PAW Manual

Ver 0.9

山形大学理工学研究科修士課程 1 年
伊藤 祐輔

2001 年 2 月 27 日

目 次

1 はじめに **3**
1.1 立ち上げ方と各 Window の名称と役割 3
1.2 おまけ 4

2 データ解析 (基礎編) **5**
2.1 QVT、もしくは ADC2QVT を使用したデータの解析方法 5
2.1.1 実際の解析方法 5
2.1.2 データの保存方法 12
2.1.3 HBOOK の開き方 14
2.2 一般的な ascii データの解析方法 15
2.2.1 vector を使った解析 15
2.2.2 ntuple を使った解析 21
2.2.3 histogram を用いた解析 21

3 いろんな設定 **22**
3.1 option と set 22
3.2 macro file 24
3.3 .Xresources 25
3.4 その他の便利なもの 26

A Handmade HBOOK **27**

1　はじめに

PAW とは、**P***hysics* **A***nalysis* **W***arkstation* の頭文字を取ったものです。その名前のとおり、work station を用いたデータ解析ソフトで、主に実験や simulation により得られたデータを解析する為に用いるアプリケーションで便利な物だと世界中の PAW ユーザーの人々は思ってます (たぶん)。この PAW の簡単な使い方についての説明をしますが、とりあえずそんなに難しい事は書きません。って言うか書けません。より、詳しく知りたい人は研究室にある正規の manual を読んだりして勉強して下さい。最新情報やより詳しい情報は CERN[1] にあるはずなので興味のある人はそちらへどうぞ。

1.1　立ち上げ方と各 Window の名称と役割

PAW は、どこのディレクトリに居ても立ち上げる事が出来ます。立ち上げ方はまず kterm を開き、そこで prompt の出ている状態で

[**ito**]ksnucl:~/files/experiment/fiber/attenuation> paw ++ &

と打つと PAW が立ち上がります (但し上で二重下線を引いた場所は prompt で、立ち上げた場所は/home2/ito/files/experiment/fiber/attenuation の場合を例として挙げています)。この最後の&は必要という訳じゃないけど、あった方が良いので出来るだけ付けて下さい。ここで 3 つの Window が立ち上がります。各 Window の名前は

1. 左上: Executive Window
2. 左下: Main Browser
3. 右上: Graphics

ですが、とりあえずこの名前は覚えなくても良いです。多分これをしっかりと覚えている人はまず居ないと思いますので。。。。。この名前についていろいろと書く代わりに、これらの基本的な役割を書いておきます。

1. *Executive Window*:
   ここは直接 command を打ち込む為の Window です。さらにそれらを打ち込んだ log も残っているので、それを見る事も出来ます。
2. *Main Browser*:
   current directory(現在いるディレクトリ) にあるファイル (hbook,kumac... *etc*) を示している。これらはダブルクリック[2]をする事により使用する事が出来る。
3. *Graphics*:
   この Window は名前のとおり、graph 等を表示する為の Window です。表示する方法等は後半の方で紹介する。また表示中には、絵を右クリックする事で様々な処理をする事が出来るようになります。

[1] http://www.cern.ch

[2] current directory までの絶対パス名の中のいずれかのディレクトリ名が大文字を含む場合にはこの方法ではダメなので他の方法を用いなければいけない。これについては 2.1 節参照。

1.2 おまけ

(1) PAW は立ち上げたディレクトリの file は開きやすいが他のディレクトリの file を開くにはちょっと手間がかかる。。。と言う設定となっているので、なるべく使いたい file があるディレクトリで立ち上げた方が良いです。[3]但し、上でも書いたとおりどこでも立ち上げる事は出来ます。基本的にはそうした方が良いよ。。。って言うだけです。

(2) PAW 使用後には、立ち上げたディレクトリの中に **last.kumac**[**old**] と言う file が作られます。これは PAW 使用時の log(使ったコマンド記録用)file です。なので、もし大事な事をした後の **last.kumac** は名前を変えて保存して置くと後々助かる場合も多いと思います。実際に後で見直す場合に一番活用するのは、この *last.kumac* という file なんじゃないでしょうか。。。。
ただ、普段から頻繁に取っておいても何の得も無いし、さらに使う予定のない場合には単なる邪魔な file でしかないので普段はあまり気にしなくてもよいです。。。。

(3) 少し慣れてきたら、PAW 立ち上げ時に読み込まれるファイル$HOME/.pawlogon.kumac と言うファイルや*.kumac という macro file を活用すると便利です。[4]
ただし、この$HOME とは各自のホームディレクトリ (login した時に入るディレクトリの事) を表してます。

(4) この manual 中では二重下線 (<u>=</u>) を引いた場所は prompt、一重下線 (<u>-</u>) を引いた場所は省略可能範囲を表しています。さらに (<u>□</u>) は実際に打ち込んで練習して欲しいところを示しています。以上のような事も参考にして下さい。

(5) 最後に、PAW を終了させるには Executive Window で"exit" と打つか、同じく Executive Window の右上の File と言う場所をクリックしてさらにそこの"Exit" と言う場所もクリックすれば終了出来ます。どうしても終了出来ない場合には PAW が暴走している場合も考えられますので、top や ps と言う command を使ってそのプロセス番号を見付けて強制終了させてください。そのままにしておくと、他の人に迷惑がかかる場合がありますので。。。。

- ▷ top を使う場合
  - (i) top を立ち上げて u(user) と打った後に自分の login name を入れる
  - (ii) そこで PAW に関する process を見付け、その id を控える
  - (iii) kill -9 [process id]
- ▷ ps を使う場合
  - (i) ps aux | grep [user name] | grep paw
  - (ii) そこで暴走しているっぽい PAW に関する process を見付け、その id を控える
  - (iii) kill -9 [process id]

このどちらでもいいですが、どちらかと言えば ps を使ったものがお薦めです。理由は特になし。

[3]立ち上げ後に Current directory を移動する方法もあります
[4]第 3.2 節参照

2　データ解析 (基礎編)

2.1　QVT、もしくは ADC2QVT を使用したデータの解析方法

2.1.1　実際の解析方法

実験で得られたデータに対しても、いろいろな面で PAW は役に立ちます。はじめに QVT を用いてデータを取った場合、もしくは CAMAC でデータを取ってそのデータを ADC2QVT(千葉さん作成) を用いて変換したデータなどに利用できる解析方法です。これらのデータは 1 × 1024 の形式で保存してあるので、その事を念頭に入れておいて下さい。なので、この方法を用いる前にこれが実行出来るかどうかを調べる為に以下の事を確認して下さい。

(確認) データが何 channel 分あるのかを調べる。まずは prompt が出ている時に

[**ito**]ksnucl:~/files/experiment/fiber/attenuation> wc -l file-name

と入力する。それに対して"1024" と表示されれば OK です。この"wc" と言う command は、file の文字数などを見るための command であり、"-l" を付ける事で行数だけを調べる事が出来ます。データは 1 行に 1 channel 分書き出されるので、行数を調べる事により channel 数がどれだけ取れているのかと言う確認になります。もし 1024 行 (channel) で無い場合は中身を見てみるなり、誰かに相談するなりしましょう。

(解析方法)

(1) vector を作る
　　⇓
(2) vector に file を読み込ませる
　　⇓
(3) histogram を作る
　　⇓
(4) vector を histogram に落とす
　　⇓
(5) histogram を plot してみる
　　⇓　　⇑ 結果に不満
(6) histogram に対して fitting する
　　⇓ 結果に満足
(7) HBOOK FILE に保存して終了

以上の事を詳しく説明する。

(1) にある vector とは数学の行列、コンピューター言語の配列に相当するものです。つまり、データ群を一つにまとめて格納しておく為の箱のようなものです。例えば以下のようなデータがあったとします。このデータは見て分かるとおり $6 \times 3 = 18$ のデータの集

0	1	2	3	4	5
0.92	4.15	9.78	14.46	17.26	21.90
0.5	1.0	0.75	1.25	1.0	1.5

まりです。この場合にこのデータを格納する為にちょうどいい大きさの vector というのは 6×3 の形のものであり、それは以下のようにして作る事が出来ます。

$$\underline{vector}/\underline{create}\ name(6, 3)$$

この名前は任意のもので良いです (ただし他の vector の名前とかぶらなければ . . . です。もし同じ名前を付けてしまうと最初に作って置いた方は、後から作った方に上書きされてしまうので注意しましょう)。
では、これを踏まえて実際のデータを入れる場合の vector を作ってみましょう。データの中身を見てみれば分かるとおり、数値が一列に並んでいます。その数は 1024。よって上の場合と同様にして

$$\underline{vector}/\underline{create}\ rahha(1, 1024)$$

が正解です。ところが、行列の列数がこの場合は 1 個しかありませんのでこの 1 は省略する事が出来ます。よって

$$\boxed{\underline{\text{vector}}/\underline{\text{create}}\ \text{rahha}(1024)}$$

と書いても大丈夫ですし、さらにこっちの方が楽チンです。って言う訳で使う時はこちらの方がお奨めです。

これで (1) が出来たので次に進みます。
次にこの作った vector に実際のデータを取り込みます (step(2))。この command は次のようになります。

$$\boxed{\underline{\text{vector}}/\underline{\text{read}}\ \text{rahha ohha.dat}}$$

ohha.dat は file 名ですので間違わないように。
次に (3),(4) について。
上の方で作った vector を histogram に変換するのですが、まずは hisotgram を作ります。

$$\underline{histogram}/\underline{create}/1d\ \ 'hid'\ \ 'name'\ \ 'total\ ch'\ \ 'start\ ch'\ \ 'last\ ch'\ \ 'maximum\ bin\ contents'$$

各単語とその意味、さらにその具体的な値は以下を参照して下さい。

- ▷ *hid* : histogram に付ける好きな id number
- ▷ *name* : histogram に付ける好きな名前
- ▷ *total ch* : 全チャンネル数 (1024 を使用)
- ▷ *start ch* : 最初のチャンネル (0. を使用)
- ▷ *last ch* : 最後のチャンネル (1024. を使用)
- ▷ *maximum bin contents* : (0. を使用)

これらの具体的な値を入れたものは

histogram/create/1d '55' 'adc spectrum' '1024' '0.' '1024.' '0.'

となる。この中で histogram の名前のところ以外は、 ' (single quotation) でくくらなくても良い。またこれを覚えるのは面倒くさいのでとりあえずは最初の command のところだけ打ってしまえば、あとは PAW からの手助けが出ます。それは新しい Window が出てそれに従って値を代入していくものですが、慣れるまではそちらに頼った方がいいかも知れません。

さらに次に進んで、この histogram に前に作った vector を代入します。この command は

$$histogram/put/content\ \ 'hid'\ \ 'vector\ name'$$

となります。ので、上の場合には

histogram/put/content '55' 'rahha'

となる。

よってこれで実際にデータを plot する準備は整ったので、とりあえずどのような histogram なのか見てみましょう。そのための command は

$$histogram/plot\ \ 'hid'(start\ ch : last\ ch)$$

- hid : histogram に付けた好きな id number
- $start\ ch$: plot する最初のチャンネル
- $last\ ch$: plot する最後のチャンネル

command 後半の start ch と last ch については、範囲設定したくない時には必要ないです。よって、これらを今まで使ってきた値で 0channel から 1000channel までの範囲で plot するには

histogram/plot '55'(0.:1000)

これで右側の graphics Window に自分が見たい (見たかった)histogram が出力されているはずです。もし見にくい場合は

$$histogram/set/max\ \ 'hid'\ \ 'max\ hight'$$

とすれば、$max\ hight$ に設定されている値を最高値とした histogram を見る事が出来ます。

histogram/set/max '55' '1000'

これをした場合としなかった場合の差を以下 (8 ページ) に示す。
よってこの command を実行すれば、このように見やすく変わるということが分かる。

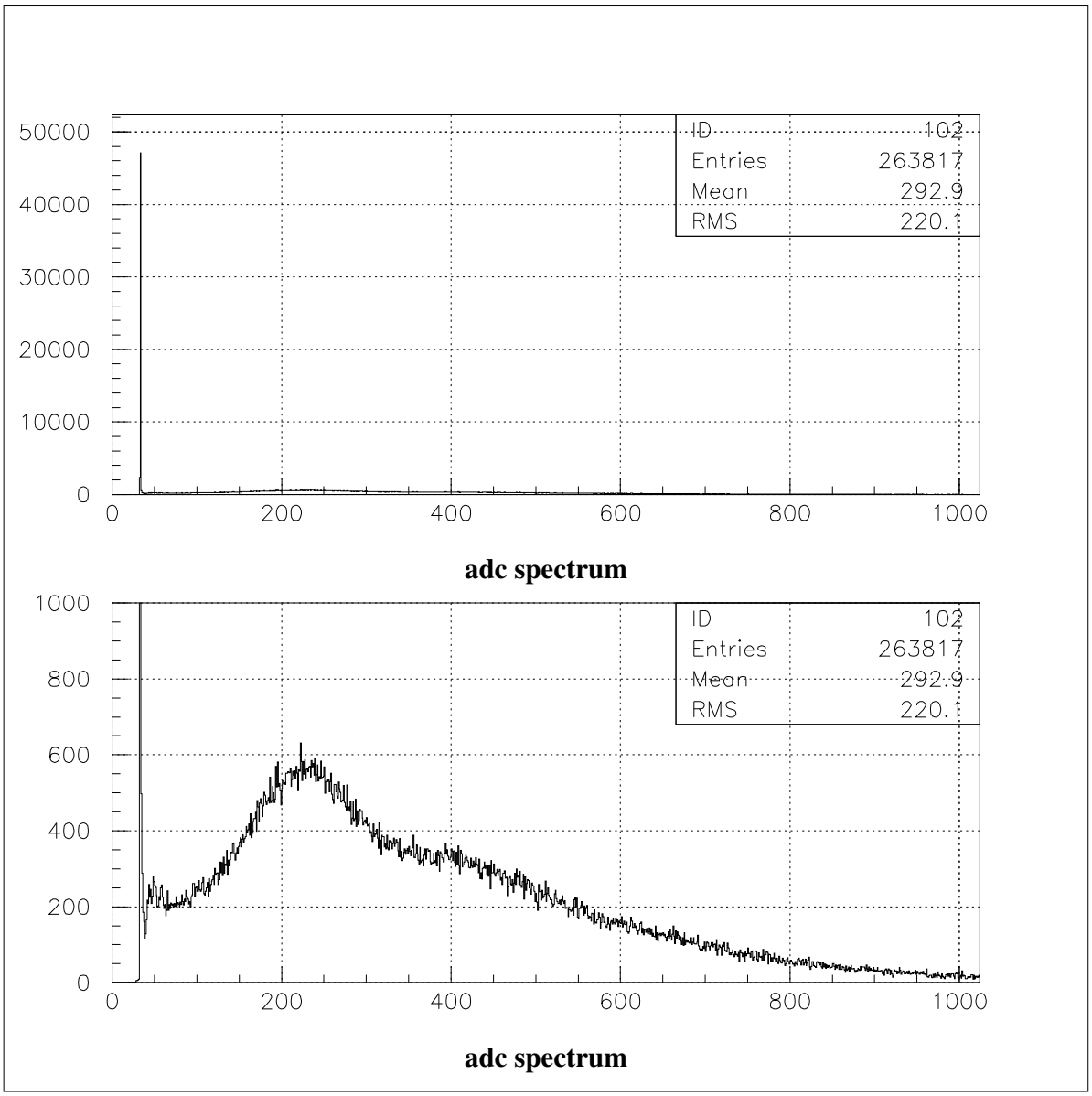

図 1: 上：h/se/ma command 使用前　　下：h/se/ma command 使用後

(6) の histogram の fit に関しては、この後でいろいろと説明します。
次に挙げるのは photon 数を見積もる為の fitting の例です。これを参考にして説明をしていきます。

$$\underline{vector/create\ \ p_{min}(n_{p_{min}})\ \ r\ \ pmin_1\ \ pmin_2 \ldots\ldots\ \ pmin_{n_{p_{min}}}}$$
$$\Downarrow$$
$$\underline{vector/create\ \ p_{max}(n_{p_{max}})\ \ r\ \ pmax_1\ \ pmax_2 \ldots\ldots\ \ pmax_{n_{p_{max}}}}$$
$$\Downarrow$$
$$\underline{vector/create\ \ par(n_{par})\ \ r\ \ par_1\ \ par_2 \ldots\ldots\ \ par_{n_{par}}}$$
$$\Downarrow$$
$$\underline{histogram/fit\ \ hid(start\ ch : last\ ch)\ \ function\ \ B\ \ n_{par}\ \ par\ \ \ p_{min}\ \ p_{max}}$$

このような手順を踏むと histogram に fitting をかける事が出来ます。
上から順に説明して行きます。
まず一番上についてですが、これは P_min と言う名前の vector を作っています。そしてそれには $n_{p_{min}}$ 個の数値 $(pmin_1, pmin_2, \ldots)$ が初期値として代入されます。そしてその下とそのまた下も同様にして P_max, par と言う vector が作成されています。それらにもまた初期値がしっかりと代入されています。ここで初期値の前にある r というのはそれらが実数 (real) である事を示しています。なので、各値が例え整数であってもなるべく 1 の位の桁の下には小数点を打つようにした方がいいです。
そして最後の command は実際に fitting する為のものです。細かいそれぞれの役割は以下のようになります。

h/fit . . . . . histogram/fit の略 (PAW のコマンド)

hid(start ch:last ch)
. . . . . fit をかけたい histogram の id とその範囲 (start ch から last ch まで) を指定。

function. . . . . fitting 関数が書かれている fortran file

n_{par} par. . . . . parameter の数と実際に parameter として使う vector を指定

B∼pmin pmax
. . . . . B とはある範囲内に parameter の値を束縛 (bound) する為のオプションであり、その範囲を指定するのが pmin と pmax である。vector の名前にもそれを示してあるが、上限は pmax で下限は pmin である。

このようにさまざまな command を組み合わせて fitting をかけています。以下に実際の fitting までの実行例を載せておきますので参考にして下さい。またこの fitting の手順は”fitmemo.txt” と言う file にいろいろと書き記されているので、そちらも参考にして下さい。

1. vec/cre pmin2(5) r 5*0
   ⇓
2. vec/cre pmax2(5) r 10000000 10 100 200 300
   ⇓
3. vec/cre par2(5) r 100000 3. 5.8 116. 64.
   ⇓
4. h/fit 101(20:200) func2.f B 5 par2 ! pmin2 pmax2

実際にデータに fitting をかける際には、fitting 関数が書かれている fortran file を誰かから貰っておきましょう。そうでなければ自分で書いてみてもいいと思います。参考として事項に実際の file を載せておきます。これは光電子増倍管に photon が 3 個以下入った場合に得られる spectrum に対して fitting をかけたい場合に用いる fitting 関数です。

最後にまとめとして、これらの流れをまとめておきます。

◇ vector/create rahha(1024)

⇓

◇ vector/read rahha ohha.dat

⇓

◇ histogram/create/1d ’55’ ’adc spectrum’ ’1024’ ’0.’ ’1024.’ ’0.’

⇓

◇ histogram/put/content ’55’ ’rahha’

⇓

◇ vec/cre pmin2(5) r 5*0

⇓

◇ vec/cre pmax2(5) r 10000000 10 100 200 300

⇓

◇ vec/cre par2(5) r 100000 3. 5.8 116. 64.

⇓

◇ h/fit 101(20:200) func2.f B 5 par2 ! pmin2 pmax2

これがこの節の一番のまとめとなります。

func.f

```
c
c     プラスチックシンチレーター＆光ファイバーによるフォトン数
c     測定の Spectrum の fitting 関数
c     光子が 3 つ以下の場合 (ペデスタル無し)

      FUNCTION FUNC3(X)
c
      COMMON/PAWPAR/PAR2(5)
c
c     For scintillation light
c     par2(1): Normalization Factor
c     par2(2): Average number of photo-electrons
c     par2(3): Sigma
c     par2(4): Peak interval
c     par2(5): Position of the pedestal peak
c
          integer I,J,SUM

      FUNC2 = 0.0
      do I=1,15
        SUM=1
        do J=1,I
          SUM = SUM * J
        end do

        v=x-par2(4)*float(I)-par2(1)
        w=2.0*float(I)*par2(3)**2
        y=-1.0*v**2/w
        FUNC3 = FUNC3 + par2(5)*exp(-1.0*par2(2))
     &        *par2(2)**I
     &        /(float(SUM)*sqrt(2.0*3.14159*float(I))*par2(3))
     &        *exp(y)
      end do
      return
      end
```

以上の方法により、ADC2QVT(QVT) で作成した file は解析可能となります。

2.1.2 データの保存方法

ここまでで、データを histogram 化してそれを解析する事までは出来るようになったと思います。次にこのデータを保存して file としておく事で、いつでも以前に作った histogram を使用出来るようにする為の方法を紹介します。
データを格納するのは hbook file と呼ばれる file です。これは PAW で使用される file の一種であり、これに histogram を格納しておく事が出来ます。以下ではこの手順について書いていきます。

(i) PAW の中において、データを格納する為の memory 領域を確保して、その中に既存、若しくは新規の hbook file を展開する。
⇓

(ii) その memory の場所に移動する。
⇓

(iii) PAW の作業 memory 中にある histogram の中で、保存したいものを選び出して先ほど確保した memory に全て出力する。
⇓

(iv) file を閉じる。

各手順については以下で説明する。

まず、(i) はこのような command で実行されます。

$$\underline{\underline{histogram}/file \;\; number_{favorite} \;\; hbook \; file \; name \;\; argument \; value}$$

- ⋆ $number_{favorite}$:memory に付ける好きな番号
- ⋆ $hbook\ name$:作りたい (使いたい)file name
- ⋆ $argument\ value$:新しく作る . . . . . -N
  既存の file を使う . . . -U

これがしっかりと出来たかどうかについては、$Main\ Browser$ の PAWC と書いたところの下に新しく LUN??(??は $number_{favorite}$) と言うものが増えていたら OK です。何故なら、その LUN??がというものが割り当てられた memory 領域そのものに相当するからです。もしダメな場合 (file が開かない, その番号は使われている $ldots$ と Executive Window で怒られる) にはある理由があります。
その時の理由として多いのは、$number_{favorite}$ が既に使われている番号である場合です。この時の対処としては、この番号を換えてやると良いです。出来るだけ 20 以上であまり大きくない番号を使う事をお奨めします。
また最後になりますが、以上の事をほとんど覚えていなくてもとにかく"$h/file$" とだけ覚えていれば、あとは出てくる window の指示にに従って同様の作業が出来るので、無理に覚えていなくても良いです。だけど、作業効率の観点から覚えておいて損はしないと思います。

(ii) は二通りのやり方があります。
1 つは memory 領域を確保した時に出てきた LUN??と言うものをクリックする方法です。これは簡単なので分からないと言う事もないと思います。これが実行されると $Main\ Browser$ に LUN??に入っている histogram(新規作成した場合には何もない) が表示されます。
もう一つの方法は、$Executive\ Window$ において以下の command を実行する方法です。

$$cdir \quad //LUN??$$

これをする事により、上でやった直接クリックする場合と同じ効果が得られます。

(iii) の作業は (ii) と連動している。(ii) の移動が終わったら、すぐに

$$\underline{histogram}/list$$

と打って、その確保した memory 上に展開した file(hbook) の中にある histogram がどれくらいあるのかを調べましょう。もし、新しく作った hbook file のはずが中に histogram が入っていたらそれは他の元からある file を使った事になるので、すぐに閉じましょう。閉じる方法は (iv) を参照して下さい。とりあえず展開した hbook を確認したら、それの中に出力したい histogram の id(hid) を確認して

$$hrout \quad hid$$

と打ちます。もし、全部出力したいのであれば hid の代わりに 0 を入れれば全て出力されます。再び $h/list$ と打ち込めば出力されたのが入っているのかどうか確認できます。

(iv) は、確保された memory 領域を閉じるという作業を示しています。

$$\underline{close} \quad number_{favorite}$$

これで好きな LUN??を閉じる事が出来ます。こちらでもすべての LUN??を閉じる為には $number_{favorite}$ の代わりに 0 を選べば全て閉じられます。

- 前節で作った histogram を保存する

◇ histogram/file 55 quark.hbook -N

⇓

◇ cdir LUN55

or

LUN55 をクリック

⇓

◇ histogram/list

⇓

◇ hrout 0

⇓

◇ close 55

2.1.3 HBOOK の開き方

ここまでで hbook を作る事が出来るようになったと思います。なので次はこの file を使用する方法を説明します。
開く方法には二通りあります。

1. *Main Browser* の中の HBOOK のところにある hbook file のアイコンを直接クリックする。

2. *Executive Window* で command で file を指定して開く。

このどちらでも hbook を開く事が出来るが refclick1 に関しては、開かない場合がある。それは前のところでも書きましたが、PAW を立ち上げている Directory までの絶対パスの中に大文字を含む場合です。この理由としては、普通 PAW が大文字と小文字を区別できないと思います (多分)。けどしっかりと指定してやると (command ラインから指定する) アホな PAW でもしっかりと理解してくれるのである。

$$\underline{histogram}/open \;\; number_{favorite} \;\; \underline{hbook\ file\ name}$$

実際に打ち込む時は以下のようにする。

```
h/file   55  ./quark.hbook
```

以上のようにして hbook を開く事が出来る。あとは開かれた LUN55 へ移動して中にあるアイコンをクリックするなり、command ラインから plot を実行すれば histogram を見る事が出来る。

2.2 一般的な ascii データの解析方法

一般的な . . . と書いてますが、ただデータが複数の列に存在すると言うだけで、根本的には 2.1.1 節で説明した内容と代わりは無いです。ちょっと応用しただけにしておきます。さらに詳しいのは付録?? (??ページ) に書きますのでそちらを参考にして下さい。

2.2.1 vector を使った解析

2.1.1 節では、データを histogram にして解析したのですが、ここでは vector のままで解析する方法について説明します。
まずは以下 (16 ページ) のようなデータを考えましょう。このデータはホール素子を回転させた時に得られた各方向の電圧のデータです。電圧 (V_x, V_y, V_z) はそれぞれ回転角の関数となっており、それらは以下のような式に従います。

$$V = RB_nI - PB_t{}^2I\sin 2\psi + V_0$$

B_n:磁場の素子平面に垂直な方向 (**n**ormal conmponet) の大きさ
B_t:磁場の素子平面方向 (**t**angential component) の大きさ
R:ホール係数 , I:電流の大きさ , P:ブレナーホール係数
ψ:I と $\mathbf{B}_t$ の為す角度 , V_0:residual voltage

この式で以上のデータに対して fitting をかける事を考えます。
上の式 (1 式) にはパラメータとなるべき値が幾つか存在します。これらの各値に対して以下のようにパラメータを割り当てて、fitting の式とします。

$$Par(1) : RB_nI \quad , \; Par(2) : angular\, misalignment \quad , \; Par(3) : \theta\text{の初期位相}$$
$$Par(4) : PB_t{}^2I \quad , \; Par(5) : residual\, voltage \quad , \; Par(6) : \psi\text{の初期位相}$$

1 . 電圧値が小さい場合の fitting の式

$$\begin{aligned} y = {} & Par(1)\, \sin\left(Par(2)\right) \cos\left(x + Par(3)\right) \\ & + Par(4)\left[1 - \left(\sin^2\left(Par\,(2)\right)\, \cos^2\left(x + Par\,(3)\right)\right)\right] \left(\sin(2\left(x + Par(6)\right))\right) \\ & + Par(5) \end{aligned}$$

2 . 電圧値が大きい場合の fitting の式

$$\begin{aligned} y = {} & Par(1)\, \cos\left(Par(2)\right) \cos\left(x + Par(3)\right) \\ & + Par(4)\, \left[\cos^2\left(x + Par(3)\right)\left(2\sin^2\left(Par(2)\right)\, \cos^2\left(x + Par(3)\right) + \cos^2\phi\right) - 1\right] \\ & + Par(5) \end{aligned}$$

これらの各パラメータの割り当て理由等は、昔の僕のレポートを参考にして下さい[5]。このような関数は file に書いておき、その file 名を参照する事で使用出来ます。この式で fitting をかけた結果は以下表 2(18 ページ) のようになります。この結果は PAW の Executive Window に表示されるので、確認しながら fitting をかけていくのが良いと思います。実際に fitting がかかっている様子としては図 2(17 ページ) のような感じになります。
この場合、B_x には 6 パラメータ fitting がかけており、また B_y, B_z には 5 パラメータ fitting をかけています。この時の手順は次 (19 ページ) のようになるので、これを参考にして vector での解析の手助けにして下さい。ついでのついでに funcs.f と funcb.f も載せておきます (20 ページ参照)。

[5] /home2/ito/files/studies/article1.tex 参照

No	回転角 $(degree)$[$1]	$V_x(mV)$[$2]	$V_y(mV)$[$3]	$V_z(mV)$[$4]
1	0.05	-0.0201	14.8249	52.5100
2	-9.08	-0.0968	23.0651	49.4130
3	-17.98	-0.1933	30.5588	45.1850
4	-27.08	-0.3153	37.5185	39.7213
5	-35.99	-0.4546	43.4111	33.4517
6	-45.08	-0.6136	48.3748	26.2404
7	-54.01	-0.7783	52.0238	18.5879
8	-63.08	-0.9443	54.4563	10.3843
9	-72.00	-1.0952	55.4790	2.0980
10	-81.07	-1.2256	55.1330	-6.3833
11	-90.01	-1.3215	53.4178	-14.6018
12	-99.06	-1.3769	50.3320	-22.6189
13	-107.99	-1.3820	46.0674	-30.0108
14	-117.05	-1.3325	40.6019	-36.8215
15	-125.99	-1.2283	34.2392	-42.6555
16	-135.06	-1.0677	26.9390	-47.5410
17	-143.95	-0.8619	19.1647	-51.1570
18	-153.03	-0.6127	10.7705	-53.5670
19	-161.93	-0.3430	2.3238	-54.5920
20	-171.05	-0.0544	-6.4383	-54.2510
21	-179.91	0.2191	-14.8046	-52.5760
22	-189.01	0.4781	-23.0826	-49.5200
23	-197.91	0.6973	-30.6341	-45.3260
24	-207.03	0.8776	-37.6602	-39.8828
25	-215.91	1.0024	-43.5837	-33.6350
26	-224.99	1.0780	-48.5697	-26.4323
27	-233.90	1.1014	-52.2487	-18.7856
28	-242.99	1.0801	-54.7023	-10.5303
29	-251.90	1.0210	-55.7360	-2.2329
30	-260.95	0.9294	-55.3790	6.2782
31	-269.88	0.8202	-53.6547	14.5157
32	-278.96	0.6972	-50.5212	22.5998
33	-287.90	0.5747	-46.2095	30.0206
34	-296.97	0.4570	-40.6751	36.8671

表 1: ホール素子から得られた電圧データ

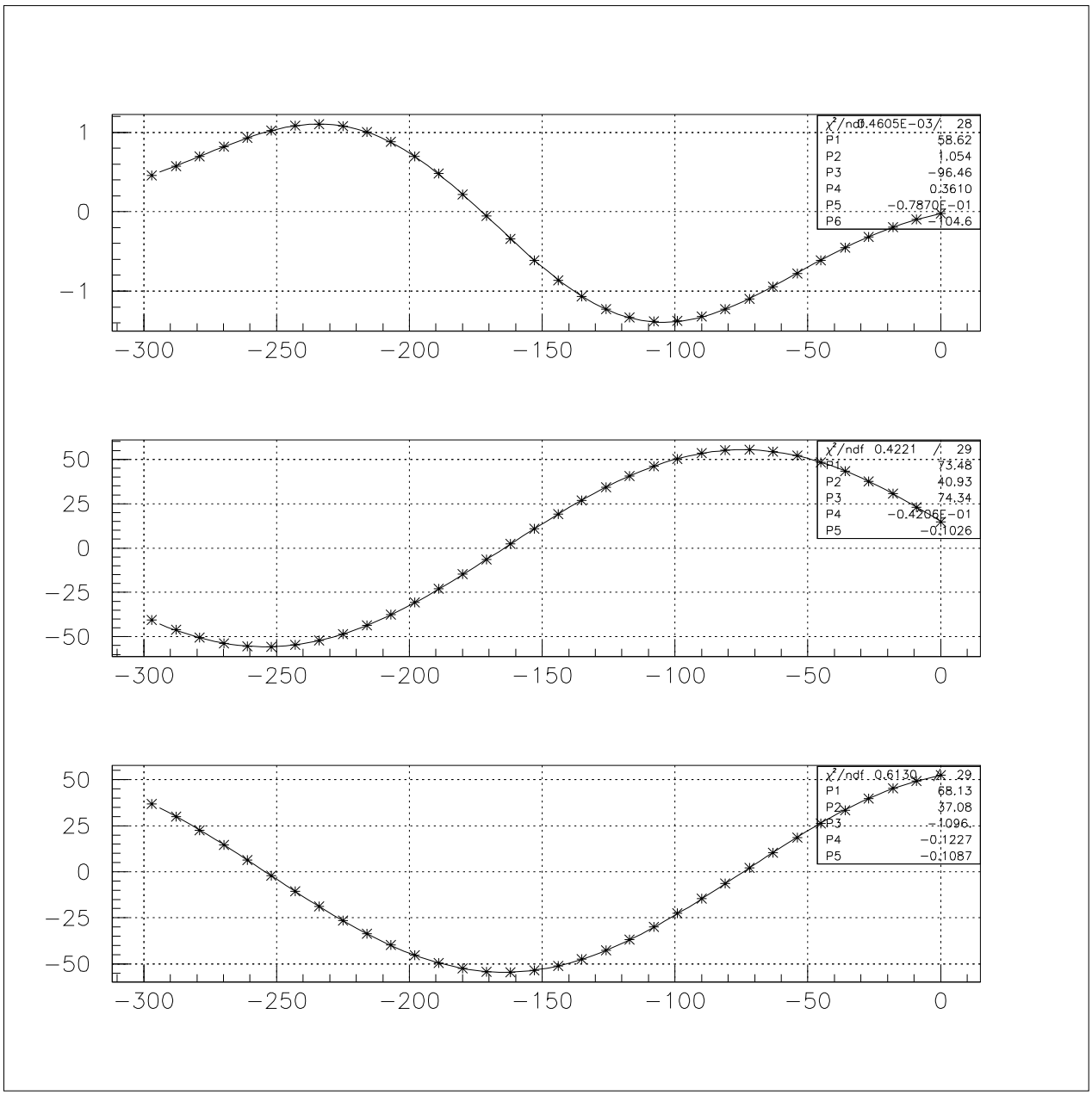

図 2: 上:B_x**vs** *rotation angle* 中:B_y**vs** *rotation angle* 下:B_z**vs** *rotation angle*

Bx

PARAMETER NAME	VALUE	APPROXIMATE ERROR	STEP SIZE	FIRST DERIVATIVE
P1	55.511	186.12	0.83755E-01	-0.82558E-07
P2	1.1129	3.7322	0.34006E-03	-0.10204E-04
P3	263.54	15.276	0.66670E-01	-0.10156E-05
P4	0.36103	0.24985	0.33937E-03	-0.12996E-04
P5	-0.78707E-01	0.18855	0.12200E-02	0.59800E-05
P6	-284.65	21.458	0.36005	0.83709E-06

$CHISQUARE =$ $0.1643E-04$ $NPFIT = 34$

By

PARAMETER NAME	VALUE	APPROXIMATE ERROR	STEP SIZE	FIRST DERIVATIVE
P1	55.540	3.3811	0.70253E-01	0.58362E-02
P2	1.4919	130.43	0.75895	-0.10320E-03
P3	74.342	0.27053	0.18807E-01	0.23819E-02
P4	-0.65112E-01	0.39087	0.49339E-01	-0.12572E-02
P5	-0.11785	0.20284	0.14538E-02	-0.48164E-02

$CHISQUARE =$ $0.1452E-01$ $NPFIT =$34

Bz

PARAMETER NAME	VALUE	APPROXIMATE ERROR	STEP SIZE	FIRST DERIVATIVE
P1	54.400	3.5300	0.68811E-01	0.17023E-02
P2	2.3700	88.373	0.11976E-01	0.33205E-02
P3	-15.747	0.23588	0.19919E-01	-0.11828E-01
P4	-0.15026	1.7269	0.23103E-02	-0.67234E-02
P5	-0.13155	0.94774	0.15523E-02	0.37858E-02

$CHISQUARE =$ $0.2169E-01$ $NPFIT = 34$

表 2: 各関数で fitting をかけた結果

♢ vector での解析に用いる command 例

---

vector/create temp (7,34)
⇓
vector/read temp y570a.dat
⇓
vector/create par(6) r 55. 1. 10. 0.35 0.1 10.
⇓
vector/create pmax(6) r 60. 1.8 360. 0.5 1. 360.
⇓
vector/create pmin(6) r 50. 0.8 -360. 0.1 0.01 -360.
⇓
vector/fit temp(1) temp(2) ! funcs.f *B 6 par 0.1 pmin pmax
⇓
vector/fit temp(1) temp(2) ! funcs.f * 6 par 0.1 pmin pmax
⇓
vec/create par(5) r 53. 0.9 10. 0.388 0.01
⇓
vector/create pmax(5) r 60. 2. 360. 0.5 0.5
⇓
vector/create pmin(5) r 50. 0.5 -360 0.01 0.001
⇓
vector/fit temp(1) temp(3) ! funcb.f *B 6 par 0.1 pmin pmax
⇓
vector/fit temp(1) temp(3) ! funcb.f * 6 par 0.1 pmin pmax
⇓
vector/fit temp(1) temp(4) ! funcb.f *B 6 par 0.1 pmin pmax
⇓
vector/fit temp(1) temp(4) ! funcb.f * 6 par 0.1 pmin pmax

□ *funcs.f*

```
c     paw で fitting をかける時に使う関数式
c     回転軸が平面の法線方向と並行でありそれが磁場と垂直の時の fitting の式
c     n –⟩ rotation

      FUNCTION FUNCS(X)
      COMMON/PAWPAR/PAR(6)


       FUNCS = PAR(1)*sin(0.0174533*PAR(2))*
     $     cos(0.0174533*(X+PAR(3)))
     $     +PAR(4)*
     $     (1-((sin(0.0174533*PAR(2)))**2)
     $     *((cos(0.0174533*(X+PAR(3))))**2))
     $     *sin(2.0*(0.0174533*(X+PAR(6))))
     $     +PAR(5)

      return

      end
```

□ *funcb.f*

```
c     paw で fitting をかける時に使う関数式
c     回転軸が平面方向であり、それが磁場と垂直の時の fitting の式
c     Hall effect –⟩ big , planar Hall effect –⟩ small

      FUNCTION FUNCB(X)

      COMMON/PAWPAR/PAR(5)


      FUNCB = PAR(1)*cos(0.0174533*PAR(2))
     $ *cos(0.0174533*(X+PAR(3)))
     $ +PAR(4)*(((cos(0.0174533*(X+PAR(3))
     $ ))**2)*
     $ ((2*(sin(0.0174533*PAR(2)))**2)*
     $ ((cos(0.0174533*(X+PAR(3))))**2)+
     $ ((cos(0.0174533*PAR(2)))**2))-1)
     $ +PAR(5)

      return

      end
```

2.2.2 ntuple を使った解析

まだやった事がないので分かりません。分かり次第書こうと思います。
(2001 年 2 月 27 日)

2.2.3 histogram を用いた解析

これは一般的なのはまだまだ分からないので、準備が出来次第書きます。
(2001 年 2 月 27 日)

3　いろんな設定

多少 PAW に慣れてくると、いろいろと自分好みに設定を変えたりしたくなってくるものです。そんな時の設定の変え方を、及びその保存の仕方をここに書いておきます。役に立つ物もありますが、中にはかなり役に立たないものもあるので、そこらへんは各自で判断して読んでください。とりあえず 3.1 節と 3.2 節だけは役に立つと思います。

3.1　option と set

設定を変える command として"set","option" と言う command があります。これの中でもよく使われるものについてのみ説明します (使い方は後述)。これ以上細かいものは、正規の manual の各対応ページ (option は P284, set は P286) を参照して下さい。

(1) option

(i) grid:
図にグリッド (線) を入れます。

(ii) fit :
データに対して関数フィットをした場合に、その結果 (各パラメータの値) を図中に出力してくれます。

(iii) date:
図中に今日の日付を入れてくれます。

(iv) stat:
図中にデータの統計的情報を表示してくれます。

(v) logx:
x 方向を log scale にします。

(vi) logy:
y 方向を log scale にします。

(vii) linx:
x 方向を linear scale にします。

(viii) liny:
y 方向を linear scale にします。

(2) set

(i) grid:
グリッドのタイプを設定する。

(ii) date:
日付を表示する場所を設定する。

(iii) csiz:
コメントの大きさを設定する。

(iv) cfon:
コメントのフォントを設定する。

(v) gsiz:
タイトルの大きさを設定する。

(vi) gfon:
タイトルのフォントを設定する。

(vii) tsiz:
ヒストグラムのタイトルの大きさを設定する。

(viii) tfon:
ヒストグラムのタイトルのフォントを設定する。

(ix) fcol:
関数をフィルするエリアの色を設定する。

(x) fwid:
関数をフィルするエリアの太さを設定する。

(xi) ygti:
タイトルを書き入れる場所を設定する。

この option,set の使い方は PAW の Executive Window の command を入れるところで

$$set\ \ command\ \ value$$

もしくは

$$\underline{option}\ \ command$$

と入れる事で実行されます。ただし、下線部分は省略可です。
またすべての値を default 値に戻すには、それぞれの command として"*"を指定して実行して下さい。

3.2 macro file

上の 3.1 節でやったように、PAW は様々な設定を各自好きな時に変える事が出来ます。ただ、これらの中にはいつでも好きな時にこの設定を使用したり、さらには設定を常に保持しておきたいものもあると思います。そのような設定は file として取って置く事が出来るし、それらを macro file として実行する事により好きな時、若しくは常にそれらの設定を使用する事が出来ます。

一般的な macro file と同様にして、PAW のマクロファイルもコマンドを羅列したものとなります。特に宣言等も要りません。細かい話をすれば、いろいろと出てきますが、とりあえずコマンドを並べてしまえばそれが macro file となります。よって、これに自分が設定したい option や set を書き込んでおけば、それらを実行するだけで各種設定が各自の思うがままになります。
ただし、ファイルの拡張子は *kumac* にして下さい。それでないと実行出来ない恐れがあります。実行の仕方は

exec [macro file]

と PAW の Executive Window で打ち込むと実行されます。[6]
macro file の特殊な物として".pawlogon.kumac" と言う file があります。これは PAW 立ち上げ時に必ず読み込まれる macro file です。これに、常に設定しておきたい事を書いておけば、常にその設定はされる事になります。そのような設定があれば、ホームディレクトリに".pawlogon.kumac" を置いておくといいでしょう。

[6] この exec を省略出来る場合もありますが、それはまだまだ先のお話としておきます。気が向いた時に書き加えておきます。

3.3 .Xresources

ここは完全なおまけです。この設定に関しては僕よりも詳しい人がいますので、分からない事があるのならば、そちらの人に聞いて下さい。

PAW を立ち上げると、3 つの Window が立ち上がりますがこれらの色というものは決まった色をしています。実はこれらの色を変える為 (本来は違うけど) の file が".Xresources" と言う file です。これは$HOME に置いておくと勝手に読み込んで色の設定を自動で行ってくれます。以下に僕の file を載せておくので、これと同様の内容を書いてホームディレクトリに置いて下さい。ただし、無くても構わない file であるので、強制ではありません。

- KxPaw++*Background: lavender
- Paw++*Background: lavender
- kterm*Background: SaddleBrown
- kterm*Foreground: snow1
- kterm*cursorColor: SandyBrown
- xemacs*Foreground: hotpink

上で下線を引いた場所が PAW に関する設定をしている場所です。他のは kterm や emacs の色の設定なので参考程度に載せておきました。
また、どのような色があるのかを見るには kterm を立ち上げて、どこでもいいので

[**ito**]ksnucl:~ >gcolorsel &

と言うふうに"gcolorsel" と言うアプリケーションを立ち上げて調べてください。これにはいろんな色の名前が付いています。ただし、これらを設定する上で注意してほしいのは文字等が見えにくい色を設定するのは止めた方がいいと言う事です。それを一番重視して、かつ好きな色に出来れば PAW に取り組む姿勢もかなり変わる事と思います。とりあえず自分に合う色を探す事からはじめましょう。
この設定に関して分からない事があれば有賀君へどうぞ。

3.4　その他の便利なもの

最後にその他にあるもので便利な command を書いておきます。

- ▷ zone
- ▷ defaults

また思い出したらつけ足しておきます。
まずは zone について。。。
zone は histogram,vector 等を plot する際に並べて plot したい場合に有用な command です。使用例は以下のようになります。

$$zone \;\; [i] \;\; [j]$$

これで i(列の数),j(行の数) だけの plot area が確保されます。つまり 4 枚の histogram を均等に張り付けたいのであれば

$$zone \;\; 2 \;\; 2$$

と打てば良いです。同様にして 3 枚の histogram を均等に張り付けたいのであれば

$$zone \;\; 1 \;\; 3$$

と言うようになります。これを元に戻したい時には何も引数を指定せずに $zone$ とすれば default 値である $zone \;\; 1 \;\; 1$ の働きをしてくれます。
均等でない場合は多少面倒になりますので、それはまたあとで。

次は defaults について。

A Handmade HBOOK

今までは、データを Executive Window で commnad を打つ事からのみ HBOOK file を作成してきました。しかし、ascii データが存在した場合にそれを解析する方法として、PAW を通すのではなくて fortran を使って直接 HBOOK file を作る方法があります。この方法について以下で説明してみます[7]。
ここで fortran で書くと書いてしまいましたが、普通に全て最初からやろうとすると HBOOK file の仕組み、fortran でそれを実現する為の知識などなどたくさんの事が要求されるので、なかなか大変です。はっきり言って僕にはまだ無理だと思います。そこで、ある他からの手助けと言うものを使って作りましょう。ここで言うある他からの手助けとは"CERN Library" の事です。それの中には HBOOK file を作る際に非常に有用な関数が入っています。これらを使えば HBOOK file を簡単 (ある程度) に作れます。ちなみにこれを使わないで C 言語を使って書いたものもあります[8]ので僕が理解したらそれもここに付け加えます。ただし今は fortran で HBOOk を作成する方向でいきます。まずは基礎編って事で。

上でも書いたように、fortran でさまざまな関数を使う為には library を参照させる必要があります。これは C 言語で言えば stdio.h,math.h. . . *etc* と同じようなものです。しかし、C 言語のように file 内部に参照させる file 名を書き込むのではなくて compile,link の際に参照出来るように指定して以上の事を実行します。その方法とは

```
'g77 -L$CERN_ROOT/lib -lgraflib -lgrafX11 -lpacklib -lmathlib
                                -lkernlib -lpawlib -lnsl -lrpcsvc'
```

と言う option を g77 に付ける事により実現されます。他の compile(frt などなど) については知りません。そのうち必要に駆られたり気が向いたら調べてみます。とりあえず書き上げた file を compile, link する際には必ず以上の option を付けて下さい (ただし全部が必要だとはいいません。多分ですが一部は必要じゃないのかも知れないなあって思ってます。これもヒマな時に調べておきます)。また、これを compile するたびに打ち込むのは嫌だ!って人は login 時に読み込まれる file(.tcshrc,.tcshrc.user ,.alias . . .) のどれかに alias(command の別名付け) を書き込んでおけば良いと思います。その方法は楽しい UNIX など読んでみて下さい。

では次に file の本体の書き方に関しての説明です。
まず、理屈を言ってもわかりにくいので最初に、sample file を例として 28 ページに載せておきます。全てこれを基本として説明していきます。fortran の知識が無くても他の言語等を多少学んでいたりすれば余裕だと思います。知らなくてもきっと大丈夫です。どうしても分からないという人は郡司先生の情報処理用のテキストをあげます。僕はこれで fortran を学びました。とてもわかりやすいと思います。あとは研究室にある fortran の本、もしくは図書館にも多少 fortran 関係の本が置いてあります。面倒くさい人は手近な人を捕まえて聞くのも良いです。いい人がいたら活用しましょう。
とにかく、深くは説明しませんが簡単には fortran についても説明します。

[7]もともとの参考文献としては "KEK の UNIX ワークステーションによるデータ解析第 6.0 版を参照しました。興味があればそちらも見て下さい

[8]2001 年 2 月現在 M2 の千葉さん作
自分でも見てみたいって言う人は **/home2/ito/analysis/fill_for_c** を参考にして下さい。

```
      program histogram
      common / pawc /h(3000000)
      call hlimit(3000000)

c     hbookfile initialize
      call hbook_init()

c     Data fill and hbookfile create
      call hbook_fill()
      end

      subroutine hbook_init()
      common / pawc / h(3000000)

      call hbook1(1,'Incident Energy',500,0.,500.,0)
      call hbook1(2,'Deposit Energy at BaF2 ',500,0.,500.,0)
      call hbook1(3,'Deposit Energy at dE',500,0.,20.,0)
      call hbook1(4,'Deposit Energy at target',500,0.,5.,0)
      call hbook2(5,'target_Dep-Incident_E',500,0.,5.,500.,0.,500.,0)

      return
      end

      subroutine hbook_fill()
      integer i,j
      real data(4)
      common / pawc / h(3000000)
      open(10,file='lns1.dat',status='old')

 5    read(10,10,end=20) data(1),data(2),data(3),data(4)

 10   format(3(F11.4,1X))

       call hf1(1,data(1),1.)
       call hf1(2,data(2),1.)
       call hf1(3,data(3),1.)
       call hf1(4,data(4),1.)
       call hf2(5,data(1),data(4),1.)

      goto 5
 20   continue
      close(10)
      call hrput(0,'randombeam1.hbook','N')

      return
      end
```

まず、

```
program histogram
common / pawc /h(3000000)
call hlimit(3000000)
```

の部分ですが、これはあまり気にしなくても良いです。数字もあまり小さくならず、かつ大きすぎずくらいにしましょう。ここでは program の名前を決めて、そして確保する memory 量とその限度を決めているのです。名前でなんとなく分かりますね。あまり詳しくは知らないのでこのくらいにしておきます。

ちなみに余談ですが"c"と言う単語が 1 列目につくと、それはコメントの宣言となります。つまりその行はコメントアウトされるのです。だいたい、自分で書いてて忘れそうなところはコメントで何かを書き残しているので、そういうのも活用して下さい。
ここで急にこの事を書いたのは、sample の中にある

```
c     hbookfile initialize
      call hbook_init()
```

もしくは

```
c     Data fill and hbookfile create
      call hbook_fill()
      end
```

というのもその類いの事をしているからです。上のものは hbook を初期化する為の関数ですよと言う事の印として (*hbook_init* 関数)、下のものは hbook に fill していく (histogram 化する事) 関数 (*hbook_fill* 関数) ですよと言う印としてコメントを残しています。

```
      subroutine hbook_init()
```

次に上のこれは subroutine として hbook_init を呼び出しています。subroutine とは関数の一種です。と、言うか関数そのものですね。fortran の達人は、各自自分で好きな関数を書いてそれらを subroutine として呼び出して使います。詳しくは達人に聞いて下さい。

次は実際に histogram(hbook に入れる) を形だけ作ります。

```
call hbook1(1,'Incident Energy',500,0.,500.,0)
call hbook1(2,'Deposit Energy at BaF2 ',500,0.,500.,0)
call hbook1(3,'Deposit Energy at dE',500,0.,20.,0)
call hbook1(4,'Deposit Energy at target',500,0.,5.,0)
call hbook2(5,'target_Dep-Incident_E',500,0.,5.,500.,0.,500.,0)
```

ここでは *hbook*1, *hbook*2 の二つがありますがそれらは次元を表わしている。もし一列のデータを histogram 化したいのであれば hbook1 を、二列のデータを histogram 化したいのであれば hbook2 を使用します。これらは多少書き方が異なりますが、当然基本的にはほとんどいっしょのものです。この関数に渡す引数の書き方は以下のようになります。

$call\ \ hbook1(hid,'title', N_{bin}, Min, Max, Buffer_{bin})$

or

$call\ \ hbook2(hid,'title', N_{bin}(x), Min(x), Max(x), N_{bin}(y), Min(y), Max(y), Buffer_{bin})$

hid : histogram id(好きな数)

title : histogram の下に付ける好きな名前

N_{bin} : bin(channel) の数

$Buffer_{bin}$: bin の buffer size(0. で default 値)

Min : 最低値

Max : 最高値

N_{bin} は bin(channel) の数で、これを大きくするときめ細かい histogram を得る事が出来るし、逆に小さくすれば荒い histogram を得る事が出来ます。また、これをしっかりと決めれば軸の一単位がエネルギー単位 (MeV,GeV...) や時間単位 (nsec,psec...) に対応させる事が出来ます。最低、最高値は histogram の各軸方向の取る最低、最高値です。これを変えても histogram の顔は変わって見えます。最後に重みというのは各 channel 毎にかける重さの事であり普段はまず、基準となる値 (0.) 以外は考えなくてもいいです。
　ではどんどん先に進みます。

```
subroutine hbook_fill()
integer i,j
real data(4)
common / pawc / h(3000000)
open(10,file='lns1.dat',status='old')
```

一番上のものは前のものと同様に subroutine を呼んでいます。次とその次はただの変数宣言。その次 (common...) は一番最初にやった memory の確保です。
では、その次。これは実際に fill したいデータが格納されている file を開くと為の手順です。¨10というのは開いた時に付ける番号、次が file 名、で、最後の¨statusと言うのがその file が最初からあったのか、今から作るのかと言う区別をする為の一種の flag のようなものです。最後は無ければ無くても良いらしいです。でも一応付けておいた方が安心ですね。
これで解析したい file を開く事までは出来ました。あと少しで hbook file が作れそうです。
ちなみにここでは既存の lns1.dat と言う file を開いています。
それでは次。

```
 5    read(10,10,end=20) data(1),data(2),data(3),data(4)

 10   format(3(F11.4,1X))
```

ここではこの式に対して label が付けられています。よってこの番号を指定する事でここに戻ってきたり、ここを参照したりする事が出来ます。
そしてここでやっているのはデータの読み込みです。¨5はこの action の行の label, 最初の¨10は開いた file につけた番号。後の"10"はその読み込みの為の format 指定を示しています。

format 文に関しては頭の”10”は label の為、3 付きの () でくくっているのは同じ format のデータが 3 つあると言うことです。そしてその中身”F11.4”とは”浮動小数点形式のデータで、全体としては１１桁を割り当てられていて小数点以下は４桁”と言うデータを表わす為の format です。ここで format を指定し間違えると、fortran は compile error を起こす場合があるので、十分注意して下さい。
あと、1X とはスペース１個の事です。２個なら”2X”です。あと、それらを区切るのは”,”ですのでお忘れないように。。。
そろそろ最後です。

```
       call hf1(1,data(1),1.)
       call hf1(2,data(2),1.)
       call hf1(3,data(3),1.)
       call hf1(4,data(4),1.)
       call hf2(5,data(1),data(4),1.)
      goto 5
 20   continue
```

上の 5 つはみんないっしょの事をしています。違うのは fill しているデータが違うだけです。この hf1,hf2 には以下のように引数を渡して使います。

$$call \quad hf1(hid, data, weight)$$
$$or$$
$$call \quad hf2(hid, data1, data2, weight)$$

hid : histogram id(さっき作った中から指定したい id を入れる)

data : fill したいデータ名

data1 : x 方向に fill したいデータ名

data2 : y 方向に fill したいデータ名

これでデータの一部を histogram の１データとして補充した事になります。あとは fortran で指定した loop ですべてのデータを histogram に加えれば 99%は完了です。
これで最後です。

```
      close(10)
      call hrput(0,'randombeam1.hbook','N')
```

ここ (close(10)) で最初に開いた file を閉じています。もし使用し終わった file を閉じないでおくと memory 不足により core を出して処理が終わってしまう場合があります。なので、file はなるべく使う時に開いて使い終わったら閉じるようにしましょう。この close の使い方としては引数として file を開いた時に使った番号を渡します。ここで注意してほしいのは、引数として”0”を渡すと今現在開かれている file が閉じられると言うことです。最後にすべて閉じる場合以外は、あまり’0’ は使わない方がいいかもしれません。

そしてその次では作ったhistogramを全てrandombeam1.hbookに出力しています。”hrput()”とはそのための関数です。”hrput()”へ渡す引数は以下のようになります。

$$hrput(hid,'hbook\ name','status')$$

hid：出力したいhistogramのid

hbook name：出力先のhbookの名前

status：上で指定したhbookが既存のものか新規のものかを決めるflag
新規の場合は’N’, 既存の場合には’A’を指定

ここで’status’で’N’を指定して既存のfileを開くと、前からあるfileは空になってしまうので注意して下さい。大事なfileに追加出力する時にはなるべくbackupをとってからにした方が良いと思います。

以上のような過程を経て、hbook fileを作ることが出来ます。これらは前の方で説明したようにPAWで見ることが出来るので、あとはそちらで同様にして解析しましょう。

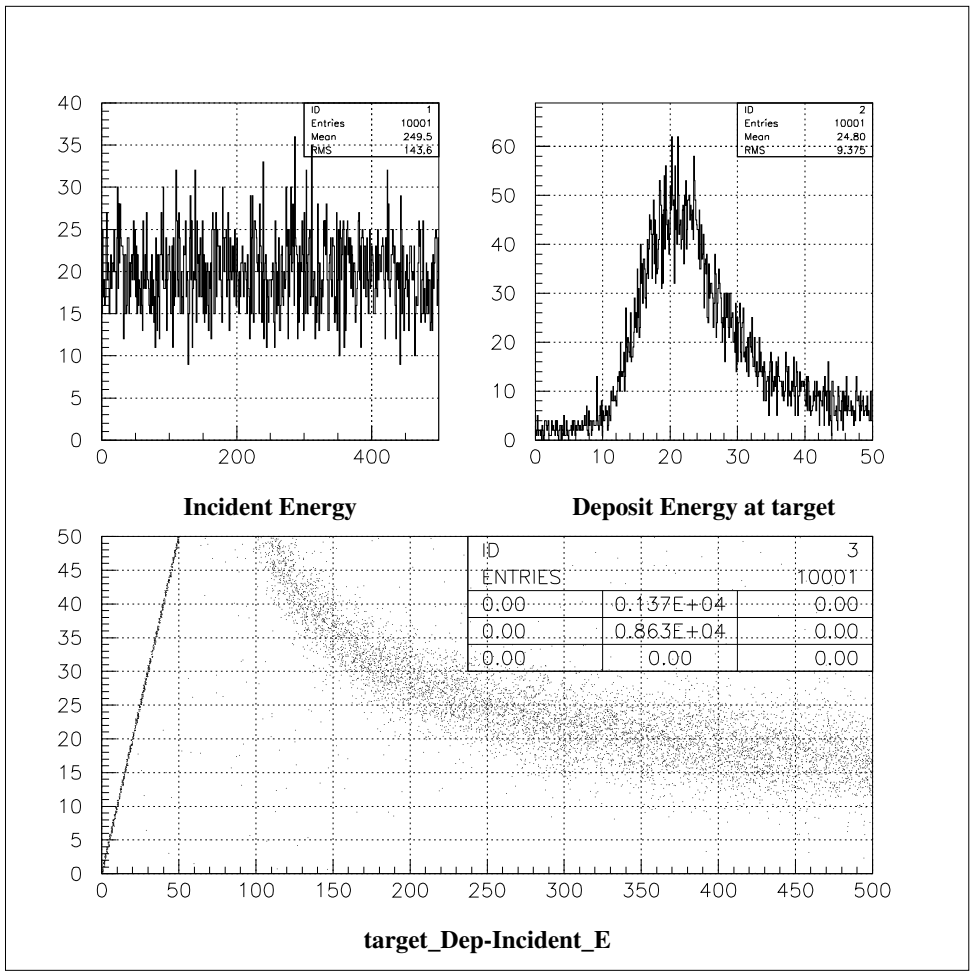

図 3: 上のfortran fileを使ってfillした例