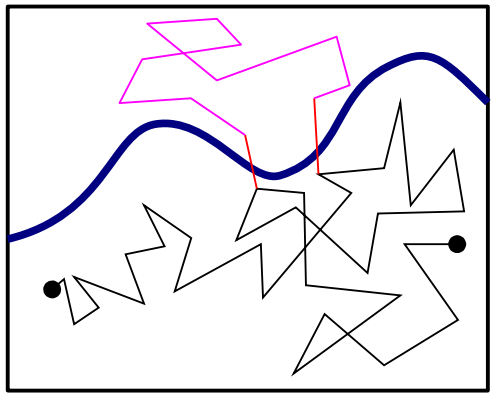
加權 (weighted) 取樣



考慮 n > 2 的狀態具有不用的權重
4, 3, 2, 3, 4, 3, 2, 1, 0, 1, 2, 3, 2, 3, 4, 5, 4, 3, 2, 3

$$\langle O \rangle_W = \sum_s W(s) O(s)$$

利用排拒 (rejection) 來調控數列出現在不同區域的機率

Metropolis–Hastings 演算法

$Q(s, s')$: 正常的取樣序列中 state s 到 state s' 的機率

$$A = \frac{W(s')Q(s', s)}{W(s)Q(s, s')}$$

$s \to s'$ 的接受機率: $\min(A, 1)$

* 符合細部平衡

細部平衡 (detailed balance)

$T(s, s')$: 躍遷率 (transition rate)

$$W(s)\ T(s, s') = W(s')\ T(s', s)$$

Boltzmann 分佈

溫度：在平衡狀態下，系統能量每增加一單位時，系統可能狀態數目增加倍數對數值的倒數

$P_T(s)$: 系統出現在 state s 的機率， $E(s)$: state s 的能量

$$P_T(s) = \frac{e^{-E(s)/kT}}{Z(T)} \qquad Z(T) = \sum_s e^{-E(s)/kT}$$

$Z(T)$: 配分函數 (partition function)

物理量的計算

$$\langle O \rangle_T = \sum_s O(s) P_T(s)$$

平均能量

$$\langle E \rangle_T = \frac{1}{Z(T)} \sum_s E(s) e^{-\beta E(s)} = \frac{\partial}{\partial \beta} \ln Z(T)$$

$$\beta \equiv 1/(kT)$$

Metropolis–Hastings 演算法

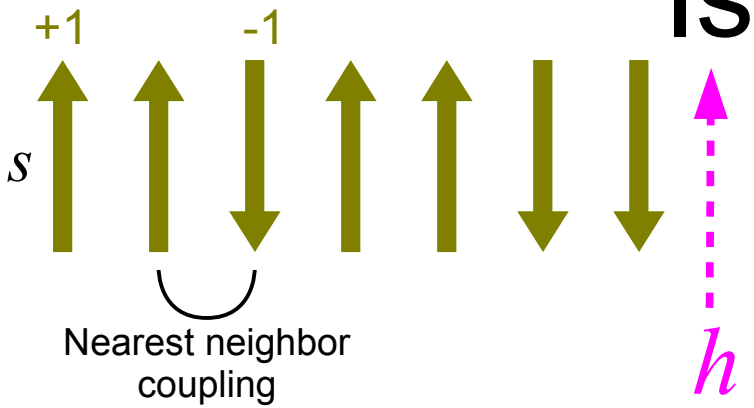
狀態更新的接受機率： $\min(A, 1)$

$$A = e^{-\Delta E/kT}$$

ΔE: 能量的增加量

Ising model

- Ferromagnet
- Lattice gas
- Binary alloy



$$E = -J \sum_{\langle i\,j \rangle} s_i s_j - h \sum_i s_i$$

$$Z = \sum_{\{s_i\}} e^{-\beta E(\boldsymbol{s})} = \sum_{\{s_i\}} e^{\beta J \sum_{\langle i\,j \rangle} s_i s_j + \beta h \sum_i s_i}$$

$$\beta J \sum_i s_i s_{i+1} + \frac{1}{2}\beta h \sum_i (s_i + s_{i+1})$$

1D

Transfer matrix: $T = \begin{pmatrix} e^{\beta(J+h)} & e^{-\beta J} \\ e^{-\beta J} & e^{\beta(J-h)} \end{pmatrix}$

Periodic boundary condition:

$$Z = \mathrm{Tr}(T^N)$$

Transfer matrix 的 eigenvalues:

$$\lambda_\pm = e^{\beta J}\left[\cosh(\beta h) \pm \sqrt{\sinh^2(\beta h) + e^{-4\beta J}}\right]$$

Partition function: $Z = \lambda_+^N + \lambda_-^N \approx \lambda_+^N$

Free energy: $A = \ln Z / \beta$

$$U = \frac{\partial}{\partial \beta} \ln Z \underset{h \to 0}{=} N J \tanh(\beta J)$$

$$M = \frac{\partial}{\partial h} A = N \frac{\sinh(\beta h)}{\sqrt{\sinh^2(\beta h) + e^{-4\beta J}}}$$

2D

$$M = \begin{cases} 0 & \text{for } T > T_c \\ N\left[1 - \sinh^{-4}(2\beta J)\right]^{1/8} & \text{otherwise} \end{cases}$$

$$T_c \approx 2.269185\, J$$

實作 Ising 模型

```
#include "ran_nr.hh"
#include <iostream>
#include <cmath>
RanNR rng(123);
int * spins; // 0 or 1
size_t n_spins = 0;
double beta, field;
// tracking system characters
int uspin, walls;
void setup(size_t n)
{
        n_spins = n;
        spins = new int [n_spins];
        for (size_t i = 0; i < n_spins; i ++) spins[i] = 1;
        uspin = n_spins;
        walls = 0;
}
void update()
{
        size_t i = size_t(rng.uniform() * n_spins);
        int ns = spins[(i + n_spins - 1) % n_spins]
                + spins[(i + 1) % n_spins];
        double dE = 2 * spins[i] * (ns + field) * beta;
        if (rng.uniform() < exp(- dE)) {
                spins[i] = - spins[i];
                uspin += spins[i];
                walls -= ns * spins[i];
        }
}
void mc_step()
{
        for (size_t i = 0; i < n_spins; i ++) update();
}
```

```
double m_cnt;
double m_eng;
double m_mag;
void measure()
{
        m_cnt ++;
        m_eng += walls;
        m_mag += uspin;
}
int main()
{
        using namespace std;
        size_t sz;
        cin >> beta >> field >> sz;
        setup(sz);
        m_cnt = 0;
        m_eng = 0;
        m_mag = 0;
        for (size_t i = 0; i < 1000; i ++) mc_step(); // warm up
        for (size_t i = 0; i < 10000; i ++) {
                mc_step();
                measure();
        }
        cout << "eng="
                << m_eng / m_cnt / n_spins - 1 << '\n';
        cout << "mag="
                << 2 * m_mag / m_cnt / n_spins - 1 << '\n';
        return 0;
}
```

程式碼物件化

程式介面 (interface)

```cpp
// ising1.hh
#include "system.hh"
#include "ran_nr.hh"
class Ising1 :
        public System
{
        // parameters
        double beta;
        double field;
        // state
        int * spins; // 0 or 1
        size_t n_spins;
        RanNR rng;
        // tracking system characters
        int uspin;
        int walls;
        // measurements
        int warm_up;
        double m_cnt;
        double m_eng;
        double m_mag;
        void set_params(double b, double h);
        void update();
        // overrides
        void initialize();
        void mc_step();
        void set_params(const std::vector<double> & p);
        double get_mcnt() const {return m_cnt;}
        std::vector<double> get_results() const;
public:
        Ising1(size_t n, const RanNR & r = RanNR());
        ~Ising1();
};
```

```cpp
// system.hh
#ifndef SYSTEM_HH
#define SYSTEM_HH
#include <iostream>
#include <vector>
class System
{
public:
        virtual void initialize() = 0;
        virtual void mc_step() = 0;
        virtual void set_params(const std::vector<double> & params) = 0;
        virtual double get_mcnt() const = 0;
        virtual std::vector<double> get_results() const = 0;
};
extern System * get_system(size_t n);
#endif // SYSTEM_
```

- Inheritance
- Virtual member

```cpp
// use_sys.cc
#include "system.hh"
int main () {
        size_t sz; double beta, field;
        std::cin >> sz >> beta >> field;
        System * sys = get_system(1024);
        std::vector<double> pm(2);
        pm[0] = beta; pm[1] = field;
        sys->set_params(pm); sys->initialize();
        while (sys->get_mcnt() < 10000) sys->mc_step();
        std::vector<double> r = sys->get_results();
        for (size_t i = 0; i < r.size(); i ++) std::cout << '\t' << r[i];
        std::cout << '\n'; return 0;
}
```

程式介面

class System

class Percolation

class Ising2

class Ising1

use_sys.cc

ave_sys.cc

animate_sys.cc

ising1.cc

```
// ising1.cc
#include "ising1.hh"
#include <cmath>
Ising1::Ising1(size_t n, const RanNR & r) :
        rng(r)
{
        n_spins = n; spins = new int [n_spins];
        initialize();
}
Ising1::~Ising1()
{
        delete [] spins;
}
void Ising1::set_params(double b, double h)
{
        beta = b; field = h;
}
void Ising1::initialize()
{
        for (size_t i = 0; i < n_spins; i ++) spins[i] = 1;
        uspin = n_spins; walls = 0; warm_up = n_spins;
        m_cnt = 0; m_eng = 0; m_mag = 0;
}
void Ising1::update()
{
        size_t i = size_t(rng.uniform() * n_spins);
        int ns = spins[(i + n_spins - 1) % n_spins] + spins[(i + 1) % n_spins];
        double dE = 2 * spins[i] * (ns + field) * beta;
        if (rng.uniform() < exp(- dE)) {
                spins[i] = - spins[i];
                uspin += spins[i]; walls -= ns * spins[i];
        }
}
```

```
void Ising1::mc_step()
{
        for (size_t i = 0; i < n_spins; i ++) update();
        if (warm_up > 0) warm_up --;
        else {m_cnt ++; m_eng += walls; m_mag += uspin;}
}
void Ising1::set_params(const std::vector<double> & p)
{
        set_params(p[0], p[1]);
}
std::vector<double> Ising1::get_results() const
{
        std::vector<double> r;
        r.push_back(m_eng / m_cnt / n_spins - 1);
        r.push_back(2 * m_mag / m_cnt / n_spins - 1);
        return r;
}
```

```
System * get_system(size_t n)
{
        Ising1 * i = new Ising1(n);
        return i;
}
```

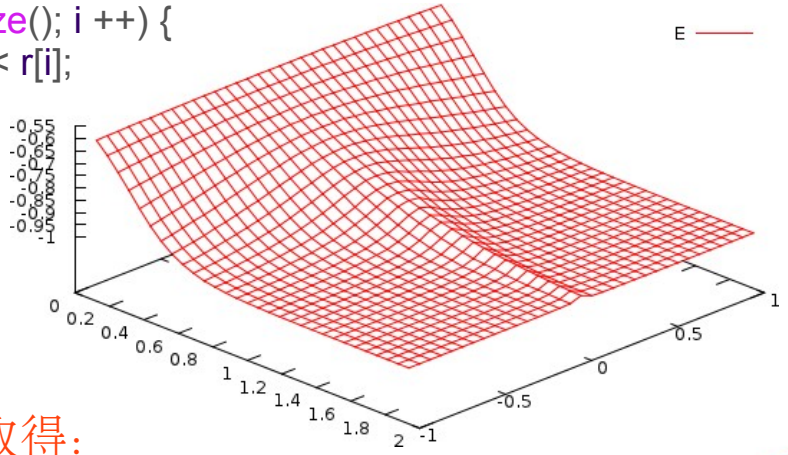
掃描 Phase space

```cpp
// ave_sys.cc
#include "system.hh"
int main ()
{
        System * sys = get_system(1024);
        std::vector<double> pm(2);
        for (double b = 0.125; b < 2; b += 0.0625) {
                for (double h = - 1; h <= 1; h += 0.0625) {
                        pm[0] = b; pm[1] = h;
                        sys->set_params(pm);
                        sys->initialize();
                        while (sys->get_mcnt() < 1000) sys->mc_step();
                        std::vector<double> r = sys->get_results();
                        std::cout << b << '\t' << h;
                        for (size_t i = 0; i < r.size(); i ++) {
                                std::cout << '\t' << r[i];
                        }
                        std::cout << std::endl;
                }
                std::cout << '\n';
        }
        return 0;
}
```

Makefile

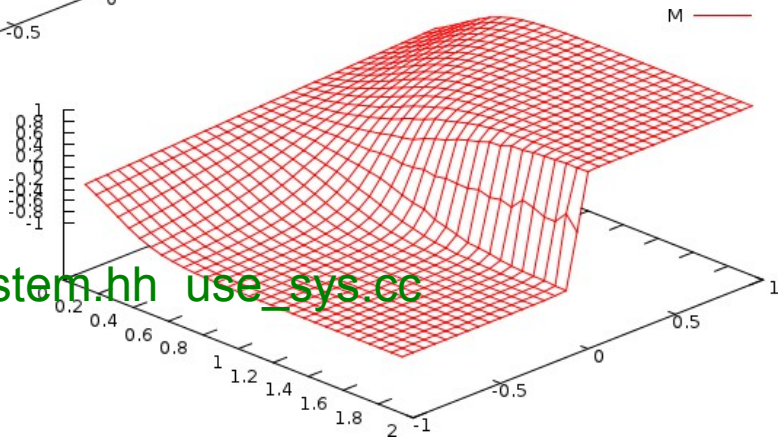
```
all:

all: use_ising1
use_ising1: use_sys.cc ising1.cc
        $(CXX) $^ -o $@
all: ave_ising1
ave_ising1: ave_sys.cc ising1.cc
        $(CXX) $^ -o $@
```



程式碼可在 CP1 SSH Server 上取得:

```
cp1@area:~$ cp -r /usr/local/src/i1 i1
cp1@area:~$ cd i1
cp1@area:~/i1$ ls
ave_sys.cc  ising1.cc  ising1.hh  Makefile  ran_nr.hh  system.hh  use_sys.cc
cp1@area:~/i1$ make
g++ use_sys.cc ising1.cc -o use_ising1
g++ ave_sys.cc ising1.cc -o ave_ising1
```



本週習題

1. 寫程式實作 2D Ising 模型。令 $J = 1$，計算一 20×20 的週期邊界系統在 $\beta = 1 \sim 4$ 及 $h = -1 \sim 1$ 時的能量 E 及磁化量 M。用得到的結果來作它們對 β, h 的 3D 圖形。