Simulation et Optimisation – TP2

# Introduction

Dans ce travail pratique, il nous est demandé de mettre en œuvre et de comparer plusieurs méthodes de Monte-Carlo :

1. Méthode d’intégration par échantillonnage uniforme,
2. Méthode d’intégration par échantillonnage préférentiel,
3. Méthode d’intégration par échantillonnage uniforme avec variable de contrôle.

Ces méthodes seront utilisées dans le but de déterminer l’aire de la fonction

Sur l’intervalle [0,15].

Dans un premier temps, nous testerons l’implémentation des différentes méthodes en générant un certain nombre de données et en analysant les résultats obtenus. Une fois ceci fait, nous passerons l’analyse des performances de chacune de ces méthodes en observant la rapidité de convergence vers la valeur de l’aire de l’intégrale.

## Notes préalables

* Le code a été compilé en C++11 en utilisant mingw 4.8.1.
* Les mesures ont été effectuées avec un processeur Intel®Core™ i7-2760QM CPU @ 2.40GHz.

# Test de l’implémentation

TODO

## Premier jeu de données – Uniforme (5,15)

*Espérance*: **10**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Méthode | Unité mesurée | Moyenne | Écart-type | IC | Largeur IC |
| 1 | Réalisations | 9. 99876 | 2.88805 | [9.9931,10.0044] | 0.0113212 |
| Temps [s] | 0.0564822 | 0.002313 | [0.0564181,0.0565463] | 0.000128226 |
| 2 | Réalisations | 10.0035 | 2.88826 | [9.9978,10.0091] | 0.011322 |
| Temps [s] | 0.0806404 | 0.00373474 | [0.0805369,0.0807439] | 0.000207043 |
| 3 | Réalisations | 9.99826 | 2.88586 | [9.9926,10.0039] | 0.0113126 |
| Temps [s] | 0.0538264 | 0.0022246 | [0.0537647,0.0538881] | 0.000123326 |

En regardant ce tableau, on remarque tout d’abord que les résultats des trois méthodes sont très proches les uns des autres. On constate également que les moyennes obtenues sont relativement proches de l’espérance. En jetant un œil aux 3 intervalles de confiance (à 95%), nous pouvons constater que l’espérance théorique s’y trouve (bien entendu, il aurait pu arriver qu’elle soit en dehors de l’IC, mais elle se retrouve statistiquement 95% du temps dedans). Tout ceci peut nous laisser penser aux premiers abords que l’implémentation des algorithmes est correcte.

Nous nous intéresserons aux temps d’exécution dans la section dédiée à l’analyse des performances.

## Deuxième jeu de données – Mélange de deux variables triangulaires

*Espérance*: **8.5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Méthode | Unité mesurée | Moyenne | Écart-type | IC | Largeur IC |
| 1 | Réalisations | 8.49921 | 4.62769 | [8.49014,8.50828] | 0.0181405 |
| Temps [s] | 0.167804 | 0.00859517 | [0.167566,0.168042] | 0.000476492 |
| 2 | Réalisations | 8.4945 | 4.62789 | [8.48543,8.50357] | 0.0181413 |
| Temps [s] | 0.0875782 | 0.00301226 | [0.0874947,0.0876617] | 0.000166991 |
| 3 | Réalisations | 8.49832 | 4.62784 | [8.48925,8.50739] | 0.0181411 |
| Temps [s] | 0.0682266 | 0.00263762 | [0.0681535,0.0682997] | 0.000146222 |

A nouveau, on peut remarquer que les résultats des trois méthodes sont similaires, que la moyenne est proche de l’espérance et que cette dernière se trouve dans les trois intervalles de confiance.

## Troisième jeu de données – Profil plutôt plat

*Espérance*: **10.758**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Méthode | Unité mesurée | Moyenne | Écart-type | IC | Largeur IC |
| 1 | Réalisations | 10.7559 | 5.22366 | [10.7456,10.7661] | 0.0204767 |
| Temps [s] | 0.0793284 | 0.00326109 | [0.079238,0.0794188] | 0.000180785 |
| 2 | Réalisations | 10.7412 | 5.20638 | [10.731,10.7514] | 0.020409 |
| Temps [s] | 0.0922574 | 0.0027182 | [0.0921821,0.0923327] | 0.000150689 |
| 3 | Réalisations | 10.7552 | 5.22255 | [10.745,10.7655] | 0.0204724 |
| Temps [s] | 0.0745272 | 0.00268177 | [0.0744529,0.0746015] | 0.00014867 |

On peut faire les mêmes commentaires que précédemment ici aussi, la seule exception étant l’espérance n’étant pas dans l’IC (relatif aux réalisations de la variable aléatoire de la méthode 2). On remarque alors un exemple où l’espérance n’est pas dans l’intervalle de confiance.

## Quatrième jeu de données – Profil accidenté

*Espérance*: **10.2616**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Méthode | Unité mesurée | Moyenne | Écart-type | IC | Largeur IC |
| 1 | Réalisations | 10.2598 | 5.70906 | [10.2486,10.2709] | 0.0223795 |
| Temps [s] | 0.257825 | 0.00595176 | [0.25766,0.25799] | 0.000329949 |
| 2 | Réalisations | 10.2584 | 5.70699 | [10.2472,10.2696] | 0.0223714 |
| Temps [s] | 0.0954716 | 0.00334572 | [0.0953789,0.0955643] | 0.000185477 |
| 3 | Réalisations | 10.259 | 5.70972 | [10.2478,10.2702] | 0.0223821 |
| Temps [s] | 0.08005 | 0.00264168 | [0.0799768,0.0801232] | 0.000146448 |

A nouveau, les mêmes remarques peuvent être faites au sujet de ce tableau.

# Analyse des performances

## Comportement des fonctions

Pour **l’acceptation-rejet** « bête et méchante », il faut enfermer la fonction dans le rectangle , avec étant le maximum des ordonnées des points constituant la fonction. Un point (X,Y) sera généré dans ce rectangle. On accepte X si Y est sous la fonction à l’abscisse X. Un profil plat implique que la fonction sera plutôt proche du « haut » du rectangle. Dans le cas de l’uniforme, comme chaque ordonnée des points de l’uniforme est égal à , le point généré sera forcément accepté (il ne peut être généré au-dessus de la fonction). Il s’agit donc d’un des cas idéaux d’utilisation de cette méthode.

Pour résumer, quand le profil de la fonction est relativement « plat », l’acceptation-rejet reste en lice avec les autres algorithmes (comme sur les figures 1 et 3 ci-après). Dans le cas contraire (beaucoup de variations entre les ordonnées des points), c’est la catastrophe. Le rectangle englobant la fonction se trouve être énorme, donc on devra générer beaucoup de points pour tomber sous la fonction (figures 2 et surtout 4).

Pour la **méthode des mélanges couplée à l’approche géométrique**, on devrait avoir des temps d’exécution approximativement les mêmes, qu’importe la fonction. En effet, pour cette méthode, pour un « morceau » de fonction donné, on génère un point (X,Y) et on l’accepte à chaque fois : s’il est sous le morceau de fonction, on retourne X et dans le cas contraire on effectue une symétrie sur X. En observant les figures de 1 à 4, on peut constater que les temps convergent effectivement vers une même valeur, ce qui valide notre hypothèse.

Enfin, pour la **méthode des mélanges couplée à la méthode des fonctions inverses**, on a deux cas pour un morceau de fonction donné: soit les ordonnées des points définissant le morceau de fonction sont égales et le calcul est relativement simple, soit elles sont différentes et le calcul est un peu plus complexe (racine carrée, division, etc).

On s’attend donc à avoir pour l’uniforme un des meilleurs temps d’exécution et également pour les autres fonctions (le temps sera plus grand que l’uniforme). C’est bien la tendance que l’on peut constater sur les graphiques : le temps est le meilleur pour l’uniforme (figure 1) et légèrement moins bien pour le reste des fonctions (figures 2 à 4). Dans ce dernier cas, il est approximativement le même pour toutes.

Finalement, on remarque que, en vertu des idées exprimées précédemment, pour un profil plutôt plat, les méthodes 1 et 3 sont idéales et que pour un profil accidenté, la méthode 1 est catastrophique mais la 2 et 3 sont plus appropriées. La méthode 3 est visiblement la meilleure et à utiliser dans toutes les situations.

Figure 1 Figure 2

Figure 3 Figure 4

Note : on remarque sur les figures que la borne inférieure et supérieure des IC sont très proches. Comme on veut représenter tous les IC en utilisant une même échelle, on ne voit malheureusement pas la différence entre la borne inférieure, supérieure et la moyenne…

On peut tout de même constater que les IC ont tous une largeur similaire et que l’on peut ainsi les comparer, ce qui permet de bien se rendre compte de l’efficacité des 3 algorithmes.