7. モンテカルロ法

常微分方程式の解法は、初期状態を決め、近似(オイラー法、RK 法など)によって次の点を求め、その点を使ってまたその次の点を求め…のように進めていけば解が求まった。これまで見てきた数値計算のアルゴリズムは、すべてこのような決定論的(deterministic)な手法であったが、コンピュータの利点を生かした確率論的(stochastic)な手法もよく用いられる。

確率論的手法とは何か? 例えば、さいころの 1 の目が出る確率は、6 面のうちで 1 面なので、決定論的(かつ解析的)に 1/6 と計算できる。一方、このように頭で考えずに実際にさいころを何度も振って実験し、実際に 1 が出た回数をすべて数え上げ、さいころを振った回数で割っても、確率は求まる。これが確率論的な手法だ。もちろん、さいころを振る回数が少なければ誤差が多くて正しくない答えになるが、大規模な計算が得意なコンピュータ上で、十分多くの回数だけさいころを振れば(つまり乱数を発生させれば)、答えが 1/6 に収束していくはずだ。このような確率論的手法を総称してモンテカルロ法という[1]。 なお、モンテカルロ法を行うためには、用いる乱数が理想的な一様分布であることが前提となるが、ここではコンピュータとソフトウェアによって、それが保証されているものとして進めていく。

7.1. 放射性崩壊のシミュレーション

放射性元素が、崩壊する速さは、残存する原子数 N に比例する。つまり、

$$\frac{dN}{dt} = -\lambda N$$

λ が定数なら解析的に解けるし、常微分方程式(ODE)だから、RK 法などの数値計算で解ける。

しかし、よくよく考えてみれば、上のような微分方程式で記述できるのは、たくさんの粒子を統計的に扱った場合の話であって、実際の崩壊の過程は、「ある粒子が崩壊したのか、しないままなのか」という二つに一つの確率過程だ。したがって、粒子の数が減ってくれば、微分方程式の解から外れてくる。そこで、この確率過程をモンテカルロ法によってシミュレーションしてみよう。

年代測定に良く使われる炭素の同位体 ^{14}C を例にする。$\lambda=1.210^{-4}$/年なので、1 年ごとに乱数を振ると、計算時間がかかりすぎるため、100 年に一度乱数を振るようにした。考え方のポイントは、いくつか残っている粒子それぞれに注目して、それぞれについて、0 から 1 までの範囲の乱数を作る(=さいころを振る)。もしその乱数がλより大きければ崩壊せず、小さければ崩壊した、と判定する。崩壊したと判定されれば、その粒子を消す。つまり、存在する粒子数を 1 つ減らす。もし、崩壊しなければ、時間を 1 単位(今の場合は 100 年)増やして、また判定する。これを粒子が崩壊するまで繰り返す。そうして、最後の 1 個が崩壊するまで、乱数を振っては、判定、を繰り返すわけ。では、具体的にプログラムを作ってみよう。

```
%radioactive.m
clear; lamb=1.2e-2; %炭素 14 では、1.2e-4/年
for m=1:5     %最初の個数を 10 から 10^5 個まで変える
  t=0;           %時間を初期化(単位は 100 年)
  Num(1)=10^m;   %最初の個数
  Nleft=Num(1);  %残っている個数=最初の個数
  while Nleft >1  %残った粒子数が>1 なら繰り返す
    t=t+1;          %時間を 1 単位(100 年)進める
    r=rand(Nleft,1); %乱数(値は 0 から 1 まで)を
       %要素に持ち、サイズが NLeft×1 のベクトルを作る
    Nleft=sum(r>lamb); %残っている個数を求める
    Num(t)=Nleft;
        %残っている個数を時間 t の関数として Num に記録
  end  %while の end。Nelft>1 なら while の行に戻る
  time=(1:t)*100; %時間ベクトルを作る
  decay=Num(1)*exp(-time*lamb/1e2);
         %比較のため、解析解を求める
  plot(time,Num,time,decay,'r');
  hold on          %図の重ね書き開始
end       %for ループの end
legend('Monte Carlo', 'Exp decay');
xlabel('Time (Year)'); ylabel('Numbers');
title('radioactive.m');
axis([0 110000 1 1e5]);
hold off          %図の重ね書き終了
```

まず、`rand(Nleft,1)`で、サイズが `Nleft×1` のベクトルを乱数で適当に作る。これで、`Nleft` 個の残存粒子一つ一つについて、さいころを振ったことになる。理解しにくい部分は、`Nleft=sum(r>lamb)`の中の `r>lamb` かもしれない。これは、数学としての不等式ではなく、`r` の要素のうち、`lamb` より大きければ、`0` そうでなければ、`1` を返すベクトルだ。例として

[1]ギャンブルで有名!?なモナコのモンテカルロから名付けられたそうだ。

`r=[0.1 0.8 0.2 0.4]`、`lamb=0.3` とすると、`r>lamb` とは、`[0 1 0 1]`というベクトルのことである。それが分かれば、その合計 `sum` を取ることで、残存している個数(今の例なら=2)が求まることが理解できると思う。

radioactive.m を実行すると、図 7.1 のような結果が得られるはず。当然ながら結果は実行するたびに異なる。残存する粒子数が多ければ semilog プロット上で直線的に、つまり指数関数的に減少していくが、粒子数が少なくなってくると、確率的な過程になって、直線から外れてくる。おおざっぱに言って、100 個程度を下回ると、解析解である指数関数の減衰で近似できなくなっていく様子が分かるだろう。

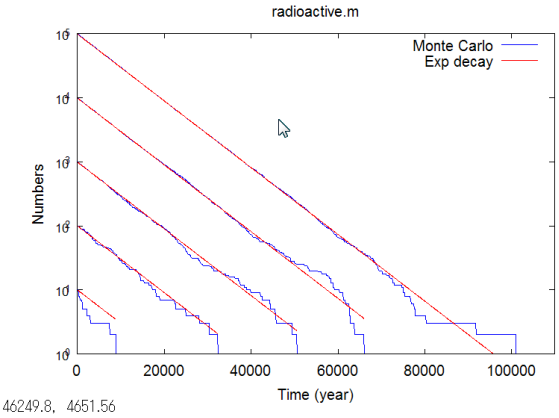

図 7.1 ^{14}C が崩壊していく様子のシミュレーション

7.2. πを求める

モンテカルロ法は確率的な事象のシミュレーションに使われる一方、確率とは関係なさそうな問題を解くことにも使える。その例として、モンテカルロ法で円周率πを求めてみよう。

まず半径 1 の円と、一辺が 2 の正方形を考える(図 7.2)。この正方形の中に無作為に点を X 個ばらまいて、点が円の中に入った個数 C を数え上げ、それを X で割れば、円に入る確率 $P=C/X$ が求まる。円と正方形の面積の比はπ:4 なので、$\pi=4P$ でπが求まるという仕組みだ。

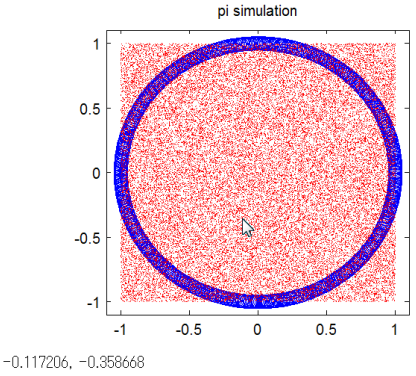

図 7.2 πを求めるモンテカルロシミュレーションの様子

乱数を作るためには、`unifrnd` を使う。これは、例えば、`unifrnd(a,b,m,n)`とすると、`a` から `b` までのランダムな値を要素に持つサイズ `m×n` の行列を出力するものだ。無作為の点(x,y)を `unifrand(-1,1,40000,2)` で 40,000 個をばらまき、それをプロットしたものが図 7.2 だ。プログラムは piplot.m。

```
%piplot.m
t=0:0.01:2*pi;
plot(cos(t),sin(t),'bo'); %半径 1 の円を描く
hold on   %重ね書き
xy=unifrnd(-1,1,40000,2); %1 列目が x,2 列目が y
plot(xy(:,1),xy(:,2),'r.')
axis([-1.1 1.1 -1.1 1.1]);
axis equal; hold off
title('pi simulation');
```

実際のシミュレーションでは、同時にばらまく回数 `x` を 10^6 にする。これを大きくすれば精度が上がるのだが、パソコンのメモリーの限界(4GB? 多くても 12GB?)に近づくので、おそらく 10^7 から 10^8 を超えるぐらいになると、メモリーエラーになる。そのため、「10^6 個の同時ばらまき試行」を N 回繰り返し、その平均値を取ることにしている。パソコンのメモリーが少なくてエラーが出る場合は、`x` をさらに減らして `1e5` にすると良い。

```
%pi_estimation.m
clear; format long   %倍精度で表示
X=1e6; %1e6 個の乱数を同時に作る
N=10; %「10^6 個の同時ばらまき試行」の回数
for n=1:N
  xy=unifrnd(-1,1,X,2);%無作為の点(x,y)を X 個選ぶ
  r=xy(:,1).^2+xy(:,2).^2;  %原点からの距離
  C=sum(r<1);       %r<1(円の内側)だった点の数の合計
  pi_once(n)=4*C/X; %一回の試行で求まるπ
end
PI=sum(pi_once)/N   % N 回求めたπの平均値
error=(pi-PI)/PI    %pi からの誤差
```

C=sum(r<1)の行の r<1 が分かりにくいかもしれないが、これは、「r の要素が 1 より小さいなら 1、そうでなければ 0 を要素に持ち、サイズが x のベクトル」である。

pi_estimation.m を実行すると、3.14 に近い値が得られるはず。より正確に求めたい場合は、この試行回数 N を増やせばよい。ちなみに私のパソコンでは N=6000、X=10^7 にして、約 1 時間程度で、3.1415926192667 という値が得られた。ちなみに、モンテカルロ法は、πを数値計算で求める方法としては、必ずしも良いアルゴリズムとは言えず、もっと精度良く求める方法は別に存在する。

7.3. モンテカルロ法で重積分を解く

先の例ではπを求めたわけだが、実際には円の面積を求めていたことに気がついただろうか? 円の面積は 2 重積分を使って求めるわけだが、確率論的なモンテカルロ法でも面積を求めることができる、つまり、重積分を解くことができるのだ。次に重積分の例として、球の体積をモンテカルロ法によって数値計算して見よう。球の体積を解析的に手計算しようとすると、極座標表示にして、3 重積分する必要があり、ちょっと面倒だが、モンテカルロ積分の場合は、とても簡単で、プログラム的にも、πを求めたときとほとんど同じだ。(sphere.m)

```
% sphere.m
clear; format long; X=1e6; N=10;
for n=1:N
    xyz=unifrnd(-1,1,X,3);%三次元座標の乱数生成
    r=xyz(:,1).^2+xyz(:,2).^2+xyz(:,3).^2;
    C=sum(r<1); S_once(n)=C/X*2^3;
end
Volume=sum(S_once)/N
error=abs((4/3*pi - Volume)/(4/3*pi))
```

これを実行すると、Volume=4.18….、error=7.5..e-004 というような答えが出る。error は球の体積の厳密解 4/3πからの誤差を表している。この程度の計算でも、大体 0.1%以下の誤差で球の体積が求まっていることが分かる。

これまで積分をどのように実行するかについては紹介していないが、これまでの経験から、連続関数の積分は、数値計算では数列の和で求められそうだということは、容易に想像してもらえるだろう。さらに、重積分は和記号が複数になる、ということも納得してもらえるに違いない。二重積分の場合は、

$$\int_{a_2}^{b_2}\int_{a_1}^{b_1} f(x,y)dxdy \Rightarrow \sum_{m}^{N}\sum_{l}^{N} f(x_m, y_l)$$

こんな感じだ。

単純に、n 重積分を数値計算しようとすると、n 個の和記号が出てくる。したがって、足し合わせる和の数を N とすると、計算量も N^n に比例して増加していく。仮に N=1000 として、10 重積分をすると、$(10^3)^{10}$=10^{30} という途方もない計算量になってしまう。これを真っ向から解こうとすると、現在世界最速のスーパーコンピュータを持ってしても、単純計算で 3000 年以上かかることになる…。とにかく多重積分は時間がかかるのだ。

一方、モンテカルロ積分ではどうだろうか? 乱数の個数を X とすると、モンテカルロ積分は、

$$\int_{a_n}^{b_n}\cdots\int_{a_1}^{b_1} f(x_{1i}, x_{2i}\cdots x_{2i})dx_1dx_2\cdots dx_n$$

$$\Rightarrow \frac{(b_1-a_1)\cdots(b_n-a_n)}{X}\sum_{i} f(x_{1i}, x_{2i}\cdots x_{ni})$$

となり、いくら重積分の次元 n が増えても和記号は一つのままで、計算量も、$O(N)$から増えず、計算時間は短くて済むことが直感的に理解できると思う。ただし、モンテカルロ積分は、精度がそれほど良くないことが知られていて、精度良く答えを出す

ためには、乱数の数を非常に多く取らないといけないという欠点もある。ただ、精度の問題を考慮しても、多重になればなるほど、モンテカルロ法が決定論的な積分より有利であることが知られている。素粒子実験のシミュレーションや、素粒子現象論の理論研究では、多重積分が現れることがしばしばあり、数年前に発表された量子電磁力学で重要な微細構造定数の精密な計算結果でも、10 重積分程度の数値積分が行われているそうだ。
(http://www.riken.go.jp/r-world/info/release/press/2007/070822/detail.html)

7.4. 相転移とイジングモデル

次に、モンテカルロ法を使って、相転移を考えよう。相転移とは、固体が溶けて液体になったり、強磁性体だった磁石が温度を上げると、急に磁石ではなくなったり、超伝導が常伝導になったりという現象で、物理のいろいろな場面で現れる面白い現象だ。

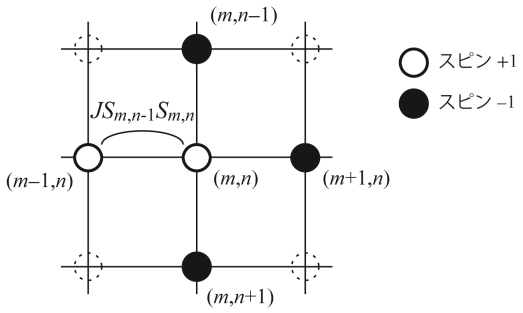

図 7.4.1 イジングモデル

それでは、実際に系を考えてみよう。モデルとしては二次元の正方格子を考えて、温度が kT のとき、格子点にスピン+1 か –1 の粒子をランダムにばらまいたとしよう。一度ばらまいたスピンは、温度と周辺のエネルギーの大小関係によって反転することができるものとする。ある点(m,n)に注目すると、格子の端でない限り、必ず周り 4 点に囲まれている。そこで、周り 4 点のスピン間の相互作用だけ考えて、それ以上離れたスピンの相互作用は考えないとしよう。スピン同士の相互作用(これを交換相互作用という)の係数を J とすれば、点(m,n)のハミルトニアンは、

$$H_{m,n} = -J(S_{m-1,n} + S_{m+1,n} + S_{m,n-1} + S_{m,n+1})S_{m,n}$$

となる。このようなモデルをイジングモデルという。この式が何を言っているか全く分からないという人のために、かみ砕いて説明すると、まず、ハミルトニアン=エネルギーだと思って欲しい。J は係数なので、適当な単位系を考えれば J=1 とできる。また、ばらまいたスピン $S_{\text{添え字}}$は、+1 か−1 かの二択になる。たとえば、図 7.4.1 の場合、中心がスピン+1 で、その周りに–1 が 3 つ、+1 が一つあるので、ハミルトニアンは、

$$H_{m,n} = -(+1-1-1-1)\times 1 = 2$$

と決まる。つまり、点 m,n におけるハミルトニアン=エネルギーは"2"ということだ。仮に、点(m,n)のスピン $S_{m,n}$ が–1 だったらどうだろう。今度は符号が反対になるので $H_{m,n}$ =–2 だ。つまり、元の図の 7.4.1 の状態の方がエネルギーが高い。これは、周り 3 点のスピンは–1 なので、点(m,n)のスピンも+1 でいるよりは、–1 に反転した方がエネルギー的に得だ、ということを示している。J が正の場合は、なるべくスピンは揃いたがる(白白白や黒黒黒と並ぶのが安定)傾向がある。逆に J が負の場合は、隣のスピンと逆向きになった方が(白黒白黒と並ぶ方が)、エネルギーが低くなって安定だ。

仮に点(m,n)のスピンが反転した場合、反転する前と後のエネルギー差ΔE は$\Delta E=H_{m,n\,後}-H_{m,n\,前}=-2H_{m,n\,前}$になる。(今の例では–4)。そこで、点$(m,n)$のスピンが反転するか、しないかの確率を、統計力学で出てくる確率因子（ボルツマン因子）$p=\exp(-\Delta E/kT)$で決めることにする。つまり、スピンが反転して得をするエネルギーが、温度 kT に比べて大きければ大きいほど、反転させる確率 p を高く設定するということだ。具体的には、0 から 1 までを取り得る一様乱数 r を発生させ、$r<p$ なら反転させ、$r>p$ なら反転させないと決める。これを**メトロポリス法**という。ここまでは、点(m,n)の話し。この"スピンが反転するかしないかの判定"を全格子点ではなく、適当に選んだ格子点について行う。判定が終わったら、`for` ループで時間を一つ進める。これを何度も繰り返すと、徐々に系が熱平衡状態に近づいていくはずだ。

ここから実際のプログラムの説明に入ろう。まずは、ハミルトニアン H を求める関数 IsingH.m を用意しておく。縦横の格子点の数を N とし、$N\times N$ の格子点にスピンをばらまいくことにす

る。また状態を表すのに、+1 か–1 を要素にもつ $N \times N$ の行列 S を使う。つまり、$S_{m,n}$ とは、行列 S の (m,n)要素ということ。

```
%IsingH.m
function [dE Eavg]=IsingH(S,J,B)
[N N]=size(S);
Si=circshift(S,[-1 0])+circshift(S,[1 0])+ ...
    circshift(S,[0 -1])+circshift(S,[0 1]);
H=-(J.*Si + B).*S; %ハミルトニアン(エネルギー)
dE=-2*H; %エネルギー差
Eavg=sum(sum(H))/N^2; %各点のエネルギーの平均
```

ここで使っている、**circshift** は、ある行列の行、あるいは列をシフトさせたいときに使う。たとえば、

```
> A=[1 2 3; 4 5 6; 7 8 9]
A=
1 2 3
4 5 6
7 8 9
```

としたとき、**B=circshift(A,[0 1])** は、

```
> B=circshift(A,[0 1])
B =
3 1 2
6 4 5
9 7 8
```

となり、列が一つずつ右へシフトし、最終列は最初の列に移動する。これをうまく使うと、$(S_{m,n-1} + S_{m-1,n} + S_{m,n+1} + S_{m,n+1})$ の部分が簡単に記述できる。この最後の行、あるいは列を最初に持ってきたりすることによって、周期境界条件を課すこともできる。(ついでに、プログラムでは外部から垂直磁場をかけられるように、磁場 **B** という入力を関数 **IsingH** に入れ、ハミルトニアンの中にも、**-B.*S** という項を含めた。)

メインプログラムの方は、IsingModel.m のようになる。先に述べたように、スピンを反転させる確率は **p=exp(-dE/kT)** によって決める。ただし、実際にスピン反転の判定を行うのは、全格子点のなかで、1/5 の確率でランダムに選び出すことにする (**rand(N)<1/5** の部分)。**Flip** はスピンを反転させるための行列である。なお、**input** という関数を使って、パラメータ **J**, **B**, **S** を対話的に入力できるようにしてみた。

```
% IsingModel.m
clear; N=50;
J=input(' Jを入力してください  J =');
kT=input(' 温度を入力してください  kT=');
B=input(' 磁場を入力してください  B=');
S=(rand(N)<0.5)*(-2)+1; %ランダムな初期状態を作る
nMax=100;    %for ループ回数
m=5;         %作図する頻度を m 回に 1 回とする
for n=1:nMax
  [dE Eavg(n)]=IsingE(S,J,B); %エネルギーを求める
  P=exp(-dE/kT);  %スピン反転を判定する確率因子
  Flip=(rand(N)<P).*(rand(N)<1/5)*(-2)+1;
  S=S.*Flip;       %反転させた新しい状態 S を作る。
  if  n==1 || mod(n,m)==0
     %mod は n/m の余りで、これが 0 のときだけ図を表示
   pcolor(S);       %カラープロット
   axis([1 N 1 N],'square');
   title(sprintf('Iteration=%g, ...
                  Energy=%g',n,Esum(n)));
   caxis([-1.5 1.5]);
   drawnow; %リアルタイムで表示
   end
end
```

まず、温度 **kT=1** として、実行すると、図 7.4.2 のような図が現れる (実際にはパラパラマンガのように図が現れるはず)。白がスピン+1、黒が–1 の点を示している。

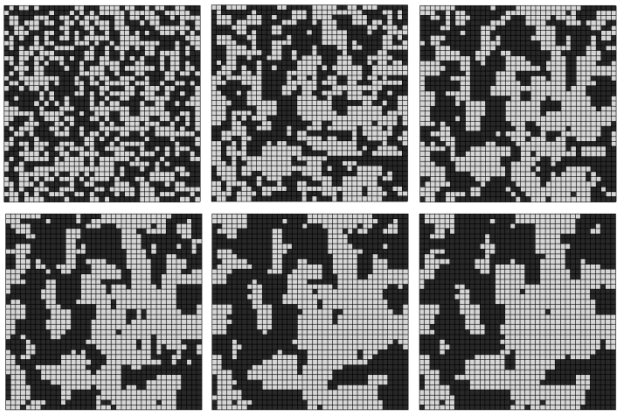

図 7.4.2 モンテカルロ法によるイジングモデルシミュレーション(**kT=1**)

最初は、ランダムにばらまいたはずのスピン(図 7.4.2 の左上)が、時間が経過するたびに、スピンが揃った比較的大きな領域(ドメイン)ができてきて、それが次第に成長していくことが分かる(同右下)。ここで注目したいのは、4 つの隣同士という短距離の相互作用しか式に入れていないのに、不思議なことに!? ″大きな島(ドメイン)″という「長距離の秩序」が次第に現れていく点である。

次に、温度を kT=1 から kT=5 に上げてみたらどうだろうか。図 7.4.3 が示すように、今度は、反復を 120 回繰り返しても、目立った領域（ドメイン)は現れない。実際にシミュレーションしてみると分かるが、今の例では、kT=2.5 付近が相転移温度で、それより低温では長距離秩序が現れ(つまり鉄、コバルト、ニッケルのような強磁性的)、それ以上だと(常磁性的)となる。

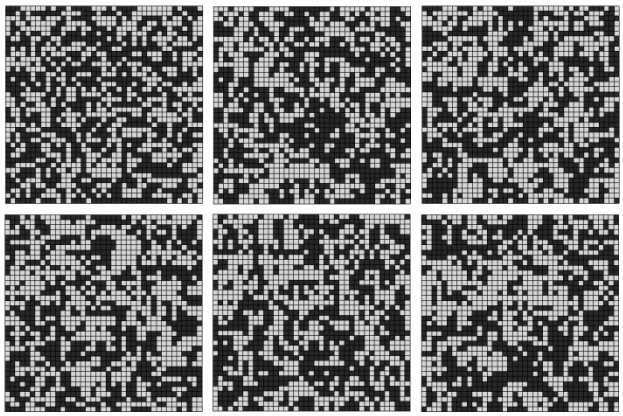

図 7.4.3 モンテカルロ法によるイジングモデルシミュレーション(**kT=5**)

今の例では、スピン間の相互作用 J は 1、つまり、J>0 なので、ドメイン構造を持った強磁性体が現れるが、J<0 にすると、反強磁性体となり、スピンの+1 と–1 が交互に並んでチェッカーフラッグのような模様が現れる。また、磁場を 0 以外にすると、エネルギーの縮退が解け、全領域でスピン 1 か–1 のみになる様子も分かる。このプログラムは比較的短いわりに、いろいろな物理現象をシミュレーションできるので、パラメータをいじって試してみてほしい。

今までの話は、スピンの+1 か–1 で考えて、強磁性体の相転移をシミュレートしてみたわけだが、強磁性体なんて興味ない、という人のために、少し違った見方をして、スピンの代わりに、電気の on、off で考えたらどうだろう。われわれの脳は、脳細胞でできているが、その脳細胞の活動は突き詰めて言えば、脳細胞の電位が上がること(発火)で機能している。脳細胞は、シナプスという"電線"で繋がっているので、イジングモデルの各点を脳細胞、スピンを脳細胞の発火、結合係数 J を脳細胞同士の結合の強さと考えれば、イジングモデルで脳の神経回路をモデル化することもできる。このようなモデルは一般的にはニューラルネットワークと呼ばれる。特にモンテカルロ法を使ったものは確率的ニューラルネットワークと呼ばれ、脳の連想記憶と関係が深い。また画像のノイズ除去や、誤り符号訂正などにも利用されている。(興味があれば、「スピングラス理論と情報統計力学(西森秀稔著)など読んでみると面白いかも)