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。**

講義（演習）に関して

- 講義ノートの以下の部分は、講義（演習）では行いません
  - １３章：データ処理
  - １４章：非線形関数の最小２乗法

- 上記の内容は、基本的には gnuplot のフィッティングで出来ますので、gnuplot を活用して下さい。

- 但し、統計処理や誤差論に関する部分は、**実験系の研究室に特別研究や大学院で進むことを考えている人**には役立ちますので、時間のある時に一読しておいて下さい。

- **レポート課題として、上記の章の内容に取り組むことは、全く問題ありません。**

数値計算法

第十一回
乱数とモンテカルロ法

若狭　智嗣

粒子物理学講座

乱数とモンテカルロ法

- **モンテカルロ法**
  - 乱数を用いて
    - 現象の解析を行う（モンテカルロ・シミュレーション）
    - 方程式を解く（モンテカルロ積分）
  - von Neumannらによる核反応シミュレーションが始まり
- **乱数とは**
  - サイコロを振るように予測の出来ない独立な数列を作ること
  - 有名な国営賭博場のあるモナコ公国の都市名にちなみ、「モンテカルロ法」と名づけられた
- **物理での適用範囲**
  - 原子核同士の核反応
  - 鎖状分子の変形
  - 流体の動力学

FORTRANでの乱数の発生方法

- （組み込み）関数による乱数の発生

rand(*iflag*)	[0,1] の範囲の一様乱数を単精度実数で返す
drand(*iflag*)	[0,1] の範囲の一様乱数を倍精度実数で返す
irand(*iflag*)	[0,2147483647] の範囲の一様整数乱数を返す

ここで整数引数 *iflag* の意味は次の通り。

0	疑似乱数列の次の乱数を返す
1	疑似乱数列の生成を再開し、最初の乱数を返す
その他	*iflag* を「種」として疑似乱数列を生成し、最初の乱数を返す

教育用計算機システム上の FORTRAN 77 処理系でも乱数発生用の組み込み関数を持っており使用可能である。但し、関数を external 宣言しておく必要がある (external について知りたいものは、FORTRAN の文法書や解説書を参考にして欲しい)。

- 具体的なプログラム法
  - real rand　　! 関数randの使用を宣言（単精度型）
  - external rand　　! 教育用システムでは必要
  - rand(0)　　! [0.1]の一様乱数が発生

演習

- 以下のコードを入力し、乱数を画面に表示させよ。
  また、出力をリダイレクションし、gnuplotで表示せよ。
  （乱数らしい分布をしてるか？）

```
      program rand_test
      implicit none
c loop:
      integer   i          ! loop用変数
c function:
      real      rand
      external  rand       ! 教育用システムでは必要
c begin:
      do i=1,20
         write(*,'(I3,F9.5)') i,rand(0)
      end do
      stop
      end
```

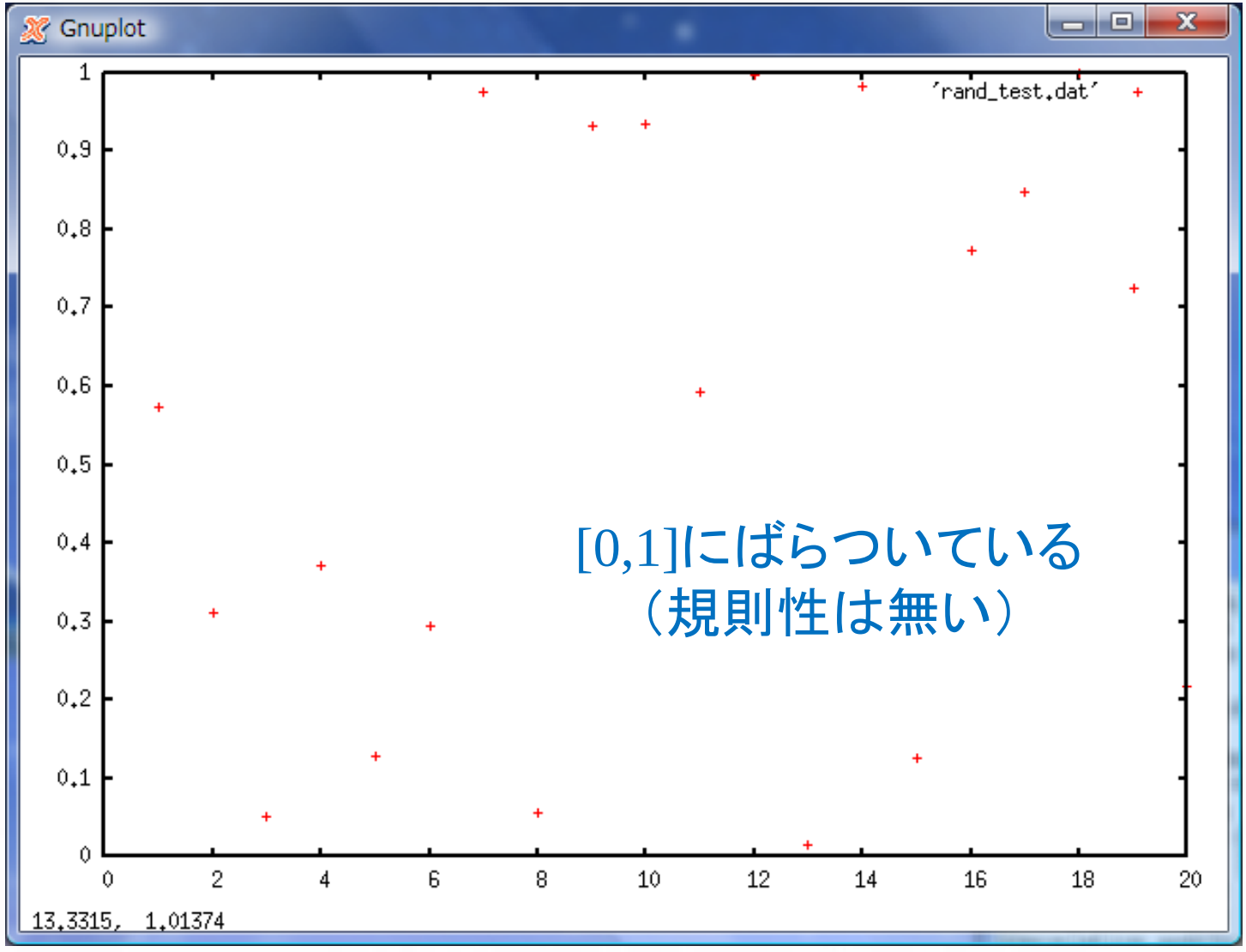
整数型(3桁):何番目か

浮動小数点型（全体9桁、点以下5桁）

i番目の乱数

例

1	0.57489
2	0.31059
3	0.05080
4	0.37071
5	0.12831
6	0.29469
7	0.97466
8	0.05695
9	0.93105
10	0.93497
11	0.59166
12	0.99647
13	0.01512
14	0.98369
15	0.12734
16	0.77209
17	0.84771
18	0.99869
19	0.72398
20	0.21692



[0,1]にばらついている
（規則性は無い）

与えられた分布関数に従う乱数の発生方法

- **逆変換法**
  - 確率密度関数 $f(x)$ と $g(y)$ の一般的関係

$$f(x)dx = g(y)dy$$

  - $f(x)$ が一様乱数のとき

$$\int^{x} f(x')dx' = x$$

$$\int^{y} f(y')dy' = G(y)$$

一様乱数 $x = G(y)$

  - 分布関数 $g(y)$ に従う乱数 y

$$y = G^{-1}(x)$$

→確率密度関数$g(y)$の積分$G(y)$が求まれば、
一様乱数xを用いて、分布関数$g(y)$に従う乱数yが生成可能

演習　指数関数に従う乱数

- **指数関数**

$$g(y) = \lambda \exp(-\lambda y)$$

- **粒子の崩壊（ベータ崩壊、ガンマ崩壊などの放射線）のシミュレーションに有効**
  - 放射線の強さ（放射能）は指数関数的に減少する
  - 量子力学により統計的に振る舞う
    - ある時刻での放射能を正確には予言出来ない
    - （予言出来ない）乱数を用いたシミュレーションが有益
- **上記の関数は確率密度関数になっている**
  - 確率密度関数＝定義域に対する積分値が1に規格化されている
  - 今の場合、定義域は[0，∞]

$$\int_0^\infty \lambda \exp(-\lambda y) dy = \left[-\exp(-\lambda y) \right]_0^\infty = 1$$

演習　指数関数に従う乱数

$$G(y)=\int_0^y \lambda \exp(-\lambda y')dy' = \left[-\exp(-\lambda y)\right]_0^y = 1-\exp(-\lambda y)$$

$$x = G(y) = 1-\exp(-\lambda y)$$

$$y = -\frac{1}{\lambda}\ln(1-x)$$

→ 一様乱数 x から指数関数分布に従う乱数 y が得られる（ハズ）

一様乱数から指数関数分布に従う乱数を得る関数

expran.f

```
      real function expran(lambda)
      implicit none
c i/o:
      real       lambda
c function:
      real       rand
      external   rand
c begin:
      expran = -1.0/lambda*log(1.0-rand(0))
      return
      end
```

←指数関数の肩の係数
（メイン・プログラムから渡す）

←一様乱数([0,1])

一様乱数(ran(0))から指数関数分布expranへ

指数関数に従う乱数を出力するメインプログラム

expran_test.f

```
      program expran_test
      implicit none
c const:
      integer   N          ← 出力する個数(20個)
      parameter(N=20)
      real      lambda     ← λの値
      parameter(lambda=3.1416)
c loop:
      integer   i
c function:
      real      expran     ← 自作した関数(実数型)
c begin:
      do i=1,N
        write(*,'(f8.4)') expran(lambda)
      end do
      stop
      end
```

浮動小数点型で、全体で8桁、小数点以下4桁表示

計算結果の例

0.2723
0.1184
0.0166
0.1474
0.0437
0.1111
1.1699
0.0187
0.8513
0.8699
0.2851
1.7977
0.0048
1.3102
0.0434
0.4707
0.5991
2.1134
0.4098
0.0778

✓0.0–0.5　13個
✓0.5–1.0　3個
✓1.0–1.5　2個
✓1.5–2.0　1個
✓2.0–2.5　1個

指数関数らしい分布

本当にλ=3.1416の指数分布か？

コメント

rand が生成する乱数は、実際は乱数数列の値
✓乱数数列は予め定まっている
✓予め定まってはいるが規則性は無い（疑似乱数）
現在のプログラムでは、必ず乱数数列の１番目から開始
✓毎回、同じ結果になる（真の乱数ではない）
✓隣の人とも同じ結果
これを回避するには、何番目から開始するかを工夫する
✓日付や時間から決めれば毎回異なる

λ =3.1416の指数分布か確認してみる

- 生成した乱数の分布関数が指数関数であれば、
  乱数の値が 1/λ　以下である確率は、

$$P = \int_0^{1/\lambda} \lambda \exp(-\lambda y) dy = \left[-\exp(-\lambda y) \right]_0^{1/\lambda} = 1 - \frac{1}{e} = 0.6321\cdots$$

- 乱数を N 個発生させて、値が 1/λ である確率を
  計算すればよい。
  - 乱数（確率的）であるので、Nが小さい時はズレている
  - Nを大きくすれば、期待値（0.6321）に近づくはず

プログラム例
expran_int.f

N を順次大きくし、
期待値に近づくか
確かめよ
（但し百万以下）

```
      program expran_int
      implicit none
c const:
      integer   N
      parameter(N=20)
      real      lambda
      parameter(lambda=3.1416)
c loop:
      integer   i
c function:
      real      expran
      integer   nok  ← 生成した乱数が 1/λ 以下の数
c begin:
      nok = 0  ← 0 に初期化
      do i=1,N   乱数 expran(lamnda) が 1.0/lambda 以下の時
        if (expran(lambda).lt.1.0/lambda) then
          nok = nok + 1  ← nok を1増やす
        endif
      end do
      write(*,'(f8.4)') real(nok)/real(N)
      stop
      end
```

N 個発生した乱数の内、
1/λ 以下であった割合　= nok/N

結果の例

- N=　　20　　P=0.5500
- N=　　200　　P=0.6150
- N=　　2000　　P=0.6395
- N=　　20000　　P=0.6333
- N=　　200000　　P=0.6330
- N=2000000　　P=0.6322

期待値 0.6321 に近づいている

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

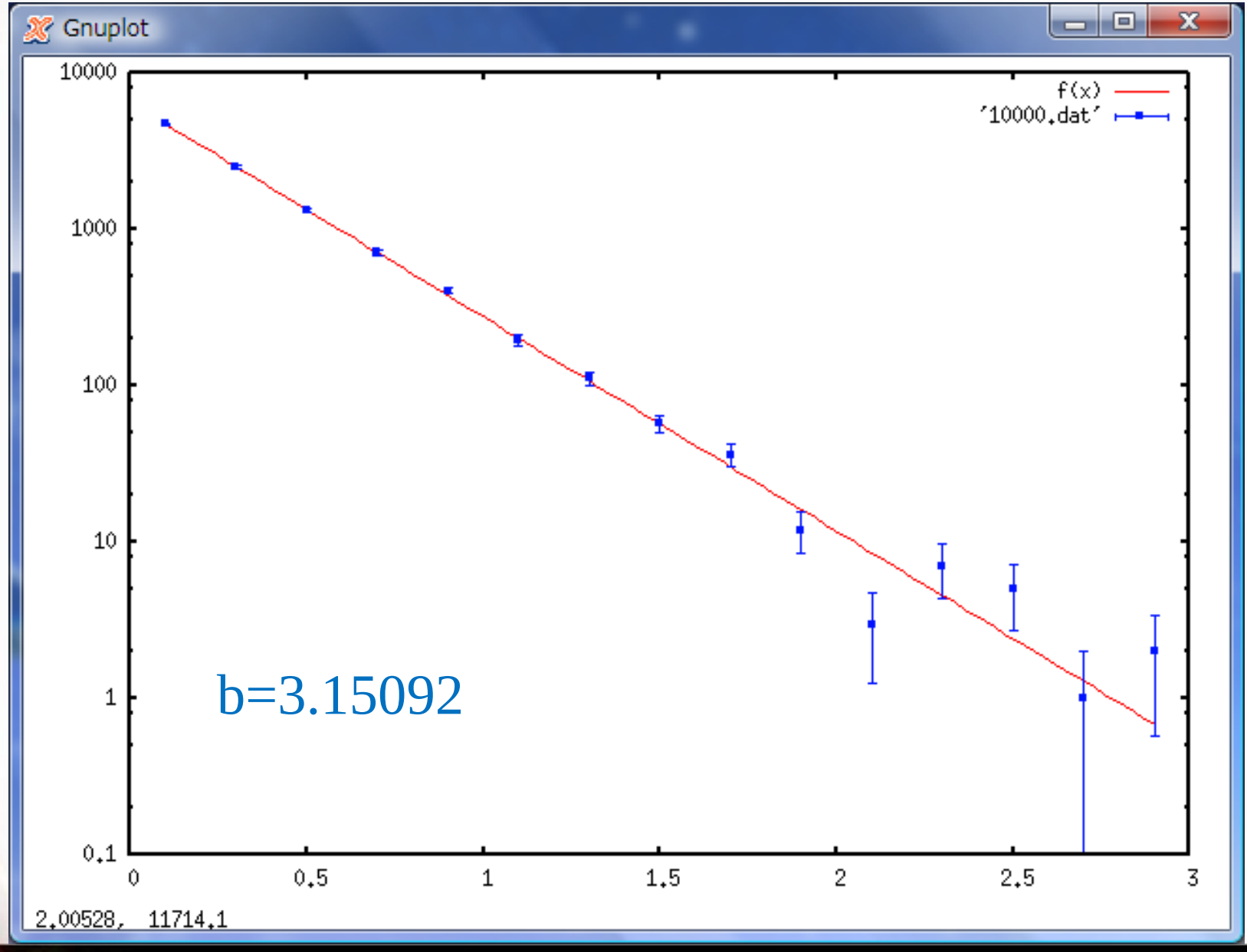
演習

- **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

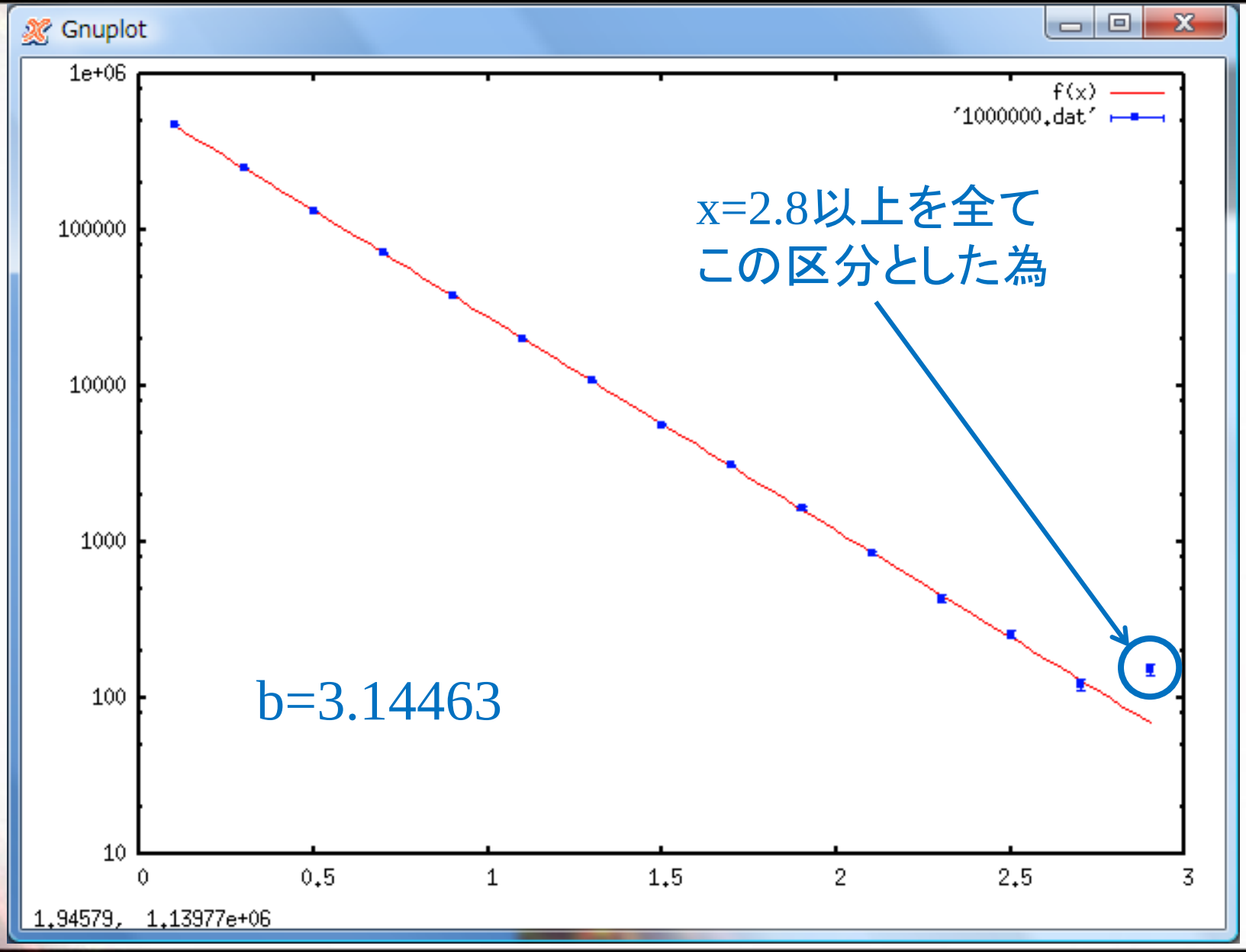
$$f(x) = a * \exp(-b * x)$$

    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