靜態 Schrödinger 方程式

(time-independent)

$$\frac{d^2\phi}{dx^2}+\frac{2m}{\hbar}[E-V(x)]\phi=0$$

差分展開

$$\frac{d\phi}{dx}\approx\frac{\phi_{i+1}-\phi_i}{a}\approx\frac{\phi_i-\phi_{i-1}}{a}\qquad\frac{d^2\phi}{dx^2}\approx\frac{\phi_{i+1}+\phi_{i-1}-2\phi_i}{a^2},\quad a\equiv x_{i+1}-x_i$$

$$\phi_{i+1}=2\phi_i-\phi_{i-1}-a^2\frac{2m}{\hbar}[E-V(x)]\phi_i$$ Euler 方法

Numerov 方法（純二次，線性），誤差 $O(a^4)$

$$\phi_{i+1}=\frac{(24-10a^2f_i)\phi_i-(12+a^2f_{i-1})\phi_{i-1}}{12+a^2f_{i+1}}\qquad f(x)=\frac{2m}{\hbar}[E-V(x)]$$

以函數為參數

把函數位址當型別

傳送函數位址

用位址呼叫函數

一次化的 Schrödinger 方程

有相同 signature 的函數

```
#include <iostream>
#include <cmath>
using namespace std;
typedef void (*SlopeFunc)(double t, double * y, double * f);
void runge_kutta(double t, double * y, double h, SlopeFunc sf, int ydim)
{
     double k0[ydim], k1[ydim], k2[ydim], k3[ydim], yy[ydim];
     (*sf)(t, y, k0);
     for (int i = 0; i < ydim; i ++) yy[i] = y[i] + k0[i] * h / 2;
     (*sf)(t + h / 2, yy, k1);
     for (int i = 0; i < ydim; i ++) yy[i] = y[i] + k1[i] * h / 2;
     (*sf)(t + h / 2, yy, k2);
     for (int i = 0; i < ydim; i ++) yy[i] = y[i] + k2[i] * h;
     (*sf)(t + h, yy, k3);
     for (int i = 0; i < ydim; i ++) y[i] += (k0[i] + 2 * (k1[i] + k2[i]) + k3[i]) * h / 6;
}

double potential(double x)
{
     if (x > - 1 && x < 1) return - 44;
     return 0;
}

double energy;

void schrodinger(double x, double * y, double * f)
{ // y[0]: \varphi, y[1]: \varphi'
     f[0] = y[1];
     f[1] = y[0] * (potential(x) - energy);
}

int main() {
     double xmin = - 4; double xmax = 4;
     int npts = 2000;
     cin >> energy;
     double y[2];
     y[0] = 0.01;
     // y[1] = sqrt(- energy) * y[0];
     y[1] = 0;
     int nn = 0;
     double y0 = y[0];
     double dx = (xmax - xmin) / npts;
     for (int i = 0; i < npts; i ++) {
          double x = xmin + i * dx;
          cout << x << '\t' << y[0] << '\t' << nn << '\n';
          runge_kutta(x, y, dx, &schrodinger, 2);
          if (y0 * y[0] < 0) nn ++;
          y0 = y[0];
     }
     return 0;
}
```

邊界值問題

$$\frac{d\vec{y}}{dt} = f(\vec{y}, t)$$

$$y_1(0) = A_0, \quad y_1(1) = A_1$$

線性方程

$$\frac{d\vec{y}}{dt} = F(t)\vec{y} + G(t)$$

可以利用線性組合來求解:

$v[s\,\alpha_1+(1\text{-}s)\alpha_2]=s\,v(\alpha_1)+(1\text{-}s)v(\alpha_2)$

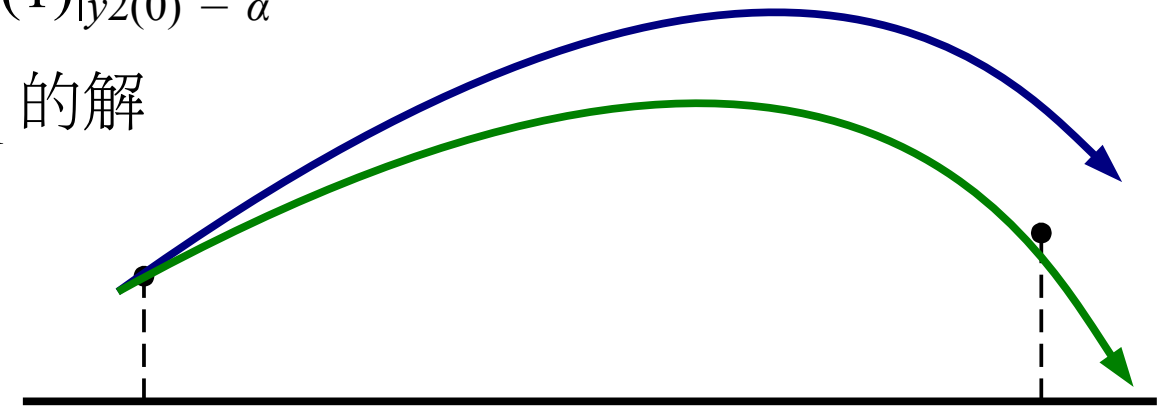
射擊（shooting）法

任取一 $y_2(0) = \alpha$ 值

積分到 $t=1$ 求 $v(\alpha) \equiv y_1(1)|_{y2(0)\,=\,\alpha}$

用求根法找到 $v(\alpha) = A_1$ 的解

由大值積分到小值時須注意數值誤差。例如: Schrödinger 方程式。



c++ 陣列實作細節

```
int a[3][5]={{0,1,2,3,4},{5,6,7,8,9},{10,11,12,13,14}};
```

陣列 (array) 到指標 (pointer) 的轉換：

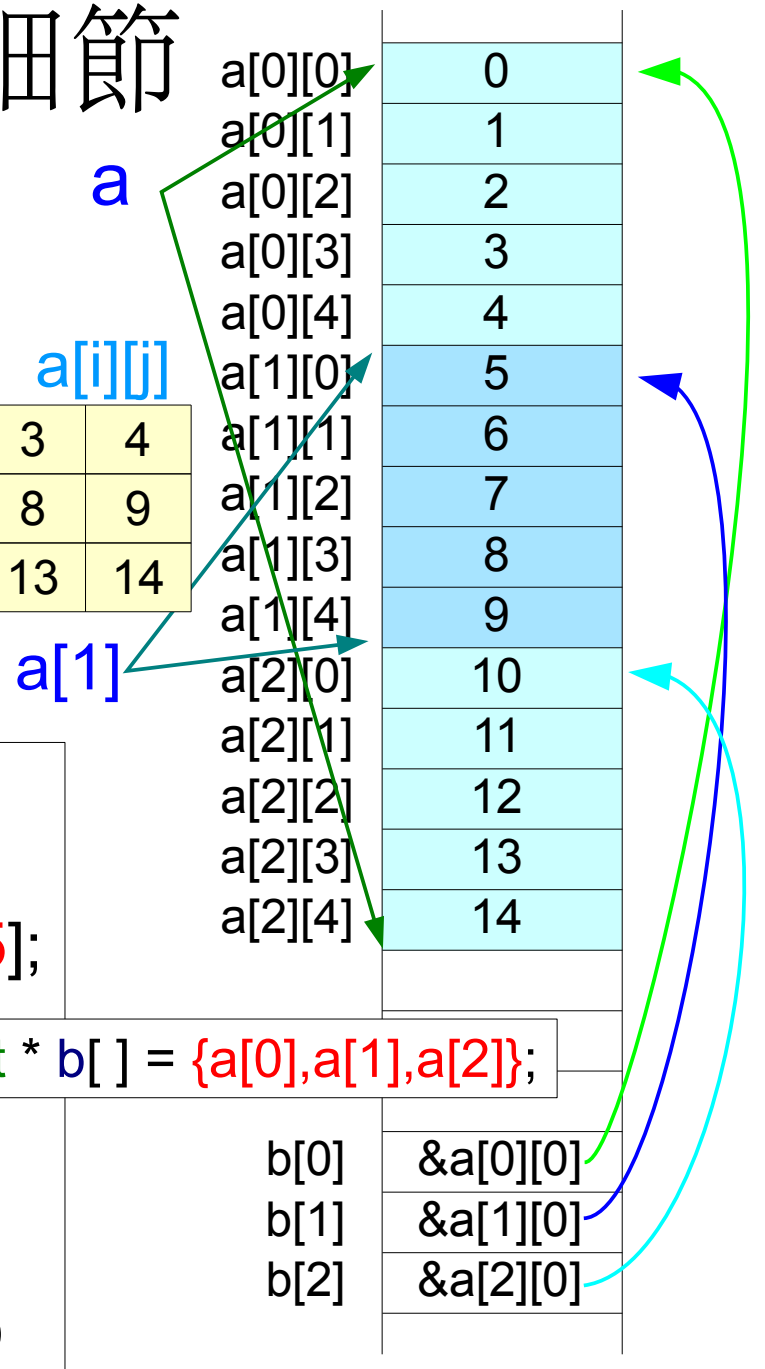
取首位元素的地址

```
Type array[Num];
Type * pointer = array;
```

⇒ `pointer = & array[0];`

i \ j →	0	1	2	3	4
	5	6	7	8	9
	10	11	12	13	14

a[i][j]



手動建置的多維陣列：

```
int * * c = new int * [3];
for (size_t i = 0;i < 3;i ++) c[i] = new int[5];
```

釋回：

```
for (size_t i = 0;i < 3;i ++) delete [] c[i];
delete [] c;
```

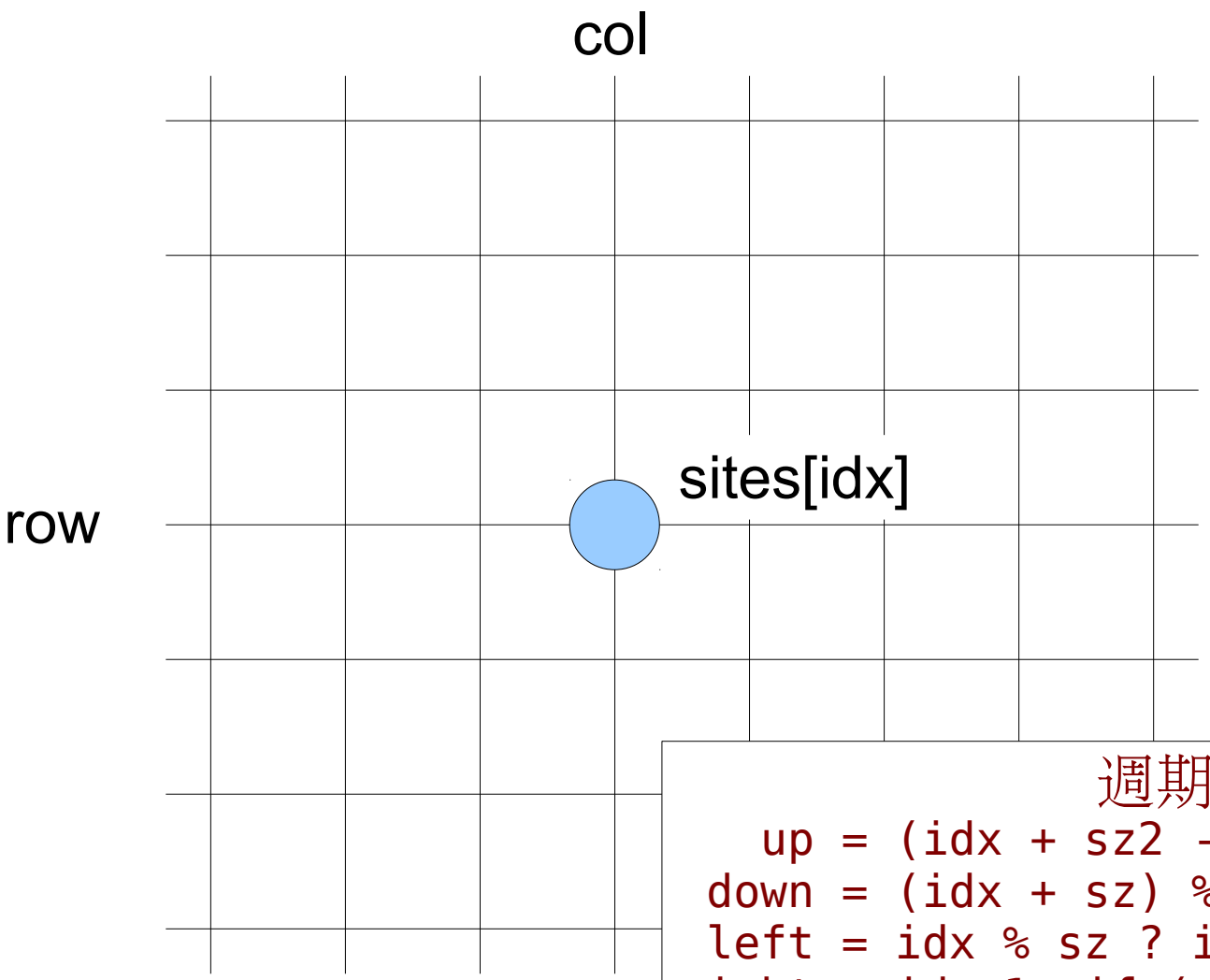
使用：　“ c[2][4] = 6;”　（dereference 兩次）

二維格點系統

```
size_t sz2 = sz * sz;
bool sites[sz2];
```

線性化的索引

```
idx=row*sz+col
```



鄰居索引

上: idx-sz

下: idx+sz

左: idx-1

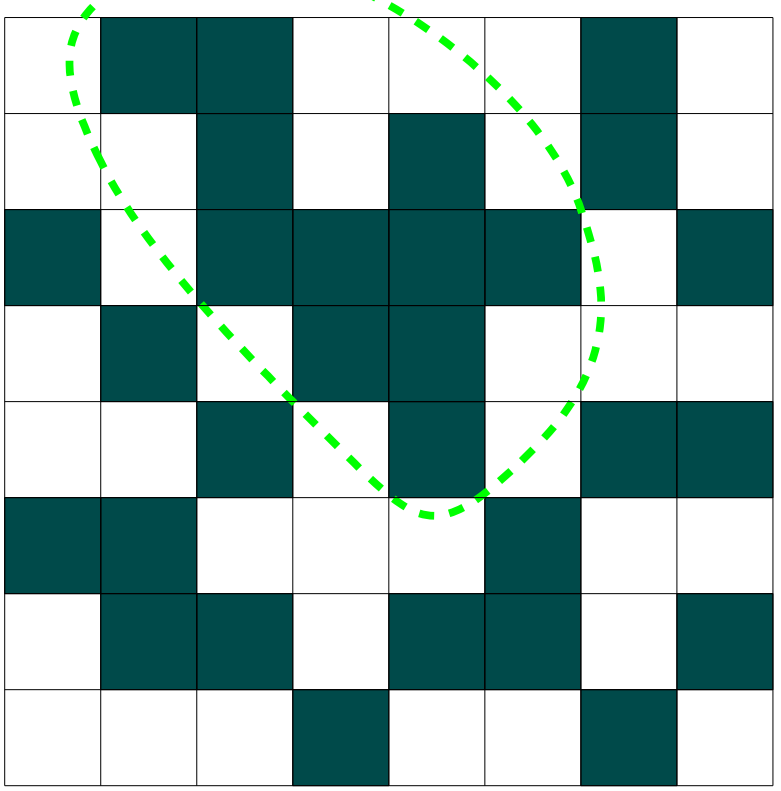
右: idx+1

週期性邊界

```
   up = (idx + sz2 - sz) % sz2;
 down = (idx + sz) % sz2;
 left = idx % sz ? idx - 1 : idx + sz - 1;
right = idx+1; if (right%sz==0) right-=sz;
```

二維滲流（Percolation）

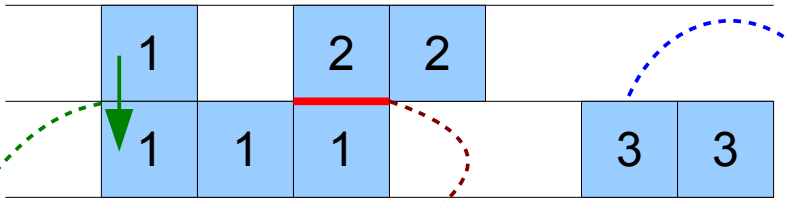
簇集
cluster



```
for (size_t i = 0; i < sz2; i ++)
    sites[i] = rng.uniform() < p;
```

Hoshen-Kopelman 演算法

```
int lab[sz2];  // 位置所屬簇集
int nlab[sz2]; // 簇集大小或真簇集
```



簇集結合，id 小的成為 id 大的真簇集

```
nlab[1] += nlab[2]; // 大小結合
nlab[2] = - 1;      // 指定真簇集
```

簇集增大

```
nlab[1] ++;
lab[new_site] = 1;
```

新簇集

```
lab[new_site] = 3;
nlab[3] = 1;
```

找所屬真簇集

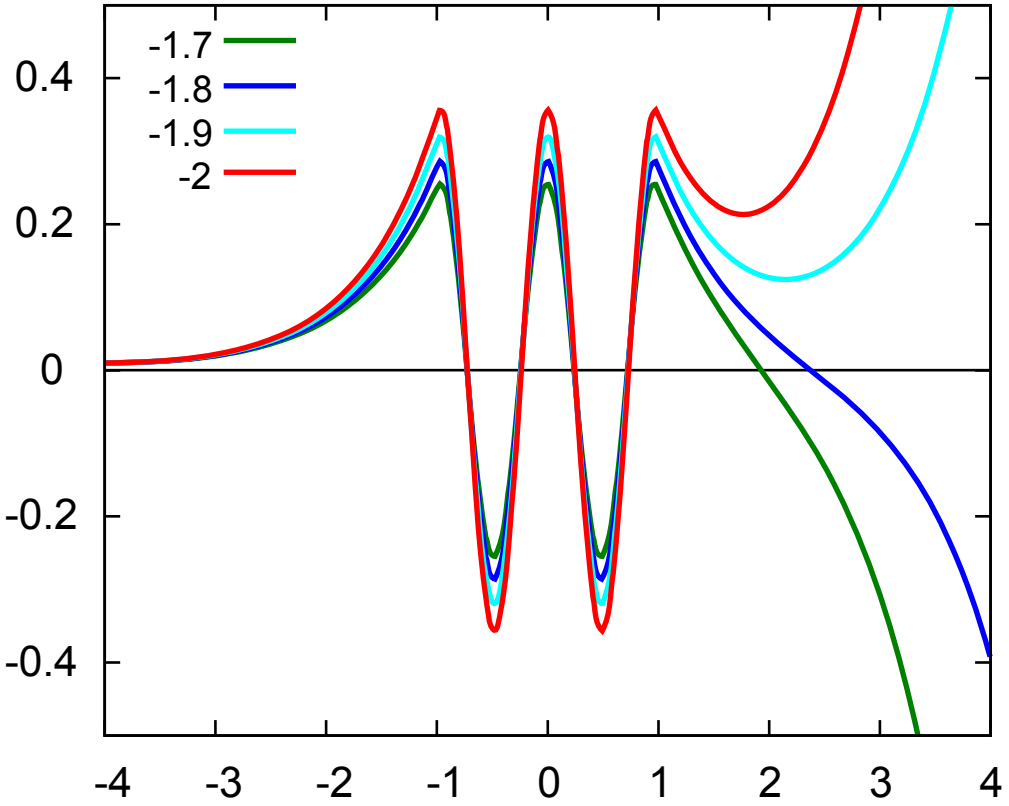
```
int l = lab[site];
while (nlab[l] <= 0) l = - nlab[l];
```

本週作業

1. 寫程式以 RK4 數值方法積分一維 Schrodinger 方程式。假定位能井 $V(x) = -44\theta(1-|x|)$，用射擊法在 $x \in [-4,4]$ 的區間中找尋束縛態的本徵能量及本徵函數（約有五組，函數畫在一張圖即可）。［隨意題］與理論值比較。
2. 實作 Hoshen-Kopelman 演算法：寫程式輸入二維點位滲流 (site percolation) 的點格大小 (lattice size) 及點位佔有機率 p，用 1000 個 seed 的亂數序列來求平均最大簇集 (cluster) 大小。［如無法用 H-K 法，可用多次掃描來標定 site 所屬的 cluster。］
3. 列出三個你到目前在所修過課程中遇到最有趣的物理問題。［如果沒有物理問題，其他科學或益智問題亦可。］

作業輸出範例

在 E_4 附近的 RK4 積分




```
cp1@area:~$ echo 16 0.6 7 |./hoshen_kopelman
# size 16, p=0.6, seed 7
                1                   2       3       3       3   3
      1   1   1   1           4       3   3   3   3   3   3   3
          1   1   1       5       3   3   3   3       3   3
  1   1   1   1   1               3   3       3   3       3
  1   1                   6       3   3       3   3       3   3   3
  1       7                   8               3   3   3           3
  1                   8   8   8       9               3   3   3   3
  1       8   8   8   8   8                   10      3   3       3
      8   8   8   8   8       11      10  10      3   3       3
  3           8           11  11      10      3   3   3       3
  3   3       8       12      11              3       3   3
  3   3   3           12  12      3   3   3   3   3   3   3   3
  3   3                       3   3   3   3   3   3       3
  3       3       3   3   3       3   3   3   3   3           3
  3   3   3   3   3       3   3   3   3   3   3   3   3   3
  3   3   3   3   3       3   3   3   3   3           3       13
# found 13 cluters.
# largest cluster has 109 sites.
cp1@area:~$
```