Algorithme de Metropolis

1. Introduction

Les méthodes de Monte-Carlo sont utilisées en physique statistique pour effectuer des simulations numériques de systèmes à l'équilibre thermodynamique. Ce document présente l'algorithme de Metropolis, qui permet de parcourir les microétats du système en suivant la distribution de Boltzmann. Un exemple simple est présenté : un système formé d'atomes fixes, ayant des niveaux d'énergie discrets.

2. Méthodes de Monte-Carlo

En physique statistique, on est fréquemment amené à calculer des valeurs moyennes de grandeurs observables. Pour un système en contact avec un thermostat de température T, la probabilité d'un microétat μ dont l'énergie est E_μ est proportionnelle au facteur de Boltzmann :

$$\exp\left(-\frac{E_\mu}{k_B T}\right) = e^{-\beta E_\mu} \tag{1}$$

où T est la température du thermostat et k_B la constante de Boltzmann.

La valeur moyenne (l'espérance) d'une grandeur Q est alors :

$$< Q >= \frac{\sum_\mu Q_\mu e^{-\beta E_\mu}}{\sum_\mu e^{-\beta E_\mu}} \tag{2}$$

Le dénominateur, noté Z, est la somme du facteur de Boltzmann pour tous les états :

$$Z = \sum_\mu e^{-\beta E_\mu} \tag{3}$$

Le nombre de microétats d'un système complexe est gigantesque. Par exemple pour un petit système formé de 1000 particules pouvant prendre chacune 2 états, le nombre total de microétats du système est 2^{1000}. Il est donc inenvisageable de calculer numériquement cette somme en cherchant à parcourir tous les microétats.

Les méthodes de Monte-Carlo sont des méthodes numériques consistant à calculer ce type de somme en choisissant aléatoirement des termes de la somme. Une première approche consiste à choisir les états aléatoirement avec une densité de probabilité uniforme, c'est-à-dire sans privilégier un état plutôt qu'un autre. Cette méthode est extrêmement inefficace (et même inapplicable), car la très grande majorité des états ont dans la distribution de Boltzmann une probabilité négligeable, leur énergie étant grande devant $k_B T$.

Il faut donc choisir les états aléatoirement avec une probabilité non uniforme, en favorisant si possible ceux qui ont une contribution relativement importante à la somme. Notons p_μ la probabilité de sélection d'un état μ. Soit M le nombre d'états sélectionnés. On montre ([1]) que la meilleure estimation de la valeur moyenne (celle qui donne la variance la plus faible) est :

$$Q_M = \frac{\sum_{i=1}^{M} Q_{\mu_i} p_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^{M} p_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}} \tag{4}$$

L'échantillonnage préférentiel consiste à privilégier les états qui ont une probabilité importante dans la distribution de Boltzmann. Le meilleur résultat est obtenu lorsque les états sont sélectionnés justement avec la distribution de Boltzmann, c'est-à-dire

$$p_\mu = Z^{-1} e^{-\beta E_\mu} \tag{5}$$

Dans ce cas, l'estimation de la valeur moyenne est simplement donnée par :

$$Q_M = \frac{1}{M} \sum_{i=1}^{M} Q_{\mu_i} \tag{6}$$

Pour générer ces états, les méthodes de Monte-Carlo utilisées en physique statistique utilisent une chaîne de Markov. Une chaîne de Markov est une suite d'états générés aléatoirement tels que la probabilité de transition d'un état à l'autre ne dépend que de ces deux états (et pas du passé). Notons $P(\mu \to \nu)$ cette probabilité.

L'objectif est d'obtenir une chaîne d'états qui tende vers la distribution (5). La chaîne de Markov doit vérifier deux propriétés importantes. La première est l'ergodicité : tous les états du système doivent être accessibles, cela afin d'éviter que la chaîne ne parcoure qu'un sous ensemble des états possibles. La deuxième propriété est le principe de l'équilibre détaillé ([1]) :

$$p_\mu P(\mu \to \nu) = p_\nu P(\nu \to \mu) \tag{7}$$

Ce principe signifie que le nombre de passages de l'état μ vers l'état ν est en moyenne égal aux nombres de passages de l'état ν vers l'état μ. Dans le cas de la distribution de Boltzmann, cette condition s'écrit :

$$\frac{P(\mu \to \nu)}{P(\nu \to \mu)} = e^{-\beta(E_\nu - E_\mu)} \tag{8}$$

3. Algorithme de Metropolis

Il est très difficile de générer directement une chaîne de Markov vérifiant la relation précédente (8). Metropolis et Rosenbluth ([2]) ont inventé un algorithme qui consiste à procéder en deux étapes pour passer d'un état de la chaîne au suivant : une étape de sélection d'un nouvel état, suivie d'une étape d'acceptation (ou de refus). Dans [2], la méthode est appliquée au modèle des sphères dures ; il s'agit d'une des premières expériences de calcul numérique, réalisées sur l'ordinateur MANIAC. La méthode a été depuis développée sur d'autres modèles, en particulier le modèle d'Ising du ferromagnétisme, et constitue aujourd'hui la méthode de Monte-Carlo de référence pour la physique statistique, et même dans d'autres domaines.

La probabilité de transition est décomposée de la manière suivante :

$$P(\mu \to \nu) = S(\mu \to \nu) A(\mu \to \nu) \tag{9}$$

Le premier terme S est la probabilité de sélection d'un état à partir du précédent. Le second terme est la probabilité d'acceptation de la transition. Dans la méthode de Metropolis originale, la probabilité de sélection est choisie constante : toutes les transitions sont équiprobables. La transition est une modification élémentaire du système, par exemple le déplacement d'un seule atome, ou le basculement d'un seul spin dans le modèle d'Ising. Il reste alors à établir des probabilités d'acceptation vérifiant :

$$\frac{A(\mu \to \nu)}{A(\nu \to \mu)} = e^{-\beta(E_\nu - E_\mu)} \tag{10}$$

La probabilité d'acceptation doit être la plus grande possible, afin d'éviter le rejet d'un trop grand nombre de transitions. Supposons que $E_\nu > E_\mu$. La probabilité d'acceptation la plus grande est celle de l'état ν vers l'état μ, c'est-à-dire celle de l'état de plus grande énergie vers l'état de plus faible énergie. Cette probabilité est choisie égale à la valeur maximale, c'est-à-dire 1. L'autre probabilité d'acceptation, celle de l'état de faible énergie vers l'état de forte énergie, est donc :

$$A(\mu \to \nu) = e^{-\beta(E_\nu - E_\mu)} \tag{11}$$

Pour implémenter la méthode de Metropolis, il faut bien sûr un générateur de nombres pseudo-aléatoires, pour l'étape de sélection et pour l'étape d'acceptation (lorsque l'énergie augmente).

Finalement, l'algorithme complet permettant de passer d'un état à un autre dans la chaîne de Markov est le suivant :

- ▷ Sélection d'une modification élémentaire de l'état du système, par exemple déplacement aléatoire d'un atome choisi aléatoirement, et calcul de la variation d'énergie ΔE du système associée à cette modification. Cette sélection doit respecter l'équiprobabilité de toutes les transitions.
- ▷ Si $\Delta E <= 0$, la transition sélectionnée est acceptée.
- ▷ Si $\Delta E > 0$, un nombre réel aléatoire x est tiré avec une probabilité uniforme dans l'intervalle $[0, 1[$. Si $x < e^{-\beta \Delta E}$, la transition sélectionnée est acceptée. Dans le cas contraire elle est refusée et le nouvel état est identique au précédent.

L'estimation de la moyenne est faite en utilisant la somme (6) appliquée aux états de la chaîne de Markov. Il faut remarquer qu'un état identique au précédent est bien compté dans la somme.

4. Exemple

4.a. Modèle

Pour tester l'algorithme de Metropolis sur un modèle très simple, considérons un système de N atomes fixes. On peut imaginer ces atomes alignés, bien que cela n'ait pas d'importance dans ce modèle. L'atome d'indice i a une énergie e_i qui peut prendre des valeurs entières, en partant de zéro. L'énergie du système est la somme de ces énergies :

$$E_\mu = \sum_{i=0}^{N-1} e_i \tag{12}$$

L'énergie d'un atome est donc un entier, qui par ailleurs suffit à définir son état. Comme l'énergie de chaque atome n'est pas limitée, il en est de même de l'énergie du système tout entier. Il y a donc une infinité d'états dans ce système, ce qui n'est pas gênant pour l'algorithme de Metropolis.

Physiquement, les atomes échangent de l'énergie avec leurs voisins sous l'effet de l'agitation thermique. La seule grandeur macroscopique observable dans ce modèle est l'énergie totale, dont on cherche la moyenne en fonction de la température :

$$< E >= \frac{\sum_\mu E_\mu e^{-\beta E_\mu}}{\sum_\mu e^{-\beta E_\mu}} \tag{13}$$

Le changement élémentaire effectué à chaque étape de l'algorithme est le changement d'énergie d'une unité, sur un atome choisi au hasard. Le changement d'énergie peut être $+1$ ou -1, cette dernière valeur étant impossible si l'énergie est nulle (l'atome est alors dans l'état fondamental). Il n'y a donc que deux valeurs possibles pour la probabilité d'acceptation : 1 et $e^{-\beta}$.

4.b. Implémentation

L'état des atomes est stocké dans un tableau à une dimension. On définit une classe dont le constructeur prend en argument le nombre d'atomes. Les énergies des atomes sont initialisées à 0 (ce qui correspond à une température nulle).

testMetropolis.py

```
import numpy
import random
import math

class Atomes:
    def __init__(self,N):
        self.N = N
        self.etat = numpy.zeros(N)
        self.E = 0
```

La première fonction de la classe permet de définir la température. Le coefficient β est l'inverse de la température. La probabilité $e^{-\beta}$ est calculée. Elle est nulle si la température est nulle.

```
    def temperature(self,T):
        self.T = T
        if T==0:
            self.A = 0.0
        else:
            self.A = math.exp(-1.0/T)
```

La fonction suivante effectue un pas de l'algorithme de Metropolis. L'atome est choisi au hasard, tout comme la variation d'énergie (-1 ou $+1$) :

```
    def metropolis(self):
        i = random.randint(0,self.N-1)
        if self.etat[i] == 0:
            dE = 1
        else:
            if random.randint(1,2)==1:
                dE = -1
            else:
                dE = 1
        if dE<=0:
            self.etat[i] += dE
            self.E = self.E+dE
        else:
            x = random.random()
            if x<self.A:
                self.etat[i] += dE
                self.E = self.E+dE
```

La fonction suivante effectue une boucle de calcul et calcule la moyenne de l'énergie et l'écart-type :

```
    def boucle(self,n):
        en = numpy.zeros(n)
        for k in range(n):
            en[k] = self.E
            self.metropolis()
        return (en,numpy.mean(en),numpy.std(en))
```

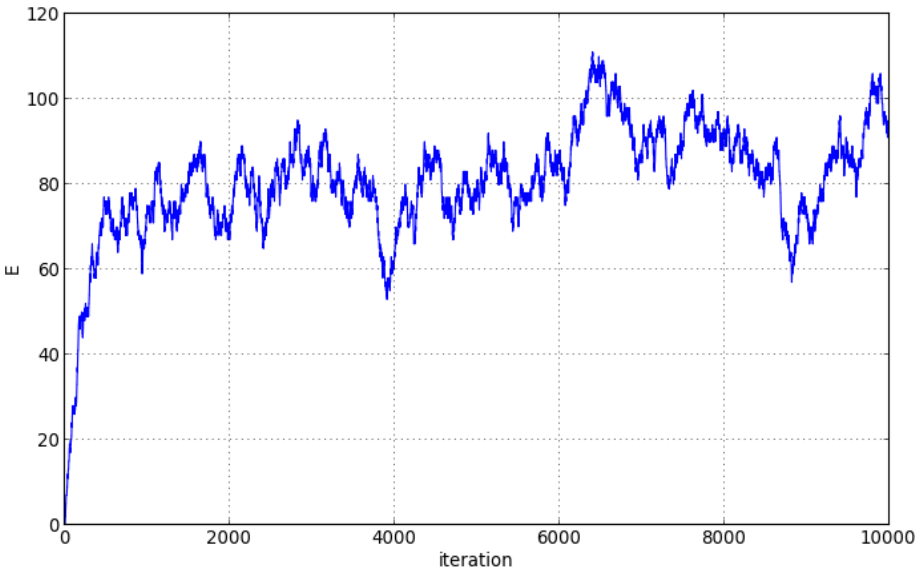
4.c. Résultats

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

Voyons tout d'abord l'évolution de l'énergie pour une température donnée, pour un système de 100 atomes :

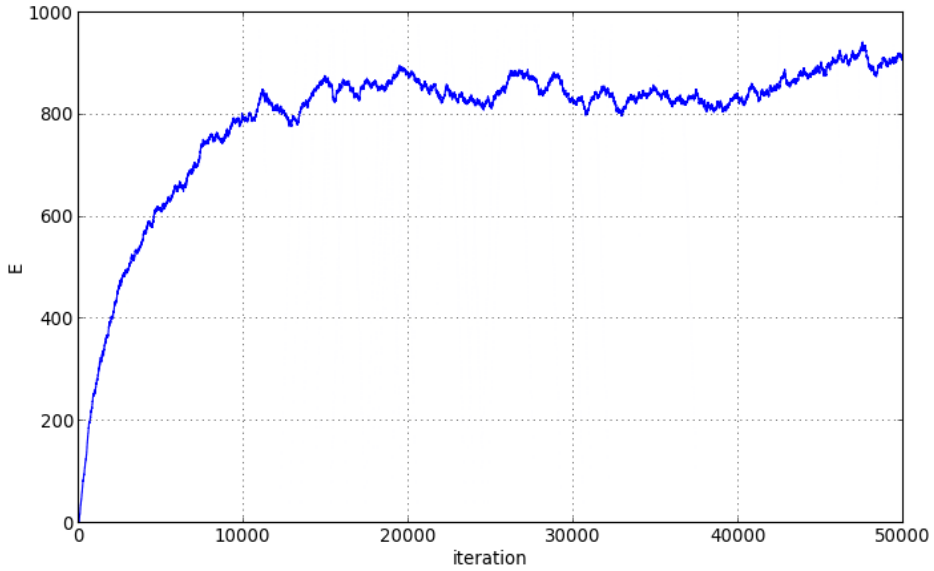
```
N=100
atomes = testMetropolis.Atomes(N)
atomes.temperature(1.0)
(en,E,delta)=atomes.boucle(10000)
figure(figsize=(10,6))
plot(en)
ylabel('E')
xlabel('iteration')
```

```
grid()
```



On observe l'évolution vers l'état d'équilibre. Les fluctuations d'énergie restent très importantes. Augmentons le nombre d'atomes :

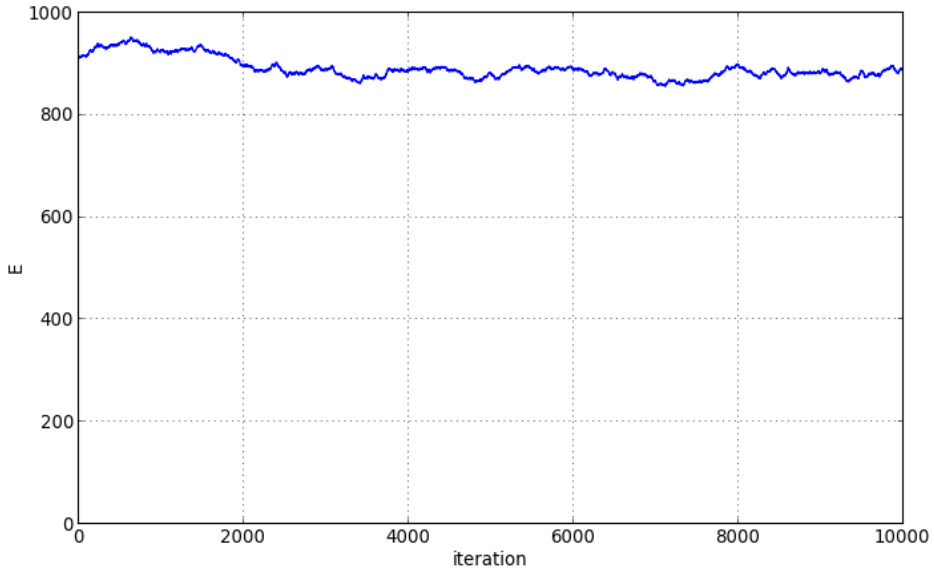
```
N=1000
atomes = testMetropolis.Atomes(N)
atomes.temperature(1.0)
(en,E,delta)=atomes.boucle(50000)
figure(figsize=(10,6))
plot(en)
ylabel('E')
xlabel('iteration')
grid()
```



Avec ce système 10 fois plus grand, la convergence vers l'état d'équilibre est plus longue mais les fluctuations d'énergie à l'équilibre sont plus modérées.

Lorsque l'équilibre est considéré atteint, on peut faire des itérations pour calculer l'énergie moyenne et l'écart-type :

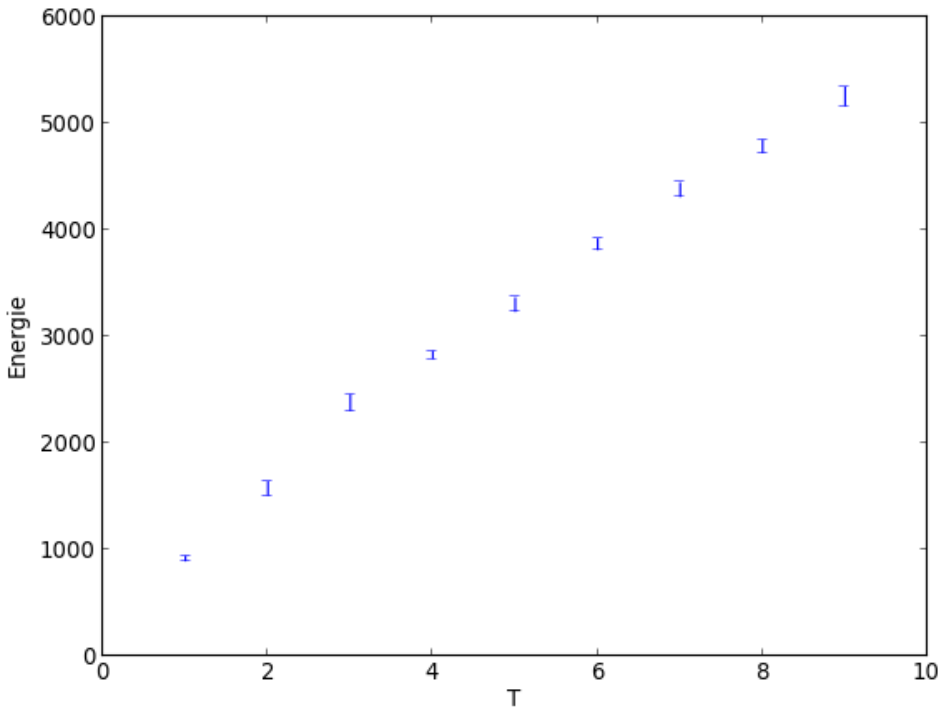
```
(en,E,delta)=atomes.boucle(10000)
figure(figsize=(10,6))
plot(en)
axis([0,10000,0,1000])
ylabel('E')
xlabel('iteration')
grid()
```



```
print((E,delta))
--> (891.08979999999997, 20.690334360758893)
```

Une simulation complète consiste à calculer la moyenne et l'écart type de l'énergie pour différentes valeurs de la température. On procède par température croissante. Pour chaque nouvelle température, on procède à une mise en équilibre en partant du dernier état obtenu avec la température précédente. Cette méthode permet d'arriver très rapidement à l'équilibre.

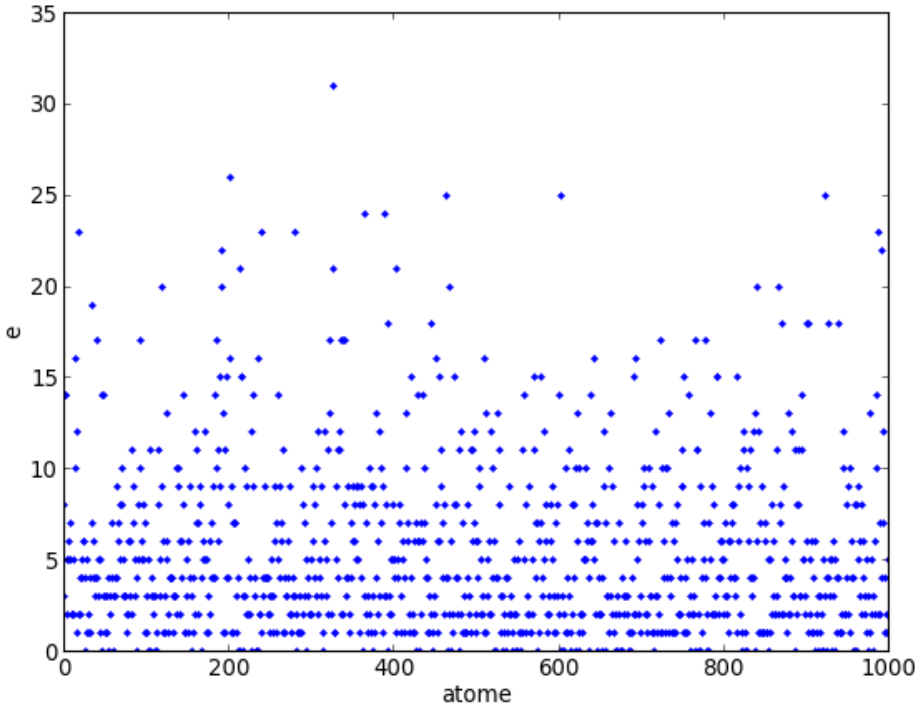
```
temp = numpy.arange(1.0,10.0,1.0)
n = temp.size
Energie = numpy.zeros(n)
Delta = numpy.zeros(n)
for k in range(n):
    atomes.temperature(temp[k])
    atomes.boucle(10000)
    (en,E,delta) = atomes.boucle(10000)
    Energie[k] = E
    Delta[k] = delta

figure(figsize=(8,6))
errorbar(temp,Energie,yerr=Delta,fmt=None)
axis([0,10,0,6000])
ylabel("Energie")
xlabel("T")
axis()
```



La température est une fonction croissante linéaire de la température.

Voyons l'état du système pour la dernière température calculée et pour la dernière configuration obtenue :

```
figure(figsize=(8,6))
plot(atomes.etat,'.')
xlabel('atome')
ylabel('e')
axis()
```



Les atomes sont dans des états très différents. Certains sont à l'état fondamental, d'autres sont dans un état fortement excité. La moyenne de l'énergie par atome est l'énergie moyenne du système divisée par le nombre d'atomes, soit environ $5000/1000 = 5$.

Références

[1] M.E.J. Newman, G.T. Barkema, *Monte Carlo methods in statisticals physics*, (Oxford university press, 1999)

[2] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, *Equation of state calculations by fast computing machines*, (J. Chem. Phys. vol. 21 No. 6, 1953)