Algorithme de Metropolis Hastings

Ouvrez un navigateur internet à l'adresse

http ://cermics.enpc.fr/∼jourdain/MC/MonteCarlo.html

L'objectif de ce TP est d'illustrer numériquement l'algorithme de Metropolis Hastings sur l'exemple du modèle de Fermi en physique statistique.

Dans ce modèle, il y a I types de sites. Pour $i \in \{1, \dots, I\}$, on note m_i le nombre de sites de type i et V_i le niveau d'énergie commun de ces sites. Chacun de ces sites est occupé par une particule au plus indépendamment des autres sites et ce avec probabilité $p_i = e^{-\frac{V_i}{k_B T}}$ où k_B est la constante de Boltzmann. Ainsi le nombre $X(i)$ de sites de type i occupés suit la loi binomiale de paramètre (m_i, p_i) que nous noterons μ_i. Les variables aléatoires $X(i)$ sont indépendantes. On note $S = \sum_{i=1}^I X(i)$ le nombre total de particules. Pour accéder à la description micro-canonique, on souhaite simuler suivant la loi conditionnelle de $(X(1), \dots, X(I))$ sachant $S = s$ où $s \in \mathbb{N}^*$:

$$\pi(x(1), \dots, x(I)) = 1_{\{x(1)+\dots+x(I)=s\}} \frac{\prod_{i=1}^I \mu_i(x(i))}{\mu_1 \star \dots \star \mu_I(s)}.$$

Notons que le calcul de la constante de normalisation de cette loi

$$\mu_1 \star \dots \star \mu_I(s) = \sum_{\substack{(x(1),\dots,x(I)) \in \mathbb{N}^I \\ \sum_{i=1}^I x(i) = s}} \prod_{i=1}^I \mu_i(x(i)) = \mathbb{P}(S = s)$$

est difficile. L'algorithme de Metropolis Hastings permet de simuler suivant π sans calculer cette constante.

Dépassant le modèle de Fermi, nous allons nous intéresser plus généralement au cas de lois μ_i sur $\mathbb{N}$ quelconques. Dans le cas où, pour tout $i \in \{1, \dots, I\}$, μ_i est la loi de Poisson de paramètre λ_i ($\mu_i(x(i)) = e^{-\lambda_i} \frac{\lambda_i^{x(i)}}{x(i)!}$ pour tout $x(i) \in \mathbb{N}$), S suit la loi de Poisson de paramètre $\Lambda = \sum_{i=1}^I \lambda_i$ et π est la loi multinomiale de paramètre $(s, \frac{\lambda_1}{\Lambda}, \dots, \frac{\lambda_I}{\Lambda})$:

$$\pi(x(1), \dots, x(I)) = 1_{\{x(1)+\dots+x(I)=s\}} \frac{s!}{\prod_{i=1}^I x(i)!} \prod_{i=1}^I \left(\frac{\lambda_i}{\Lambda}\right)^{x(i)}.$$

Le caractère explicite de π pour ce choix des μ_i va permettre de vérifier le bon fonctionnement de l'algorithme de Metropolis Hastings.

Notons $E = \{\mathbf{x} = (x(1), \dots, x(I)) \in \mathbb{N}^I : \sum_{i=1}^I x(i) = s\}$ et $\mathbf{e}_i$ le i-ème vecteur de la base canonique de $\mathbb{R}^I$. Le noyau de proposition sur E que nous allons utiliser dans l'algorithme de Metropolis Hastings est

$$Q(\mathbf{x}, \mathbf{y}) = \begin{cases} \frac{x(i)}{s(I-1)} \text{ si } \mathbf{y} = \mathbf{x} - \mathbf{e}_i + \mathbf{e}_j \text{ avec } 1 \le i \ne j \le I \text{ et } x(i) \ge 1, \\ 0 \text{ sinon.} \end{cases}$$

1. Soit $\mathbf{x} \in E$ et $(\mathcal{I}, \mathcal{J})$ un couple aléatoire à valeurs dans $\{1, \dots, I\}^2$ tel que pour tout $i, j \in \{1, \dots, I\}$, $\mathbb{P}(\mathcal{I} = i) = \frac{x(i)}{s}$ et $\mathbb{P}(\mathcal{J} = j | \mathcal{I} = i) = \frac{1_{j \neq i}}{I-1}$. Vérifier que $\mathbf{x} - \mathbf{e}_{\mathcal{I}} + \mathbf{e}_{\mathcal{J}}$ est distribué suivant $Q(\mathbf{x}, .)$. Expliquer pourquoi les trois lignes en python

   ```
   i=np.sum(np.cumsum(X)<s*np.random.random(1))
   j=int((I-1)*np.random.random(1))
   j=j+(j>=i)
   ```

   permettent de tirer un couple distribué suivant la loi de $(\mathcal{I} - 1, \mathcal{J} - 1)$ associée à $(\mathbf{x}(1), \dots, \mathbf{x}(I)) = (X(0), \dots, X(I-1))$ ou les trois lignes en scilab

   ```
   i=1+sum(cumsum(X)<s*rand(1));
   j=1+floor((I-1)*rand(1));
   j=j+(j>=i);
   ```

   permettent de tirer un couple distribué suivant la loi de $(\mathcal{I}, \mathcal{J})$ associée à la valeur courante X de $\mathbf{x}$.
2. Vérifier que la probabilité d'acceptation d'un mouvement de $\mathbf{x}$ à $\mathbf{x} - \mathbf{e}_i + \mathbf{e}_j$ avec $1 \leq i \neq j \leq I$ et $x(i) \geq 1$ est donnée par

$$\alpha(\mathbf{x}, \mathbf{x} - \mathbf{e}_i + \mathbf{e}_j) = \frac{(x(j)+1)\mu_i(x(i)-1)\mu_j(x(j)+1)}{x(i)\mu_i(x(i))\mu_j(x(j))} \wedge 1.$$

3. Vérifier que, dans le cas des lois de Poisson,

$$\alpha(\mathbf{x}, \mathbf{x} - \mathbf{e}_i + \mathbf{e}_j) = \frac{\lambda_j}{\lambda_i} \wedge 1.$$

4. Dans le programme *mhlgn_Q.py/mhlgn_Q.sce*, implémenter à chaque itération de l'algorithme de Metropolis-Hastings le choix du couple $(\mathcal{I}, \mathcal{J})$ proposé plus haut ainsi que l'acceptation du mouvement proposé avec probabilité $\alpha(X, X - \mathbf{e}_i + \mathbf{e}_j)$. Le programme trace en noir l'évolution $n \mapsto \frac{1}{n} \sum_{k=0}^{n-1} X_k(\ell)$ où $\lambda_\ell = \max_{1 \leq i \leq I} \lambda_i$ et la compare avec sa valeur limite $\frac{\lambda_\ell s}{\Lambda}$ en vert ainsi qu'avec l'évolution de la moyenne empirique de tirages i.i.d. suivant la loi binomiale $\mathcal{B}(s, \frac{\lambda_\ell}{\Lambda})$ en bleu. Notons que $\mathcal{B}(s, \frac{\lambda_\ell}{\Lambda})$ est la loi de la ℓ-ème composante de $\mathbf{x}$ sous π. Relancer le programme avec 10000 puis 100000 itérations.
5. Dans le programme *mhmargin_Q.py/ mhmargin_Q.sce*, les itérations de l'algorithme de Metropolis-Hastings sont vectorisées de manière à faire évoluer simultanément et efficacement M trajectoires indépendantes de l'algorithme. Mettre à jour le calcul du vecteur *compt*, qui donne la loi empirique de $X_n(\ell)$ sur ces M trajectoires (en python penser à convertir $X[m, \ell]$ en entier par $\text{int}(X[m, \ell])$). Pour des valeurs de n en progression géométrique de raison 4, le programme compare cette loi empirique et la loi binomiale $\mathcal{B}(s, \frac{\lambda_\ell}{\Lambda})$ de la ℓ-ème composante de $\mathbf{x}$ sous π.
6. Dans le programme *mhtlc_Q.py/mhtlc_Q.sce*, stocker dans le vecteur *kappa* la somme des valeurs de la ℓ-ème colonne de la matrice X au cours des itérations et tracer l'histogramme d'une transformation affine bien choisie de ce vecteur pour visualiser le théorème de la limite centrale pour $f(\mathbf{x}) = \mathbf{x}(\ell)$.