AYPLOT

柳瀬 章

平成 19 年 9 月 26 日

1 はじめに

空間群のプログラム TSPACE、立体透視図のプログラム TPERSP はいずれもバンド計算を実行したり、その結果を、効率良く理解し表現するために 1970 年代の後半の時代に開発された。後者の TPERSP は、フェルミ面と結晶構造の作図用に開発された。計算センターにある計算機を共同利用するのが、計算機の普通の使い方であった 1980 年代までは、どこのセンターにも作図用の装置がおいてあった。この時代は計算機を使うのには自分でプログラム (ほとんど FORTARN) を作ってそれをコンパイルして実行していた。それぞれの装置は独自の形式のデーターで駆動されるようになっていたようで、それを使うユーザーはセンターが発行したマニュアルにしたがって、自分のプログラムの中に CALL PLOT(X,Y,IP) とか CALL LINE2(X,Y,IP) のような作図用のサブルーチンを CALL する文をいれて、作図プログラムを作成していた。TSPACE,TPERSP はこの時代に開発された。したがって結晶構造を画いたり、バンド図を作図する時には、当然のようにこの形式で行ってきた。

しかし時代とともに、センターから作図専用の装置が姿を消していった。現在では計算結果などを図にするときは、ほとんどプリンターが使われている。そのためのソフトもワークステーション、パソコン等のシステムに合せて、AVS,GNUPLOT,AdobeIllustrator などさまざまなものが開発され広く使われている。これらを FORTRAN から使用するためには、それぞれのソフトに合せた書式でデーターファイルを作らなければならない。筆者の勉強不足のせいか、1970 年代に TPERSP で、できたことを、これらのソフトを使用してはできないことがある。幸い筆者の環境では、昔のようにドラフターに出力するつもりで FORTRAN のプログラムを走らせれば、適当なファイルが出力されて、簡単な手続きでプリンターに出力できるようになっていた。これは TPERSP の利用者により作成された、pig システムのおかげである。このような環境はほとんどの FORTRAN のユーザーは持っておられることを前提にして「空間群のプログラム TSPACE」の本を書いた時に作図関連の機能についての記述をおこなった。しかしこのことはそれほどあたりまえではないことが、読者の方からの通信で知らされた。最近の状況として、ポストスクリプトの形式で出力すれば、広範なプリンターやソフトが正常に扱うようになっている。ポストスクリプトはその解説書によれば、一つの計算機言語であり、これを解読するソフトやプリンターは、この言語のインタープリター機能を持っている。そこでこの機能を生かして、昔の計算センターにあったようなプログラムセットを今回「AYPLOT」として作成した。直線を引くことと、文字を書くことだけの機能であるが、TSPACE、TPERSP の応用にはこれで十分である。

ここで説明するサブルーチンはポストスクリプトの文法に精通した上で作られたものではなく、反対にマニュアルをとばし読みして、必要な機能を拾い出した結果である。一応 PostScript リファレンス・マニュアル第２版の記述にしたがっている。個々のサブルーチンの詳しい説明の前に、簡単な例と作図結果をまず示すことから始める。続いて、今回開発したサブルーチンの機能を、出力であるポストスクリプトの説明を含めて行う。

2　簡単な例

プログラム 2.1 は AYPLOT で直線と文字を画いた利用例である。結果は図 1 である。それぞれのサブルーチンが何をしているかは、名前から理解されると考える。

プログラム 2.1

```
      CALL AYPSTR(85)
      CALL AYORIG(20.0,30.0)
      CALL MOVETO(10.0,10.0)
      CALL LINETO(65.0,10.0)
      CALL MOVETO(10.0,10.0)
      CALL LINETO(10.0,20.0)
      CALL AWRITE(30,'(ABCDEFG)',9,10.0,10.0,0)
      CALL AYPEND
      STOP
      END
```

図 1: 直線と文字

プログラム 2.2 は AYPLOT で三角関数のグラフを画いている。結果は図 2 であるがこの図では文字は、AdobeIllustrator に取り込んだあとで加えてある。この図は一見曲線に見えているものが、良く見ると直線の集まりであることが見てとれるように画いてある。ここでは 41 点で画いているが、この程度の寸法で 200 点をこえればどう見ても曲線になる。

プログラム 2.2

```
      CALL AYPSTR(86)
      CALL AYORIG(20.0,30.0)
      PI=4.0*ATAN(1.0)
      XM=140.0
      HXM=0.5*XM
      YM=120.0
      HYM=YM*0.5
      CALL MOVETO(0.0,0.0)
      CALL LINETO(XM,0.0)
      CALL LINETO(XM,YM)
      CALL LINETO(0.0,YM)
      CALL LINETO(0.0,0.0)
      CALL LINEWD(0.5)
      CALL MOVETO(HXM,0.0)
      CALL LINETO(HXM,YM)
      CALL MOVETO(0.0,HYM)
      CALL LINETO(XM,HYM)
      CALL LINEWD(1.5)
      DO 1 I=-20,20
         X=(PI*I)/20.0
         XX=XM*(I+20)/40.0
         YY=(SIN(X)+1.0)*HYM
         IF(I.EQ.-20) CALL MOVETO(XX,YY)
         IF(I.GT.-20) CALL LINETO(XX,YY)
1     CONTINUE
      CALL LINEWD(1.3)
      CALL SETDAS(5,3,1)
      DO 2 I=-20,20
         X=(PI*I)/20.0
         XX=XM*(I+20)/40.0
         YY=(COS(X)+1.0)*HYM
```

```
      IF(I.EQ.-20) CALL MOVETO(XX,YY)
      IF(I.GT.-20) CALL LINETO(XX,YY)
2     CONTINUE
      CALL AYPEND
      STOP
      END
```

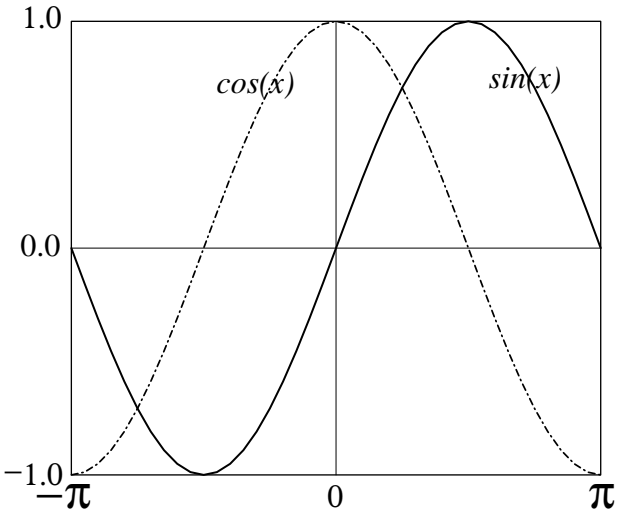

図 2: 三角関数のグラフ

3 AYPSTR & AYPEND

```
      CALL AYPSTR(ILLF)
```

AYPLOT を利用するプログラムは最初に AYPSTR を CALL し、最後に AYPEND を CALL する。AYPSTR の整数型の引数は出力ファイルの機番を指定する。処理系によっては、対応する OPEN 文が必要になる。

AYPSTR は出力としての ps ープログラムにヘッダーをつける役割を受け持つ。ps ープログラム 3.1 はこの部分の出力である。%で始まる行はコメント行である。したがって最初の 4 行はなくてもよいはずである。実際ほとんどのソフトやプリンターはこれがなくても動作する。ただし AdobeIllustrator は二行のコメント行がないと、なぜか図をあらわすものとは解釈しないで、テキストとしてあつかってしまう。コメント行の最初は、ほとんどのポストスクリプト関連のソフトの出力の 1 行目についているので、ここでもポストスクリプトの創始者である Adobe 社に敬意をはらってつけてある。2、3 行めは AYPLOT で以下の図が作られたことを示し、このソフトが作られた時期を示している。

ps-プログラム 3.1

```
%!PS-Adobe-2.0
%%Creator AYPLOT
%%CreationDate:2001:01:31   by A.Yanase
%%EndComments
/m {moveto} def
/l {lineto} def
/S {stroke} def
```

```
/w {setlinewidth} def
newpath
1 setlinecap
1 setlinejoin
    1.00    1.00 scale
  1.00 w
```

次の 4 行はこれから頻繁に使う演算子を簡略な表現に変えるために、”def” 演算子を使っている。各演算子の機能については節 5 で説明する。newpath は「カレントパスを空に初期化して、カレントポイントを未定義状態にする。」という演算子である。このカレントパス、カレントポイントについても節 5 で説明する。

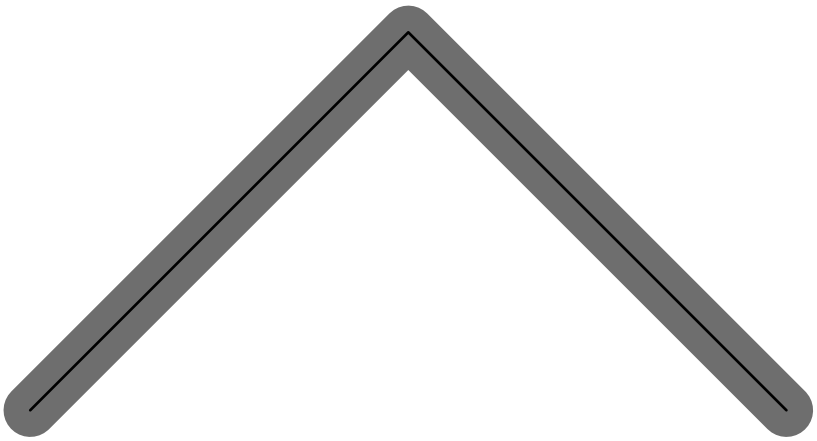

図 3: 線の端とつなぎ目

1 setlinecap、1 setlinejoin　では、図 3 のように、線の端と、つなぎめに円弧を補うように指定している。この図のような太い線を画くことは AYPLOT では想定していないので、このパラメーターはこのように固定することにした。

```
      CALL AYPEND
```

ps-プログラム 3.2

```
 S
showpage
%%EOF
```

AYPLOT を使用する FORTRAN プログラムの最後につける AYPEND は出力ファイルに締めくくりをあたえる。ps ープログラム 3.2 はこの部分の出力である。”showpage” は大事な演算子でこれがないと、プリンターは出力を行はない。

4 AYORIG

```
      CALL AYORIG(X,Y)
```

原点を移動するサブルーチンである。ポストスクリプトでは原則として、出力する紙の左下を図の原点にするようになっている。しかしグラフの目盛りを左側に画きたい時には負の価の座標も使用したほうが一

般には便利である。このサブルーチンを CALL して、原点を右上にずらしておけば、負の領域も使用できるようになる。

AYPLOT で作った ps ープログラムは、バージョン 7,8,9 の AdobeIllustrator が読みこんでくれるが、それぞれ原点の扱いが異なっている。バージョン 7 はそのまま読み込むが少し大きい図はエラーになって読んでくれない。バージョン 8 は非常に大きなスペースを用意してその中の適当な位置（大体左下）に置いてくれる。ナビゲータの窓に黒い点のように見えているのが読み込んだ図なので、そこをクリックしてまん中に移動する必要がある。バージョン 9 では図の範囲を判断して、ちょうど真ん中になるように置いてくれる。

プログラム 4.1

```
SUBROUTINE AYORIG(X,Y)
COMMON/PSP/ILLF,IUDM,XAM,YAM,XO,YO,JSTART
XA=X*(7200.0/2539.9)+XO
YA=Y*(7200.0/2539.9)+YO
IF(XA.LT.0.0.OR.XA.GT.9999.9) THEN
     WRITE(6,*) ' STOP in AYORIG(XA)',XA
     STOP
END IF
IF(YA.LT.0.0.OR.YA.GT.9999.9) THEN
     WRITE(6,*) ' STOP in AYORIG(YA)',YA
     STOP
END IF
XO=XA
YO=YA
RETURN
END
```

プログラム 4.1 にこのサブルーチンの全体を示している。仮引数 X,Y は mm を単位にして、原点の移動を指示する。このサブルーチンの中にある (7200.0/2539.9) の因子は mm をポストスクリプトの単位である「ポイント」に換算している。ここでは設定された原点が適切であるかどうかだけを判定して、それを COMMON 領域の変数 XO,YO に SAVE するだけで、出力は行っていない。なお COMMON 領域の最初の変数 ILLF は、AYPSTR が指定した出力ファイルの機番である。他の COMMON 変数については、節 5 で説明する。

5　線を画くサブルーチン

```
CALL MOVETO(X,Y)
CALL LINETO(X,Y)
```

サブルーチン MOVETO はペンの移動を行い、LINETO は線を引く。ps ープログラム 3.1 にある m こと moveto はペン移動の演算子で、l こと lineto は線を引く演算子である。単純に mm 単位の X,Y を「ポイント」px, py に換算して、それぞれに対して

```
px py m
px py l
```

のいづれかを選んで出力すればよさそうである。ところが、AdobeIllustrator は連続する moveto 演算はエラーとして処理を中断してしまう。もちろん連続する moveto 演算は最後のものだけが有効で、他のものは不要である。しかし、これは PostScript リファレンス・マニュアルがこの不要なものは無視するとしているのとは、明らかに取扱いが違っている。AdobeIllustrator の真意は分からないが、AYPLOT はこの意向にしたがって、連続する moveto 演算を避けるようにしている。サブルーチン MOVETO(X,Y) では何

も出力をしないで、COMMON 変数 XAM,YAM に SAVE するだけにする。LINETO(X,Y) が CALL されると、格納してあった XAM,YAM をつかって moveto 演算を出力し、つづいて lineto 演算を出力する。COMMON 変数 IUDM は直前のサブルーチン CALL が MOVETO か LINETO のどちらであったかを記録している。もちろん直前が LINETO であれば、moveto 演算を出力しない。

節 3.1 の newpath 演算の説明にあらわれた、カレントポイントというのは moveto 演算と lineto 演算によって運ばれてきたペンの現在位置をあらわす。いきなり lineto 演算を行うとエラーになる。カレントポイントが不定で、どこから線を引いてよいか不明であるためである。つまり、CALL LINETO(X,Y) の直前は CALL MOVETO(X,Y) か CALL LINETO(X,Y) でなければならない。カレントパスは moveto 演算と lineto 演算によって画かれた現在のペンの軌跡のことである。このカレントパスを図にする演算が S こと stroke 演算である。この演算はカレントパスを図にすると共に、newpath 演算つまり「カレントパスを空に初期化して、カレントポイントを未定義状態にする。」を実行する。AYPLOT ではサブルーチン LINETO の中で、moveto 演算を出力する直前にこれを出力する。これで、たまっていたパスを処理して新しいパスを始めることができる。AYPEND から出力される、ps-プログラム 3.2 にある S 演算は最後のパスを図にするためにつけてある。COMMON 変数 JSTART は S 演算が連続しないように監視する目的で置くことにしたが、実際にはこのことに起因する問題が生じないことが分かったので厳密にこれを行うようにはなっていない。カレントパスがないのに、S 演算があらわれることがある。

線幅を制御するサブルーチンとして

```
CALL LINEWD(WIDTH)
```

が用意されている。これが CALL された以降に作成されるパスは WIDTH(単位ポイント) で指定された線幅で画かれる。WIDTH の価の有効範囲は出力するプリンターの分解能にもよるが、非常に細い線として 0.1 位までは実用になるようである。このサブルーチンは w こと setlinewidth 演算を出力する。ps ープログラム 3.1 の最後にある 1.00 w で、デフォルトの線の幅が 1 ポイントに設定されている。

破線を画くためには

```
CALL SETDAS(IS1,IS2,IS3)
```

を使用する。例えば　CALL SETDAS(5,3,1) で 5 ポイントの線の次に 3 ポイントの空白、次に 1 ポイントの線と 3 ポイントの空白が続く 1 点鎖線を指定できる。IS3=0 の場合は IS1 の長さの線と、IS2 の長さの空白からなる破線になる。また IS2 も 0 の場合は、IS1 の長さの線と、IS1 の長さの空白からなる破線になる。IS1=0 の場合は実線を指定したことになる。このサブルーチンが CALL された以降に作成されるパスはこれで指定された線種になる。このサブルーチンは setdash 演算を出力する。図 3 の説明にあるように、線の端には線の太さを直径にした半円がついている。この分だけ破線や、 1 点鎖線の空白部分が狭くなることを想定して、SETDAS の引数を決める必要がある。

6　文字の出力

プログラム 2.1 の

```
CALL AWRITE(30,'(ABCDEFG)',9,10.0,10.0,0)
```

に対応する出力は ps-プログラム 6.1 である。

ps-プログラム 6.1

```
S
 85.04 113.39 m
```

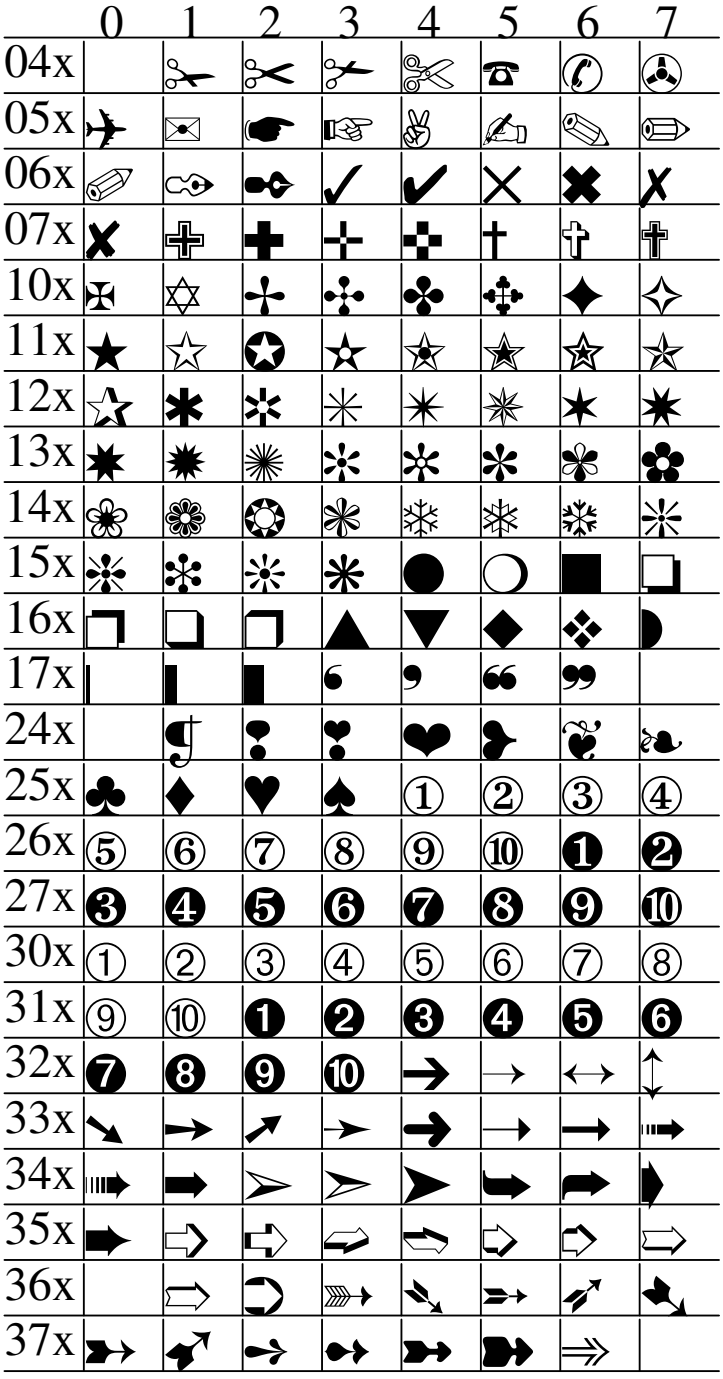

図 4: サブルーチン ZDFONT のコード表

```
/Times-Roman findfont
30 scalefont
setfont
(\(ABCDEFG\)) show
```

最初に stroke 演算でカレントパスを処理したあと、カレントポイントを文字を書く位置に移動して、findfont 演算で フォント名を指定し、scalefont 演算でフォントのサイズを指定する。setfont 演算で使用する文字のフォントが決定される。書く文字は () ではさんで示してオペレータ show を使う。これがポストスクリプトで文字を書く手続きになる。(や) が書きたい文字列にあるときは、そのまま書けば ps-プログラム 6.1 のようにサブルーチン AWRITE が処理をする。上の具体例で示したように、

```
CALL AWRITE(IPOINT,AA,NA,X,Y,IANG)
```

で文字列を出力できる。最初の IPOINT は文字のサイズを指定する。次の AA が書きたい文字列を表す文字型変数または文字型定数である。NA は書きたい文字数、X,Y は文字列の左下の座標である。図 1 には同じ座標で直線が引いてあるので、この座標が実際にどの位置を指しているかを見ることができる。最後の変数 IANG は規準線と水平方向との角度で、単位を度であたえる。

```
CALL CWRITE(IPOINT,A,X,Y)
```

は一文字だけを書きたい時に使用する。ここでは字を傾けることはできない。

```
CALL GWRITE(IPOINT,A,X,Y)
```

はギリシャ文字を書く。A はたとえば α を書きたい時には 'a' とすればよい。アルファベットとの対応で少し分かりにくいものを次にあげる。χ が 'c' に、γ が 'g' に、η が 'h' に、θ が 'q' に、ψ が 'y' に対応している。もちろん ϕ は'f' に対応している。このサブルーチンは symbol という名前のフォントを使用している。

```
CALL ZDFONT(IPOINT,A,X,Y)
```

は図 4 に示す特種文字を書くために使用する。ここで A は 3 文字で構成される文字型定数か、変数である。図 4 はこの対応表である。それぞれの文字は枠の左下によせて描かれているが、この線の交点が X,Y で指定されている座標である。この図に示されているのは ZapfDingbats フォントと呼ばれるセットで表の左端が上位桁でその x の部分に上端にある下位桁を組み合わせる。たとえば、CALL ZDFONT(10,'252',X,Y) とすればハート型の記号が１０ポイントの大きさで、X,Y の位置に画かれる。

7　数値の書き出し

```
CALL IWRITE(IPOINT,II,IW,X,Y,IANG)
```

で、整数値 II が IW の桁で出力される。

```
CALL FLWRIT(IPOINT,FD,IW,ID,X,Y,IANG)
```

で、実数値 FD が IW の文字数、小数点以下 ID 桁で出力される。

プログラム 7.1 は FLWRIT, IWRITE を用いたグラフの目盛りを出力するプログラムである。結果は図 5 である。

プログラム 7.1

```
      CALL AYPSTR(81)
      CALL AYORIG(20.0,80.0)
      CALL MOVETO(0.0,0.0)
      CALL LINETO(150.0,0.0)
      DO 1 I=0,7
         X=I*20.0
         CALL MOVETO(X,0.0)
         CALL LINETO(X,-5.0)
         E=I*0.1
         CALL FLWRIT(20,E,3,1,X-2.0,-5.0,-90)
 1    CONTINUE
      CALL AWRITE(25,'Ry',2,150.0,-5.0,-90)
      CALL AYORIG(0.0,7.0)
      CALL MOVETO(0.0,0.0)
      CALL LINETO(150.0,0.0)
      DO 2 I=0,10
         X=(I*20.0)/1.361
         CALL MOVETO(X,0.0)
         CALL LINETO(X,5.0)
         IF(I.NE.10) CALL IWRITE(20,I,2,X-2.0,Y+10.0,-90)
         IF(I.EQ.10) CALL IWRITE(20,I,2,X-2.0,Y+12.0,-90)
 2    CONTINUE
      CALL AWRITE(25,'eV',2,155.0,12.0,-90)
      CALL AYPEND
      STOP
      END
```

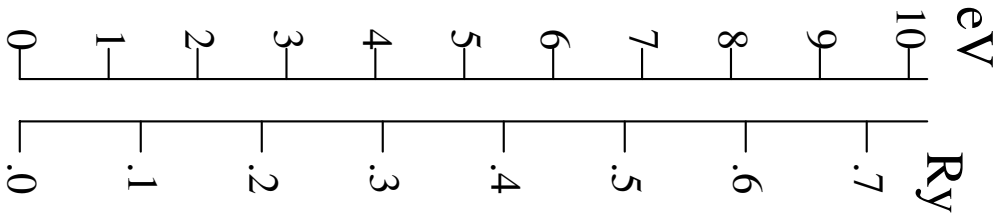

図 5: **eV と Ry の換算**

8 グラフの上に記号をつける

グラフの上にデータ点などを示すための記号を画きたいときには

```
CALL NRMARK(IPOINT,A,X,Y)
CALL ZDMARK(IPOINT,A,X,Y)
CALL CNMARK(IPOINT,I,X,Y)
```

のどれかを使えばよい。NRMARK は normal roman のマークの意味で前に出てきた CWRITE と同じプログラムで X,Y が文字の中心になるようにしたサブルーチンである。 ZDMARK も同様に ZapfDingbats フォントをマークにするもので、ZDFONT を少し変えたものである。CNMARK はサークルナンバーのマークの意味で、整数型の I に 1 から２０までの数値を与えると、対応する数字を○でかこんだ記号が画かれる。この CNMARK は２バイトコードのフォントを使っているので、数値計算に特化していて、漢字を扱わないシステムでは正常に出力されない。しかしほとんどのプリンターは漢字コードを知っているので、たとえモニターの上の図が乱れていてもプリンターに送れば正常にプリントされることが多い。

9　おわりに

以上で AYPLOT のサブルーチンはすべてである、バンド図を描く AYBAND、結晶構造を描く TSCS-DTMN の説明は完成しだいこのページに順次あげていく予定である。バンド図を描く AYBAND のソースは ayband.f としてこのページに含めてある。これは「空間群のプログラム TSPACE」の本の付属のディスクに収納した bndplt.f を AYPLOT 用に書き換えたものである。銅とニッケルの必要データと作図例もつけてあるので参照されたい。ただこのプログラムは非共型の空間群の場合にカーブの勾配を正しく表さないことがあるので、現在改良中である、