Télécharger le fichier metropolis.py à l'adresse

`http://www.math.u-psud.fr/~meliot/markov/metropolis.py`

Ouvrir un terminal, puis lancer Sage avec la commande `sage -n jupyter`. Copier l'intégralité du contenu de `metropolis.py` dans une cellule de la feuille de calcul, et l'exécuter (Shift+Enter).

1 Modèle d'Ising

1. Créer une configuration de spins sur une grille de taille $N \times N = 10 \times 10$ avec la commande `I = IsingConfiguration(10)`. Expérimenter les méthodes suivantes (pour appeler une méthode `method` sur `I`, on tapera `I.method()`, et `I.method(arg)` si la méthode admet pour arguments `arg`) :

- `plot()` dessine la configuration de spins, le rouge correspondant aux spins $+1$ et le gris aux spins -1.
- `energy()` calcule l'énergie de la configuration (on pourra éventuellement vérifier le calcul sur un exemple de plus petite taille).
- `magnetization()` calcule la somme de tous les spins.
- `random_site()` retourne un site aléatoire de la grille.
- `switch_site(y, u, beta)` modifie le site y en

$$\begin{cases} +1 & \text{si } u < \frac{\exp(\beta \sum_{x\sim y} I(x))}{2\cosh(\beta \sum_{x\sim y} I(x))}, \\ -1 & \text{sinon.} \end{cases}$$

- `transform(n, beta)` applique n transitions markoviennes de paramètre β à la configuration. Essayer d'abord avec un petit nombre de transitions ($n < 10$), puis un plus grand nombre $n \sim 1000$. On prendra par exemple $\beta = 0.3$, et on dessinera la nouvelle configuration entre chaque essai. Commenter.

2. Pour chaque valeur de β dans $\{0.2, 0.3, 0.4, 0.5, 0.6\}$, approximer la loi $\mathbb{P}^{\text{Ising}}_{50,\beta}$ en appliquant $n = 100000$ transitions de paramètre β à une configuration de taille $N = 50$. On pourra utiliser la commande `MarkovIsingConfiguration(N, n, beta)`. Comparer les résultats.

3. Le programme `MarkovIsingConfiguration(N, n, beta)` admet comme argument optionnel

`boundary="positive"` ou `boundary="war"`

qui correspondent à la loi d'Ising conditionnellement au fait que la frontière soit entièrement positive, ou conditionnellement au fait que le bord gauche soit positif et que le bord droit soit négatif. Expérimentez ces deux options.

4. Le programme `RandomIsingConfiguration(N, beta)` utilise la méthode de couplage pour simuler exactement $\mathbb{P}^{\text{Ising}}_{N,\beta}$. Expérimentez-le (on commencera avec N suffisamment petit et $\beta < 0.5$, et on pourra faire suivre `N,beta` de l'argument optionnel `verbose=True` pour avoir un indicateur de progression du programme). Vérifier que le code applique bien la méthode de couplage.

2 Surfaces aléatoires

1. Créer un empilement vide dans le cube de taille $N \times N \times N = 5 \times 5 \times 5$ avec la commande `P = DimerCovering(5)`. Expérimenter les méthodes suivantes :
   - `plot()` dessine l'empilement. Cette méthode admet deux arguments optionnels `rotate` et `beautiful`, par défaut égaux à `True`. En particulier, la méthode `plot(True,False)` dessine la configuration de dimères correspondant à l'empilement.
   - `draw_honeycomb()` donne le système de coordonnées du réseau hexagonal sous-jacent à l'empilement.
   - `volume()` calcule le volume de l'empilement de cubes.
   - `switchup(Hexagon(a,b))` ajoute une case / effectue une rotation des dimères au niveau de l'hexagone de coordonnées a et b, si cela est possible. On définit de façon analogue la méthode `switchdown(Hexagon(a,b))`. Construire manuellement un empilement de volume au moins égal à 5. Vérifier que chaque ajout ou retrait de cube correspond à une rotation d'un hexagone dans la configuration de dimères.
   - `transform(n,q)` applique n transitions markoviennes de paramètre q à la configuration. Essayer d'abord avec un petit nombre de transitions ($n < 50$), puis un plus grand nombre $n \sim 1000$. On prendra par exemple $q = 1$ ou $q = 2$, et on dessinera le nouvel empilement entre chaque essai. Commenter.

2. Pour chaque valeur de q dans $\{0.5, 0.9, 1\}$, simuler la loi $\mathbb{P}^{\text{Dimer}}_{5,q}$ en appliquant 10000 transitions de paramètre q à un empilement de taille $5 \times 5 \times 5$. Commenter. Essayer en taille plus grande.

3. Le programme `RandomDimerCovering(N,q)` simule exactement la loi $\mathbb{P}^{\text{Dimer}}_{N,q}$ avec la méthode de couplage. Expérimentez (on pourra de nouveau utiliser l'argument optionnel `verbose=True` pour avoir une idée du temps de calcul).