PAW Which Even Monkeys Can Use

H.Kitamura & His Company

平成9年 8月 19日

preface

このPAW Which Even Monkies Can Useは、ＰＡＷを使ったことがない人のために作りました。直訳すると「猿でも使えるＰＡＷ」ということで、某マンガ家の作品のタイトルをぱくってまして、そのまんまやん！という声もちらほらとあります。まあそんなことはどうでもよくて、とりあえず「そのまま打ったら絵が書けた」というマニュアル作りを目指してみました。

使い方は、ここに書いてある通りにやってしまえばよくって、**bold**で書かれてるところはそのまま、*itaric*のところは適当に変えてもいいというふうにしておきました。ちなみにvectorとかntupleなどとばか丁寧に書いてますが、わかりやすくするためで、実は、こいつらはvとかntというように省略することが可能です。(当然省略できないものもあります)

と、えらそうに書いてますが、僕も使い始めたばっかりで、そんなにわかってないので変なところもあると思いますので、そういう時は、ほんまもののマニュアルをみてください。

最後にＰＡＷを終わるときに、quitとするのを忘れないようにしましょう。

北村 尚

苦情・文句などがありましたら、e-mailで

`kitamura@phys01.phys.kobe-u.ac.jp`

まで、御一報下さい。直接文句をいいに来て頂いても結構です。それから、神戸大の小浜氏がhomepageを作ってくれました。

`http://phys01.phys.kobe-u.ac.jp:2080/~kohama/sarupaw.html`

です。たまに(１年に数回くらい)updateしてますので、適当に持って行って下さい。

目 次

Chapter 1

Welcome to PAW

それではまずＰＡＷに入ってみましょう。

```
phys01{kitamura}2> paw
```

と打つと、次のが出てきて、

```
****************************************************
*                                                  *
*           W E L C O M E    to   P A W            *
*                                                  *
*           Version 1.10/00  8 November 1990       *
*                                                  *
****************************************************
Workstation type (?=HELP) <CR>=1 :
```

そのあとに、return key を押すと、

```
PAW>
```

となって、ＰＡＷに入れました。ＰＡＷに入れたら、さっそくＰＡＷから出てみましょう。

```
PAW>
```

が出ている時、

```
PAW> quit
```

と打つと、ＰＡＷから出ることができます。もう一度ＰＡＷに入りたいときは、

```
phys01{kitamura}3> paw
```

とすると、入ることができます。

(注意)

ＰＡＷを quit せずにウインドウを閉じると画面が、

```
PAW> PAW> PAW> PAW> PAW> PAW> PAW> PAW> PAW>
```

となってしまうので、気をつけましょう。

Chapter 2

ntuple and hist

2.1 ヒストグラムの書き方

2.1.1 １次元のヒストグラム

まず ntuple と hist の説明をします。これらはヒストグラムをつくりたい時に使います。とりあえず

1.376
0.83
. . .

という様なデータを用意して下さい。では、ヒストグラムを描いてみましょう。

```
PAW> ntuple/create 100 ’title’ 1 ’ ’ 1000 name
```

ntuple/create	ntuple をつくります。
100	つくった ntuple のＩＤナンバー (どんな数字でもいい)
’title’	ntuple の名前
1	データの種類の数
’ ’	気にしないでつけておきましょう。
1000	データの数
name	データの名前 (ヒストグラムにでます)

```
PAW> ntuple/read 100 filename.dat
```

さっき作った ntuple(ＩＤナンバーは 100) にデータ (filename.dat) を読み込ませます。

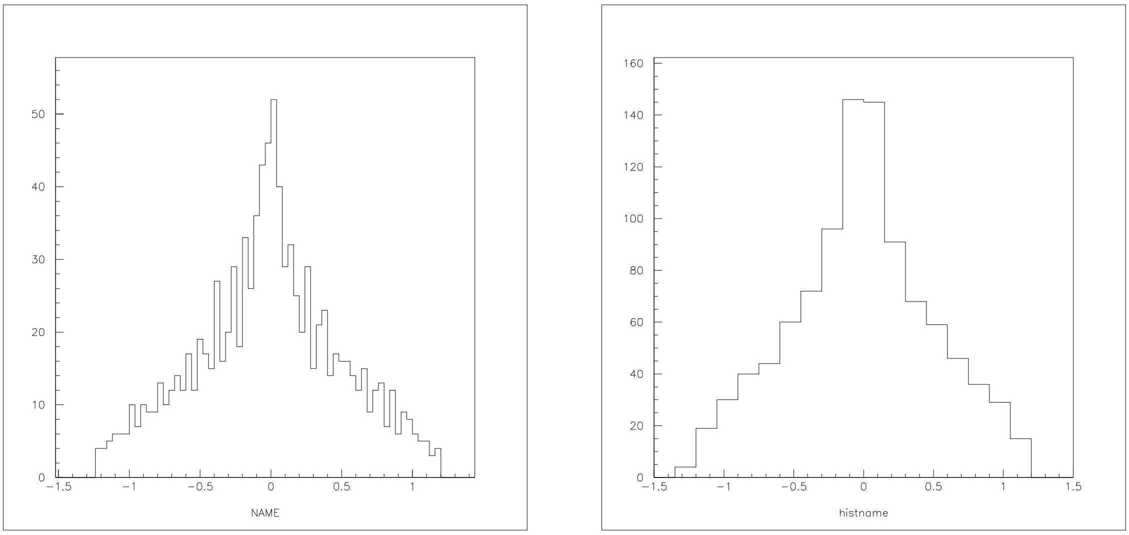
```
PAW> ntuple/plot 100.name
```

ヒストグラムが書けます。(図 2.1(a))

次に hist を使って、領域や bin 幅などを指定のできるヒストグラムを書きます。今までに書いたヒストグラムでデータの範囲を見ておいてください。

```
PAW> hist/create/1dhist 110 ’histname’ 20 -1.5 1.5
```

hist/create/1dhist	１次元の hist を作ります。
110	つくった hist のＩＤナンバー (さっきの ntuple のＩＤナンバー 100 と違う数字です)
’histname’	データの名前 (ヒストグラムの下に表示されます)
20 -1.5 1.5	20 はヒストグラムの階級の数 -1.5 1.5 は、ヒストグラムを書きたい範囲



(a) (b)

Figure 2.1: (a)ntuple で書いたヒストグラム,(b)hist で書いたヒストグラム

```
PAW> ntuple/project 110 100
```

100 は ntuple のＩＤナンバーで、110 は hist のＩＤナンバーです。hist にさっきの ntuple のデータを入れると思ってください。

```
PAW> hist/plot 110
```

とすると、ヒストグラムがかけます。(図 2.1(b))

また、いろいろなヒストグラムがかけて

```
PAW> hist/plot 110 e
```

e をつけるとエラーバーが書けます。(図 2.2(a))

```
PAW> hist/plot 110 b
```

とすると、細めのヒストグラム (棒状) が書けます。(図 2.2(b))

```
PAW> hist/plot 110 l
```

ヒストグラムが折れ線グラフになります。(図 2.2(c))

```
PAW> hist/plot 110 c
```

なめらかな線のグラフになります。(図 2.2(d))

2.1.2 ヒストグラムの fitting

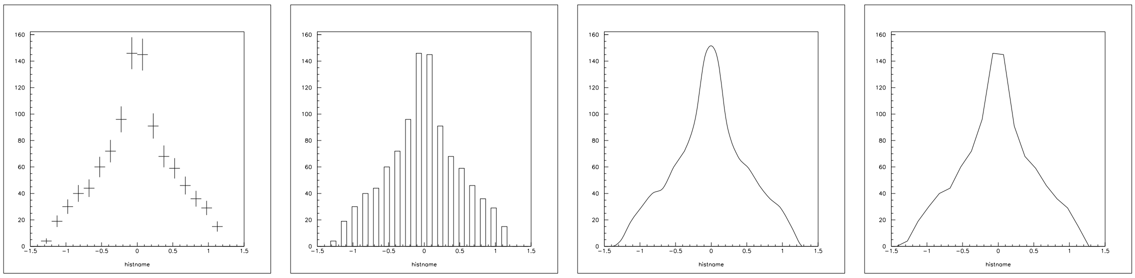
説明します。Histogram ID が *110* というのがあるとして、

```
PAW> hist/fit 110 g
```

とすると、Gaussian で fitting します。最後の g というのが Gaussian で fitting するというオプションです (図 2.2)。

```
PAW> hist/fit 110 e
```

```
PAW> hist/fit 110 p
```



(a) (b) (c) (d)

Figure 2.2: hist で書いたさまざまなヒストグラム. (a) エラーバー, (b) 細めのヒストグラム, (c) 折れ線グラフ, (d) なめらかな線

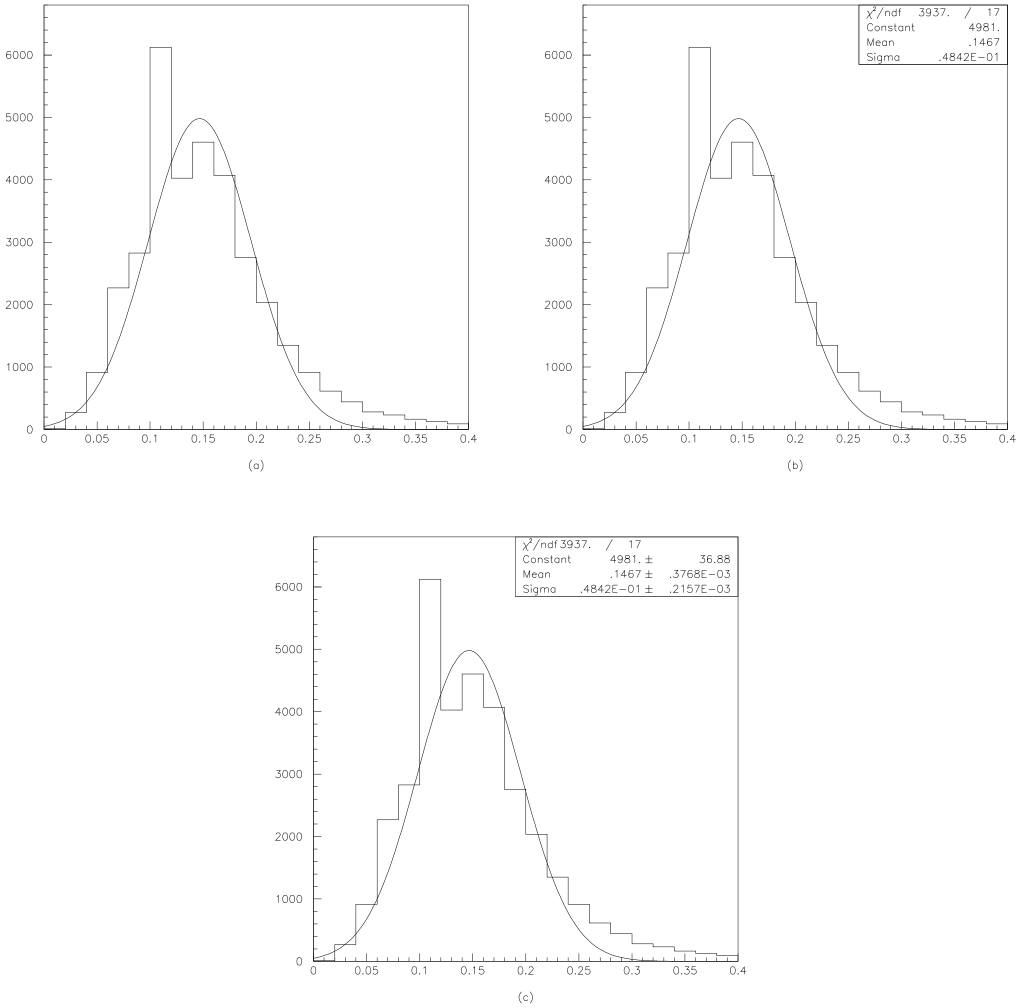


Figure 2.3: ヒストグラムでの fitting

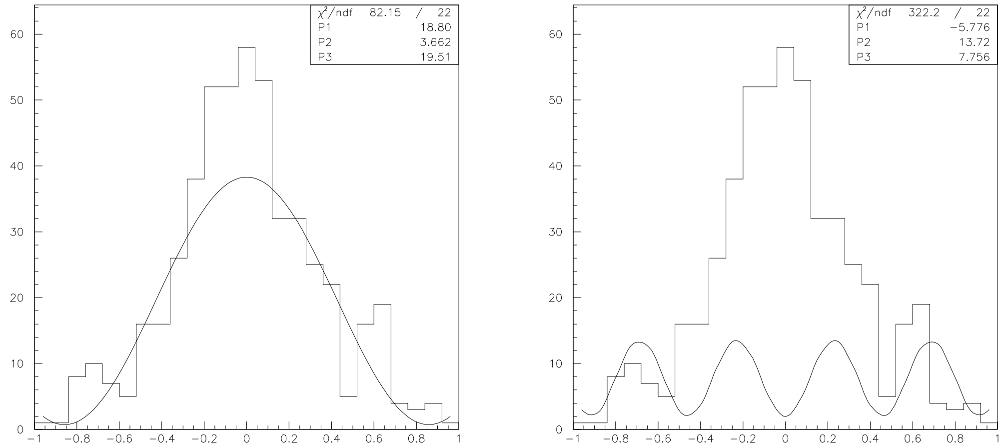


Figure 2.4: cos の fitting

```
PAW> hist/fit 110 p2
```

e は exponential fitting で、p とか p2 というのはそれぞれ１次 ・2 次のべき関数で、pn (n は任意) とすると n 次になります。

fitting した時に、統計の情報を図の上に出力したい時には

```
PAW> option fit
```

```
PAW> hist/fit 110 g
```

とすると、右上にχ^2、Constant、Mean、Sigma が表示されます (図 2.3(b))。さらにそれぞれの誤差もつけたい場合には

```
PAW> set fit 111
```

と打っておきます (図 2.3(c))。

任意の関数での fitting

ここでは、任意の関数の fitting をやってみます。例として、

$$p_1 \cos(p_2 x) + p_3 \tag{2.1}$$

という関数でやってみます。まず、以下のような FORTRAN のファイル cosfit.f を作ります。

```
function cosfit(x)
common /pawpar/ par(3)
cosfit=par(1)*cos(par(2)*x)+par(3)
end
```

これは FPRTRAN ですので行の頭を 6 文字くらいあける必要があります。で、あらかじめ ID = 100 という hist を作っておくとして、

```
PAW> vector/create par(3) R 10 1 10
```

ここで、vector/create については次の vector の章を見て下さい。この par に fitting の結果がはいります。次の、R 10 1 10 は初期値です。それで、

```
PAW> hist/fit 100 cosfit.f ! 3 par
```

で、fitting します。100 はヒストグラムの ID、cosfit.f はファイル名、! はオプションは適当にというやつで、3 はパラメータの数、par は上で定義した vector です。結果が図 2.4の左図です。右の図は初期値を

```
PAW> vector/create par(3) R 10 10 10
```

としたもので、変な初期値をほり込むとむちゃくちゃになります。

2.1.3 2 次元のヒストグラム

それでは、2 つのデータを使って、2 次元のヒストグラムを書きます。データは、次のようなものです。

```
1.376 -0.26
0.83 1.192
...
```

2 次元の場合でも ntuple を使うのは同じで

```
PAW> ntuple/create 120 ’title’ 2 ’ ’ 1000 first second
```

ここで、1 次元の場合と違うのは *2* と *first second* というところだけで、*2* はデータの数、*first second* は 1 番目と 2 番目のデータの名前になっています。2 次元の hist は

```
PAW> 2d 130 ’title’ 20 -1.5 1.5 20 -1.5 1.5
```

と、します。この場合、1 次元のときと違うのは *20 -1.5 1.5 20 -1.5 1.5* とするところで、はじめの *20 -1.5 1.5* は x 軸の設定で、あとのほうは y 軸の設定です。

```
PAW> ntuple/read 120 filename.dat
```

```
PAW> ntuple/project 130 120.second%first
```

こうすると hist の時と同様に、ntuple のデータが 2d に入ります。但し first はｘ軸に、second はｙ軸に入ります。

```
PAW> hist/plot 130
```

これで 2 次元のヒストグラムが書けました。(図 2.5(A))

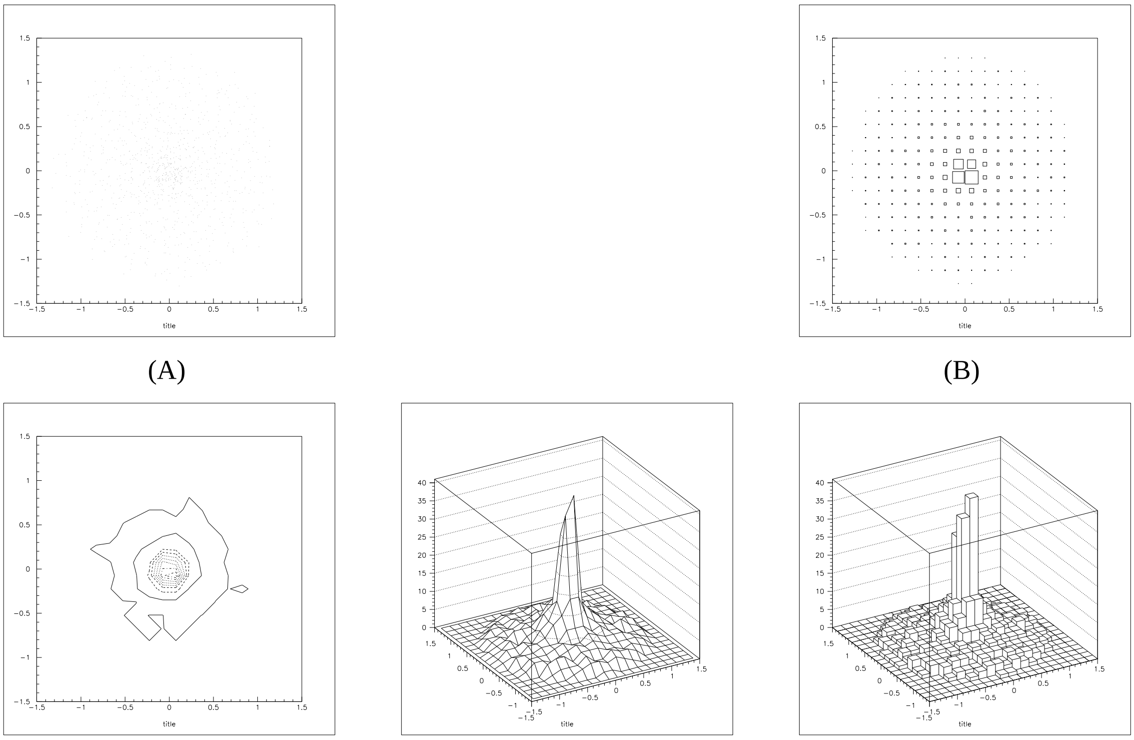
```
PAW> hist/plot 130 box
```

```
PAW> hist/plot 130 contour
```

```
PAW> hist/plot 130 surf
```

```
PAW> hist/plot 130 lego
```

などと打つと、いろいろな図が書けます。(図 2.5(B)~(E))



(A) (B)

(C) (D) (E)

Figure 2.5: 2次元のヒストグラム. (A) ドット (何もない時), (B) ボックス (box), (C) 等高線 (contour), (D)surface(surf), (E) レゴ (lego)

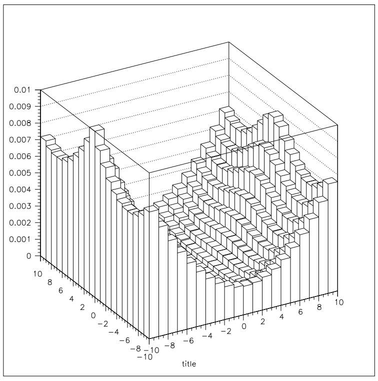


Figure 2.6: ３次元の plot

2.2 ntuple を使った scatter plot

2.2.1 ２次元 scatter plot

2.2.2 ３次元 scatter plot

３次元の plot の書き方は data がｘ、ｙ、ｚの順に並んでるとすると ntuple のフォーマットで

```
PAW> ntuple/create 100 'title' 3 ' ' 400 x y z
```

```
PAW> ntuple/read 100 filename.dat
```

```
PAW> ntuple/plot 100.z%y%x
```

で、３次元座標上で点をうちます。さらに

```
PAW> 2dhist 200 'title' 20 -10 10 20 -10 10
```

```
PAW> ntuple/plot 100.y%x z ! -200 ! lego
```

とすると、lego plot が書けます。(図 2.6)

Chapter 3

vector

3.1 1D vector

ここでは値の推移を見たい時に使う、vectorの説明をします。ntupleの時と同じ様なデータファイルを用意して下さい。それでは始めます。

3.1.1 plot

まずは１次元のplotから

```
PAW> vector/create first(100)
```

```
PAW> vector/create second(100)
```

firstとsecondというvectorを作ります。first(100)の100は、読み込むデータの数です。

```
PAW> vector/read first,second filename.dat
```

さっき作ったfirstとsecondというvectorにfilename.datのデータを読み込ませます。このとき１番目のデータはfirst、２番目のデータはsecondに入ります。

```
PAW> vector/draw first
```

この様に打つとfirstのデータがヒストグラムの様に書けます(図3.1(A))。同様に

```
PAW> vector/draw second
```

とすると、secondのデータが書けます。(図3.1(B))

```
PAW> vector/draw first ! b
```

とすると、ヒストグラムが細くなります(図3.1(C))。

```
PAW> vector/draw first ! l
```

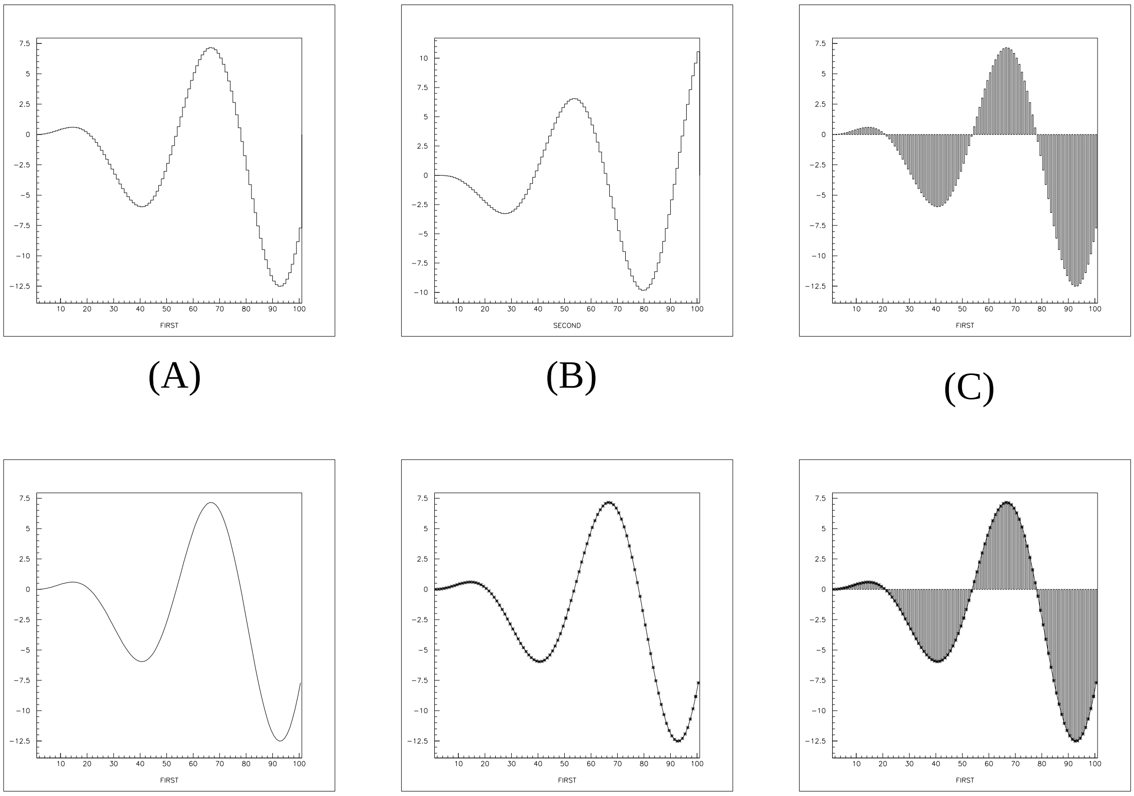
とすると、折れ線になります(図3.1(D))。

```
PAW> vector/draw first ! l*
```

とすると、折れ線に”＊”のplotがつきます。(図3.1(E))

```
PAW> vector/draw first ! bl*
```

こうすると細いヒストグラムと折れ線とplotがつきます。(図3.1(f))



(A) (B) (C)

(D) (E) (F)

Figure 3.1: vector/draw で書いた図。(A),(B):vector/drawn のみ、(C):細いヒストグラム、(D):折れ線、(E):折れ線と" ＊ "、(F):細いヒストグラムと折れ線と" ＊ "

3.1.2 scatter plot

次に2種類のデータを使って、2次元のplotを作ります。

```
PAW> graph 100 first second l
```

firstとsecondの位置の変化をを平面上に折れ線でつなぎます。100はデータの数です。

```
PAW> igset mtyp 29
```

```
PAW> graph 100 first second p
```

で、データがplotされます。igset mtypというのは後の「せこいわざ」のところで説明しますが、plotする点を色々と変えるという命令で、別になくてもかまいません。こうすると、勝手にさっきの折れ線の上に点をplotするので、plotだけにしたい時は

```
PAW> graph 100 first second apw
```

とします。点を折れ線でつなぐのは、

```
PAW> graph 100 first second alpw
```

とします。

3.1.3 エラーバーつきのscatter plot

エラーバーをつけるには、x方向、y方向のエラーの大きさのvectorとしてfirst_eとsecond_eを用意して、

```
PAW> gr/hp/errors first second first_e second_e 100
```

です。最後の100はgraphコマンドと一緒でデータの数です。

3.1.4 scatter plotのfitting

vectorのfittingは、hist/fitの場合と同じで、Gaussian fittingだと

```
PAW> vector/fit first second second_e g
```

second_eはy方向のvectorであるsecondのエラーを表すvectorです。

3.2 vectorとhist

vectorでもヒストグラムを書くことができて、

```
PAW> vector/plot first
```

そしてvectorからhistも作れて、

```
PAW> hist/create/1dhist 150 'first' 10 -12.5 7.5
```

```
PAW> vector/hfill first 150
```

とすれば

```
PAW> hist/plot 150
```

で、かけます。その他にhistとvectorを変換するコマンドは

- histの内容をvectorに代入する。
  ```
  PAW> hist/get_vect/contents 150 first
  ```

- histのエラーをvectorに代入する。
  ```
  PAW> hist/get_vect/error 150 first
  ```

- `hist` の各 `bin` の中央の値を `vector` に代入する。
  ```
  PAW> hist/get_vect/abscissa 150 first
  ```

- `vector` の内容を `hist` に代入する。
  ```
  PAW> hist/put_vect/contents 150 first
  ```

- `vector` の内容を `hist` のエラーに代入する。
  ```
  PAW> hist/put_vect/error 150 first
  ```

などもあります。

ひとこと

ここでは vector/read で、data 数が二つのものを例としてあげましたが、例えば data 数が 5 個の file では vector/create で v1,v2,v3,v4,v5 と作って

```
PAW> vector/read v1,v2,v3,v4,v5 filename.dat
```

などと、1 度に読み込むことも可能です。

3.3 2D vector

今までは 1 次元の vector のことでしたが、例えば 20 × 20 のデータがあって、それを 3 次元的に描きたいときは、2 次元の vector を使います。データファイルは、

X(x=1,y=1)
X(x=2,y=1)
⋮
X(x=20,y=1)
X(x=1,y=2)
⋮
X(x=20,y=20)

というように 1 列に並べてください。

2D vector の create は、

```
PAW> vector/create vect_name(20,20)
```

と、します。そして

```
PAW> vector/read vect_name data_file_name
```

で、データを data_file_name から読み込ませます。それから 2D hist を作って、

```
PAW> 2d 130 'title' 20 -1.5 1.5 20 -1.5 1.5
```

ここに vector の値を代入します。

```
PAW> hist/put_vect/contents 130 vect_name
```

こうすると、あとは

```
PAW> hist/plot 130
```

と、2次元ヒストグラムと同じになります。

3.4 SIGMA

vector は、SIGMA を使うと簡単に計算ができます。例えば、vec1 という vector があるとして、この sin を計算したい場合

```
PAW> sigma vec2=sin(vec1)
```

とすると、vec2 には vec1 の sin が格納されます。で、当然ながら四則演算 (+ - * /)、expornential (**) も可能で

```
PAW> sigma vec3=vec2+3.0*sin(vec1)
```

なんてことも出来ます。

その他、便利な function があります。
たとえば、

- vec1 = (1, 2, 3, 4, 5, 6, 7, 8, 9,10) というような数列を作るには

  ```
  PAW> sigma vec1=array(10,1#10)
  ```

- vector の difference をとるのは (最終項は最後の3成分の quadratic extrapolatoin)

  ```
  PAW> sigma vec2=diff(vec1)
  ```

  vec2=(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

- vector v1 の最大値を vector vec2 にいれる。

  ```
  PAW> sigma vec2=maxv(vec1)
  ```

  vec2=(10,10,10,10,10,10,10,10,10,10)

- 同じく最小値の場合

  ```
  PAW> sigma vec2=minv(vec1)
  ```

  vec2=(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

- 積算していく $(vec2(n) = \sum_{i=1}^{n} vec1(i))$

  ```
  PAW> sigma vec2=sumv(vec1)
  ```

  vec2=(1, 3, 6,10,15,21,28,36,45,55)

なんかがあります。

Chapter 4

function

ＰＡＷでは、色々な関数(１次元・２次元)をplotすることができます。関数はＦＯＲＴＲＡＮの関数で入力します。まずは１変数の関数から書きます。

```
PAW> fun/plot x*sin(x)*exp(-0.1*x) -10 10
```

関数

$$f(x) = x \sin x e^{-0.1x} \tag{4.1}$$

のplotを-10から10までの範囲で描きます(図4.1)。

次に、２変数関数を描きます(図4.2)。

```
PAW> fun2 200 abs(sin(x)/x)*(cos(y)*y) 40 -6 6 40 -6 6
```

関数

$$f(x, y) = \left| \frac{\sin x}{x} \right| y \cos y \tag{4.2}$$

のplotをhistの時と同様にｘ軸方向に-6から6まで40個ずつ、ｙ軸方向に-6から6まで40個ずつ描きます。(abs()というのはＦＯＲＴＲＡＮで絶対値のことです。)

```
PAW> hist/plot 200
PAW> hist/plot 200 contour
PAW> hist/plot 200 surf
PAW> hist/plot 200 lego
PAW> hist/plot 200 box
```

などはhistの時と同様に書けます。

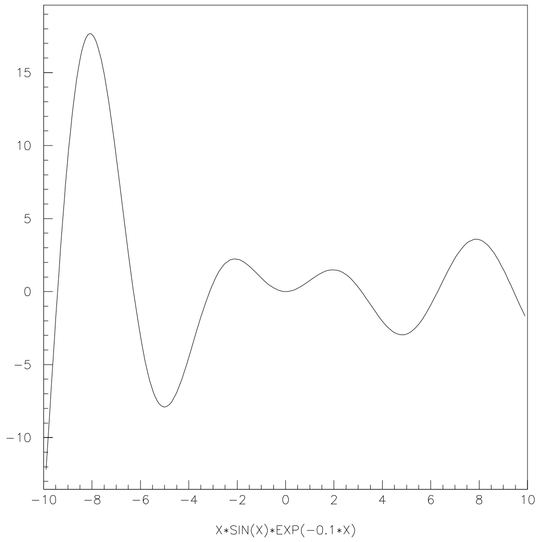


Figure 4.1: 1変数関数のプロット。$f(x) = x \sin x \, e^{-0.1x}$

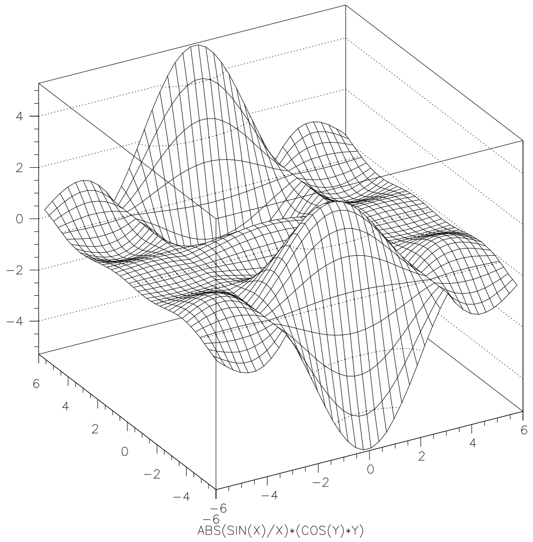


Figure 4.2: 2変数関数のプロット。$f(x,y) = \left|\frac{\sin x}{x}\right| y \cos y$

Chapter 5

graphics

ここでは、グラフなどの座標上に色々な図形を書く方法を紹介します。まずは座標上の２点(1,2)、(3,4)を直線で結ぶには

```
PAW> line 1 3 2 4
```

と、打って下さい。順番が少しおかしいようにみえますが、line の後はｘの始点と終点でその次にｙの始点と終点を書きます。これと同様にして

```
PAW> box 1 3 2 4
```

とすると、左下が(1,2)、右上が(3,4)の長方形を書きます。直線を矢印にするには

```
PAW> arrow 1 3 2 4
```

とすると、(1,2)から(3,4)の方に矢印(⟶)を書きます。⟷というふうな向かい合った矢印が書きたいときは、

```
PAW> arrow 1 3 2 4 -0.4
```

と、-0.4をつけます。‘ - ’ は向かい合う矢印を指定して、0.4は矢印の大きさを指定します。(何もつけないときは、0.4になっています。)

座標上に円を書きたいときは(ここでは中心が(4,5)で、半径が2)

```
PAW> arc 4 5 2
```

とします。ちなみに座標のｘ軸とｙ軸のスケールが違うと、楕円しか書けません。

座標上に(例えば(2,3)に)点を打つ場合は

```
PAW> key 2 3
```

点をマークにしたい場合は

```
PAW> key 2 3 20
```

最後の20というのは、マーカーの種類です。その点の横に文字をいれることもできて

```
PAW> key 2 3 20 'Hello World!'
```

Chapter 6

せこいわざ

基本的な図の書き方ができたら、次はその図に手を加えてみましょう。

6.1 画面分割

一画面上に複数個の図を書きたい時は、

```
PAW> zone 2 2
```

画面上にｘ軸方向に２個 (最初の２)、ｙ方向に２個 (２番目の２) のグラフを書きます。(全部で４個)
途中でグラフの数を変えたいときは、

```
PAW> zone 1 2 2 s
```

ｘ軸方向に１個、ｙ方向に２個という書式に変えて、次の２は変えた書式の２番目から書くということで、s というのは今まで書いていた図の上に書くということです。

6.2 座標軸

ＰＡＷでは、かってに座標軸を書いてくれますが、思った通りの座標軸を書くには

```
PAW> null 10 100 10 30
```

null はグラフの座標を書けという命令で、10 100 はｘ軸の範囲、10 30 はｙ軸の範囲です。こうすると座標軸だけを書いて、

```
PAW> hist/plot 110 S
```

などとして、重ね書きします。

```
PAW> option logx
```

```
PAW> option logy
```

とやると、次に描く図からｘ軸・ｙ軸がそれぞれ log 目盛りになります。もとに戻すには

```
PAW> option linx
```

```
PAW> option liny
```

とすればいいです。また次のをすると

```
PAW> option grid
```

座標軸上に grid を引きます。

6.3 title

ＰＡＷで書いた図の上に、どーんとしたタイトル (global title) を付けるには、

```
PAW> title_global ‘global title’
```

しかし、このままではせこいタイトルしか書けないので、

```
PAW> set gfon -70
PAW> set gsiz 1.2
PAW> set ygti 0.5
PAW> title_global ‘global title’
```

と打つといいです。set gfon -70 というのは、global title の書体を -70 にして、set gsiz 1.2 は文字の大きさを 1.2 cm にして、set ygti 0.5 は global title の位置を上から 0.5cm あけるということです。

グラフなどを書いた時に、かってにタイトルが付いてきたと思いますが、このタイトルも

```
PAW> set tsiz 0.5
PAW> set tfon -12
PAW> set yhti 1.0
```

とすると、書体、大きさ、タイトルのｙ軸からの距離 (cm) が指定できます。

ｘ軸、ｙ軸のタイトル (axis title) をつけるには

```
PAW> atitle ‘x axis title’ ‘y axis title’
```

と打ちます。global title の時と同様に

```
PAW> set asiz 0.4
PAW> set lfon -60
PAW> set ylab 0.5
PAW> set xlab 1.0
PAW> atitle ‘x axis title’ ‘y axis title’
```

などと打つと、文字の書体や大きさが指定できます。set ylab 0.5 はｘ軸のタイトルのｙ軸からの距離、set xlab 1.0 はｙ軸のタイトルのｘ軸からの距離になることに注意して下さい。

グラフの中に文字 (text) を書きたいときは

```
PAW> itx 10.0 20.2 ‘text’
```

あるいは

```
PAW> text 10.0 20.2 ‘text’
```

こうすると、グラフ上の点 (10.0 , 20.2) を先頭として text と表示します。書体や大きさを変えるのは、今までのとちょっと違っていてよく分からないので、マニュアルを読んで下さい。

6.4 ヒストグラムの修飾

ヒストグラムを描く時に、色々とできて、

```
PAW> set htyp 244
```

```
PAW> hist/plot 110
```

とすると、ヒストグラムに模様がつきます。これを用いると

```
PAW> set htyp -3
PAW> hist/plot 110
PAW> set htyp 244
PAW> hist/plot 120 s
```

とやれば、見やすくヒストグラムをかさねられます。

ヒストグラムを3番目のbinから8番目のbinまでだけを表示したい場合には

```
PAW> hist/plot 110(3:8)
```

とします。同様に始めから4番目のbin、4番目のbinから終わりまでの時は

```
PAW> hist/plot 110(:4)
PAW> hist/plot 110(4:)
```

とすればいいです。

```
PAW> hist/plot 110
PAW> set htyp -3
PAW> hist/plot 110(2:5)
```

などというのもできます。

ヒストグラムの太さを変えたいときは

```
PAW> opt bar
PAW> igset baro 0.2
PAW> igset barw 0.6
```

とします。opt bar はヒストグラムの太さを変えるということで、igset baro 0.2 はヒストグラムの間隔で（0から1）、igset barw 0.6 はヒストグラムの太さ（0から1）です。(もともとのヒストグラムの太さを1としています。)

```
PAW> opt bar
PAW> igset baro 0.1
PAW> igset barw 0.3
PAW> hist/plot 110
PAW> igset baro 0.6
PAW> set htyp 444
PAW> hist/plot 120 s
```

こんなのもできます。

6.5 折れ線グラフの修飾

vector や hist で折れ線グラフを書くときの説明をします。

```
PAW> set dmod 2
PAW> vector/draw first ! l
```

こうすると、線のタイプが変えられます。

```
PAW> igset mtyp 29
```

```
PAW> vector/draw first ! lp
```

igset mtyp 29 というのは、plot する点のタイプを変える命令です。こうすると折れ線グラフに plot が入ります。

vector でも書く点の指定ができて、

```
PAW> vector/draw first(12:19)
```

と、hist の時と同様に出来ます。

6.6 igset,option,set

ここまでに色々な igset,option,set を使いましたが、他にも色々な物があります。それでごちゃごちゃといじると分からなくなるので、

```
PAW> igset
```

```
PAW> opt
```

```
PAW> set
```

と打つと、それぞれの初期設定、現在の設定、そして説明文がつきます (英語で)。それぞれを初期設定に戻したい場合は

```
PAW> set *
```

とします。

6.7 help

PAW には online help 機能がついています。例えば

```
PAW> hist/plot
```

とかすると、help file がでてきます。ほとんどの環境では vi がたちあがると思います。ちなみに、vi を close するときは:q です。

6.8 カラー

contour map とかを書く時に、はでな色使い (?) をする方法です。ここでは 64 色のグラデーションの例を書いておきます。

```
PAW> set ncol 64
```

```
PAW> palette 1
```

```
PAW> contour 200 64 3
```

Chapter 7

printout and macrofile

7.1 print out の仕方

ＰＡＷで書いた絵を print out するには

```
PAW> fort/file 3 filename.ps
PAW> graph/meta 3 -111
       ...
     絵を書く
PAW> hist/plot 120
     など
       ...
PAW> fort/close 3
```

こうすると、filename.ps というファイルができるので、これを print out します。このとき気をつけてほしいのは、fort/file ... と打ってから絵を書くということで、手順としてはまず vector/create... などというように vector や ntuple ,hist などを作ってから、fort/file... として、vector/draw... というふうに図を書いて行くといいと思います。

EPS ファイルの作り方

TeX などにファイルを取り込むときに使う EPS(Encapsulated PostScript) ファイルを作る時には

```
PAW> graph/meta 3 -113
```

とします。

7.2 macro file の作成

ＰＡＷで絵を書くときにいちいち打つのがめんどくさい時は、macro file を作ると便利です。例えば macroname.kumac というファイルを作ります。(.kumac は必ずつけておいて下さい）

```
macro macroname [1] [2]
```

```
  set gfon -70
  set gsiz 1.5
  set ygti 0.5
  set lfon -30
  set asiz 0.5
  set tfon -10
  set tsiz 1.0
  set yhti 1.0
  set htyp 244
  title_gloval 'macro file test'
  nt/cr 100 'test' 1 ' ' [1] data
  nt/re 100 [2]
  1d 110 'data' 50 0 100
  nt/pro 110 100.data
  h/pl 110
  atitle ' ' 'count'
return
```

というようにします。[1] [2] というのはそこに色々な文字をいれることが出来ます。ここでは [2] はデータの入っているファイル、[1] はデータの個数になっています。[1] [2] などは、必要がなければ使わなくてもいいです。

ＰＡＷで絵を書くには、

PAW> **exec** *macroname 1000 filename.dat*

というふうに、exec macrofile の名前 (.kumac はなくてもいい) [1] [2] の順番にいれます。

last.kumac

ＰＡＷを終了すると、last.kumac というファイルが実行したディレクトリに生成されます (もし last.kumac 存在していたら last.kumac は last.kumacold というファイルに変わって、新たに last.kumac が生成されます)。これは終了時までにＰＡＷで実行したコマンドの履歴ファイルで、これを使って macro file を書くと結構便利かと思います。