du trapèze

Interpolation par une fonction P_n ***de degré*** $n = 1$

On souhaite évaluer

$$\int_{x_0}^{x_n} f(x)dx$$

où $f(x)$ est une fonction connue seulement en deux points, ou encore une fonction n'ayant pas de primitive. La solution qui vient tout de suite à l'esprit consiste a remplacé $f(x)$ par le polynôme degré 1, passant par les points $(x_0, f(x_0))$ et $(x_1, f(x_1))$. La valeur approximative de l'intégrale correspond à l'aire sous la courbe de l'intégrale. Cette aire forme un trapèze qui donne son nom *La méthode du Trapèzes*. Évidemment, l'approximation est grossière et on peut déjà soupçonner que le résultat sera peu précis.

Proposition 2

Dans le cas de degré $n = 1$ *le support d'interpolation est réduit à* $\{x_0, x_1\}$. *Si* f *est de classe* $\mathcal{C}^2$ *sur* $[a, b]$ *la valeur approchée est*

$$I_1(f) = \int_a^b [f[x_0] + f[x_0, x_1](x - x_0)]dx$$

et l'erreur d'intégration est

$$R_1(f) = \int_a^b f[x_0, x_1, x](x - x_0)(x - x_1)dx$$

Démonstration.
Ici $I_1(f) = \int_a^b P_1(x)dx$ avec $P_1(x)$ désigne le polynôme d'interpolation sur $\{x_0, x_1\}$ alors,

$$\begin{aligned} I_1(f) &= \int_a^b P_1(x)dx \\ &= \int_a^b f(x_0)\frac{(x - x_1)}{(x_0 - x_1)} + f(x_1)\frac{(x - x_0)}{(x_1 - x_0)} \\ &= \int_a^b f(x_0) + \frac{f(x_0) - f(x_1)}{(x_0 - x_1)}(x - x_0) \\ &= \int_a^b [f[x_0] + f[x_0, x_1](x - x_0)]\, dx \end{aligned}$$

d'où le résultat.
Le caractère $\mathcal{C}^2$ garantit la continuité de l'application $x \longmapsto (x-x_0)(x-x_1)$. d'où l'existence de $R_1(f)$. □

En choisissant $x_0 = a$ et $x_1 = b$, il vient

Corollaire 3 (Méthode du Trapèzes)

Si f est de classe $\mathcal{C}^2([a,b])$ alors, la valeur approchée vaut

$$I_T = (b-a)\left(\frac{f(a)+f(b)}{2}\right) \tag{1.5}$$

cette méthode est d'ordre 1.
L'erreur d'intégration vaut

$$R_T = \frac{(b-a)^3}{12} f''(\xi), \quad \xi \in]a,b[.$$

Démonstration.
- Calcul de I_T en terme de la valeur approchée, $I_T = \int_a^b P_1(x)dx$ où P_1 désigne le polynôme d'interpolation de f sur $\{a,b\}$. Ainsi

$$\begin{aligned} I_T &= \int_a^b f[a]dx + \int_a^b f[a,b](x-a)dx \\ &= (b-a)f(a) + \frac{f(b)-f(a)}{b-a}\frac{(b-a)^2}{2} \\ &= (b-a)\left(\frac{f(a)+f(b)}{2}\right) \end{aligned}$$

d'où le résultat annoncé, qu'on aurait pu déduire d'une interpolation géométrique via l'aire du trapèze :

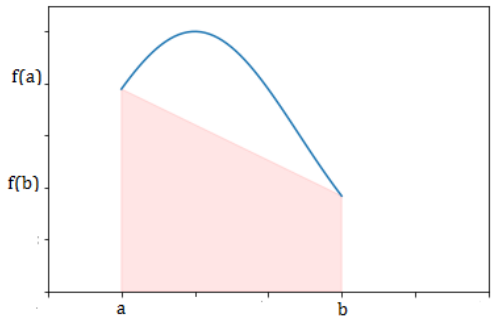


Fig. 1.5 – Méthode du trapèze (n=1)

- pour l'erreur d'intégration

$$R_T = \int_a^b f[a,b,x](x-a)(x-b)dx$$

Ici $\psi_1(x) = (x-a)(x-b) \leq 0$ donc elle ne change pas de signe dans $]a,b[$ alors, d'après le théorème (1.3) et le lemme (1.1), $\exists\ \xi \in [a,b]$ tell que

$$\begin{aligned}
R_T &= f[a,b,\xi] \int_a^b (x-a)(x-b)dx \\
&= \frac{f''(\xi)}{2!} \int_a^b (x-a)(x-b)dx \\
&= \frac{f''(\xi)}{2!} \left[\frac{x^3}{3} - \frac{a+b}{2}x^2 + abx\right]_a^b \\
&= \frac{f''(\xi)}{2!} \left[\frac{1}{3}(b^3-a^3) - \frac{1}{2}(a+b)(b^2-a^2) + ab(b-a)\right] \\
&= \frac{f''(\xi)}{2!} \frac{(b-a)}{6} \left[-b^2 - a^2 + 2ab\right] \\
&= \frac{(b-a)^3}{12} f''(\xi)
\end{aligned}$$

qui est le résultat. □

Remarque 1.4

-Cette méthode nécessite deux évaluations de la fonction f (en a et en b). Elle est donc en gros deux fois plus lente que les méthodes précédentes.
-L'erreur R_T n'est pas connue car la valeur de $\xi \in [a,b]$ reste indéterminée. Cependant, on peut la majorer par la plus grande valeur de la dérivée troisième sur le domaine considéré, alors pour $h = b - a$:

$$|R_T| \leq \frac{h^3}{12} \sup_{[a,b]}(|f''|)$$

-Les remarques sur l'erreur sont les mêmes que pour la méthode du point milieu. En précision, cette méthode est donc équivalente à celle du point milieu ($R_T \simeq R_M$), mais elle est deux fois plus lente.

L'interpolation quadratique : quadrature de Simpson

Interpolation par une fonction P_n, de degré $n = 2$
Un travail préliminaire s'impose ; reprenons le même raisonnement que dans la méthode des traits, mais cette fois-ci en utilisant un polynôme de degré 2 dont la courbe passe par $(x_0, f(x_0)), (x_1, f(x_1))$ et $(x_2, f(x_2))$, le polynôme dont l'expression donnée par la formule de Newton [5] par

$$P_2(x) = f[x_0] + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1)$$

Servons ensuite de l'approximation

$$\begin{aligned}
\int_{x_0}^{x_2} f(x)dx &\approx \int_{x_0}^{x_2} P_2(x)dx \\
&= \int_{x_0}^{x_2} \left\{f(x_0) + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1)\right\} dx
\end{aligned}$$

En se plaçant dans le cas où les abscisses sont également équidistantes, on pose de nouveau $s = \dfrac{x-x_0}{h}$, ce qui entraîne

$$(x - x_i) = (s-i)h$$

si bien que

$$\begin{aligned}\int_{x_0}^{x_2} f(x) &= \int_0^2 \big(f(x_0) + f[x_0,x_1]hs + f[x_0,x_1,x_2]h^2 s(s-1)\big)\, hds \\ &= \int_0^2 \left(f(x_0) + \frac{f(x_1)-f(x_0)}{h}hs + \frac{f(x_2)-2f(x_1)+f(x_0)}{2h^2}h^2 s(s-1)\right) hds \\ &= \frac{h}{3}(f(x_0)+4f(x_1)+f(x_2))\end{aligned}$$

c'est-à-dire

$$\int_0^2 f(x)dx \approx \frac{h}{3}[f(x_0)+4f(x_1)+f(x_2)]$$

On a donc le résultat suivant :

Proposition 3

Dans le cas de degré $n = 2$, le support d'interpolation est réduit à $\{x_0, x_1, x_2\}$. Si f est de classe $\mathcal{C}^3$ sur $[a, b]$. la valeur approchée est :

$$I_2(f) = \int_a^b f[x_0]dx + \int_a^b f[x_0,x_1](x-x_0)dx + \int_a^b f[x_0,x_1,x_2](x-x_0)(x-x_1)dx$$

et on commet l'erreur d'intégration suivante :

$$R_2(f) = \int_a^b f[x_0,x_1,x_2,x](x-x_0)(x-x_1)(x-x_2)dx$$

Démonstration.
Similaire à celle de la proposition (2). □

En choisissant $x_0 = a$, $x_1 = \dfrac{a+b}{2}$ et $x_2 = b$, il vient

Corollaire 4 (Méthode du Simpson)

Si f est de classe $\mathcal{C}^4([a, b])$, alors, la valeur approchée vaut

$$I_S = \frac{b-a}{6}(f(a)+4f(m)+f(b)), \quad avec \quad m = \frac{a+b}{2}$$

Cette méthode est aussi d'ordre 3. L'erreur d'intégration commise vaut

$$R_S = -\frac{(b-a)^5}{90}f^{(4)}(\xi), \quad \xi \in]a,b[\tag{1.6}$$

Démonstration.
voire [14] □

Remarque 1.5
-Cette méthode nécessite trois évaluations de la fonction f (en $x_0 = a, x_1 = (a+b)/2$ et $x_2 = b$). Elle est donc en gros 3 fois plus lente que les méthodes à 1 point.
- L'erreur R_S n'est pas connue car la valeur de $\xi \in [a, b]$ reste indéterminée. Cependant, on

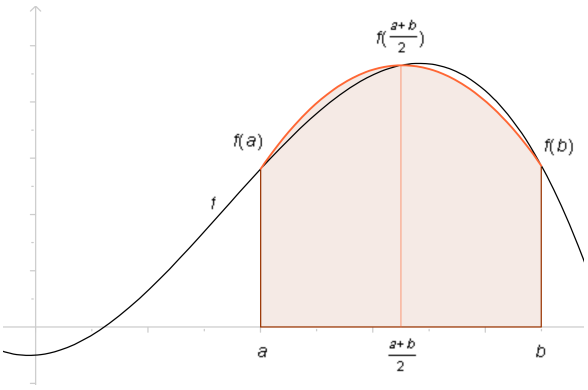


Fig. 1.6 – Méthode de Simpson simple (n=2)

peut la majorer par la plus grande valeur de la dérivée quatrième sur le domaine considéré, alors pour $h = b - a$ *:*

$$|R_S| \leq \frac{h^5}{90} \sup_{[a,b]}(|f^{(4)}|)$$

- Cette méthode d'intégration est exacte pour les fonctions f *polynomiales d'ordre* 3 *(car elles vérifient* $f^{(4)} = 0$*), ce qui inclut en particulier les fonctions constantes, les fonctions affines, et les paraboles par exemple. Plus généralement elle est d'autant plus précise que les variations de* f *sont faibles (* $f^{(4)}$ *petit).*
-Plus l'intervalle $[a, b]$ *est petit, plus l'erreur est faible. Cette erreur décroit en* h^5 *lorsque* h *diminue, c'est-à-dire beaucoup plus rapidement que les méthodes précédentes.*
- Ainsi,pour des intervalles $[a, b]$ *suffisamment petits, la méthode de Simpson est toujours plus précise que les méthodes précédentes.*

Nous allons donner maintenant quelques méthodes d'intégration numérique composées.

1.3.2 Méthodes d'ordres plus élevés

Plus généralement, on peut construire des approximations en utilisant des polynômes d'ordre quelconque. Le polynôme d'ordre n passant par $n + 1$ points régulièrement espacés entre a et b s'exprime en fonction des polynômes de Lagrange $L_k(x), k \in [0, n]$:

$$P_n(x) = \sum_{k=0}^{n} f_k L_k(x) \quad \text{avec} \quad L_k(x) = \prod_{j=0, j\neq k} \frac{x - x_j}{x_k - x_j}$$

L'intégrale approchée $I_n(f) = \int_a^b P_n(x)dx$ peut alors se calculer, et donne :

$$I_n(f) = (b - a) \sum_{k=0}^{n} w_k f_k \quad \text{avec} \quad w_k = \frac{\int_a^b L_k(x)dx}{b - a}$$

On rappelons quelque remarque important : -Une telle méthode nécessite $n + 1$ évaluations de la fonction f (en $x_k, k \in [0, n]$). Ainsi, plus le degré est élevé, plus la méthode est lente. L'erreur peut aussi se calculer (mais ça devient franchement compliqué), et on peut montrer que :

$$\text{si } n \text{ est impair : } \quad \exists\, \xi \in [a, b] \quad R_n = \frac{h^{n+2}}{C_n} f^{(n+1)}(\xi) \quad \text{c-à-d} \quad |R_n| \leq \frac{h^{n+2}}{C_n} \sup_{[a,b]} \left(\left|f^{(n+1)}\right|\right)$$

si n est pair : $\quad \exists\, \xi \in [a,b] \;\; R_n = \dfrac{h^{n+3}}{C_n} f^{(n+2)}(\xi) \quad$ c-à-d $\quad |R_n| \leq \dfrac{h^{n+3}}{C_n} \sup_{[a,b]} \left(\left|f^{(n+2)}\right|\right)$

où C_n est un coefficient qui dépend de l'ordre du polynôme.
-Une telle méthode est exacte pour les polynômes de degré n, car ceux-ci vérifient $f^{(n+1)} = 0$ (et même en fait pour des polynômes de degré $n+1$ si n est pair). Sinon, elle est d'autant plus précise que la fonction varie peu sur le domaine $[a,b]$.
-Plus le domaine $[a,b]$ est petit, plus l'erreur est faible, cette erreur décroit en h^{n+2} lorsque h diminue (et h^{n+3} si n est pair), c'est-à-dire beaucoup plus rapidement que les méthodes utilisant des polynômes de degré plus faible.

On trouve ainsi parfois référence a méthodes suivantes :

Méthode de Rectangles :
- À droite

$$I_n(f) = \frac{b-a}{n} \sum_{k=1}^{n} f(x_k)$$

- À gauche

$$I_n(f) = \frac{b-a}{n} \sum_{k=1}^{n} f(x_{k-1})$$

- Sur un intervalle uniforme

$$I_n(f) = \frac{b-a}{n} \sum_{k=1}^{n} f(x_k)$$

et l'erreur est majorée par

$$R_r \leq \frac{(b-a)^2}{n} \sup_{x\in[a,b]} |f^{'}(x)|$$

Méthode de Points milieu :

$$I_n(f) = \frac{b-a}{n} \sum_{k=1}^{n} f(\frac{x_{k+1} - x_k}{2})$$

et l'erreur est majorée par

$$R_M \leq \frac{(b-a)^5}{2880n^4} \sup_{x\in[a,b]} |f^{(4)}(x)|$$

Méthode de Trapèzes :

$$I_n(f) = \frac{b-a}{2n} \sum_{k=1}^{n} f(x_{k-1}) + f(x_k)$$

et l'erreur est majorée par

$$R_T \leq \frac{(b-a)^3}{12n^2} \sup_{x\in[a,b]} |f^{''}(x)|$$

Méthode de Simpson :

$$I_n(f) = \frac{b-a}{6n} \sum_{k=1}^{n} f(x_{k-1}) + 4f(x_{k-1/2}) + f(x_k)$$

et l'erreur est majorée par

$$R_S \leq \frac{(b-a)^3}{12n^2} \sup_{x\in[a,b]} |f''(x)|$$

Méthode de Simpson $3/8$ **(degré** $n = 3$**)**

$$I_3(f) = (b-a)\frac{f_0 + 3f_1 + 3f_2 + f_3}{8}$$

$$R_S(f) = -\frac{3}{80}h^5 f^{(4)}(\xi)$$

Les méthodes usuelles de quadrature (méthode de Simpson, méthodes de Trapèze...) sont efficaces pour le calcul d'intégrales à une dimension et se généralisent facilement à plusieurs dimensions lorsque le domaine d'intégration est simple (hypercube, par exemple). En pratique cependant, les coefficients deviennent de plus en plus grands lorsque l'ordre du polynôme augmente. De plus, à partir du degré $n = 7$, certains coefficients deviennent négatifs. Elles ne sont donc jamais utilisées pour $n > 7$.

Par exemple : la méthode de Simpson, qui est la méthode plus rapide que les méthodes précédentes, Lorsque la dimension d est grand, la convergence, en fonction du nombre N de points où la fonction est évaluée (ce qui détermine le temps) devient très lente. Pour la règle de Simpson, la convergence devient plus lente que $1/N$ lorsque d est supérieur à 7. Nous voyons apparaître l'intérêt d'une méthode, où l'erreur est en $1/N$ quelle que soit la dimension d cette méthode est la méthode de Monte-Carlo.

Chapitre 2

Méthode de Monte-Carlo

Les méthodes de Monte-Carlo sont particulièrement utilisées pour calculer des intégrales en dimensions plus grandes que 1 (en particulier, pour calculer des surfaces, des volumes, etc.). Dans ce chapitre on s'intéresse à la description de la méthode de Monte-Carlo. On débutera en donnant le principe générale de la méthode, puis en étudiera plus pratiquement la convergence, l'estimation de la variance d'un calcul et on définira l'intervalle de confiance. Enfin on va présent des méthode de réduction de variance.

Maintenant on va donner quelque définition élémentaire[13]

Définition 2.1

i) Une probabilité est une application sur $\mathcal{P}(\Omega)$; l'ensemble des parties de Ω (où Ω est l'espace fondamental), telle que :
1. $0 \leq P(A) \leq 1$*, pour tout événement* $A \subset \Omega$
2. $P(A) = \sum_{\omega \in A} P(\omega)$*, pour tout événement* A
3. $P(\Omega) = \sum_{\omega \in A} P(\omega) = 1$.

ii) Une variable aléatoire X est une fonction définie sur l'espace fondamental, qui associe une valeur numérique à chaque résultat de l'expérience aléatoire étudiée. Ainsi, à chaque événement élémentaire ω, on associe un nombre $X(\omega)$.

iii) Soit X une v.a.. On appelle fonction de répartition de X la fonction de $\mathbb{R}$ dans $[0,1]$, définie pour tout $x \in \mathbb{R}$ par

$$F(x) = P[X \leq x]$$

iv) Soit X un variable aléatoire (v.a.) absolument continue de fonction de répartition F. On appelle fonction de densité de X noté f la dérivée de F qu'il vérifier :
1. $f(x) \geq 0, \forall x \in \mathbb{R}$
2. $F(x) = \int_{-\infty}^{x} f(t)dt$
3.$\int_{-\infty}^{+\infty} f(t)dt = 1$

v) L'espérance ou moyenne d'une v.a. discrète X est le réel

$$E[X] = \sum_{k} P[X = k]$$

où on somme sur toutes les valeurs k que peut prendre X.

vi) Une v.a. X suit une loi uniforme sur $[a, b]$, si elle admet pour densité

$$f(x) = \begin{cases} \frac{1}{b-a} & si\ \ a < x < b \\ 0 & sinon \end{cases}$$

2.1 Principe

2.1.1 Intégration unidimensionnelle

Pour utiliser une méthode de Monte-Carlo [9], on doit tout d'abord mettre sous la forme d'une espérance la quantité que l'on cherche à calculer, à l'issu de cette étape, il reste à calculer cette quantité par une espérance $E(X)$ de la variable aléatoire X. Pour ce calcul, il convient de savoir simuler une variable aléatoire selon la loi de X. On dispose alors d'une suite $(x_i)_{1\leq i\leq n}$ de n réalisations de la variable aléatoire X, On approxime alors $E(X)$ par :

$$E(X) \simeq \frac{1}{n}(x_1 + ... + x_N)$$

Nous souhaitons calculer l'intégrale[8]

$$I(h) = \int_a^b h(x)dx \tag{2.1}$$

Nous exprimons tout d'abord la fonction $h(x)$ sous la forme

$$h(x) = g(x)f(x)$$

où $f(x)$ est une densité sur (a, b). Ainsi, par définition de l'espérance :

$$\begin{aligned} I(h) &= \int_a^b g(x)f(x)dx \\ &= E[g(X)] \end{aligned}$$

où X est la variable aléatoire avec fonction de densité de probabilité $f(x)$. Par la suite, si $x_1, ..., x_n$ sont des observations simulées de la densité $f(x)$, alors

$$I_n(h) = \frac{1}{n}\sum_{i=1}^{n} g(x_i) \tag{2.2}$$

constitue une estimation de l'intégrale $I(h)$.

Par souci de simplicité et parce qu'il est facile d'obtenir un échantillon aléatoire d'une loi uniforme, nous posons en général $X \sim \mathcal{U}(a, b)$, soit

$$f(x) = \frac{1}{b-a},\ \ a < x < b.$$

Dans ce cas, $g(x) = (b - a)h(x)$ donc l'intégrale de départ (2.1) se réécrit

$$\begin{aligned} I(h) &= (b-a)\int_a^b h(x)\frac{1}{b-a}dx \\ &= (b-a)E[h(x)]. \end{aligned} \tag{2.3}$$

L'estimation (2.2) de l'intégrale devient alors

$$I_n(h) = \frac{b-a}{n} \sum_{i=1}^{n} h(x_i)$$

où $x_1, ..., x_n$ est un échantillon aléatoire d'une loi $\mathcal{U}(a, b)$.
Par changement de variable posons

$$u = \frac{x-a}{b-a}, \quad du = \frac{dx}{b-a}$$

équivaux à

$$x = a + (b-a)u, \quad dx = (b-a)du$$

on remplace dans l'intégrale (2.3). On obtient alors

$$\begin{aligned} I(h) &= (b-a) \int_0^1 h(a + (b-a)u)du \\ &= (b-a)E[h(a + (b-a)U)] \end{aligned}$$

où $U \sim \mathcal{U}(0, 1)$. Une estimation de l'intégrale est donc

$$I_n(f) = \frac{b-a}{n} \sum_{i=1}^{n} h(a + (b-a)u_i)$$

où $u_1, ..., u_n$ est un échantillon aléatoire d'une loi $\mathcal{U}(0, 1)$.
Nous devons utiliser la technique du changement de variable lorsque le domaine est infini.

2.1.2 Intégration multidimensionnelles

Le calcul de l'intégrale multidimensionnelles $I(f) = \int_\Omega f(x)dx$ avec $\Omega \in \mathbb{R}^d, d \in \mathbb{N}$ par la méthode de Monte-Carlo se ramène à résoudre l'intégrale suivante :

$$I(f) = \int_{[0,1]^d} f(x)dx$$

Dans ce cas, la méthode de Monte-Carlo consiste à écrire cette intégrale sous forme d'espérance de f qui est la généralisation de la moyenne de f sur $[0, 1]^d$, et avec u une variable aléatoire suivant la loi uniforme sur $[0, 1]^d$:

$$I(f) = E(f(u))$$

Donc on doit :

1. Mise $I(f)$ sous forme d'espérance : on pose $X = f(U_1, ..., U_d)$ où $U_1, ..., U_d$ sont des réalisations de la loi uniforme sur l'intervalle $[0, 1]$, alors

$$E(X) = E(f(U_1, ..., U_d)) = I(f)$$

2. Simulation de la variable aléatoire : on suppose que l'on dispose d'une suite $(U_i)_{i\geq 1}$ de réalisations de la loi uniforme sur $[0,1]$. On pose alors $X_1 = f(U_1, ..., U_d), X_2 = f(U_{d+1}, ..., U_{2d})$,... etc. Alors les (X_i) sont des réalisations de la variable aléatoire X et

$$I(f) \simeq \frac{1}{n}(X_1 + ... + X_n)$$

On approxime l'intégrale comme suit :

$$I(f) \simeq I_n(f) = \frac{1}{n}\sum_{i=1}^{n} f(x_i) = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Les points x_i sont choisis dans l'intervalle Ω, donc quand le nombre des points n augmente l'approximation sera plus précise et on a :

$$I(f) = \int_\Omega f(x)dx = \lim_{n\to\infty} I_n(f) = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

2.1.3 Convergence de la méthode

Pour avoir une idée de l'intérêt de la méthode, il faut pouvoir évaluer l'erreur commise.

Définition 2.2
L'erreur commise définie par :

$$\varepsilon_n = E(X) - \frac{1}{n}\sum_{i=1}^{n} X_i$$

C'est la loi forte des grands nombres qui permet de justifier la convergence de la méthode.

Théorème 2.1 (Théorème de loi forte des grands nombres)
Soit $X_1, X_2, ..., X_N$ une suite de variables aléatoires réelles deux à deux indépendantes et ayant toutes la même moyenne $E[X_i] = m$ et le même écart type fini : $\sigma = E[X_i^2]$. Alors la moyenne arithmétique :

$$\frac{X_1 + X_2 + ... + X_N}{N}$$

converge "en probabilité" vers la moyenne stochastique m lorsque $n \longrightarrow 1$. C'est à dire, quel que soit le nombre positif ϵ donné (si petit soit-il) :

$$\lim_{N\longrightarrow\infty} P\left(\left|\frac{X_1 + X_2 + ... + X_N}{N} - m\right| > \epsilon\right) = 0$$

Démonstration.
Pour la démonstration voire [15] □

2.1.4 Vitesse de convergence

La vitesse de convergence [7] d'une estimation par la méthode de Monte Carlo est donnée par le théorème central limite.

Théorème 2.2 (Théorème de limite centrale)
Soit $(X_j, j \geq 1)$ une suite de variables aléatoires indépendantes et de même loi que X, telles que $E(|X^2|) < +\infty$ On note σ^2 la variance de X :

$$\sigma^2 = E((X - E(X))^2) = E(X^2) - (E(X))^2$$

Alors la suite

$$\frac{\sqrt{n}}{\sigma}\varepsilon_n = \frac{\sqrt{n}}{\sigma}\left(\frac{1}{n}\sum_{j=1}^{n} X_j - E(X)\right) = \frac{1}{\sigma\sqrt{n}}\sum_{j=1}^{n}(X_j - E(X))$$

Converge en loi vers une gaussienne centrée réduite. C'est-à-dire

$$\forall a \leq b \quad \lim_{n\to+\infty} P(\frac{\sigma}{\sqrt{n}}a \leq \varepsilon_n \leq \frac{\sigma}{\sqrt{n}}b) = \frac{1}{\sqrt{2\pi}}\int_a^b \exp^{\frac{-x^2}{2}}\, dx$$

Démonstration. Supposons au départ que $E(X) = 0$ et $\sigma^2 = 1$, posons $S_n = \sum_{j=1}^{n} X_j$

$$\frac{1}{\sigma\sqrt{n}}\sum_{j=1}^{n}(X_j - E(X)) = \frac{S_n - nE(X)}{\sigma\sqrt{n}} = \frac{S_n}{\sqrt{n}}$$

a comme fonction caractéristique :

$$\varphi_{\frac{S_n}{\sqrt{n}}}(t) = E(e^{it\frac{S_n}{\sqrt{n}}}) = E(e^{it\frac{X}{\sqrt{n}}})^n = (\varphi_X(\frac{t}{\sqrt{n}}))^n$$

comme X a carré intégrable, nous pouvons donc dériver deux fois sa fonction caractéristique

$$\varphi_X^{'}(0) = 0, \quad \varphi_X^{''}(0) = -1$$

pour tout u au voisinage de 0, un développement limité de la fonction caractéristique donne

$$\varphi_X(u) = 1 - \frac{u^2}{2} + o(u^2)$$

donc

$$\begin{aligned}\lim_{n\to\infty} \varphi_{\frac{S_n}{\sqrt{n}}}(t) &= \lim_{n\to\infty}(\varphi_X(\frac{t}{\sqrt{n}}))^n \\ &= \lim_{n\to\infty}(1 - \frac{t^2}{2n} + o(\frac{t^2}{n}))^n \\ &= e^{-\frac{t^2}{2}}\end{aligned}$$

qui est la fonction caractéristique d'un loi $\mathcal{N}(0,1)$.On en déduit alors la convergence en loi de $\frac{S_n}{\sqrt{n}}$ vers la loi gaussienne centrée réduit.
Si $E(X) \neq 0$ ou $\sigma^2 \neq 1$, posons pour tout $i \in \mathbb{N}$,

$$Y_j = \frac{X_j - E(X_j)}{\sigma}$$

les Y_j sont des variables centré réduit, on applique alors la première partie de la démonstration. [10] □

Cependant, en général, l'écart-type σ est lui aussi inconnu, il va donc falloir l'estimer.

2.1.5 Estimation de la variance

La méthode Monte-Carlo en fournit justement un estimateur à peu de frais [3] puisque basé sur le même échantillon $(X_1, ..., X_n)$, à savoir

$$\hat{\sigma}_n^2 = \frac{1}{n}\sum_{i=1}^{n} \varphi(X_i)^2 - I_n^2 \xrightarrow{p.s}_{n\to\infty} \sigma^2$$

Le théorème de la limite centrale donne un asymptotique de l'erreur ε_n, mais de nature aléatoire. Dans ce cas, la loi de l'erreur finit par ressembler à une loi gaussienne centrée. Plus précisément, le théorème de la limite centrale doit permettre de construire des intervalles de confiance, on en déduit bien entendu des intervalles de confiance asymptotiques pour I.

Définition 2.3 (Intervalles de confiance)
Soit $\alpha \in (0,1)$ fixé. Un intervalle de confiance de niveau asymptotique $1-\alpha$ pour I est

$$\left[I_n - \Phi^{-1}(1-\alpha/2)\frac{\hat{\sigma_n}}{\sqrt{n}}, I_n + \Phi^{-1}(1-\alpha/2)\frac{\hat{\sigma_n}}{\sqrt{n}}\right]$$

où $\Phi^{-1}(1-\alpha/2)$ désigne le quantile d'ordre $1-\alpha/2$ de la loi normale centrée réduite.

Typiquement $\alpha = 0.05$ donne $\Phi^{-1}(1-\alpha/2) = q_{0.975} = 1.96$, qui permet de construire un intervalle de confiance à 95% pour I.

Remarque 2.1
-Par le théorème de loi forte des grands nombres, $I_n \longrightarrow I$ quand $n \longrightarrow \infty$.
-Cette méthode ne dépend pas de la régularité de f qui doit être juste mesurable.
-Dans les applications, on remplace ε_n par une gaussienne centrée de variance $\frac{\sigma^2}{n}$.
-On présente souvent l'erreur de la méthode de Monte-Carlo, soit en donnant l'écart-type de l'erreur ε_n, soit en donnant un intervalle de confiance à 95% pour le résultat. L'intervalle de confiance de la méthode à 95% est :

$$[I_n - 1.96\frac{\sigma^2}{n}, I_n + 1.96\frac{\sigma^2}{n}]$$

-En général, le calcul de la valeur de σ est approximée par la méthode de Monte-Carlo :

$$\frac{1}{n}\sum_{i=1}^{n}[g(X_i)]^2 - \left[\frac{1}{n}\sum_{i=1}^{n} g(X_i)\right]^2 \longrightarrow \sigma^2$$

-La vitesse de convergence est en $\frac{\sigma}{\sqrt{n}}$, ce qui n'est pas très rapide, mais c'est parfois la seule méthode abordable, de plus cette vitesse ne change pas si on est en grande dimension, et elle ne dépend pas de la régularité de la fonction.

Nous avons voir que la vitesse de convergence de la méthode Monte-Carlo est de l'ordre de $\frac{\sigma}{\sqrt{n}}$. Pour améliorer cette méthode, il existe de nombreuses techniques, dites de réduction de variance, qui cherchent à diminuer la valeur de σ^2 sans augmenter significativement le temps de calcul.

2.2 Méthodes de réduction de variance

L'idée générale de ces méthodes est de donner une autre représentation sous forme d'espérance de la quantité à calculer, c'est-à-dire : $E(X) = E(Y)$ avec $Var(Y) < Var(X)$. Plusieurs techniques réduisant la variance, mais pas la vitesse de convergence (excepté pour la variante du cas de l'échantillonnage stratifié), sont présentées maintenant. Il est aussi possible de combiner plusieurs techniques de réduction de la variance, mais une telle pratique doit être réalisée avec prudence car elle ne produit pas toujours les effets escomptés : en effet, la variable aléatoire dont on cherche l'espérance n'est alors plus la même. Afin d'illustrer les méthodes, nous nous limiterons au calcul d'une intégrale $I(f) = \int_{[0;1]^d} f(x)dx$ [16].

2.2.1 Variables de contrôle

L'erreur d'approximation dans l'estimation de $I(f)$ par une méthode de Monte-Carlo standard provient des grandes variations de f. La technique de variable de contrôle réécrit l'intégrale sous la forme :

$$I(f) = \int_{[0,1]^d} g(x)dx + \int_{[0,1]^d} (f(x) - g(x))dx$$

Si g est suffisamment simple pour être intégrée formellement, on peut se concentrer sur la deuxième intégrale. De même, si g est suffisamment proche de f (si g absorbe les variations de f), on obtient une réduction conséquente de la variance. En effet, si X désigne une variable aléatoire uniformément distribuée sur $[0,1]^d$ et si l'on pose $Z = f(X) - g(X)$, on a :

$$\sigma^2(Z) = \sigma^2(f(X)) + \sigma^2(g(X)) - 2Cov(f(X), g(X))$$

Pour obtenir une réduction de la variance, il suffit donc que :

$$\sigma^2(g(X)) - 2Cov(f(X), g(X)) < 0$$

2.2.2 Variables antithétiques

Les variables antithétiques aussi appelées variables complémentaire reposent sur ce principe :
Faire une exécution en utilisant les nombres aléatoires $(u_1, ..., u_n)$ pour obtenir une échantillon d'observations aléatoires $(x_1, ..., x_n)$, puis calculer l'estimateur $I_n(f)$ faire une autre exécution en utilisant les nombres aléatoires complémentaires $(1 - u_1, ..., 1 - u_n)$ pour obtenir une échantillon d'observations aléatoires $(x_1^{'}, ..., x_n^{'})$ puis calculer l'estimateur $I_n^{'}(f)$
Utiliser ces deux exécutions pour calculer la moyenne combinée : $I_{nac}(f) = \frac{I_n(f) + I_n^{'}(f)}{2}$ qui est un estimateur sans biais et plus précis que $I_n(f)$ [4].

2.2.3 Stratification

Le principe ici est de :

1. Séparer le domaine d'intégration en plusieurs sous-domaines (J sous-domaines Ω_j)
2. simuler n_j points $(X_{1j}, ..., X_{nj})$ dans chaque Ω_j avec probabilité $P(\Omega_j)$

L'estimateur est :

$$I_n(f) = \sum_{j=1}^{J} \frac{P(\Omega_j)}{n_j} \sum_{i=1}^{n_j} f(X_{ij}).$$

La variance estimée[1] est :

$$\hat{\sigma}(I_n(f))^2 = \sum_{j=1}^{J} \frac{P(\Omega_j)^2 \hat{\sigma}_j^2}{n_j}, \;\; \hat{\sigma}_j^2 = \frac{1}{n_j} \sum_{i=1}^{n_j} f(X_{ij}^2) - \left(\frac{1}{n_j} \sum_{i=1}^{n_j} f(X_{ij}) \right)^2 .$$

.

2.2.4 Échantillonnage suivant l'importance

Échantillonnage préférentiel[1] c'est une méthode de calcul de $I_n = E[f(X)]$ qui consiste à tirer les variables selon une distribution erronée, et à compenser numériquement le résultat à postériori.
Pour en donner une idée, revenons au calcul d'une intégrale sur $[0, 1]$. Si p est une densité strictement positive sur $[0, 1]$, on peut écrire

$$\int_0^1 f(x)dx = \int \frac{f(x)}{p(x)} p(x)dx$$

Ce qui suggère la possibilité d'un autre algorithme : $I_n^{'} = \frac{1}{n} \sum_{i=1}^{n} \frac{f(Y_i)}{Y_i}$, où les Y_i sont i.i.d sur $[0, 1]$ de loi $p(x)dx$: Rappelons que si X a une densité $f_X(x)$ et Y une densité f_Y alors, $p = f_Y / f_X$.

Chapitre 3

Application

À travers ce chapitre nous allons voir l'utilisation des notions et principaux résultats de la méthode de Monte-Carlo. On va présenter des exemples du calcul d'intégrale simple et compliqué (calcul surface d'un roue) par cette méthode de plus, on va exposer une comparaison entre la méthode de Monte-Carlo et les quatre méthodes numérique de première chapitre en utilisant les deux logiciels **R** et Python.

3.1 La méthode de Monte-Carlo par R

3.1.1 Dimension un

Exemple 3.1
Soit la fonction

$$h(x) = x^{11/5}e^{-x/10}$$

posons l'intégrale

$$I = \int_2^5 x^{11/5}e^{-x/10}dx$$

Pour utiliser la méthode Monte-Carlo, nous posons [8] :

$$I = 3\int_2^5 x^{11/5}e^{-x/10}(\frac{1}{3})dx$$

Si $\{x_1, ..., x_n\}$ *est un échantillon aléatoire d'une loi* $\mathcal{U}(2,5)$*, alors*

$$I_n = \frac{3}{n}\sum_{i=1}^{n} x_i^{11/5}e^{-x_i/10}$$

est une estimation de I*. avec le code* ***R*** *(3.1) nous obtenons les résultats suivants et la figure (3.1)*

n	I
100	33.27398
1000	34.26288
10 000	34.3455
100 000	34.5762
1000 000	34.51774

Table 3.1 – Monte-Carlo par **R** dimension 1

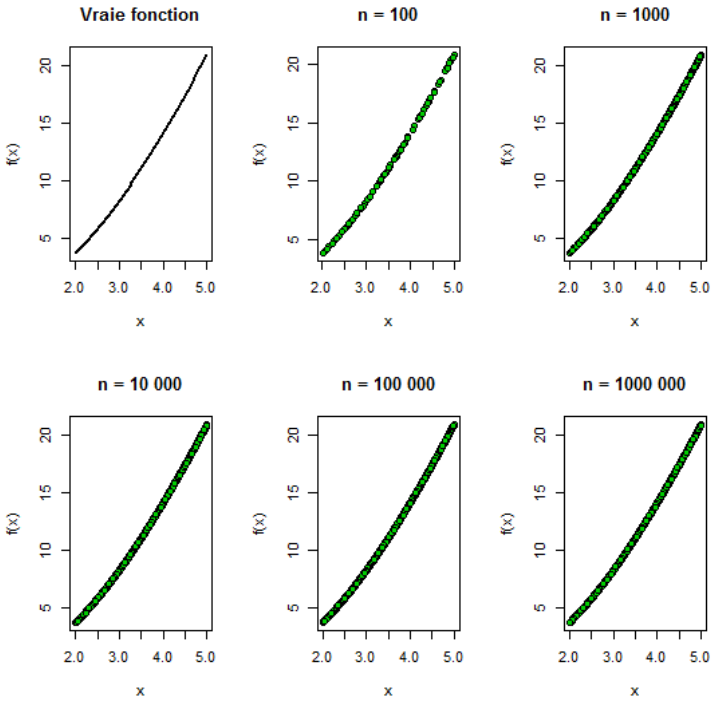


Fig. 3.1 – Monte-Carlo par **R** dimension 2

3.1.2 Dimension deux

Exemple 3.2
Soit la fonction

$$h(x,y) = \sqrt{4 - x^2 - y^2}$$

La procédure à suivre avec les intégrales multiples est la même qu'avec les intégrales simples, sauf que nous tirons des points uniformément dans autant de dimensions que nécessaire. Ainsi, dans le cas présent, nous posons :

$$\begin{aligned} I &= \int_0^{5/4}\int_0^{5/4} \sqrt{4 - x^2 - y^2}\,dydx \\ &= \frac{25}{16}\int_0^{5/4}\int_0^{5/4} \sqrt{4 - x^2 - y^2}\,\frac{1}{\left(\frac{5}{4}\right)\left(\frac{5}{4}\right)}\,dydx \\ &= \frac{25}{16}E\left[\sqrt{4 - X^2 - Y^2}\right] \end{aligned}$$

où X et Y sont des variables aléatoires indépendantes distribuées uniformément sur l'intervalle $(0, \frac{5}{4})$. Autrement dit, la densité conjointe de X et Y est uniforme sur $(0, \frac{5}{4}) \times (0, \frac{5}{4})$.

Par conséquent, une estimation de I calculée à partir d'un échantillon $\{(x_1, y_1), ..., (x_n, y_n)\}$ tiré de cette loi conjointe est

$$I_n = \frac{25}{16n} \sum_{i=1}^{n} \sqrt{4 - x_i^2 - y_i^2}.$$

*Le code **R** (3.2) effectuer quelques résultats obtenus avec **R** les calculs et les graphiques (voir la figure (3.2)).*

n	I
100	2.707254
1000	2.651387
10 000	2.669409

Table 3.2 – Monte-Carlo par **R** dimension 2

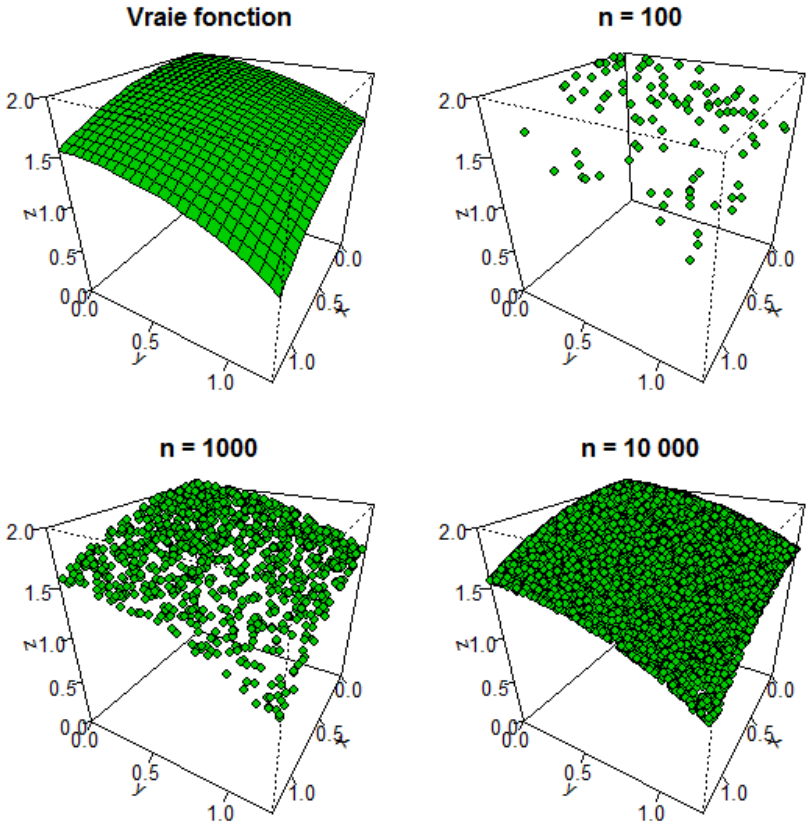


Fig. 3.2 – Monte-Carlo par **R** dimension 2

3.2 Calcule volume

Exemple 3.3
Nous allons calculer, en dimension $d = \{3, 5, 7\}$, le volume compris entre les hypersphères

de rayons R_1 et R_2. La fonction (3.4) renvoie le volume, l'estimation de la variance σ^2 On pose $R_1 = 0$ et $R_2 = 1$ En dimension 3, on trouve :

volume	σ^2	e
4.2424	15.941242240000001	0.07825591107972867

Table 3.3 – Calcul volume par la méthode de Monte-Carlo, $d = 3$

avec un intervalle de confiance à 95% :

$$[I - e, I + e] = [4.164144088920271, 4.3206559110797285]$$

En dimension 5 on obtient

volume	σ^2	e
5.1392	138.04302336	0.2302837550805041

Table 3.4 – Calcul volume par la méthode de Monte-Carlo, $d = 5$

avec un intervalle de confiance à 95% :

$$[I - e, I + e] = [4.908916244919496, 5.369483755080504]$$

Comme on le voit sur cet exemple, la variance dépend de la dimension. Augmentons le nombre d'échantillons pour obtenir un écart comparable à celui de la dimension 3. Comme l'écart évolue comme $1/\sqrt{N}$, il faut augmenter N d'un facteur 90 environ on trouve :

volume	σ^2	e
5.285511111111111	141.19972784987655	0.024550014132335506

Table 3.5 – Calcul volume par la méthode de Monte-Carlo, $d = 5$, $N = 9 \times 10^5$

avec un intervalle de confiance à 95% :

$$[I - e, I + e] = [5.260961096978775, 5.310061125243447]$$

Voyons le volume d'une coque d'épaisseur un dixième du rayon :

volume	σ^2	e
2.12064	63.3633659904	0.049337278683437805

Table 3.6 – Calcul volume par la méthode de Monte-Carlo, $d = 5$, $N = 10^5$

avec un intervalle de confiance à 95% :

$$[I - e, I + e] = [2.0713027213165622, 2.1699772786834375]$$

Voici les mêmes calculs en dimension 7 (beaucoup plus longs) :

volume	σ^2	e
4.713984	581.168306847744	0.014941941532432435

Table 3.7 – Calcul volume par la méthode de Monte-Carlo, $d = 7$, $N = 10^7$

avec un intervalle de confiance à 95% :

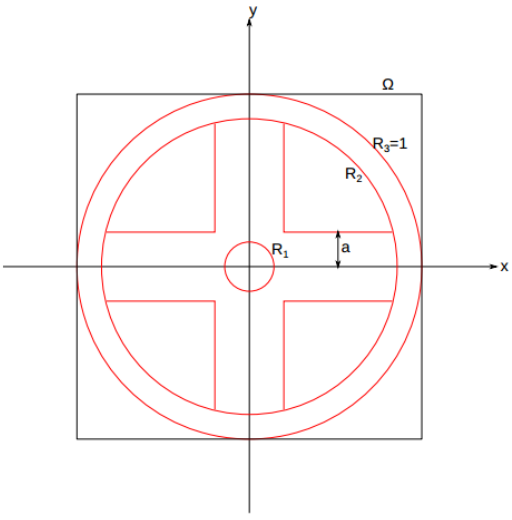
$$[I - e, I + e] = [4.699042058467567, 4.728925941532433]$$

3.3 Calcul de moments d'inertie par intégration de Monte-Carlo

3.3.1 Méthode de Monte-Carlo avec échantillonnage uniforme

Exemple 3.4

L'objectif est de calculer les moments d'inertie d'un solide en utilisant la méthode d'intégration de Monte-Carlo. On verra dans un premier temps un échantillonnage uniforme sur un domaine carré contenant le solide, avant d'envisager un échantillonnage préférentiel, qui permettra de réduire la variance de la somme calculée. On considère l'exemple d'une roue :



La roue est un cylindre d'axe Z et d'épaisseur e. Le rayon extérieur est $R_3 = 1$, le rayon intérieur est R_2. Le moyeu comporte un trou circulaire de diamètre R_1 et deux branches de largeur $2a$. On suppose que la densité de masse est uniforme.
On cherche à calculer le moment d'inertie de la roue par rapport à son axe Z :

$$J_z = \int\int\int (x^2 + y^2)\rho(x, y, z)dxdydz = e \int\int (x^2 + y^2)\rho(x, y)dxdy$$

En raison du caractère bidimensionnel de ce solide, on est ramené à un calcul d'intégrale double, et on posera $e = 1$*, la densité de masse est* $\rho = 1$ *dans la matière (dans les deux branches du moyeu et dans la roue), et nulle en dehors.*
Pour le calcul de l'intégrale, on utilisera le domaine carré $\Omega = [-1, 1] \times [-1, 1]$*. Il s'agit donc de calculer :*

$$J_z = \int_1^1 \int_1^1 (x^2 + y^2)\rho(x, y)dxdy$$

On définit la fonction à intégrer :

$$f(x, y) = (x^2 + y^2)\rho(x, y)$$

La fonction (3.5) fait le calcul de l'intégrale et donne

I	e
1.127363547054364	0.009026401041990512

Table 3.8 – Méthode de Monte-Carlo avec échantillonnage uniforme

avec un intervalle de confiance à 95% *:*

$$[I - e, I + e] = [1.1183371460123734, 1.1363899480963544]$$

Elle génère aussi une image (3.3) carrée dont le nombre de pixels sur chaque dimension est `np`*. Cette image, noire au départ, est complétée par un pixel blanc pour chaque point tiré dont la valeur de* f *est non nulle.*

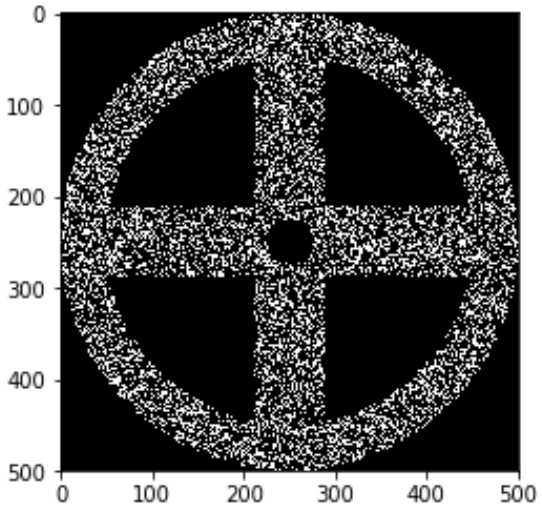


Fig. 3.3 – Méthode de Monte-Carlo avec échantillonnage uniforme

3.3.2 Méthode de Monte-Carlo avec échantillonnage préférentiel

Exemple 3.5
Une première méthode consiste à échantillonner uniformément sur le disque de rayon 1.

On doit pour cela utiliser les coordonnées polaire (r, θ) *et faire des tirages avec la densité de probabilité uniforme sur le disque :*

$$p(r, \theta) = \frac{1}{\pi}$$

Pour réaliser cet échantillonnage à partir de variables à densité uniforme, on considère la fonction de répartition bidimensionnelle :

$$F(r, \theta) = \int_0^{\theta} \int_0^{r} \frac{r^{'}}{\pi} d\theta^{'} dr^{'} = \frac{\theta}{2\pi} r^2$$

L'inversion de la fonction de répartition permet d'échantillonner r *et* θ *à partir de deux variables* (u_1, u_2) *uniformes sur l'intervalle* $[0, 1]$ *:*

$$\theta = 2\pi u_1$$

$$r = \sqrt{u_2}$$

La fonction (3.6) fait le calcul de l'intégrale et donne

I	e
1.1328106867306718	0.007304407857477749

Table 3.9 – Méthode de Monte-Carlo avec échantillonnage préférentiel

avec un intervalle de confiance à 95% *:*

$$[I - e, I + e] = [1.125506278873194, 1.1401150945881495]$$

Elle génère aussi l'image (3.4)

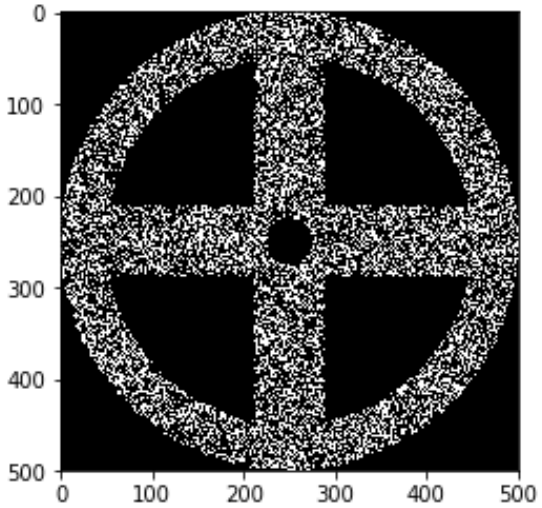


Fig. 3.4 – Méthode de Monte-Carlo avec échantillonnage préférentiel

3.4 Comparaison des cinq méthodes

Exemple 3.6

Dans cet exemple, on va faire une comparaison entre les quatre méthodes de chapitre 1 *et la méthode de Monte-Carlo sur les dimensions* 1, 2 *et* 3.

Dimension 1

Soit la fonction

$$f(x) = \sqrt{x^2 + e^{5x}}$$

On va calculer l'intégral

$$I = \int_0^1 f(x)dx \tag{3.1}$$

Le programme (3.7) fait le calcul de l'intégral (3.1) et d'erreur suivant les quatre méthodes. On prend $n = 10000$*, on obtient le résultat suivant :*

Les méthodes	I	L'erreur
Valeur intégral	4.502084721431	0.000000000000
Rectangle	4.501523571333	0.000561150098
Point Milieu	4.502084709791	0.000561150098
Simpson	4.502084721431	0.000000000000
Monte-Carlo	4.533330768805	0.031246047374

Table 3.10 – Comparaison en dimension 1

Dimension 2

Soit la fonction

$$g(x, y) = xy + y^2 + 1$$

On va calculer l'intégral

$$\int_1^2 \int_0^3 g(x, y)dxdy \tag{3.2}$$

Le programme (3.8) fait le calcul de l'intégral (3.2) et d'erreur suivant les quatre méthodes. On prend $nx = 10000$*,* $ny = 20000$*, on obtient le résultat suivant :*

Les méthodes	I	L'erreur
Valeur intégral	18.750000000000	0.000000000000
Rectangles	18.748762522505	0.001237477495
Point Milieu	18.749999994380	0.000000005620
Trapèzes	17.625000011241	1.124999988759
Simpson	18.750000003757	0.000000003757
Monte-Carlo	18.924444405693	0.174444405693

Table 3.11 – Comparaison en dimension 2

Dimension 3

Soit la fonction

$$g(x, y, z) = ze^{x^2+y^2}$$

On va calculer l'intégral

$$\int_0^1 \int_0^1 \int_0^1 g(x,y,z)dxdydz \tag{3.3}$$

Le programme (3.9) fait le calcul de l'intégral (3.3) et d'erreur suivant les quatre méthodes.
On prend $n1 = 5$, $n2 = 3$ *et* $n3 = 1$, *on obtient le résultat suivant :*

Les méthodes	I	L'erreur
Valeur intégral	1.069675064903	0.000000000000
Rectangles	0.000000000000	1.069675064903
Point milieu	1.045397685605	0.024277379298
Trapèzes	1.485974293374	0.416299228471
Simpson	1.192256554861	0.122581489959
Monte-Carlo	1.069506112470	0.000168952433

Table 3.12 – Comparaison en dimension 3

En dimension d, on est amené à calculer des intégrales dans un espace de dimension d qui peut être très grand (par exemple plusieurs millions). On peut citer l'espace des phases utilisés en physique statistique pour représenter la configuration d'un système comportant un grand nombre d'atomes. Les méthodes de quadrature sont très inefficaces, voire inutilisables, lorsque la dimension est élevée. Par exemple, pour la méthode des Rectangles, le nombre de points nécessaires pour maintenir une précision constante évolue comme N^d. Le logarithme du nombre de points est donc proportionnel à la dimension d, une loi que l'on retrouve pour toutes les méthodes de quadrature. La méthode de Monte-Carlo ne présente pas cette dépendance par rapport à la dimension. La variance de la somme calculée est :

$$var(S_N) = \frac{\sigma^2}{N}$$

La variance σ^2 de la variable f/p (p densité de probabilité) est susceptible d'augmenter avec la dimension. La méthode de Monte-Carlo a donc un avantage décisif en dimension élevée.

Conclusion et perspectives

Comme conclusion à cette modeste étude, nous retenons que la méthode de Monte-Carlo est une méthode d'approximation par introduction de procédés aléatoire. Cela permet d'estimer des valeurs numériques et de caractériser des systèmes complexes.

La méthode de Monte-Carlo est l'une des techniques de simulation offrant la possibilité de calculer des intégrales difficiles à calculer. Elle a comme grand avantage, par rapport aux méthodes d'analyse numérique classiques, d'avoir une vitesse de convergence en $o(\frac{1}{\sqrt{n}})$ (pour un échantillon de n points), donc indépendante de la dimension du problème. Du point de vu des applications, la méthode de Monte-Carlo sont aujourd'hui indispensables dans plusieurs domaines.

Cette méthode nécessite un paramètre d'entrée de nature aléatoire. Néanmoins, il doit exister des suites de nombres telles que la convergence soit plus rapide, en supprimant l'aspect aléatoire. C'est pourquoi on intéresse à la recherche des méthodes qui permis de créer des suites qui convergent plus rapidement que les suites de Monte-Carlo, ces méthodes sont les méthodes Quasi Monte-Carlo.

Bibliographie

[1] Bernard-delyon (2012). *Simulation et modélisation.* Universite Rennes I.

[2] Boubalou, M., Oukachbi, A., Ourbih, M., et al. (2015). *La comparaison entre les méthodes hypercube latin et descriptif amélioré à travers un modèle de file d'attente M/G/1.* PhD thesis, Université abderrahmane mira béjaia.

[3] Céline-Baranger, J.-M. (2012-2013). Méthode de monte-carlo.

[4] Derradji, N., Khalfi, L., Ourbih, M., et al. (2013). *Les méthodes Monte Carlo, Quasi Monte Carlo et leurs applications.* PhD thesis, Université abderrahmane mira béjaia.

[5] Enseignements-Licence-L3 (2008-2009a). *Interpolation.* Université de Caen Basse-Normandie.

[6] Enseignements-Licence-L3 (2008-2009b). *Intégration Numérique.* Université de Caen Basse-Normandie.

[7] Faivre, R., Iooss, B., Mahévas, S., Makowski, D., and Monod, H. (2016). *Analyse de sensibilité et exploration de modèles : application aux sciences de la nature et de l'environnement.* Editions Quae.

[8] Goulet, V. (2018). *Méthodes numériques en actuariat avec R,Simulation stochastique.* école d'actuariat, Université Laval.

[9] Ledra, M., Rima, B., and Fatiha, F. (2016). *La méthode Monté Carlo et ses applications.*

[10] Medouakh, F. Z. (2018). *Méthodes de Monte-Carlo.* PhD thesis.

[11] Menini, C. (18 mai 2009). inégalité des accroissements finis.

[12] Pansu, P. (12 avril 2005). *THEOREMES D'ANALYSE.*

[13] Perrut, A. (Août 2010). Cours de probabilités et statistiques.

[14] Richard, L. and Burden, J. (2001). Douglas faires. numerical analysis . thomson learning.

[15] Roger, M. (2008). Méthodes de monte-carlo. *Service de Physique de l'Etat Condensé CEA Saclay.*

[16] RUBINO, G. and TUFFIN, B. (2007). *Simulations et méthodes de Monte Carlo.* Ed. Techniques Ingénieur.

[17] Touzani, R. (octobre 2013). Introduction à l'analyse numérique.

[18] Varet, S. (2010). *Développement de méthodes statistiques pour la prédiction d'un gabarit de signature infrarouge*. PhD thesis, Université Paul Sabatier-Toulouse III.

Annexes

Estimations successives de l'intégrale par la méthode Monte-Carlo avec des échantillons de plus en plus grands.

```
f <- function(x) x^2.2 * exp(-x/10)
x <- runif(1e2, 2, 5)
3 * mean(f(x))
x <- runif(1e3, 2, 5)
3 * mean(f(x))
x <- runif(1e4, 2, 5)
3 * mean(f(x))
x <- runif(1e5, 2, 5)
3 * mean(f(x))
x <- runif(1e6, 2, 5)
3 * mean(f(x))

op <- par(mfrow = c(2, 2)) # 4 graphiques en grille 2 x 2
curve(f(x), xlim = c(2, 5), lwd = 2, main = "Vraie fonction")
x <- runif(1e2, 2, 5)
plot(x, f(x), main = "n = 100",
pch = 21, bg = 3)
x <- runif(1e3, 2, 5)
plot(x, f(x), main = "n = 1000",
pch = 21, bg = 3)
x <- runif(1e4, 2, 5)
plot(x, f(x), main = "n = 10 000",
pch = 21, bg = 3)
x <- runif(1e5, 2, 5)
plot(x, f(x), main = "n = 100 000",
pch = 21, bg = 3)
x <- runif(1e6, 2, 5)
plot(x, f(x), main = "n = 1000 000",
pch = 21, bg = 3)
par(op)                    # revenir aux paramètres par défaut
```

Code Listing 3.1 – Monte-Carlo par R dimension 1

Estimations successives de l'intégrale double par la méthode Monte-Carlo avec des échantillons de plus en plus grands.

```
u <- runif(1e2, 0, 1.25)
v <- runif(1e2, 0, 1.25)
mean(sqrt(4 - u^2 - v^2)) * 1.25^2
u <- runif(1e3, 0, 1.25)
v <- runif(1e3, 0, 1.25)
mean(sqrt(4 - u^2 - v^2)) * 1.25^2
u <- runif(1e4, 0, 1.25)
```

```
v <- runif(1e4, 0, 1.25)
mean(sqrt(4 - u^2 - v^2)) * 1.25^2
## Graphiques de la vraie fonction et des points où celle-ci
## est évaluée avec la méthode Monte Carlo. Pour faire un
## graphique en trois dimensions, nous pouvons utiliser la
## fonction 'persp'.
op <- par(mfrow = c(2, 2), mar = c(1, 1, 2, 1))
f <- function(x, y) sqrt(4 - x^2 - y^2)
x <- seq(0, 1.25, length = 25)
y <- seq(0, 1.25, length = 25)
persp(x, y, outer(x, y, f), main = "Vraie fonction",
zlim = c(0, 2), theta = 120, phi = 30, col = 3,
zlab = "z", ticktype = "detailed")
## Pour faire les trois autres graphiques, nous allons d'abord
## créer des graphiques en trois dimensions vides, puis y
## ajouter des points avec la fonction 'points'. La fonction
## 'trans3d' sert à convertir les coordonnées des points dans
## la projection utilisée par 'persp'.
u <- runif(1e2, 0, 1.25)
v <- runif(1e2, 0, 1.25)
res <- persp(x, y, matrix(NA, length(x), length(y)),
main = "n = 100",
zlim = c(0, 2), theta = 120, phi = 30,
zlab = "z", ticktype = "detailed")
points(trans3d(u, v, f(u, v), pm = res),
pch = 21, bg = 3)
u <- runif(1e3, 0, 1.25)
v <- runif(1e3, 0, 1.25)
res <- persp(x, y, matrix(NA, length(x), length(y)),
main = "n = 1000",
zlim = c(0, 2), theta = 120, phi = 30,
zlab = "z", ticktype = "detailed")
points(trans3d(u, v, f(u, v), pm = res),
pch = 21, bg = 3)
u <- runif(1e4, 0, 1.25)
v <- runif(1e4, 0, 1.25)
res <- persp(x, y, matrix(NA, length(x), length(y)),
main = "n = 10 000",
zlim = c(0, 2), theta = 120, phi = 30,
zlab = "z", ticktype = "detailed")
points(trans3d(u, v, f(u, v), pm = res),
pch = 21, bg = 3)
par(op) # revenir aux paramètres par défaut
```

Code Listing 3.2 – Monte-Carlo par R dimension 2

```
# -*- coding: utf-8 -*-
""" Created on Wed May 22 13:24:27 2019
%@author: Abekhti.A """

import numpy.random
import math
def integrationDisque(fonction,R,N):
    theta = 2*numpy.pi*numpy.random.random_sample(N)
    r = R*numpy.sqrt(numpy.random.random_sample(N))
    p = 1.0/(numpy.pi*R*R)
    somme = 0.0
    somme2 = 0.0
    for i in range(N):
        f = fonction(r[i],theta[i])
        somme += f
```

```
            somme2 += f*f
        moyenne = somme/(p*N)
        variance = somme2/(p*p*N)-moyenne*moyenne
        return (moyenne,math.sqrt(variance/N)*1.96)

def f(r,theta):
    return 1-r*r
(integrale,intervalle) = integrationDisque(f,1.0,10000)
print((integrale,intervalle))
```

Code Listing 3.3 – integration Disque

```
# -*- coding: utf-8 -*-
""" Created on Mon May  13 11:10 2019
%@author: Abekhti.A """

import random
import math
def volume(dim,R1,R2,N):
    somme = 0.0
    somme2 = 0.0
    R12 = R1*R1
    R22 = R2*R2
    p = 1.0/math.pow(2*R2,dim)
    for i in range(N):
        r2 = 0.0
        for d in range(dim):
            x = random.uniform(-R2,R2)
            r2 += x*x
        if r2>= R12 and r2<=R22:
            f = 1.0
        else:
            f = 0.0
        somme += f
        somme2 += f*f
    moyenne = somme/(p*N)
    variance = somme2/(p*p*N)-moyenne*moyenne
    return (moyenne,variance,math.sqrt(variance/N)*1.96)
(m,v,e) = volume(3,0,1,10000)
print(m,v,e)
(m,v,e) = volume(5,0,1,10000)
print(m,v,e)
(m,v,e) = volume(5,0,1,10000*90)
print(m,v,e)
(m,v,e) = volume(5,0.9,1,10**5)
print(m,v,e)
(m,v,e) = volume(7,0,1,10**7)
print(m,v,e)
```

Code Listing 3.4 – volume

```
# -*- coding: utf-8 -*-
""" Created on Mon May  6 10:18:39 2019
%@author: Abekhti.A """

import numpy
import numpy.random
from matplotlib.pyplot import *
import math

def integration2D(fonction,N,param,np):
```

```
    image = numpy.zeros((np,np))
    x = -1+2*numpy.random.random_sample(N)
    y = -1+2*numpy.random.random_sample(N)
    p = 1.0/4
    somme = 0.0
    somme2 = 0.0
    for i in range(N):
        f = fonction(x[i],y[i],param)
        somme += f
        somme2 += f*f
        if f!=0:
            image[int((y[i]+1)/2*np),int((x[i]+1)/2*np)] = 1.0
    moyenne = somme/(p*N)
    variance = somme2/(p*p*N)-moyenne*moyenne
    return (moyenne,math.sqrt(variance/N)*1.96,image)

def f_Jz(x,y,param):
    R1 = param[0]
    R2 = param[1]
    a = param[2]
    r2 = x*x+y*y
    if r2 > 1 or r2 < R1*R1:
        return 0.0
    if r2 > R2*R2:
        return r2
    if abs(x) < a:
        return r2
    if abs(y) < a:
        return r2
    return 0.0
param = [0.1,0.8,0.15]
N = 10**5
(integrale,intervalle,image) = integration2D(f_Jz,N,param,500)
figure()
imshow(image,cmap='gray')
#-----------------------------------------
print((integrale,intervalle))
```

Code Listing 3.5 – Échantillonnage uniforme

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jun 15 20:13:40 2019

@author: Abekhti. A
"""

import numpy
import numpy.random
from matplotlib.pyplot import *
import math

def integration2d_disque(fonction,N,param,np):
    image = numpy.zeros((np,np))
    theta = 2*numpy.pi*numpy.random.random_sample(N)
    r = numpy.power(numpy.random.random_sample(N),0.5)
    somme = 0.0
    somme2 = 0.0
    p = 1.0/numpy.pi
    for i in range(N):
        f = fonction(r[i],theta[i],param)/p
```

```
        somme += f
        somme2 += f*f
        if f!=0:
            image[int((r[i]*math.cos(theta[i])+1)/2*np),int((r[i]*math.sin(
    theta[i])+1)/2*np)]=1.0
    moyenne = somme/N
    variance = somme2/N-moyenne*moyenne
    return (moyenne,math.sqrt(variance/N)*1.96,image)
def f_Jz_polaire(r,theta,param):
    R1 = param[0]
    R2 = param[1]
    a = param[2]
    r2 = r*r
    if r2 > 1 or r2 < R1*R1:
        return 0.0
    if r2 > R2*R2:
        return r2
    if abs(r*math.cos(theta)) < a:
        return r2
    if abs(r*math.sin(theta)) < a:
        return r2
    return 0.0

N=10**5
param = [0.1,0.8,0.15]
(integrale,intervalle,image) = integration2d_disque(f_Jz_polaire,N,param,500)
figure()
imshow(image,cmap='gray')
print((integrale,intervalle))
```

Code Listing 3.6 – Échantillonnage préférentiel

Programme Python des méthodes (Méthode des Rectangles, Méthode des Trapèzes, Méthode de Point milieu, Méthode de Simpson et Méthode de Monte-Carlo)
Dimension 1

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 15 10:17:24 2019

@author: Abekhti.A
"""

import math
import random
import sympy

def f(x) :
    return sympy.sqrt(x**2+sympy.exp(5*x));
#********************Methode de Rectangles*********************
def Rectangle(a,b,n) :
    S=0;
    for i in range(n) :
        x=a+i*(b-a)/n;
        S=S+f(x);
    S=((b-a)/n)*S;
    return S;
#********************Methodee de Point milieux*********************
def Milieu(a,b,n) :
    S=0;
    for i in range(n) :
```

```
        x=a+i*(b-a)/n;
        y=a+(i+1)*(b-a)/n;
        x=(x+y)/2;
        S=S+f(x);
    S=((b-a)/n)*S;
    return S;
#******************Methode de Trapezes********************
def Trapeze(a,b,n):
    S=0;
    for i in range(n) :
        x=a+i*(b-a)/n;
        y=a+(i+1)*(b-a)/n;
        S=S+f(x)+f(y);
    S=((b-a)/(2*n))*S;
    return S;
#******************Methode de Simpson********************
def Simpson(a,b,n) :
    S=0;
    for i in range(n) :
        x=a+i*(b-a)/n;
        y=a+(i+1)*(b-a)/n;
        z=(x+y)/2;
        S=S+f(x)+4*f(z)+f(y);
    S=((b-a)/(6*n))*S;
    return S;
#******************Methode de Monte-Carlo********************
def Montecarlo(a,b,n) :
    S=0;
    for i in range(n) :
        x=random.uniform(0,1);
        S=S+f(x);
    S=(1/n)*S;
    return S;
#*********************
def Exact(a,b) :
    I_exact =sympy.integrate(f(x),(x, a,b))
    I=I_exact.evalf()
    return I
#****************** Comparaison********************
def Integral(a,b,n) :
    I =Exact(a,b);
    print ("|Exact integral:" "%0.12f" % I,"|")
    S =Rectangle(a,b,n);
    print ("-------------------------------------------------------------" );
    print ("|Rectangle :" "%0.12f" % S, "|erreur =""%0.12f" % abs(I-S),"|" );
    S =Milieu(a,b,n);
    print ("|Milieu    :""%0.12f" % S, "|erreur =""%0.12f" % abs(I-S),"|");
    S =Trapeze(a,b,n);
    print ("|Trapeze   :""%0.7f" % S, "|erreur =""%0.12f" % abs(I-S) );
    S =Simpson(a,b,n);
    print ("|Simpson   :""%0.12f" %S, "|erreur =""%0.12f" % abs(I-S),"|");
    S =Montecarlo(a,b,n);
    print ("|Monte carlo:""%0.12f" % S, "|erreur =""%0.12f" % abs(I-S),"|");
Integral(0,1,10000)
```

Code Listing 3.7 – Comparaisons dimension 1

Dimension 2

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 15 11:48:48 2019
```

```
4
5 @author: Abekhti.A
6 """
7
8 import random
9 import sympy
10 x,y=sympy.symbols('x y')
11 def f(x,y) :
12     return  x*y+y**2+1
13  #******************Methode de Rectangles*******************
14 def Rectangle_double1(f,a,b,c,d,nx,ny) :
15     S=0;
16     for i in range(nx) :
17         for j in range(ny) :
18             x1=a+i*(b-a)/nx;
19             x2=c+j*(d-c)/ny;
20             S=S+f(x1,x2);
21     S=((b-a)/nx)*((d-c)/ny)*S;
22     return S;
23 #******************Methode de Point milieux*******************
24 def Milieu_double1(f,a,b,c,d,nx,ny):
25     S=0;
26     for i in range(nx) :
27         for j in range(ny) :
28             x1=a+i*(b-a)/nx;
29             y1=a+(i+1)*(b-a)/nx;
30             m1=(x1+y1)/2;
31             x2=c+j*(d-c)/ny;
32             y2=c+(j+1)*(d-c)/ny;
33             m2=(x2+y2)/2;
34             S=S+f(m1,m2);
35     S=((b-a)/nx)*((d-c)/ny)*S;
36     return S
37
38 #******************Methode de Trapezes*******************
39 def Trapeze_double1(f,a,b,c,d,nx,ny):
40     S=0;
41     for i in range(nx) :
42         x1=a+i*(b-a)/nx;
43         y1=a+(i+1)*(b-a)/nx;
44         for j in range(ny) :
45             x2=a+j*(d-c)/ny;
46             y2=a+(j+1)*(d-c)/ny;
47             S=S+f(x1,x2)+f(y1,y2);
48     S=((b-a)/(2*nx))*((d-c)/(2*ny))*S;
49     return S;
50
51
52 #******************Methode de Simpson*******************
53 def Simpson_double1(f,a,b,c,d,nx,ny):
54     S=0;
55     for i in range(nx) :
56         for j in range(ny) :
57             x1=a+i*(b-a)/nx;
58             y1=a+(i+1)*(b-a)/nx;
59             z1=(x1+y1)/2;
60             x2=c+j*(d-c)/ny;
61             y2=c+(j+1)*(d-c)/ny;
62             z2=(x2+y2)/2;
63             S=S+f(x1,x2)+4*f(z1,z2)+f(y1,y2);
```

```
    S=((b-a)/(6*nx))*((d-c)/(ny))*S;
    return S;

#******************Methode de Monte-Carlo*******************
def Montecarlo_double1(f,a,b,c,d,nx) :
    S=0;
    for i in range(nx) :
        x=random.uniform(a,b)
        y=random.uniform(c,d);
        S=S+f(x,y);
    S=(((b-a)*(d-c))/(nx))*S;
    return S;
#************************Comparaison**********************
def Integral(a,b,c,d,nx,ny) :
    I_expected =sympy.integrate(f(x,y), (x,a,b), (y,c,d))
    print ("|I_expected :" "%0.12f" % I_expected ) ;
#**********************************
    print ("-------------------------------------------------------" );
    I_computed1 =Rectangle_double1(f,a,b,c, d,nx,ny) ;
    print ("|Rectangle_double1 :" "%0.12f" % I_computed1,"|erreur =""%0.12f"
   % abs(I_expected-I_computed1),"|" );
#************************
    I_computed2 =Milieu_double1(f,a,b,c, d,nx,ny) ;
    print ("| Milieu_double1 :" "%0.12f" % I_computed2,"|erreur =""%0.12f" %
   abs(I_expected-I_computed2),"|" );
#**********************
    I_computed3 =Trapeze_double1(f,a,b,c, d,nx,ny) ;
    print ("|Trapeze_double1 :" "%0.12f" % I_computed3,"|erreur =""%0.12f" %
   abs(I_expected-I_computed3),"|" );
#**********************
    I_computed4 =Simpson_double1(f,a,b,c, d,nx,ny);
    print ("|Simpson_double1 :" "%0.12f" % I_computed4,"|erreur =""%0.12f" %
   abs(I_expected-I_computed4),"|" );
#**************************
    I_computed5 =Montecarlo_double1(f,a,b,c, d,nx);
    print ("|Montecarlo_double1 :" "%0.12f" % I_computed5,"|erreur =""%0.12f"
    % abs(I_expected-I_computed5),"|" );
Integral(1,2,0,3,10000,20000)
```

Code Listing 3.8 – Comparaisons dimension 2

Dimension 3

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 15 19:24:13 2019

@author: Abekhti.A
"""

import random
import sympy
x,y,z=sympy.symbols('x y z')
def g(x,y,z) :
    return  z*sympy.exp(x**2+y**2)
#******************Methode de Rectangles*******************
def Rectangle_triple1(g,a,b,c,d,e,f,n1,n2,n3) :
    S=0;
    for i in range(n1) :
        for j in range(n2) :
            for k in range(n3) :
                x=a+i*(b-a)/n1;
```

```
                y=c+j*(d-c)/n2;
                z=e+k*(f-e)/n3;
                S=S+g(x,y,z);
    S=((b-a)/n1)*((d-c)/n2)*((f-e)/n3)*S;
    return S;
#******************Methode de Trapezes*******************
def Trapeze_triple1(g,a,b,c,d,e,f,n1,n2,n3):
    S=0;
    for i in range(n1) :
        x1=a+i*(b-a)/n1;
        y1=a+(i+1)*(b-a)/n1;
        for j in range(n2) :
            x2=c+j*(d-c)/n2;
            y2=c+(j+1)*(d-c)/n2;
            for k in range(n3) :
                x3=e+k*(f-e)/n3;
                y3=e+(k+1)*(f-e)/n3;
                S=S+g(x1,x2,x3)+g(y1,y2,y3);
    S=((b-a)/(2*n1))*((d-c)/(n2))*((f-e)/(n3))*S;
    return S;
#******************Methode de Simpson*******************
def Simpson_triple1(g,a,b,c,d,e,f,n1,n2,n3):
    S=0;
    for i in range(n1) :
        x1=a+i*(b-a)/n1;
        y1=a+(i+1)*(b-a)/n1;
        z1=(x1+y1)/2;
        for j in range(n2) :
            x2=c+j*(d-c)/n2;
            y2=c+(j+1)*(d-c)/n2;
            z2=(x2+y2)/2;
            for k in range(n3) :
                x3=e+k*(f-e)/n3;
                y3=e+(k+1)*(f-e)/n3;
                z3=(x3+y3)/2;
                S=S+g(x1,x2,x3)+4*g(z1,z2,z3)+g(y1,y2,y3);
    S=((b-a)/(6*n1))*((d-c)/(n2))*((f-e)/(n3))*S;
    return S;

#******************methode de Point milieux*******************
def Milieu_triple1(g,a,b,c,d,e,f,n1,n2,n3):
    S=0;
    for i in range(n1) :
        x1=a+i*(b-a)/n1;
        y1=a+(i+1)*(b-a)/n1;
        m1=(x1+y1)/2;
        for j in range(n2) :
            x2=c+j*(d-c)/n2;
            y2=c+(j+1)*(d-c)/n2;
            m2=(x2+y2)/2;
            for k in range(n3) :
                x3=e+k*(f-e)/n3;
                y3=e+(k+1)*(f-e)/n3;
                m3=(x3+y3)/2;
                S=S+g(m1,m2,m3);
    S=((b-a)/n1)*((d-c)/n2)*((f-e)/n3)*S;
    return S
#******************Methode de Monte-Carlo*******************
def Montecarlo_triple1(g,a,b,c,d,e,f,n1) :
    S=0;
```

```
80      for i in range(n1) :
81          x=random.uniform(a,b)
82          y=random.uniform(c,d);
83          z=random.uniform(e,f);
84          S=S+g(x,y,z);
85      S=(((b-a)*(d-c)*(f-e))/(n1))*S;
86      return S;
87
88 #************************Comparaison**********************
89 def Integral(a,b,c,d,e,f,n1,n2,n3) :
90      I_expected =sympy.integrate(g(x,y,z), (x,a,b), (y,c,d),(z,e,f))
91      print ("|I_expected :" "%0.12f" % I_expected ) ;
92 #***********************************
93      print ("-------------------------------------------------------" );
94      I_computed1 =Rectangle_triple1(g,a,b,c,d,e,f,n1,n2,n3) ;
95      print ("|Rectangle_triple1 :" "%0.12f" % I_computed1,"|erreur =""%0.12f"
        % abs(I_expected-I_computed1),"|" );
96 #************************
97      I_computed2 =Milieu_triple1(g,a,b,c, d,e,f,n1,n2,n3) ;
98      print ("| Milieu_triple1 :" "%0.12f" % I_computed2,"|erreur =""%0.12f" %
        abs(I_expected-I_computed2),"|" );
99 #************************
100     I_computed3 =Trapeze_triple1(g,a,b,c, d,e,f,n1,n2,n3) ;
101     print ("|Trapeze_triple1 :" "%0.12f" % I_computed3,"|erreur =""%0.12f" %
        abs(I_expected-I_computed3),"|" );
102 #************************
103     I_computed4 =Simpson_triple1(g,a,b,c, d,e,f,n1,n2,n3);
104     print ("|Simpson_triple1 :" "%0.12f" % I_computed4,"|erreur =""%0.12f" %
        abs(I_expected-I_computed4),"|" );
105 #****************************
106     I_computed5 =Montecarlo_triple1(g,a,b,c, d,e,f,n1);
107     print ("|Montecarlo_triple1 :" "%0.12f" % I_computed5,"|erreur =""%0.12f"
        % abs(I_expected-I_computed5),"|" );
108 Integral(0,1,0,1,0,1,5,3,1)
```

Code Listing 3.9 – Comparaisons dimension 3

ملخص

في حساب التكامل، بعض الدوال يصعب حساب تكاملها. لإيجاد قيمة تقريبية لهذا التكامل، نستخدم طرقًا عددية تقريبية مثل طريقة المستطيل، طريقة النقطة المتوسطة، طريقة شبه المنحرف وطريقة سيمبسون، لكن هذه الطرق بطيئة جدًا في الأبعاد الكبيرة.

لحل هذه المشكلة، نستخدم طريقة أخرى تسمح لنا بالعثور على قيمة تقريبية دقيقة بأبعاد كبيرة وتتقارب بسرعة هذه الطريقة هي طريقة مونت كارلو.

الكلمات المفتاحية: تقريبية، طريقة المستطيلات، طريقة النقطة المتوسطة، طريقة شبه المنحرف، طريقة سيمبسون، تتقارب، الأبعاد الكبيرة، طريقة مونت كارلو.

Résumé

Dans le calcul d'intégration, certaines fonctions son intégration est difficile à calculer. Pour trouver une valeur approximative pour cette intégration, nous recourons à des méthodes numériques approximatives telles que la méthode de Rectangles, la méthode de Point Milieu, la méthode de Trapèzes et la méthode de Simpson, mais ces méthodes sont très lentes dans les grandes dimensions.

Pour résoudre ce problème, nous utilisons une autre méthode qui nous permettons de trouver une valeur approximative précise dans les grandes dimensions et converge rapidement cette méthode est intitulée la méthode de Monte-Carlo.

Mots clés: approximative, méthode de Rectangles, méthode de Point Milieu, méthode de Trapèzes, méthode de Simpson, converge, grandes dimensions, méthode de Monte-Carlo.

Abstract

In the integration calculation, some functions its integration is difficult to calculate. To find an approximate value for this integration, we use approximate numerical methods such as the Rectangle method, the Middle Point method, the trapezoidal method and the Simpson method, but these methods are very slow in large dimensions.

To solve this problem, we use another method that allows us to find a precise approximate value in large dimensions and quickly converge this method is the Monte Carlo method.

Key words: approximate, Rectangles method, Middle Point method, trapezoidal method, Simpson method, converge, large scale, Monte Carlo method.