おしらせ

- **第一回、および二回のレポートを、学科事務室にて返却しています**
  - 各人、都合の良い時に、早めに取りに行ってください
- **課題レポート提出に関して**
  - **期限：２月９日（月）正午**
  - **提出先：物理学科事務室**
- **補講に関して**
  - **１月22日に補講を行いますが、この時間はレポート作成の時間とします（講義・演習は無し）**
  - **レポート作成に際して質問などある時はこの時間を有効に活用下さい。**

数値計算法

第十二回
乱数とモンテカルロ法
（続き）

若狭　智嗣

粒子物理学講座

より直接的に分布が指数関数的か確認する ～ 度数分布（頻度分布）を作成する。～

- **発生した乱数の度数（頻度）分布を作成し、指数関数的か確認**
  - 乱数の値域をいくつかの区間に分けて、各区間の個数を数える
  - Fortranの場合=配列を用意して、「*i* 番目の区間の個数 = *y*(*i*) 」等とする

- **20個の値を書き出した時の最大値＝2.1134 → 区間の最大値を 3.0 にとる**
  - 0.0から3.0までを、0.2刻みで分けて、各区間の個数を数える
    - 3.0より大きい場合は、最後の区間の数とする
      （原理的には∞まであり得るので）

- **以下、どのように Fortran で実現するか考えてみる**
  - あくまで1例であり、よりよい方法もある。
  - Excel 等を利用すれば、より簡便に実現できる

} Fortran の演習としては有益

度数分布の作成指針

int = 実数を整数にするFortranの関数（切り捨て）

乱数の値　x	$5x$	int(5*x)	int(5*x)+1	対応する配列
0.0～0.2	0～1	0	1	y(1)
0.2～0.4	1～2	1	2	y(2)
0.4～0.6	2～3	2	3	y(3)
⋮	⋮	⋮	⋮	⋮
2.6～2.8	13～14	13	14	y(14)
2.8～	14～	14～	15～	y(15)

✓乱数 x に対して、int(5*x)+1 番目の配列 y を1増やす

✓int(5*x)+1が16以上の場合も15番目を増やす

→ min(int(5*x)+1,15)番目を増やす

小さい方の値を返す

プログラム例

expran_check.f

```
      program expran_check
      implicit none
c const:
      integer N
      parameter(N=100)  ← 発生させる乱数の数
      integer NBIN
      parameter(NBIN=15)  ← 区間の数
c slope parameter:
      real    lambda
      parameter(lambda=3.1416)
c for histotram:
      real    y(NBIN)  ← 各区間の数を入れる配列
                         （計算の都合上、実数型で定義）
c function:
      real    expran
c loop etc:
      integer i,j
```

プログラム例　続き

expran_check.f

```
c begin:
      do i=1,NBIN
         y(i) = 0.0
      end do
      do i=1,N
         j = int(5.0*expran(lambda))+1
         y(min(j,NBIN)) = y(min(j,NBIN)) + 1.0
      end do
      do i=1,NBIN
         write(*,*) real(i)/5.0-0.1,y(i),sqrt(y(i))
      end do
      stop
      end
```

← 各区間の数を 0 に初期化

← N個の乱数を発生

←j=int(5*x)+1

対応する配列を1増やす

i番目の区分の中心値
i=1 → 0.1
i=2 → 0.3
…

ポアソン分布を
仮定した時の誤差

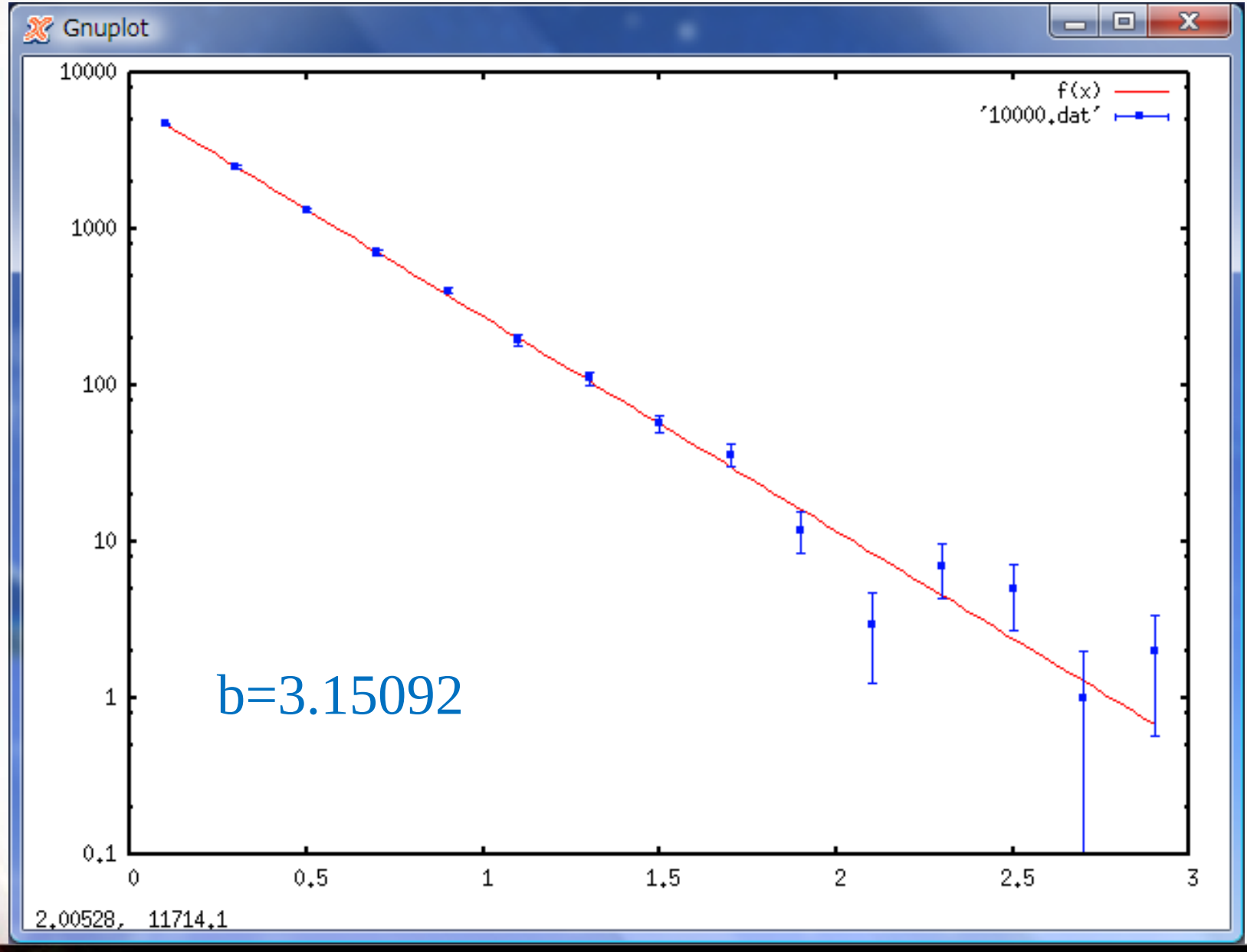
演習

- **プログラムをコンパイルし、実行せよ**
  - 結果を、リダイレクションによりデータをファイルに出力せよ
  - Gnuplot を用いて、データをプロットせよ
    - yの誤差棒を付ける事
    - 縦軸を log スケールにしてみよ
- **N=10000にして実行し、結果をファイルに出力せよ**
  - 結果を、gnuplot を用いて指数関数

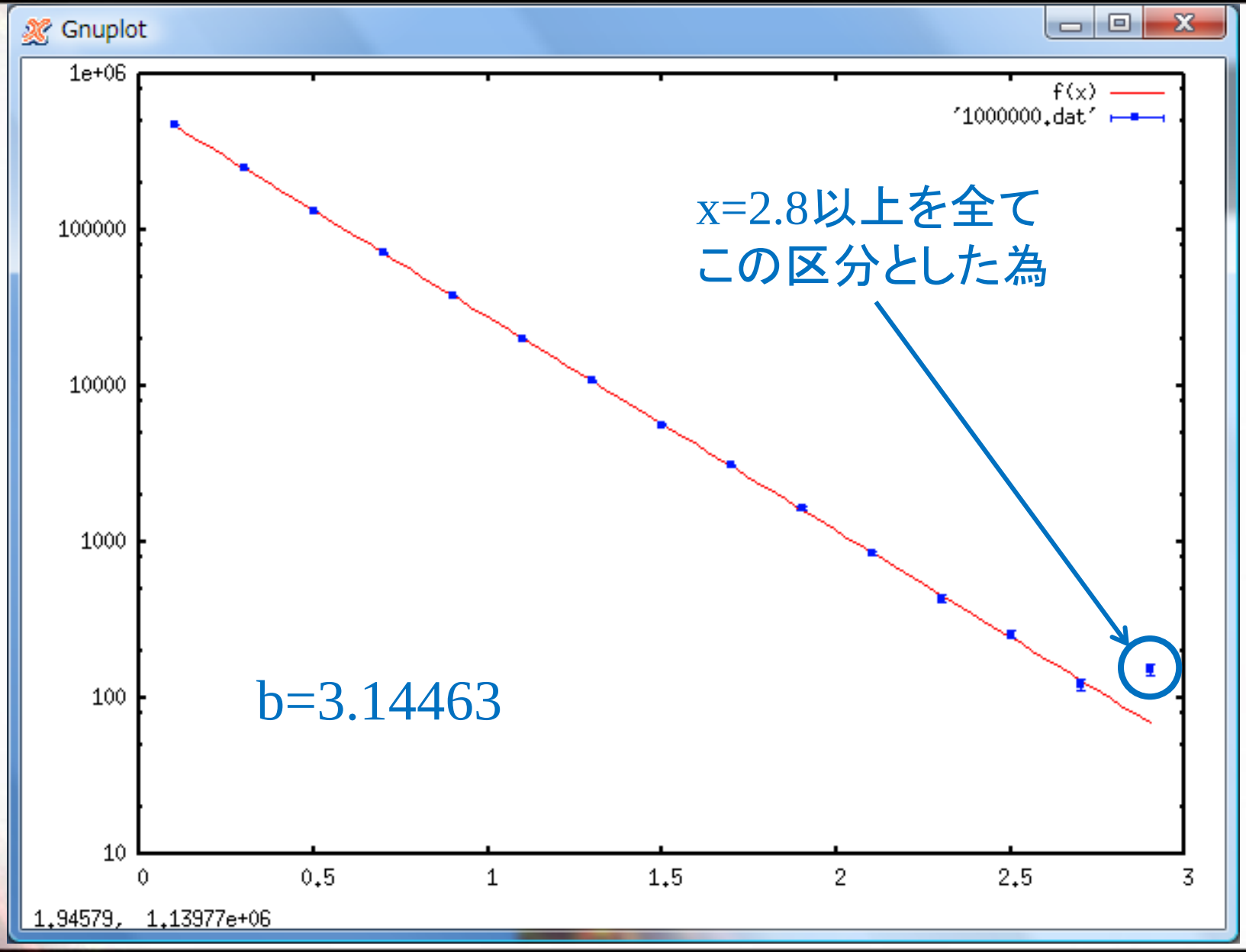
$$f(x) = a * \exp(-b * x)$$

    でフィッティングを行い、bが期待される3.1416に近いか確認せよ
  - データとフィッティングで求めた線を同時に描き、データが線に対してどのようにばらついているか確認せよ
    - 誤差棒の範囲内“程度”に線があるはず
- **N=1000000　にして前と同じ事を行い、以下の事を確認せよ**
  - b が期待される3.1416により近くなる
  - フィッティングで求めた線に対する「ばらつき」が小さくなる

N=10000の場合


Gnuplot
10000
1000
100
10
1
0.1
0
0.5
1
1.5
2
2.5
3
f(x)
'10000.dat'
b=3.15092
2.00528, 11714.1

N=1000000の場合



モンテカルロ積分

- **以下の積分を考える。**

$$I = \int_a^b f(x)dx$$

- **数値積分の台形公式**
  - 積分区間 $[a,b]$ を等間隔に分割して積分を評価
- **モンテカルロ積分では…**
  - 積分区間 $[a,b]$ に一様に発生する乱数 x_i を N 個発生

$$I = \int_a^b f(x)dx \approx \frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i) \qquad dx \approx \frac{b-a}{N}$$

  - 区分求積法と基本は同じだが、
    - 区分求積法: x_i は等間隔
    - モンテカルロ積分: x_i はランダム
- **モンテカルロ積分の精度**
  - 中心極限定理から、$O(N^{-1/2})$
    - 4次元以上の積分では台形法より有利

演習

課題

2,4,5,10 次元空間の半径 1 の (超) 球の体積をモンテカルロ法により求めよ。d 次元球のときには、$[0,1]$ に一様に分布する乱数を d 個用意して、それらの 2 乗和が 1 より小さくなる割合を求めればよい。得られた体積を解析的に求めた結果と比較せよ。余裕があれば、モンテカルロ法以外の数値積分法でも計算して、計算精度や計算時間を比較せよ。

- **上記課題の2次元（円）の場合のプログラムを作成せよ。**
  - [0,1]の一様乱数で計算する場合、円の面積の１／４（x、yが共に正の部分の面積）を求めていることになることに注意
  - 半径1の円の面積はπ なので、π を求めることに相当
    - 数値計算でπ を求める場合は、より高速な方法がある

プログラムの例

```
      program rand_2d
      implicit none
c const:
      integer  N
      parameter(N=10000)
c loop:
      integer  i,j
c function:
      real     rand
      external rand
c begin
      j = 0
      do i=1,N
         if (rand(0)**2+rand(0)**2.le.1.0) then
            j = j + 1
         endif
      end do
      write(*,'(A,F8.5)') 'PI = ',float(j)/float(N)*4.0
      stop
      end
```

← 発生する乱数の数

← 初期化

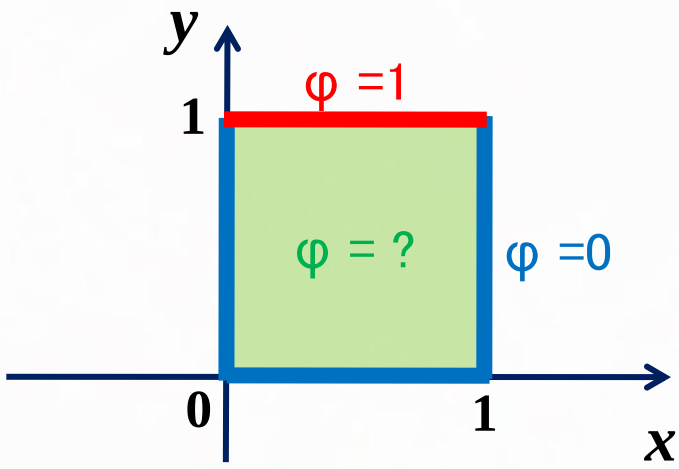
乱数を2つ発生して、
その二乗和が1以下なら

二乗和が1以下の確率×4

モンテカルロ法と境界値問題(Dirichlet問題)

- 電荷が無い場合の2次元正電場のポテンシャルφ (x,y)
  (但し、0 < x < 1, 0 < y < 1)を以下の境界条件で求めよ
  - φ (x,0) = φ (0,y) = φ (1,y) = 0
  - φ (x,1) = 1
- 解くべき方程式はラプラス方程式

$$\frac{\partial^2}{\partial x^2}\phi + \frac{\partial^2}{\partial y^2}\phi = 0$$

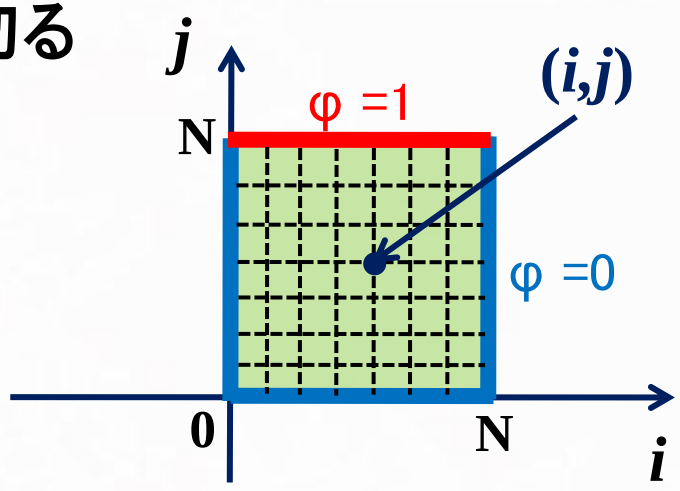


- この問題自身は、第9章「Relaxation」法で解く法が
  精度が出やすい(速い)

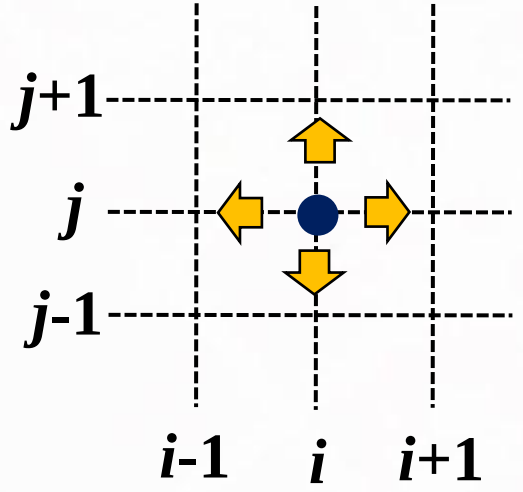
- モンテカルロ法(酔歩)でも解けるので、例題として扱う

酔歩と境界値問題(Dirichlet問題)

- 考える領域を縦・横共にN分割の格子に区切る
  - x軸 : i=0 (x=0) ～ i=N (x=1)
  - y軸 : j=0 (y=0) ～ j=N (y=1)



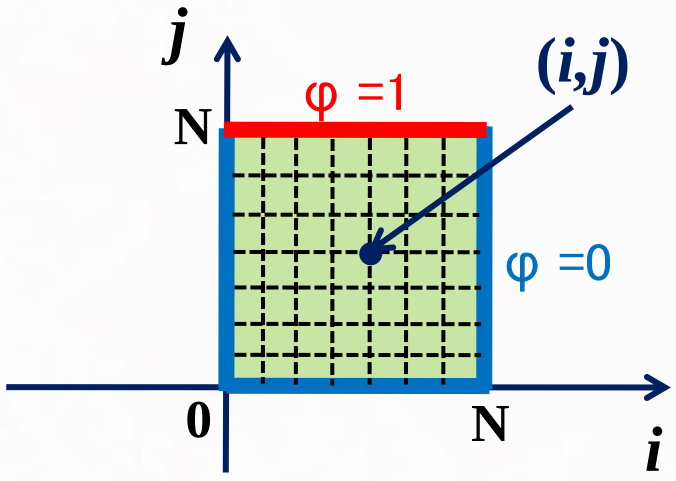
- ある点(i,j)から酔歩する
  - “必ず”上下左右どちらかの格子点に移動
  - どの格子点に移動するかはランダム(確率1/4)
  - いずれ境界に達する→境界での値をφ_kとする
    - Φ_kは今の場合、0 または 1



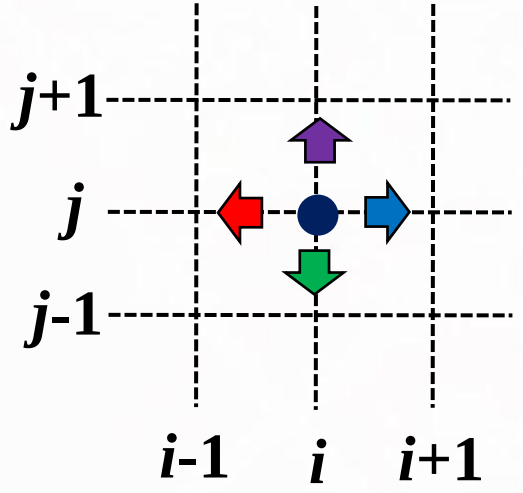
- 同じ点(i,j)からの酔歩をk回行う
  - φ_kの平均値(期待値)= 点(i,j)でのφ (i,j) ・・・証明省略
    - Φ =1の境界に近い
      →φ =1の境界に達する確率大→φ は1に近い
    - Φ =0の境界に近い
      →φ =0の境界に達する確率大→φ は0に近い

酔歩と乱数（モンテカルロ法）

- ある点(i,j)から酔歩する
  - “必ず”上下左右どちらかの格子点に移動
  - どの格子点に移動するかはランダム（確率1/4）
  - いずれ境界に達する→境界での値をφ$_k$とする
    - Φ$_k$は今の場合、0 または 1



- [0,1]の標準乱数で実現する
  - 0.00から0.25 : 左へ
  - 0.25から0.50 : 右へ
  - 0.50から0.75 : 下へ
  - 0.75から1.00 : 上へ



Fortran で実現してみる

プログラム例

/home/teacher/z6wt01in/SAMPLE/random_walk.f

```
      program random_walk
      implicit none
c const:
      integer   N
      parameter(N=50)          ! 区間の分割数
      integer   NWALK
      parameter(NWALK=50)      ! 酔歩の数
      real      H
      parameter(H=1.0/N)       ! 刻み幅
c local:
      real      phi(0:N,0:N)   ! 2次元ポテンシャル
      integer   x,y            ! 格子点
      real      walk           ! 酔歩
c function:
      real      rand
      external  rand
c loop:
      integer   i,j,k
```

実数を整数で割っている
→結果は実数

0 から N まで
→0 と N は境界上

プログラム例(続き)

/home/teacher/z6wt01in/SAMPLE/random_walk.f

```
c begin:
      do i=0,N                    ! 境界条件
         phi(i,0) = 0.0           ! phi(x,0) = 0  → x軸上(y=0)の各点
         phi(0,i) = 0.0           ! phi(0,y) = 0  → y軸上(x=0)の各点
         phi(N,i) = 0.0           ! phi(1,x) = 0  → x=1軸上の各点
         phi(i,N) = 1.0           ! phi(x,1) = 1  → y=1軸上の各点
      end do
```

プログラム例(続き)

```
do i=1,N-1                    ! 境界(x=0,1)を除く格子点(i-th)
   do j=1,N-1                 ! 境界(y=0,1)を除く格子点(j-th)
      phi(i,j) = 0.0          ! 初期化
      do k=1,NWALK
         x = i                ! 格子点(i,j)からスタート
         y = j
         do while ((x.gt.0.and.x.lt.N).and.
&                  (y.gt.0.and.y.lt.N))
            walk = rand(0) ←[0,1]一様乱数発生
            if (walk.lt.0.25) then 0.00 ～ 0.25の時
               x = x - 1 左へ
            elseif (walk.lt.0.50) then 0.25 ～ 0.50の時
               x = x + 1 右へ
            elseif (walk.lt.0.75) then 0.50 ～ 0.75の時
               y = y - 1 下へ
            else 0.75 ～ 1.00の時
               y = y + 1 上へ
            endif
         end do
         phi(i,j) = phi(i,j) + phi(x,y)
      end do
      phi(i,j) = phi(i,j)/NWALK
   end do
end do
```

NWALK回酔歩する

0<x<Nかつ0<y<Nの間酔歩し続ける（境界に達するまで）

NWALKで割って平均を出す→

たどり着いた境界の値phi(x,y)を加える

プログラム例(続き)

/home/teacher/z6wt01in/SAMPLE/random_walk.f

```
do  j=0,N
    do  i=0,N
        write(*,*) H*i,H*j,phi(i,j)
    end do
    write(*,*)
end do
stop
end
```

2次元上の全格子点(i,j)
(N+1)×(N+1)点の phi を出力

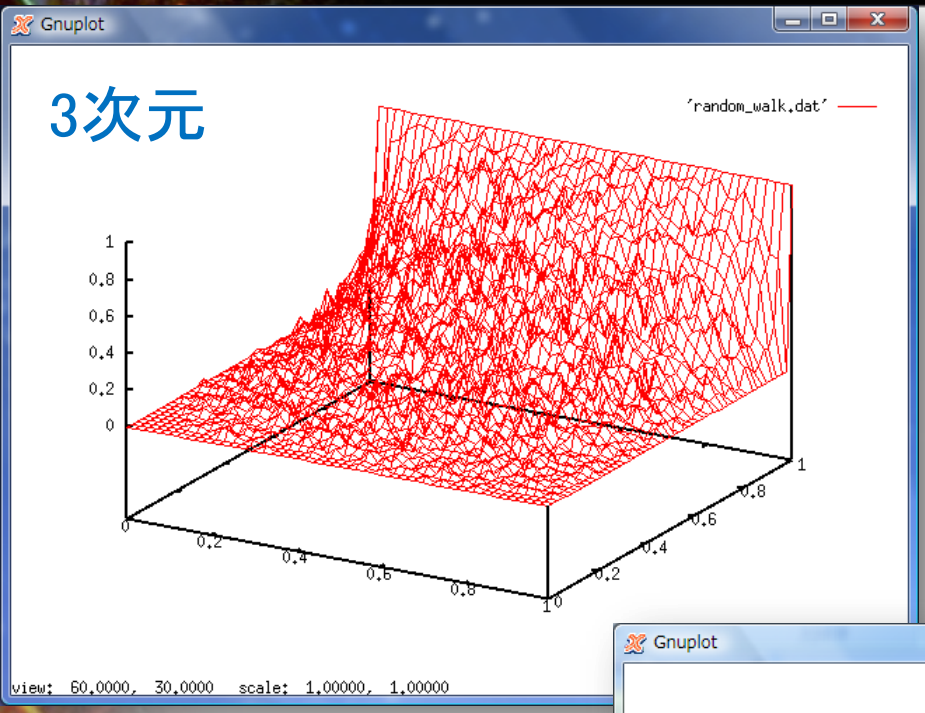
! splotの為の空行

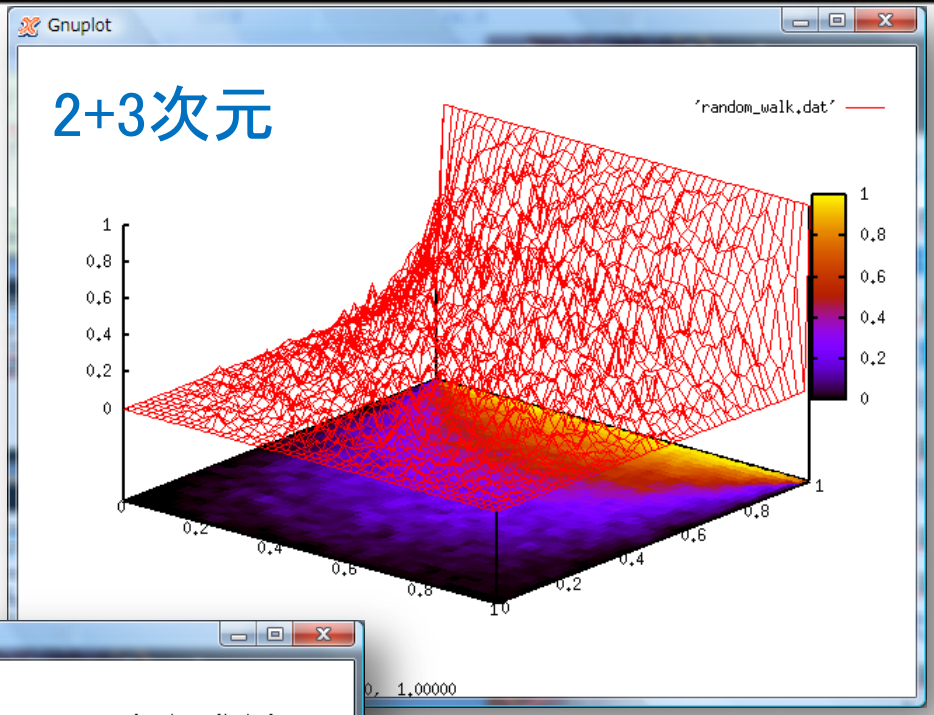
刻み幅 H をかけて
実際の格子点の座標とする

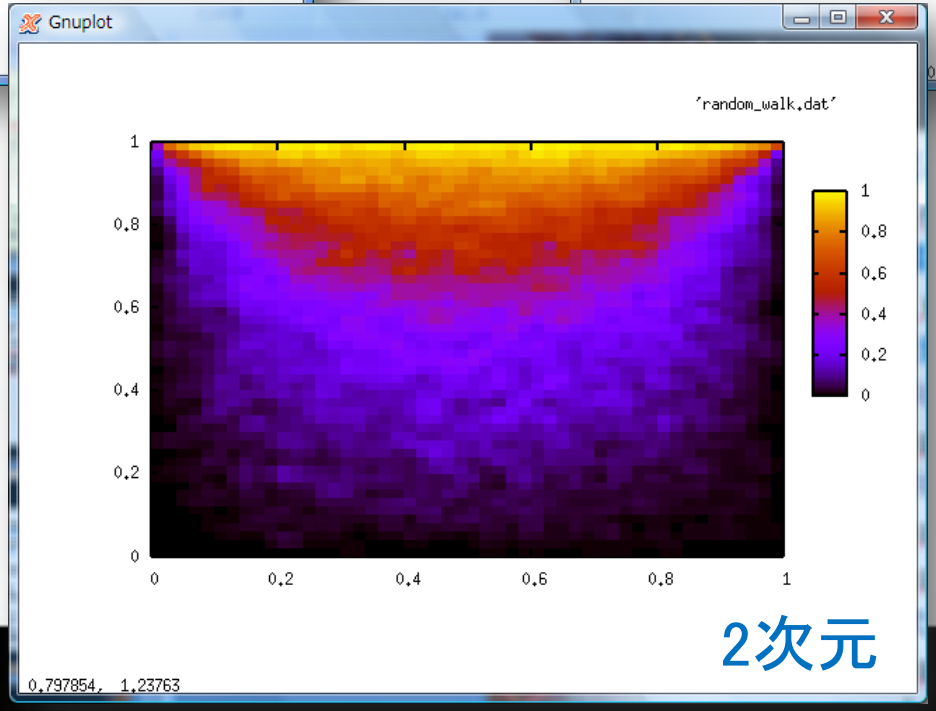
実行の仕方

- コンパイルする
  - % frt -o random_walk random_wark.f
- 実行する(計算結果をリダイレクションでファイルに出力)
  - % random_walk > random_wark.dat
- Gnuplot を起動して描画する
  - 3次元表示(z軸が電位φ)
    - gnuplot> splot ‘random_walk.dat’ with lines
  - 2次元表示(電位を色分け)
    - gnuplot> set pm3d map
    - gnuplot> splot ‘random_walk.dat’
  - 2次元+3次元表示
    - gnuplot> set view 60,40
    - gnuplot> set pm3d at b
    - gnuplot> splot ‘random_walk.dat’ with lines
- 余裕があれば、NWALKを増やして実行してみる（200以上にはしない）

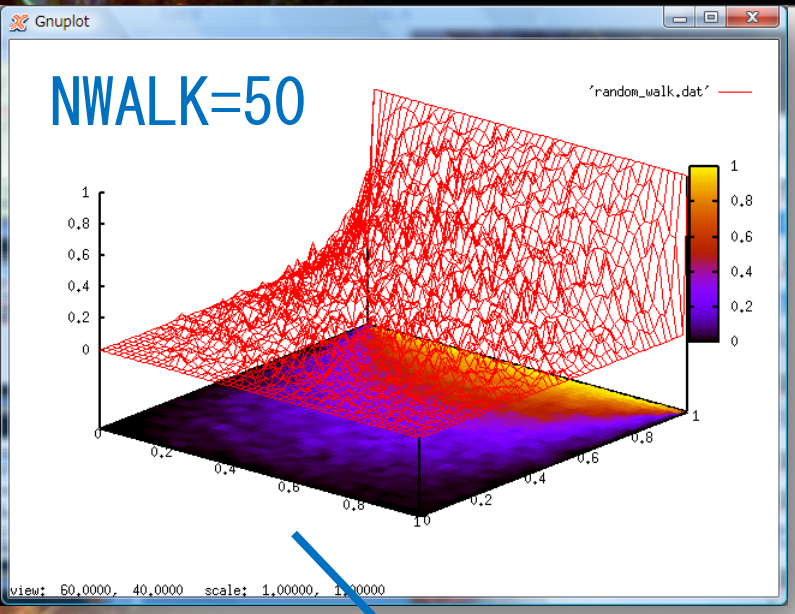
描画例

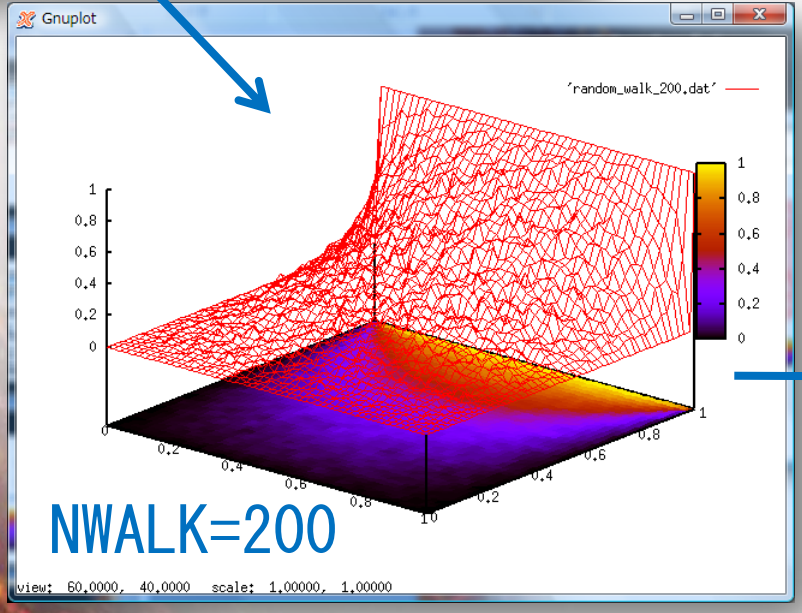

Gnuplot
3次元
'random_walk.dat'
view: 60.0000, 30.0000 scale: 1.00000, 1.00000

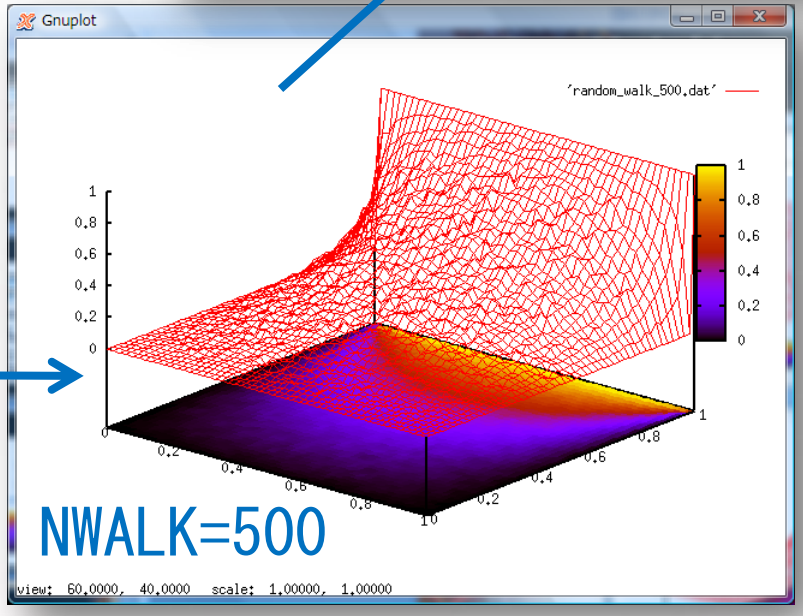

Gnuplot
2+3次元
'random_walk.dat'

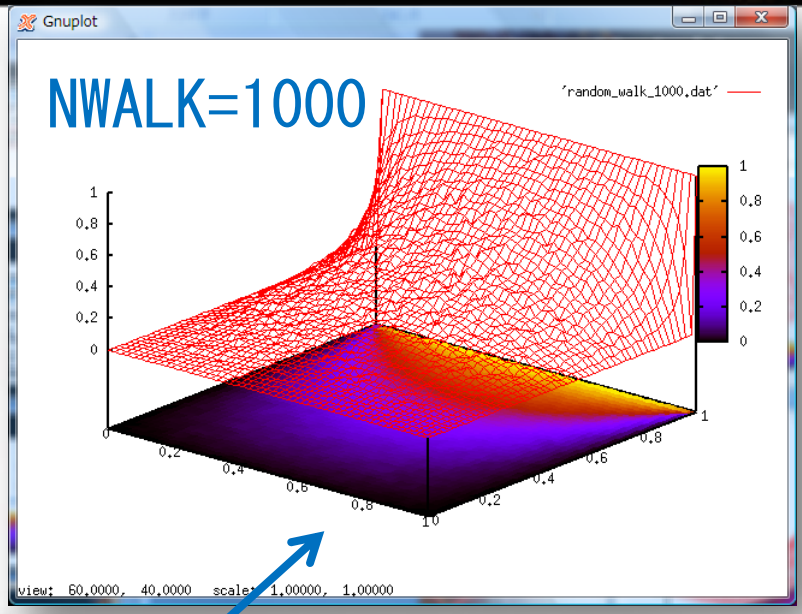

Gnuplot
'random_walk.dat'
2次元
0.797854, 1.23763

NWALK依存性


Gnuplot
NWALK=50
'random_walk.dat'
view: 60.0000, 40.0000 scale: 1.00000, 1.00000


Gnuplot
'random_walk_200.dat'
NWALK=200
view: 60.0000, 40.0000 scale: 1.00000, 1.00000


Gnuplot
'random_walk_500.dat'
NWALK=500
view: 60.0000, 40.0000 scale: 1.00000, 1.00000


Gnuplot
NWALK=1000
'random_walk_1000.dat'
view: 60.0000, 40.0000 scale: 1.00000, 1.00000

授業評価アンケートへの協力のお願い

- 配布する「授業アンケート」用紙に回答し、
  ２月19日（木）
  までに、
  物理第1講義室前の回収箱
  に提出ください。
- 科目名等は以下の通りです。アンケート用紙上部に記入下さい。
  - 授業科目コード　1312
  - 曜日時限コード　42
  - 授業科目名　　数値計算法
  - 担当教員名　　若狭　智嗣