TP de simulations de Monte-Carlo, 2019

Questions

1D Random Walk :

Montrer que la marche aléatoire conduit à des distributions de déplacements équivalente à ce qu'on observe pour la diffusion de composés chimiques.

Questions :

1. Générer et représenter une marche aléatoire 1D avec n pas pour 1 marcheur
2. Générer et représenter une marche aléatoire 1D avec n pas pour N marcheurs + tester différents n et N
3. Etudier la distribution de probabilité du dernier pas d'une marche aléatoire avec un petit nombre de pas (n=6 par exemple); comparer quantitativement avec la distribution théorique attendue
4. Etudier une marche aléatoire avec un grand nombre de pas (n=1000 par exemple); vérifier (via un fit) que l'on obtient bien une distribution normale
5. Comparer les statistiques des résultats avec le modèle de la diffusion

Questions supplémentaires:

- 6) Regarder <x> pour un grand nombre de marcheurs (que peut-on en dire ?)
- 7) Regarder <x^2> pour un grand nombre de marcheurs + fitter par une droite et discuter de la valeur de la pente (cfr. coefficient de diffusion)
- 8) Envisager d'étendre l'étude à 2 ou 3 dimensions

Code python : 1D_random_walk.py

Jeter de pièces :

Simulation de lancers de pièce

Questions :

1. Générer n lancers de pièceset en mesurer la moyenne (tester différents n)
2. Générer N moyennes de n lancers de pièce
3. Etudier la distribution de probabilité de la question 2 pour des grands N et n, fitter une loi normale et comparer avec la valeur théorique attendue de mu et sigma
4. Comparer les résultats obtenus pour différents n (en gardant N fixe), que peut-on en dire?

Questions supplémentaires

- 5) Adapter le code (de manière efficace) pour effectuer des lancers de dés au lieu des lancers de pièces
- 6) Et un dé de type tétrakihexaèdre ?

Code python : throw_coin_and_dice.py

Approximation du nombre pi - direct sampling :

Permet d'approcher pi par une méthode stochastique (Direct Sampling)

Questions :

1. Trouver une méthode stochastique pour approcher le nombre pi ⇒ Tirer n flechettes sur un carre avec de coté 2R et comptabiliser le nbre de flechettes à l'intérieur du cercle de rayon R inscrit au carré. Le rapport des deux surfaces donne pi/4.
2. Générer n lancers de flechettes sur la “cible” et approcher pi avec la méthode trouvée à la question (1)
3. Représenter graphiquement les lancers de flechettes sur la “cible”
4. Générer N approximations de pi en faisant n lancers + étudier la distribution de probabilité des N approximations (faire varier n et N)
5. Estimer l'écart quadratique moyen, sqrt(<(Nhits/N − pi/4)^2>) + représenter graphiquement. Cmt varie-t-il avec N ?

Code python : Pi_approximation_by_direct_sampling.py

Approximation du nombre pi - Markov chain :

Permet d'approcher pi par une méthode stochastique (Markov chain)

Questions :

1. Trouver une méthode stochastique utilisant une marche aléatoire pour approcher le nombre pi ⇒ Attention besoin de gérer la question du rejet (cad lorsque le prochain pas est en dehors de la “cible”), de la longueur du pas et du départ de la marche !!!
2. Approcher pi avec la méthode trouvée à la question (1) pour une marche avec un nbre de pas suffisamment grand de pas + représenter graphiquement la marche
3. Générer N approximations de pi en faisant n marches + étudier la distribution de probabilité des N approximations (faire varier n et N)
4. Estimer l'écart quadratique moyen sqrt(<(Nhits/N − pi/4)**2>) + représenter graphiquement. Cmt varie-t-il avec N ?
5. Etudier la dépendance de la condition initiale sur l'estimation de pi pour une marche suffisamment longue
6. Etudier la dépendance de la longueur du pas sur la qualité de l'approximation de pi

Questions supplémentaires

- 7) Etudier la dépendance de la longueur du pas sur le taux acception. Que peut-on en dire?

Code python : Pi_approximation_by_markov_chain.py

Pebble Game 3x3 :

Etudier Le cas d'un terrain quadrillé en 3×3 (cas plus simple que le problème précédent) et comprendre pourquoi, on empile des cailloux lorsque l'on rejette les pas qui sortent de la cible. De plus, cela permet d'introduire l'algorithm de Metropolis.

Questions :

1. Montrer théoriquement qu'empiler des cailloux permet d'uniformément parcourir le terrain de jeu
2. Verifier par simulation que cet empilement permet bien de parcourir uniformément le terrain de jeu
3. Montrer théoriquement que l'algorithme de Metropolis est en accord avec la detailled condition balance
4. Etudier un terrain ou les probabilités de presence sont inhomogenes a l'aide d'un algorithme de metropolis

Questions supplémentaires:

- 5) Montrer que si on impose des cond. périodiques à un terrain homogène, la probabilité de presence est uniforme sur l'entièreté du terrain sans utiliser de condition de rejet
- 6) Envisager l'étude sur un terrain autre que 3×3. Le code devra être adaptatif (cad ne pas devoir modifier l'entièreté du code) pour décrit un terrain NxM

Code python : Pebble3x3.py