地球流体電脳ライブラリ

サンプル集

地球流体電脳倶楽部

2018 年 07 月 20 日 (DCL-7.3.3)

目次

第1章　基本描画

1.1　概要

ここでは GRPH1 の中のユーザーインターフェイスを含むパッケージ SGPACK の基本的な機能を, いくつかの簡単なプログラム例で示す. これらのプログラムは dcl-x.x/demo/grph1/sgpack の中にあるので, 参考にしていただきたい. なお, 座標系や座標変換, 各種出力プリミティブとその属性などに関する基本用語は「GRPH1」マニュアルを参照のこと.

出力例の図は縮小されている関係上, 線の太さなどの見た目が実際の出力結果と異なる場合があるので, 注意していただきたい. また, サンプルプログラム中の ! 印は行末コメントであるが, これは FORTRAN77 の規格外の機能である. 説明の都合上, 地球流体電脳ライブラリの基本方針に反して, あえて規格外の機能を使っていることをお許しいただきたい.

1.2　V 座標系で線分を描く

これは V 座標系で線分を描くプログラム例である.

sgpk01.f:page1

- **描画準備と終了:**

  最初の SGOPN（SGpack OPeN）(以下では SGpack を単に SG と書く) と最後の SGCLS（SG CLoSe）は, それぞれ初期化および終了処理をおこなうルーチンで, この 2 つの間で SGPACK の各ルーチンを呼ぶことができる．SGOPN を実行した後, 実際の描画をはじめる前に, もう一つのルーチン SGFRM（SG FRaMe）を実行しなければならない．これは作画領域を設定するもので, 1 枚の白紙を用意すると考えていただければよい．改ページをし，新たな作画領域を設定するときにもこのルーチンを使う．

  なお, SGPWSN は使用可能なデバイス名を標準出力に書き出すルーチンである．このプログラム例のように書いておくと, 普段使い慣れていないシステムでコンパイル・実行するときにも, デバイス名のリストが出力されるので便利である．

- **線分描画 (V 座標系):**

  まず，描画可能な範囲がどのように設定されているかについて簡単に述べておく．作画例のコーナーマークで示される長方形領域は，ふつう出力画面あるいは出力用紙の四隅に描かれ，その出力装置の最大作画領域を示す．しかし，一般にこの長方形領域の縦・横比は 1 でないことが多いので，標準的なオプション設定のもとでは，コーナーマークで示される長方形領域に最大内接するような正方形領域を [0,1]×[0,1] とするように描画範囲が設定される (ここで単位は R 座標系であるが，透視変換を用いていなければ V 座標系と等しい)．最大作画領域をいっぱいに使うことも可能で，そのような例については第 2 節を参照されたい．

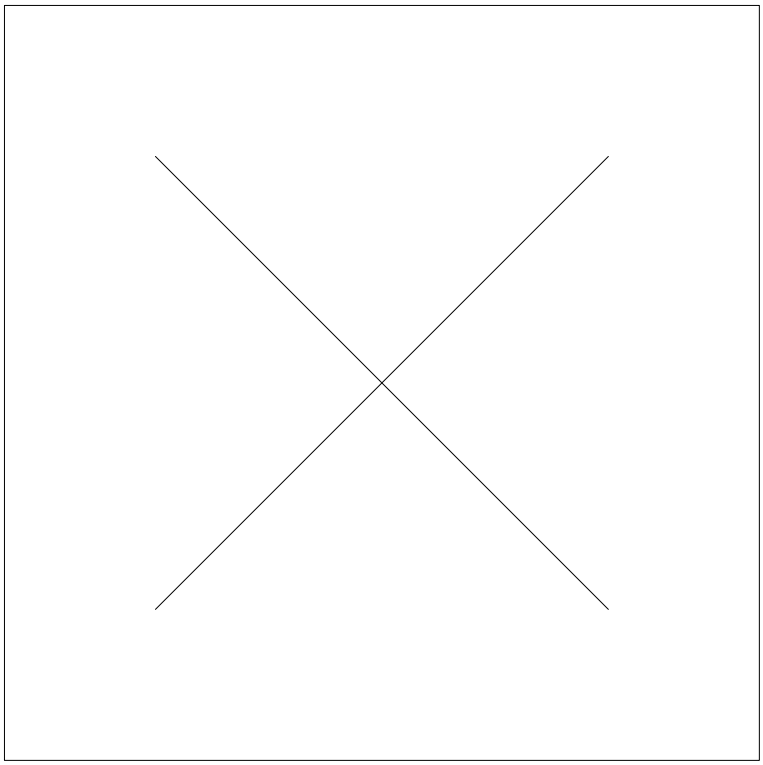
  さて，実際の線分を描画するのが SGLNV（SG LiNe V）で, これは根回し型ルーチンである．つまり線分の色と太さを指定するラインインデクスは SGSLNI（SG Set LiNe Index）によって設定する．SGLNV のルーチンの上意下達型が SGLNZV（SG LiNe ZV）である．ラインインデクスも同時に指定する点が異なるだけで SGLNV と全く同じ働きをする．作画例のうち，上側の 4 本が SGLNV による出力, 下側の 4 本が SGLNZV による出力であるが, 同じ結果が得られていることがわかる．

  なお, ラインインデクスは 3 桁の整数 (nnm) で, 上位 2 桁 (nn) が色番号, 下位 1 桁 (m) が線の太さである．ただし, どんなディバイスでも色と太さの両方が変えられるとは限らないので，色または太さしか変えられないようなディバイスでは色と太さのインデクスを読みかえる．たとえば，太さの変えられないようなディバイスで 1 桁の数字 (太さ) を指定していも, それは色のインデクスとして読みかえられ線分の色が変わる．詳しくは「GRPH1」マニュアルを参照されたい．

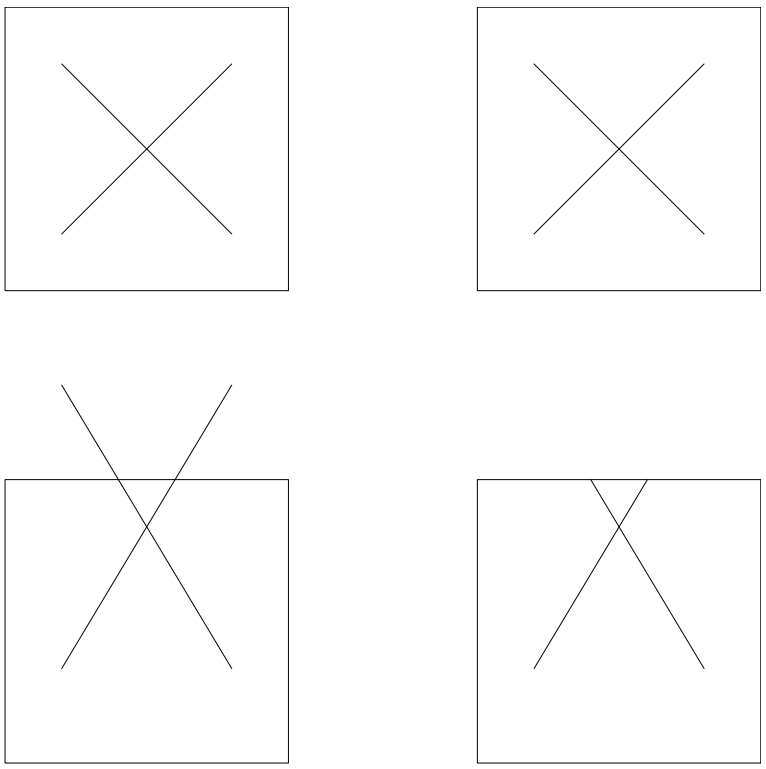
```
1  *---------------------------------------------------------------------
2        PROGRAM SGPK01
3        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
4        CALL SGPWSN                  ! <-- 使用可能なデバイス名一覧を出力
5        READ(*,*) IWS
6        CALL SGOPN(IWS)              ! <-- デバイスをオープンする.
7        CALL SGFRM                   ! <-- ページを用意する.
8        X1 = 0.1
9        X2 = 0.9
10       CALL SGLNV(X1, 0.9, X2, 0.9) ! <-- デフォルトの LINE INDEX で直線描画
11       CALL SGSLNI(2)               ! <-- LINE INDEX を 2 に設定
12       CALL SGLNV(X1, 0.8, X2, 0.8)
13       CALL SGSLNI(3)
14       CALL SGLNV(X1, 0.7, X2, 0.7)
15       CALL SGSLNI(4)
16       CALL SGLNV(X1, 0.6, X2, 0.6)
17       CALL SGLNZV(X1, 0.4, X2, 0.4, 1 ) !  <-- LINE INDEX を指定して描画
18       CALL SGLNZV(X1, 0.3, X2, 0.3, 2 )
19       CALL SGLNZV(X1, 0.2, X2, 0.2, 3 )
20       CALL SGLNZV(X1, 0.1, X2, 0.1, 4 )
21       CALL SGCLS
22       END
```

1.3 U 座標系で線分を描く

次に U 座標系で線分を描いてみよう.



sgpk02.f:page1



sgpk02.f:page2

- **ウインドウとビューポート:**

  U 座標系で線分を描くには，SGFRM を実行したあとで変換関数を定義する以下の 4 つのルーチンを呼ぶ: SGSWND (SG Set WiNDow), SGSVPT (SG Set ViewPorT), SGSTRN (SG Set TRansformation Number), SGSTRF (SG Set Transformation Function)．このうち最初の 3 つを呼ぶ順番は任意であるが，SGSTRF は必ず最後に呼ばなければならない．ここでは U 座標系の (0–1) × (0–1) の空間 (ウインドウ) を V 座標系の (0–1) × (0–1) の空間 (ビューポート) に投影するように変換関数を定義しているので, 恒等変換となり, V 座標系と U 座標系は等しくなる．(実は, SGFRM を実行した段階で, この恒等変換が定義されているので, 作画例の 1 ページ目については，SGSWND などによって変換関数を改めて指定しなくても同じ結果になる.) ここで, ウインドウとは U 座標系上に描く図形のうち我々が実際に見たい領域であり, ビューポートはその見たい領域を投影する V 座標系上の領域である．このウインドウとビューポートという言葉は今後よくでてくるのでよく覚えておいてほしい．

  また, SGSTRN の引数は変換関数番号で, 1 ならば線形座標, 4 ならば両対数座標というように座標系のタイプを指定する．変換関数に関する以上の情報は SGSTRF を呼んではじめて有効になる．そのあとに呼んでいる SLPVPR はビューポートの枠を描く SLPACK のサブルーチンである (GRPH1/SLPACK 参照).

- **線分描画 (U 座標系):**

  前出の SGSLNI によりラインインデクスを設定して, SGLNU と SGLNZU を使って実際の線分を描画

する (このほかに R 座標系で作画するルーチンもあるが, ここでは述べない). それぞれ, 根回し型と上意下達型のルーチンである. ここで, SGLNZU で指定したラインインデクスは, SGLNU のラインインデクスへ影響を及ぼさないことに注意していただきたい.

- いろいろなウインドウ / ビューポート:

  SGFRM により改ページをして, 新しいページの左上に前のページと同じ図形を描いてみる. ウインドウとビューポートの設定に関しては, SGSVPT によるビューポートの指定のみが前のページのものと異なる. ウインドウ, ビューポート, 変換関数番号に関する情報はすべて SGPACK の内部変数管理ルーチン SGpGET/SGpSET (SG Parameter GET/SET) が保持している. (以下でも同様の表記をするが, 一般に p は R/I/L/C のどれかで, それぞれ, 実数, 整数, 論理変数, 文字変数用のルーチンに対応することをあらわす.) したがって, 変更部分のみを指定した上で, SGSTRF によって変換関数を確定してやればよい (その意味では, この部分に関する SGSWND の指定も実際にはいらない). その結果, 左上に前のページと同じ図形が縮小されて描かれる.

  その右側には SGSWND のウインドウ指定で, UYMIN>UYMAX とし上下逆さまのウインドウを表示してみた.

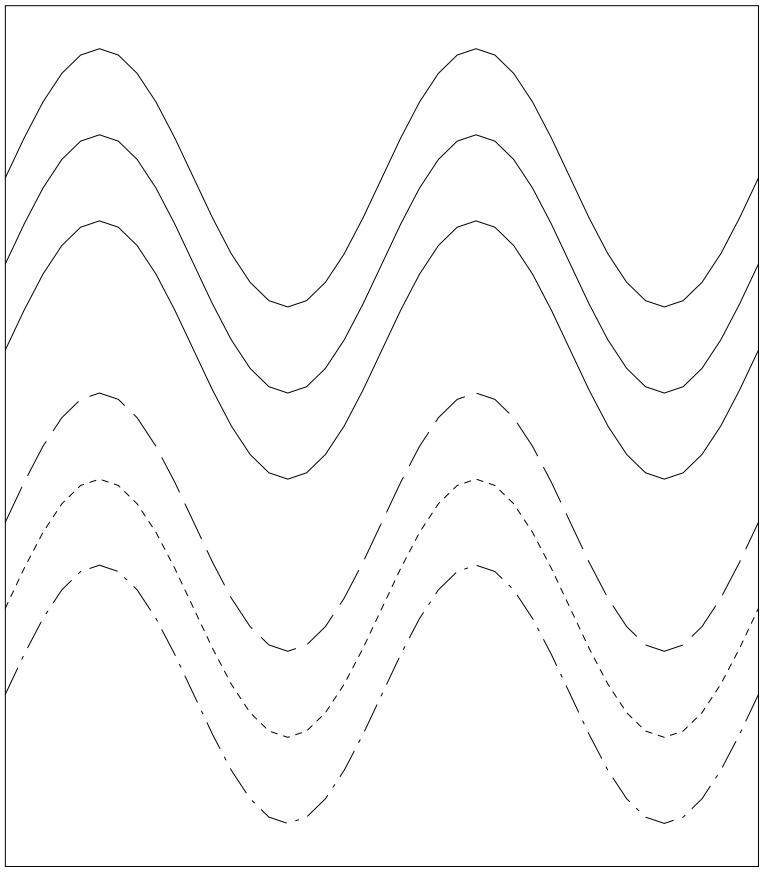
- クリッピング:

  ウインドウとは U 座標系上の我々が見たい部分であると書いたが, SGSWND で小さなウインドウを指定して図形の一部だけ表示させようとしても, これだけでは作画しようとする図形がビューポートをはみ出してしまう. ビューポートをはみ出す部分を描かないようにするためには SGLSET によって, クリッピングを制御する内部変数'LCLIP' を.TRUE. にしてやればよい.

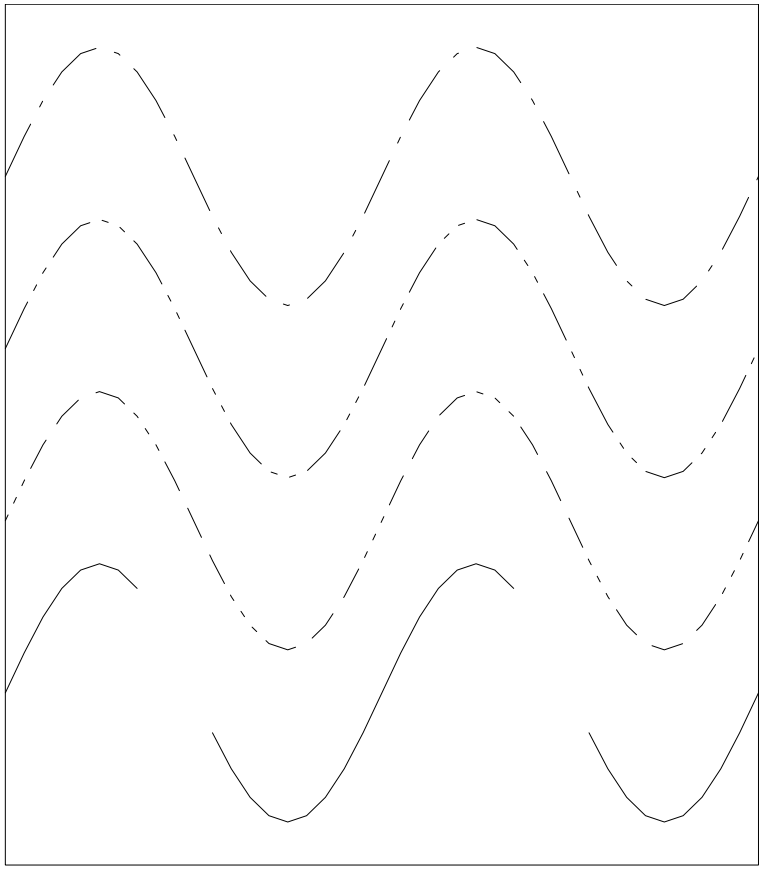
```
1   *-----------------------------------------------------------------------
2         PROGRAM SGPK02
3         CALL SWCSTX('FNAME','SGPK02')
4         CALL SWLSTX('LSEP',.TRUE.)
5         WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
6         CALL SGPWSN
7         READ(*,*) IWS
8         CALL SGOPN(IWS)
9   *---------------------------- page 1 ---------------------------------
10        CALL SGFRM
11        CALL SGSLNI(3)
12  *                 XMIN, XMAX, YMIN, YMAX
13        CALL SGSWND(  0.0,  1.0,  0.0,  1.0)   !  <--+
14        CALL SGSVPT(  0.0,  1.0,  0.0,  1.0)   !     | 変換関数の設定
15        CALL SGSTRN(1)                         !     |
16        CALL SGSTRF                            !  <--+
17        CALL SLPVPR(1)                         !  <--- 枠を描く
18        CALL SGLNZU( 0.2,  0.8,  0.8,  0.2, 2) !  <--- X 印を描く
19        CALL SGLNU ( 0.2,  0.2,  0.8,  0.8)    !  <-|
20  *---------------------------- page 2 ---------------------------------
21        CALL SGFRM
22  *                 XMIN, XMAX, YMIN, YMAX
23        CALL SGSWND(  0.0,  1.0,  0.0,  1.0)   !  <-- 正立
24        CALL SGSVPT(  0.1,  0.4,  0.6,  0.9)   !  <-- 左上
25        CALL SGSTRF
26        CALL SLPVPR(1)
27        CALL SGLNZU( 0.2,  0.8,  0.8,  0.2, 2)
28        CALL SGLNU ( 0.2,  0.2,  0.8,  0.8)
29  *   ----------------------------------------
30  *                 XMIN, XMAX, YMIN, YMAX
31        CALL SGSWND(  0.0,  1.0,  1.0,  0.0)   !  <-- 上下逆
32        CALL SGSVPT(  0.6,  0.9,  0.6,  0.9)   !  <-- 右上
33        CALL SGSTRF
34        CALL SLPVPR(1)
35        CALL SGLNZU( 0.2,  0.8,  0.8,  0.2, 2)
36        CALL SGLNU ( 0.2,  0.2,  0.8,  0.8)
37  *   ----------------------------------------
38  *                 XMIN, XMAX, YMIN, YMAX
39        CALL SGSWND(  0.0,  1.0,  0.0,  0.6)   !  <-- 小さなウインド
40        CALL SGSVPT(  0.1,  0.4,  0.1,  0.4)   !  <-- 左下
41        CALL SGSTRF
42        CALL SLPVPR(1)
43        CALL SGLNZU( 0.2,  0.8,  0.8,  0.2, 2)
44        CALL SGLNU ( 0.2,  0.2,  0.8,  0.8)
45  *   ----------------------------------------
46  *                 XMIN, XMAX, YMIN, YMAX
47        CALL SGSWND(  0.0,  1.0,  0.0,  0.6)   !  <-- 小さなウインド
48        CALL SGSVPT(  0.6,  0.9,  0.1,  0.4)   !  <-- 右下
49        CALL SGSTRF
50        CALL SLPVPR(1)
51        CALL SGLSET('LCLIP', .TRUE.)           !  <-- クリッピングの指定
52        CALL SGLNZU( 0.2,  0.8,  0.8,  0.2, 2)
53        CALL SGLNU ( 0.2,  0.2,  0.8,  0.8)
54        CALL SGCLS
55        END
```

1.4 折れ線を描く

これは折れ線ルーチンの使用例である. 折れ線ルーチンも例によって 4 つ (R 座標系での作画ルーチンも含めると 6 つ) のタイプがあるが, ここでは, U 座標系で描画する根回し型ルーチンのみを取り上げる.



sgpk03.f:page1



sgpk03.f:page2

- **折れ線描画 / ラインインデクスとラインタイプ:**

  折れ線を描くルーチンは SGPLU (SG PolyLine U) である. 折れ線ルーチンでは線分属性としてラインインデクスの他に, 実線, 点線等といったラインタイプが指定できる. これらを変更するルーチンは, それぞれ SGSPLI と SGSPLT である. 1 ページ目にはいろいろなラインインデクスとラインタイプの例を示してある.

- **サイクル長を変える:**

  SGPLU で描く点線や破線のパターンの 1 サイクルの長さは V 座標系で指定されている. したがって, 小さなビューポートで描画するとビューポートの大きさに比べてパターンが長すぎるため, 線分部分と空白部分のバランスが悪くなることがある. それとは逆に, ディスプレイ等に点線や破線を表示した場合, ディスプレイの分解能が悪いと, パターンが潰れて判別できなくなることもある. そのようなときには SGRSET で内部変数 'BITLEN' を変更すると, パターンのサイクル長を変えることができる.

- **パターンを変える:**

  GRPH1 では実線, 破線, 点線, 一点鎖線の 4 種類の線種を指定できるが, これ以外のパターンの線種も指定することができる. 実は, SGPLT の引数に 4 より大きい数値を指定すると, その整数の 2 進表現のパターンを指定したものと見なされる. すなわち, 整数の下位 16 bit のうち 1 の部分に線を描き, 0 の部分は空白となるようなパターンを設定する. このパターンを表わす整数を求めるに

は MISC1/BITLIB の中の BITPCI を使うと便利である.

さらに複雑なパターンを指定したいときには, SGISET で内部変数'NBITS' を変更することにより, パターンのビット長を 32 bit まで長くすることができる.

- **欠損値処理:**

  データの中に欠損がある場合には, MATH1/SYSLIB の中の GL*p*SET (GLobal Parameter SET) で欠損値処理の指定に関する内部変数 'LMISS' を .TRUE. にして, 欠損値処理を有効にする. このように指定すると, 座標成分 (x,y) のどちらかが 999. である点を欠損値と見なして線を結ばない. また, 有効な点が欠損値に囲まれ, 一点だけ孤立している様な場合には, 有効な点を線で結ぶことはできないので, そこには何も描かれない. 欠損値 999. の値が実際のデータ範囲に入っていて, 999. 以外の値にしたい場合には, GLRSET によって実数の欠損値をあらわす内部変数'RMISS' を変更してやればよい.

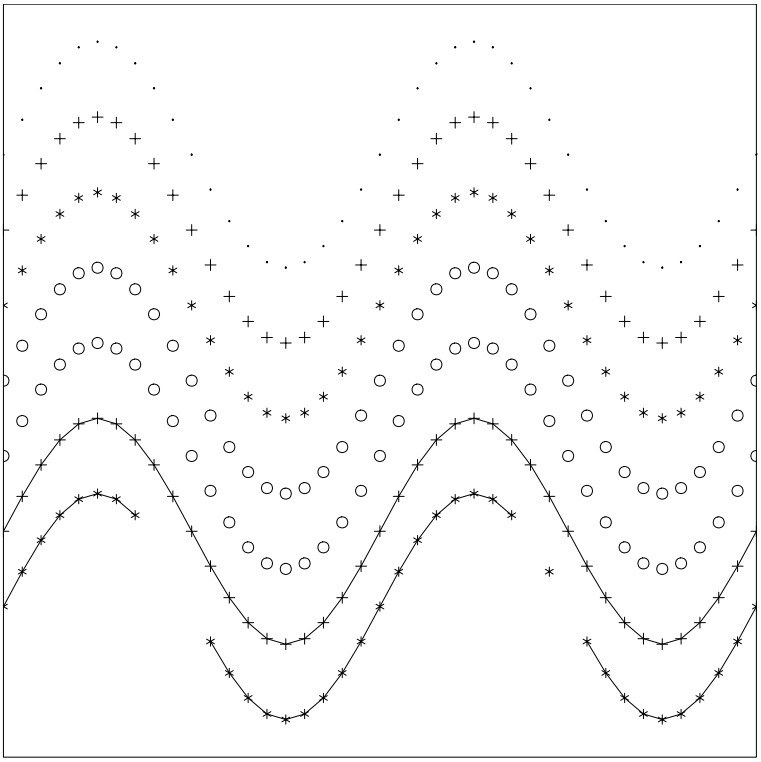
  欠損値処理の制御は, SGPACK だけでなくプログラム全体で統一的に行われなければならないため, 欠損値処理に関する内部変数の管理は SG*p*GET/SG*p*SET ではなく GL*p*GET/GL*p*SET で行われてる.

```
*---------------------------------------------------------------------
      PROGRAM SGPK03
      PARAMETER(N=41)
      DIMENSION X(N), Y(N)
      DT = 4.* 3.14159 / (N-1)
      DO 10 I=1,N
        Y(I) = SIN(DT*(I-1))*0.15
        X(I) = REAL(I-1)/REAL(N-1)
   10 CONTINUE
      CALL SWCSTX('FNAME','SGPK03')
      CALL SWLSTX('LSEP',.TRUE.)
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN(IWS)
*----------------------------- page 1 --------------------------------
      CALL SGFRM
      CALL SGSWND( 0.0,  1.0, -0.8, 0.2)
      CALL SGSVPT( 0.15, 0.85, 0.1, 0.9)
      CALL SGSTRN(1)
      CALL SGSTRF
      CALL SLPVPR(1)
      CALL SGPLU(N, X, Y)                    ! <-- 1 本目
      CALL SGSWND( 0.0,  1.0, -0.7, 0.3)
      CALL SGSTRF
      CALL SGSPLI(2)                         ! <-- Line INDEX 設定
      CALL SGPLU(N, X, Y)                    ! <-- 2 本目
      CALL SGSWND( 0.0,  1.0, -0.6, 0.4)
      CALL SGSTRF
      CALL SGSPLI(3)                         ! <-- Line INDEX 設定
      CALL SGPLU(N, X, Y)                    ! <-- 3 本目
      CALL SGSWND( 0.0,  1.0, -0.4, 0.6)
      CALL SGSTRF
      CALL SGSPLT(2)                         ! <-- Line TYPE 設定 (破線)
      CALL SGPLU(N, X, Y)                    ! <-- 4 本目
      CALL SGSWND( 0.0,  1.0, -0.3, 0.7)
      CALL SGSTRF
      CALL SGSPLT(3)                         ! <-- Line TYPE 設定 (点線)
      CALL SGPLU(N, X, Y)                    ! <-- 5 本目
      CALL SGSWND( 0.0,  1.0, -0.2, 0.8)
      CALL SGSTRF
      CALL SGSPLT(4)                         ! <-- Line TYPE 設定 (一点鎖線)
      CALL SGPLU(N, X, Y)                    ! <-- 6 本目
*----------------------------- page 2 --------------------------------
      CALL SGFRM
      CALL SGSWND( 0.0,  1.0, -0.8, 0.2)
```

```
47          CALL SGSVPT( 0.15, 0.85, 0.1, 0.9)
48          CALL SGSTRN(1)
49          CALL SGSTRF
50          CALL SLPVPR(1)
51          CALL SGRSET('BITLEN', 0.006)          ! <-- サイクル長の変更
52          CALL SGSPLT(4)
53          CALL SGPLU(N, X, Y)                   ! <-- 1 本目
54          CALL SGSWND( 0.0,  1.0, -0.6, 0.4)
55          CALL SGSTRF
56          CALL SGSPLI(2)
57          CALL BITPCI('1111111100100100', ITYPE)! <-- パターン生成
58          CALL SGSPLT(ITYPE)                    ! <-- パターン設定
59          CALL SGPLU(N, X, Y)                   ! <-- 2 本目
60          CALL SGSWND( 0.0,  1.0, -0.4, 0.6)
61          CALL SGSTRF
62          CALL SGISET('NBITS', 32)
63          CALL BITPCI('10010010011111000111110001111100', ITYPE)
64          CALL SGSPLT(ITYPE)
65          CALL SGPLU(N, X, Y)
66          CALL SGSWND( 0.0,  1.0, -0.2, 0.8)
67          CALL SGSTRF
68          CALL SGSPLI(1)
69          CALL SGSPLT(1)
70          N1=N/4
71          Y(N1-1) = 999.                        ! <-- 欠損値
72          Y(N1  ) = 999.
73          Y(N1+1) = 999.
74          N2=N1*3
75          Y(N2-1) = 999.
76          Y(N2+1) = 999.
77          CALL GLLSET('LMISS',.TRUE.)
78          CALL SGPLU(N, X, Y)                   ! <-- 4 本目
79          CALL SGCLS
80          END
```

1.5 マーカー描画

これはマーカーを描くプログラムである．ここでも U 座標系で描画する根回し型ルーチンのみを取り上げるが, V 座標系での作画ルーチンなども揃っている．



sgpk04.f:page1

- **いろいろなマーカー:**

  指定された点列を折れ線で結ぶ代わりに, 点の位置にマーカーを描くのが SGPMU（SG PolyMarker U）である．マーカーの種類は SGSPMT（SG Set PolyMarker Type） でフォントテーブルのフォント番号を指定する．マーカーの属性としてはマーカーの種類以外に, マーカーの大きさ, ラインインデクスがあり, それぞれ SGSPMS（SG Set PolyMarker Size), SGSPMI（SG Set PolyMarker Index）で指定する．また, SGLSET で内部変数 'LCLIP' を .TRUE. としクリッピングをするように指定すると, マーカーもクリッピングされる．

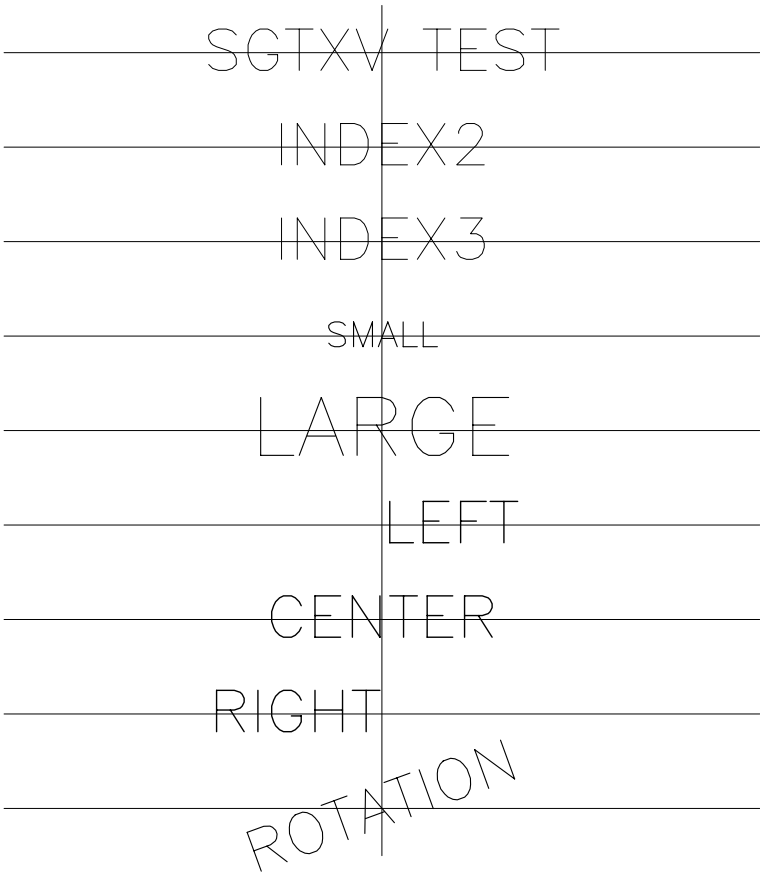
- **折れ線と重ねて描く:**

  このポリマーカー描画ルーチンは折れ線と同時に使用することが多い．折れ線と同じ様に欠損値の処理もできる．SGPLU では何も描かれなかった欠損値に囲まれた孤立点も, SGPMU では表示される．

```
*-----------------------------------------------------------------------
      PROGRAM SGPK04
      PARAMETER(N=41)
      DIMENSION X(N), Y(N)
      DT = 4.* 3.14159 / (N-1)
      DO 10 I=1,N
        Y(I) = SIN(DT*(I-1))*0.15
        X(I) = REAL(I-1)/REAL(N-1)
   10 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN(IWS)
      CALL SGFRM
      CALL SGSWND( 0.0, 1.0, -0.8, 0.2)
      CALL SGSVPT( 0.1, 0.9,  0.1, 0.9)
      CALL SGSTRN( 1)
      CALL SGSTRF
      CALL SGLSET('LCLIP', .TRUE.)
      CALL SLPVPR(1)
      CALL SGPMU(N, X, Y)                    ! <-- マーカー描画 (1 列目)
      CALL SGSWND( 0.0, 1.0, -0.7, 0.3)
      CALL SGSTRF
      CALL SGSPMT(2)                         ! <-- マーカー TYPE 設定 (+)
      CALL SGPMU(N, X, Y)                    ! <-- マーカー描画 (2 列目)
      CALL SGSWND( 0.0, 1.0, -0.6, 0.4)
      CALL SGSTRF
      CALL SGSPMT(3)                         ! <-- マーカー TYPE 設定 (*)
      CALL SGPMU(N, X, Y)                    ! <-- マーカー描画 (3 列目)
      CALL SGSWND( 0.0, 1.0, -0.5, 0.5)
      CALL SGSTRF
      CALL SGSPMT(4)                         ! <-- マーカー TYPE 設定 (o)
      CALL SGPMU(N, X, Y)                    ! <-- マーカー描画 (4 列目)
      CALL SGSWND( 0.0, 1.0, -0.4, 0.6)
      CALL SGSTRF
      CALL SGSPMI(2)                         ! <-- マーカー INDEX 設定
      CALL SGPMU(N, X, Y)                    ! <-- マーカー描画 (5 列目)
      CALL SGSWND( 0.0, 1.0, -0.3, 0.7)
      CALL SGSTRF
      CALL SGSPMI(3)
      CALL SGSPMT(2)
      CALL SGPLU(N, X, Y)                    ! <-- 折れ線描画
      CALL SGPMU(N, X, Y)                    ! <-- マーカー描画 (6 列目)
      N1=N/4
      Y(N1-1) = 999.                         ! <-- 欠損値
      Y(N1  ) = 999.
      Y(N1+1) = 999.
      N2=N1*3
      Y(N2-1) = 999.
      Y(N2+1) = 999.
      CALL SGSWND( 0.0, 1.0, -0.2, 0.8)
      CALL SGSTRF
      CALL GLLSET('LMISS',.TRUE.)
      CALL SGSPMI(2)
      CALL SGSPMT(3)
      CALL SGPLU(N, X, Y)                    ! <-- 折れ線描画
      CALL SGPMU(N, X, Y)                    ! <-- マーカー描画 (7 列目)
      CALL SGCLS
      END
```

1.6 V 座標系で文字を書く

ここでは文字を書く例を示す．文字の位置は通常「文字列の中心」の座標で指定する．指定した文字の座標と，実際に描画される文字の位置関係をわかりやすくするために，まず補助線を引いてその交点の座標を指定して文字を出力する．



sgpk05.f:page1

- **文字の出力 (V 座標系):**

  V 座標系で文字を出力するには `SGTXV (SG TeXt V)` を使う．これは根回し型のルーチンで，線分描画ルーチンと同じ様に上意下達型の `SGTXZV` もあるが，機能は全く同じなのでここでは省略する．

  まず，ここで注意すべきことは，文字列の中心が指定した座標になるように出力されることである．これは後述のパラメータで変更可能である．また，文字の高さはデフォルトで 0.05 であり，文字のピッチもそれに等しい．

  細かいことであるが，ここでいう文字の高さとは，すべての文字を入れることができる正方形の高さである．つまり，小文字の g 等の様にベースラインよりも下に伸びる文字もあるので，大文字を

書いた場合には下に少し余白が残ることになる. そのために, 実際の文字の高さは大文字では 0.05 よりも少し (12-13 %) 小さく, また, 文字の中心 (文字位置を指定する座標) も大文字の実際の中心よりも心持ち下になる.

- **文字の属性を変える:**

  文字のインデクスを変えるには SGSTXI (SG Set TeXt Index) を使う. このインデクスは線分のインデクスには影響されず, 独立に管理されていることに注意されたい.

  また, 文字の大きさは SGSTXS (SG Set TeXt Size) によって変えることができる.

- **センタリングオプション:**

  初期状態では SGTXV で指定する位置は, 文字列の中心の座標であるが, 文字列の左端の座標や, 右端の座標で指定することもできる. この指定を変えるには, SGSTXC (SG Set TeXt Centering option) を使う.

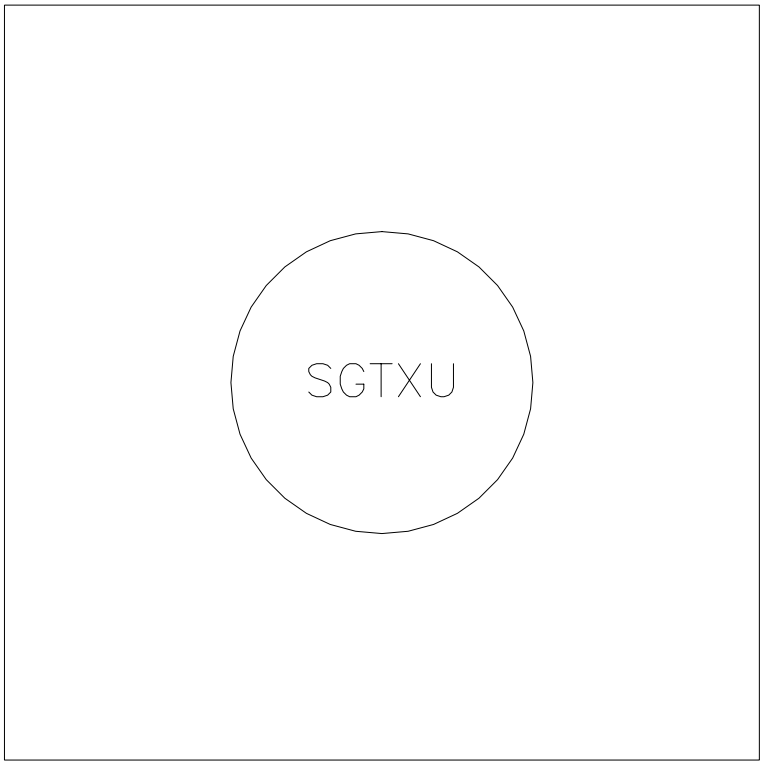
- **文字の回転:**

  文字列を回転させるためには SGSTXR (SG Set TeXt Rotation) を用いて指定した位置を中心に反時計回りの角度を指定する.

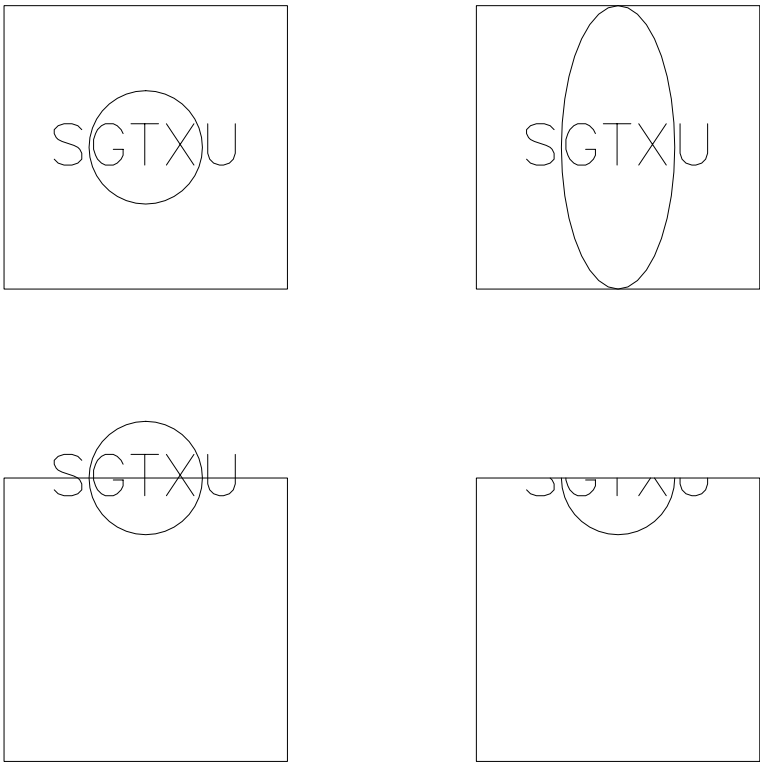
```
1   *-----------------------------------------------------------------------
2         PROGRAM SGPK05
3         PARAMETER (N=9)
4         DIMENSION Y(N)
5         WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
6         CALL SGPWSN
7         READ(*,*) IWS
8         CALL SGOPN(IWS)
9         CALL SGFRM
10        X1 = 0.1
11        X2 = 0.9
12        XC = 0.5
13        CALL SGSLNI(1)
14        DO 10 I=1, N
15          Y(I) = 0.1 * (10-I)
16          CALL SGLNV(X1, Y(I), X2, Y(I))
17    10  CONTINUE
18        CALL SGLNV(XC, 0.05, XC, 0.95)
19        CALL SGTXV(XC, Y(1), 'SGTXV TEST') ! <-- テキスト 1 段目
20        CALL SGSTXI(2)                     ! <-- テキスト INDEX 設定
21        CALL SGTXV(XC, Y(2), 'INDEX2')     ! <-- テキスト 2 段目
22        CALL SGSTXI(3)                     ! <-- テキスト INDEX 設定
23        CALL SGTXV(XC, Y(3), 'INDEX3')     ! <-- テキスト 3 段目
24        CALL SGSTXI(4)
25        CALL SGSTXS(0.03)                  ! <-- テキスト SIZE 設定 (小)
26        CALL SGTXV(XC, Y(4), 'SMALL')      ! <-- テキスト 4 段目
27        CALL SGSTXS(0.07)                  ! <-- テキスト SIZE 設定 (大)
28        CALL SGTXV(XC, Y(5), 'LARGE')      ! <-- テキスト 5 段目
29        CALL SGSTXS(0.05)
30        CALL SGSTXI(5)
31        CALL SGSTXC(-1)                    ! <-- 左揃え
32        CALL SGTXV(XC, Y(6), 'LEFT')       ! <-- テキスト 6 段目
33        CALL SGSTXC(0)                     ! <-- 中央揃え
34        CALL SGTXV(XC, Y(7), 'CENTER')     ! <-- テキスト 7 段目
35        CALL SGSTXC(1)                     ! <-- 右揃え
36        CALL SGTXV(XC, Y(8), 'RIGHT')      ! <-- テキスト 8 段目
37        CALL SGSTXC(0)
38        CALL SGSTXI(4)
39        CALL SGSTXR(20)                    ! <-- テキスト回転
40        CALL SGTXV(XC, Y(9), 'ROTATION')   ! <-- テキスト 9 段目
41        CALL SGCLS
42        END
```

1.7 U 座標系で文字を書く

これは U 座標系で文字を書くプログラムである. U 座標系での文字の描画も基本的には V 座標系空間での描画と同じであるが, 文字の大きさは V 座標系で指定するので, ビューポートの大きさが変わっても文字の大きさは変わらないという点が, 線分描画の場合と異なる.



sgpk06.f:page1



sgpk06.f:page2

- **文字の出力 (U 座標系):**

  U 座標系空間での文字の描画には SGTXU (SG TeXt U) を使う. 1 ページ目には線画図形との比較のため SGPLU で描いた円の中に文字を出力してある.

- **ビューポート / ウインドウの変更:**

  2 ページ目の左上に前のページと同じ図形を, ビューポートだけを小さくして描画した. ビューポートが小さくなったのに応じて SGPLU で描いた円は小さくなるが, 文字の大きさは 1 ページ目と変わらない.

  ウインドウを変えて図形の縦横比を変えた場合 (右上) でも文字は不変である.

- **クリッピング:**

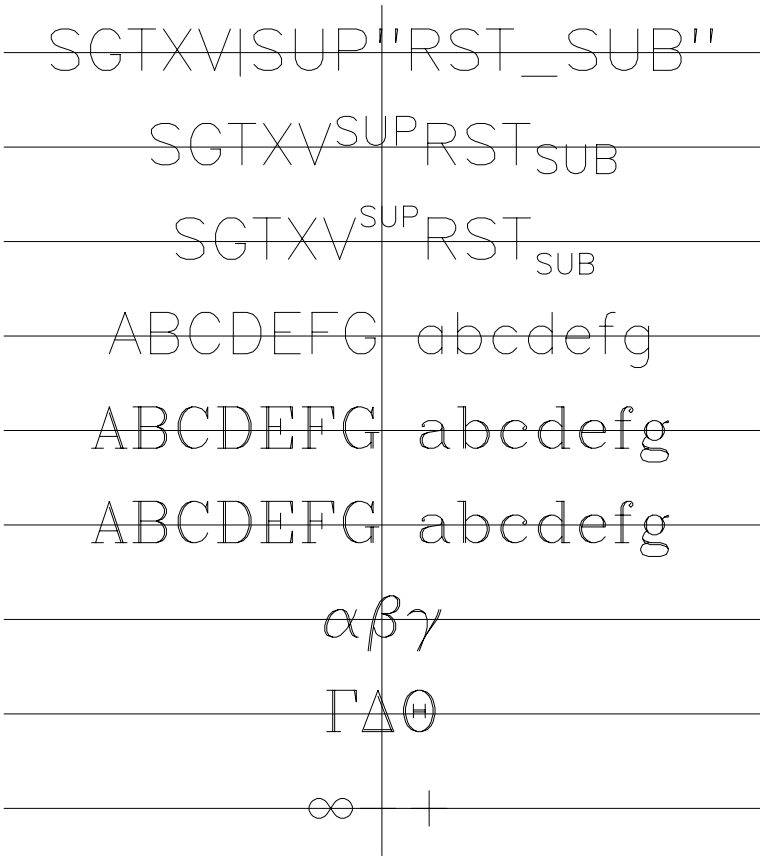
  クリッピングに関して, 文字は線画図形と同じで, SGLSET を用い内部変数'LCLIP' を .TRUE. とした指定が文字に対しても適用される.

```
*-----------------------------------------------------------------------
      PROGRAM SGPK06
      PARAMETER(N=37)
      DIMENSION X(N), Y(N)
      DT = 2.* 3.14159 / (N-1)
      R = 0.2
      DO 10 I=1, N
        X(I) = R*SIN(DT*(I-1)) + 0.5
        Y(I) = R*COS(DT*(I-1)) + 0.5
   10 CONTINUE
      XC = 0.5
      YC = 0.5
      CALL SWCSTX('FNAME','SGPK06')
      CALL SWLSTX('LSEP',.TRUE.)
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
```

```
16        CALL SGPWSN
17        READ(*,*) IWS
18        CALL SGOPN(IWS)
19  *----------------------------- page 1 ---------------------------------
20        CALL SGFRM
21        CALL SGSTXI(3)
22        CALL SGSPLI(2)
23  *                 XMIN, XMAX, YMIN, YMAX
24        CALL SGSWND( 0.,   1.,   0.,   1.)
25        CALL SGSVPT( 0.,   1.,   0.,   1.)
26        CALL SGSTRN( 1)
27        CALL SGSTRF
28        CALL SLPVPR(1)
29        CALL SGPLU(N, X, Y)                       ! <-- 円描画
30        CALL SGTXU(XC, YC, 'SGTXU')               ! <-- テキスト
31  *----------------------------- page 2 ---------------------------------
32        CALL SGFRM
33  *                  XMIN, XMAX, YMIN, YMAX
34        CALL SGSWND( 0.0,  1.0,  0.0,  1.0)       ! <-- 1x1
35        CALL SGSVPT( 0.1,  0.4,  0.6,  0.9)       ! <-- 小さな view port
36        CALL SGSTRN( 1)
37        CALL SGSTRF
38        CALL SLPVPR(1)
39        CALL SGPLU(N, X, Y)                       ! <-- 円描画
40        CALL SGTXU(XC, YC, 'SGTXU')               ! <-- テキスト
41  *                 XMIN, XMAX, YMIN, YMAX
42        CALL SGSWND( 0.0,  1.0,  0.3,  0.7)       ! <-- ゆがんだ window
43        CALL SGSVPT( 0.6,  0.9,  0.6,  0.9)
44        CALL SGSTRN( 1)
45        CALL SGSTRF
46        CALL SLPVPR(1)
47        CALL SGPLU(N, X, Y)                       ! <-- 円描画
48        CALL SGTXU(XC, YC, 'SGTXU')               ! <-- テキスト
49  *                 XMIN, XMAX, YMIN, YMAX
50        CALL SGSWND( 0.0,  1.0, -0.5,  0.5)       ! <-- はみ出し window
51        CALL SGSVPT( 0.1,  0.4,  0.1,  0.4)
52        CALL SGSTRN( 1)
53        CALL SGSTRF
54        CALL SLPVPR(1)
55        CALL SGPLU(N, X, Y)                       ! <-- 円描画
56        CALL SGTXU(XC, YC, 'SGTXU')               ! <-- テキスト
57        CALL SGLSET('LCLIP', .TRUE.)              ! <-- クリッピング
58  *                 XMIN, XMAX, YMIN, YMAX
59        CALL SGSWND( 0.0,  1.0, -0.5,  0.5)
60        CALL SGSVPT( 0.6,  0.9,  0.1,  0.4)
61        CALL SGSTRN( 1)
62        CALL SGSTRF
63        CALL SLPVPR(1)
64        CALL SGPLU(N, X, Y)                       ! <-- 円描画
65        CALL SGTXU(XC, YC, 'SGTXU')               ! <-- テキスト
66        CALL SGCLS
67        END
```

1.8 添字とギリシャ文字

これは GRPH1 で出力できるいろいろな文字の例である.



sgpk07.f:page1

- **添字:**

  GRPH1 においては | " _ の 3 文字は添字の制御に使われる. 何も指定しなければ, これらの文字も他の文字と同様に文字として出力されるが, SGLSET で内部変数'LCNTL' を .TRUE. にすると上付, 下付の添字を出力することができる. ここで注意すべきことは, 文字列が添字で終る場合でも, 添字の後に必ず" を入れて通常の文字モードに戻すことである. こうしないと, 文字列の長さが正確に求まらず, センタリングや右よせ等の処理がうまくいかない.

  また, 添字の大きさや上下の移動量は, それぞれ SGRSET で内部変数'SMALL', 'SHIFT' を指定することにより変更できる.

- **きれいなフォント:**

  GRPH1 は 2 種類のフォントを持っており, SGISET で内部変数'IFONT' を 2 とすることによって, フォント番号 2 の高品位フォントを出力することができる. この例を見ればわかるように, 高品位フォントは複数の線を引くことで, 線の太さを調節しているため, ラインインデクスで線の太

さを変えられる出力装置の場合には, 太い線を指定すると空白が潰れてそれらしい文字になる.

また, 当然のことながら高品位フォントを使うと出力量が増えるので, 出力に時間がかかるようになる.

- **ギリシャ文字と特殊記号:**

  フォントテーブルを見ればわかるように, GRPH1 は普通のアルファベットだけでなく, ギリシャ文字や特殊記号のフォントも持っている. これらを出力するには, フォント番号を文字関数 CSGI で文字コードに変換し, それをテキストとして SGTXV 等に渡せばよい.

  この文字関数 CSGI は ASCII コードを使用する計算機では CHAR 関数と同値であり, これを使っても同じ結果が得られるが, EBCDIC コードを使用する計算機では CHAR 関数で正しい文字が得られない.

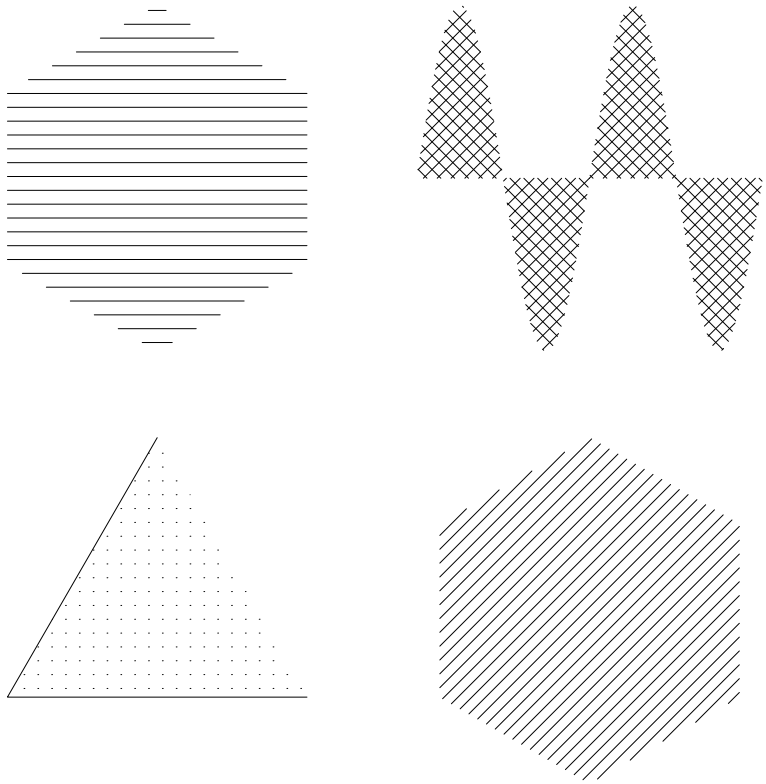
  従って, CSGI を使って文字コードを指定しておけば, どちらのコードを使用する計算機であっても正しい結果が得られる. フォント番号に関しては「GRPH1」マニュアルの付表を参照のこと.

```
*-----------------------------------------------------------------------
      PROGRAM SGPK07
      PARAMETER (N=9)
      DIMENSION Y(N)
      CHARACTER GREEK1*10, GREEK2*10, SYMBOL*10, USGI*3
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN(IWS)
      CALL SGFRM
      X1 = 0.1
      X2 = 0.9
      XC = 0.5
      CALL SGSLNI(1)
      DO 10 I=1, N
        Y(I) = 0.1 * (10-I)
        CALL SGLNV(X1, Y(I), X2, Y(I))
 10   CONTINUE
      CALL SGLNV(XC, 0.05, XC, 0.95)
*---------------------- SUPER/SUB SCRIPT ------------------------------
      CALL SGTXV(XC, Y(1), 'SGTXV|SUP"RST_SUB"')
      CALL SGSTXI(2)
      CALL SGLSET('LCNTL', .TRUE.)               ! <-- 添鐃緒申鐃緒申鐃緒申鐃夙ワ申鐃緒申鐃緒申 ON
      CALL SGTXV(XC, Y(2), 'SGTXV|SUP"RST_SUB"')
      CALL SGRSET('SHIFT', 0.5)                  ! <-- 鐃緒申鐃春ワ申鐃緒申鐃緒申鐃緒申
      CALL SGRSET('SMALL', 0.5)                  ! <-- 添鐃緒申鐃緒申鐃順き鐃緒申鐃緒申鐃緒申
      CALL SGTXV(XC, Y(3), 'SGTXV|SUP"RST_SUB"')
*----------------------- FONT SELECTION -------------------------------
      CALL SGSTXI(1)
      CALL SGSTXS(0.05)
      CALL SGTXV(XC, Y(4), 'ABCDEFG abcdefg')
      CALL SGISET('IFONT', 2)                    ! <-- 鐃緒申鐃曙い鐃淑フワ申鐃緒申鐃緒申
      CALL SGTXV(XC, Y(5), 'ABCDEFG abcdefg')
      CALL SGSTXI(3)
      CALL SGTXV(XC, Y(6), 'ABCDEFG abcdefg')
*------------------------ GREEK LETTERS -------------------------------
      GREEK1 = USGI(152)//USGI(153)//USGI(154)//USGI(155)//USGI(156)//
     #         USGI(157)//USGI(158)//USGI(159)//USGI(160)//USGI(161)
      GREEK2 = USGI(130)//USGI(131)//USGI(135)//USGI(138)//USGI(141)//
     #         USGI(143)//USGI(145)//USGI(148)//USGI(150)//USGI(151)
      CALL SGTXV(XC, Y(7), GREEK1)
      CALL SGTXV(XC, Y(8), GREEK2)
*---------------------------- SYMBOLS ---------------------------------
      SYMBOL = USGI(189)//USGI(190)//USGI(191)//USGI(192)//USGI(193)//
     #         USGI(210)//USGI(211)//USGI(212)//USGI(217)//USGI(218)
      CALL SGTXV(XC, Y(9), SYMBOL)
```

```
47          CALL SGCLS
48          END
```

1.9 網かけをする

これは指定した領域に網かけをする例である.



sgpk08.f:page1

- **領域網かけ:**

  SGTNU (SG ToNe U) は U 座標系で多角形の内部に網かけをする. 引数の形は SGPLU と同じであるが, SGTNU では指定した点列の始点と終点を結んだ多角形の中を塗りつぶす. (左下) 座標点に関して SGTNU に渡したものと同じ引数で折れ線を描いているが, 始点と終点は一致する必要のないことがわかる.

- **パターンの指定:**

  SGTNU で行う網かけのパターンの指定は SGSTNP (SG Set ToNe Pattern) で指定できる. 例によってパターンを同時に指定するルーチン SGTNZU も用意されている. トーン番号とパターンの対応については, 「GRPH1」マニュアルの付表を参照のこと.

- **複雑な多角形:**

  SGTNU/SGTNZU で指定する多角形は単連結領域である必要はない. 右上のように多角形の辺同志が交差する「ねじれた多角形」でも, その図形によってできる閉領域全てを塗りつぶす.

- **ハードフィル / ソフトフィル:**

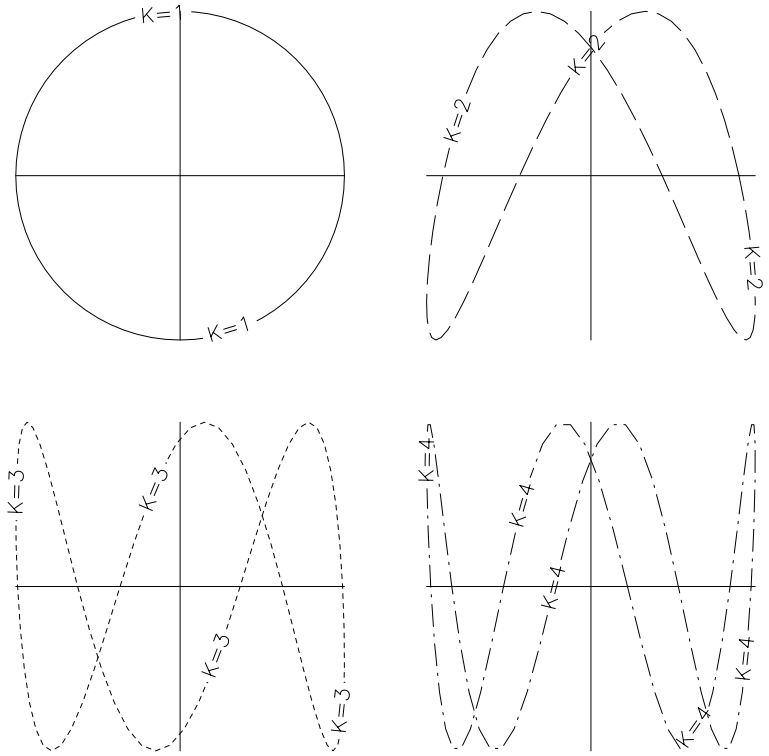
  なお, 何も指定しなければ標準的にはハードフィル (ディバイスに依存した網かけ) を行うようになっている. しかし, ハードフィルができないようなディバイス, あるいは SGLSET で内部変数

'LSOFTF' を .TRUE. にすると, ソフトフィルによる網かけをおこなう. ここでは, ハードフィルができないようながデバイスも念頭に置いて, ソフトフィルを陽に指定した.

```
*-----------------------------------------------------------------------
      PROGRAM SGPK08
      REAL UPX3(3), UPY3(3), UPX6(6), UPY6(6), UPXS(61), UPYS(61)
      A = 0.8
      TH = 3.14159 * 2 / 3
      DO 100 I=1, 3
        UPX3(I) = A*SIN(TH*I)
        UPY3(I) = A*COS(TH*I)
  100 CONTINUE
      TH = 3.14159 * 2 / 6
      DO 200 I=1, 6
        UPX6(I) = A*SIN(TH*I)
        UPY6(I) = A*COS(TH*I)
  200 CONTINUE
      TH = 3.14159 * 4 / 60
      DO 300 I=1, 61
        UPXS(I) = A*(I-31) / 30.
        UPYS(I) = A*SIN(TH*(I-1))
  300 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN(IWS)
      CALL SGLSET('LSOFTF',.TRUE.)           ! <-- ソフトフィルの指定
      CALL SGFRM
      CALL SGSWND(-1., 1., -1., 1.)
      CALL SGSVPT(0., 0.5, 0., 0.5)
      CALL SGSTRN( 1)
      CALL SGSTRF
      CALL SGPLU(3, UPX3, UPY3)
      CALL SGTNU(3, UPX3, UPY3)              ! <-- 網かけ（左下）
      CALL SGSVPT(0., 0.5, 0.5, 1.)
      CALL SGSTRF
      CALL SGSTNP(101)
      CALL SGTNU(6, UPX6, UPY6)              ! <-- 横線（左上）
      CALL SGSVPT(0.5, 1., 0., 0.5)
      CALL SGSTRF
      CALL SGTNZU(6, UPX6, UPY6, 201)        ! <-- 斜線（右下）
      CALL SGSVPT(0.5, 1., 0.5, 1.)
      CALL SGSTRF
      CALL SGTNZU(61, UPXS, UPYS, 601)       ! <-- 横線（右上）
      CALL SGCLS
      END
```

1.10 ラベルつきの折れ線を描く

ポリラインプリミティブの拡張機能として，ラベルつきの折れ線を描くこともできる．



sgpk09.f:page1

- ラベルつきの折れ線:

  SG*p*GET/SG*p*SET の管理する内部変数'LCHAR' を .TRUE. にすることによって，ポリラインプリミティブはラベルつきの折れ線を描く．ここでいうラベル付き折れ線とは, 描くべき線分のある長さを 1 サイクルとして, その一部分を空白域としその空白部分に指定した文字列を描くものである．

- ラベルの指定:

  描く折れ線にはさみこむ文字列は SGSPLC (SG Set PolyLine Character) で指定する．このルーチンによって文字列を指定し，'LCHAR' を .TRUE. とし，SGPLU などのポリラインプリミティブを呼ぶと，描かれる折れ線に SGSPLC で指定した文字が一定間隔で描かれる．注意すべきことは，'LCHAR' の指定は，線分を描くすべてのプリミティブに有効なので，必要な描画が終ったら 'LCHAR' を .FALSE. に戻しておかなくてはならない．

- ラベルの変形:

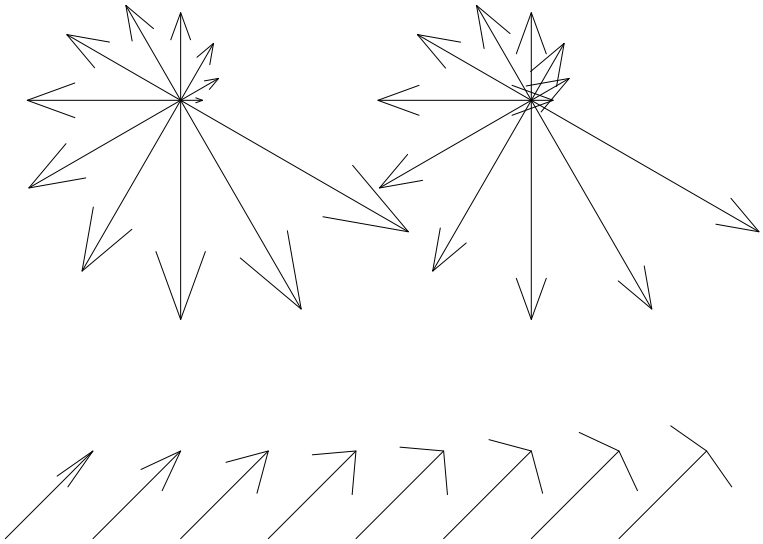
  この例のようにラベルの最後の 1 文字について，その文字番号を 1 つずつ増やしたいときは，SGNPLC (SG Next PolyLine Character) によって設定されている文字列の最後の 1 文字の文字番号を 1 つふやすことができる．つまりこの例では文字列として'K=1' が指定されているが, まず第 1 回目の呼び出しで'K=1' を描き, 以降第 i 回目の呼び出しでは'K=i' $(i = 1...4)$ が描かれてい

る. SGSPLC で文字列が設定してない場合の初期値は'A' である.

```
*---------------------------------------------------------------------
      PROGRAM SGPK09
      PARAMETER ( KMAX=4, NN=73 )
      PARAMETER ( DD=2.0, PI=3.141592 )
      REAL      XORG(KMAX), YORG(KMAX), UX(NN), UY(NN)
      DATA      XORG / 2.5, 7.5, 2.5, 7.5 /
      DATA      YORG / 7.5, 7.5, 2.5, 2.5 /
      WRITE(*,*) ' WORKSTATION ID (I) ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN( IWS )
      CALL SGFRM
      CALL SGSWND( 0.0, 10.0, 0.0, 10.0 )
      CALL SGSVPT( 0.0, 1.0, 0.0, 1.0 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      CALL SGSTXS( 0.03 )
      CALL SGSTXI( 3 )
      CALL SGSTXC( 0 )
      CALL SGSPLC( 'K=1' )
      DO 20 K = 1, KMAX
        CALL SGSLNI( 1 )
        CALL SGLNU( XORG(K)-DD, YORG(K), XORG(K)+DD, YORG(K) )
        CALL SGLNU( XORG(K), YORG(K)-DD, XORG(K), YORG(K)+DD )
        DO 10 I = 1, NN
          TH = 2*PI*(I-1)/(NN-1)
          UX(I) = XORG(K) + DD*COS(TH+(K-1)*PI/7)
          UY(I) = YORG(K) + DD*SIN(K*TH)
   10   CONTINUE
        CALL SGSPLI( 2 )
        CALL SGSPLT( K )
        CALL SGLSET( 'LCHAR', .TRUE. )
        CALL SGPLU( NN, UX, UY )
        CALL SGLSET( 'LCHAR', .FALSE. )
        CALL SGNPLC
   20 CONTINUE
      CALL SGCLS
      END
```

1.11 矢印を描く

矢印つき線分 (線分の終点から対称な 2 本の線分を付け加える) を描くサブルーチンも用意されている.



sgpk10.f:page1

- **アローサブプリミティブ:**

  アローサブプリミティブはの基本的な構成は, ラインサブプリミティブとほぼ同じである. 違いは, ラインサブプリミティブではラインインデクスしか指定しなかったが, アローサブプリミティブではラインインデクスのほかにラインタイプも指定できる. したがってこの例でも, ラインタイプに関する引数部分を消して SGLAZU (SG Line-Arrow ZU) のかわりに SGLNZU を呼べば, 矢羽の部分がないだけで同じような線分群を描くことができる.

- **矢じり部分の形状設定:**

  矢じり部分の形状は SG*p*GET/SG*p*SET が管理するいくつかの内部変数によって設定できる. 上段左側の矢印は, デフォルトの内部変数を用いて描いた. この場合, 線分部分が長くなると, それに比例して矢じりの部分も長くなる. 上段右側の矢印は, 内部変数'LPROP' を .FALSE. として描いた. このようにすると, 矢じり部分の長さが一定値となる. また下段の矢印は, 矢じり部分の線分と本体部分の線分のなす角 (内部変数'ANGLE') を変化させて描いた.

```
*---------------------------------------------------------------------
      PROGRAM SGPK10
      PARAMETER ( ND=12, DDEG=360.0/ND, DD=0.25 )
      PARAMETER ( PI=3.141592 )
      REAL      RX(ND), RY(ND)
      DO 10 N = 1, ND
        RX(N) = DD*N*COS(PI/180*DDEG*(N-1))
        RY(N) = DD*N*SIN(PI/180*DDEG*(N-1))
   10 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN( IWS )
      CALL SGFRM
      CALL SGSWND( 0.0, 10.0, 0.0, 10.0 )
      CALL SGSVPT( 0.0, 1.0, 0.0, 1.0 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      X1 = 3.0
      Y1 = 7.0
      DO 15 N = 1, ND
        CALL SGLAZU( X1, Y1, X1+RX(N), Y1+RY(N), 1, 2 )
   15 CONTINUE
      CALL SGLSET( 'LPROP', .FALSE. )
      CALL SGRSET( 'CONST', 0.05 )
      X1 = 7.0
      Y1 = 7.0
      DO 20 N = 1, ND
        CALL SGLAZU( X1, Y1, X1+RX(N), Y1+RY(N), 1, 2 )
   20 CONTINUE
      DO 25 I = 1, 8
        CALL SGRSET( 'ANGLE', 10.0*I )
        CALL SGLAZU( REAL(I), 2.0, REAL(I)+1, 3.0, 1, 2 )
   25 CONTINUE
      CALL SGCLS
      END
```

第2章　レイアウト

2.1　概要

SLPACK (LAYOUT) は, 図形の外側にマージンをとったり, 複数の図を 1 つのページにまとめたりする機能を持つパッケージである. SLPACK のルーチンは, 原則として全て SGOPN と最初に現れる SGFRM の間で呼ばれなければならない.

ここで説明するデモプログラムは dcl-x.x/demo/grph1/slpack の中にあるので, 参考にしていただきたい.

2.2　1 ページに複数の図形

同じ様な図形を沢山並べたい時, SLPACK を使うと非常に簡単にできる.

slpk01.f:page1

このプログラムでは横長の用紙を想定して, SGOPN で縦位置を選択し, SLMGN (Sgpack/Layout MarGiN) でその用紙全体のフレームでマージンをとるように指定している. さらにマージンを除いた部分を SLDIV (SL DIVide) を用いて横 3, 縦 5 に分割し, その分割された領域のなかでさらにマージンをとる. 分割された領域は縦横比が 1 ではないので, SLRAT (SL RATio) を呼んでマージンをとったあとの領域に最大内接するように 1:1 のフレームを設定する. (このプログラムでは SLRAT を呼ばなくても SGFRM が 1:1 のフレームをとってくれるが, いつもそのような好運に恵まれるわけではない. 思わぬ結果になることを避けるために, SLPACK のルーチンを使う時には常に SLRAT で縦横比を指定するようにしていただきたい.)

このように分割されたフレームを GRPH1 ではあたかも 1 枚の紙のように扱い, SGFRM の実行により, 次のフレームに自動的に移っていく. プログラムの DO ループの中は, 普通に改ページをしながら描画するのと全く同じである. SLPWWR (SL Plot Workstation Window Rectangular) はワークステーションウインドを描くユーティリティーであるが, これが分割されたフレームの中に描かれていることを見れば, GRPH1 がこの小さなフレームをワークステーションウインドと見なしていることがわかる.

また SGSVPT でビューポートを設定しなおした時と違って, 文字の大きさも分割されたフレームの大きさに応じて小さくなっていることに注意されたい.

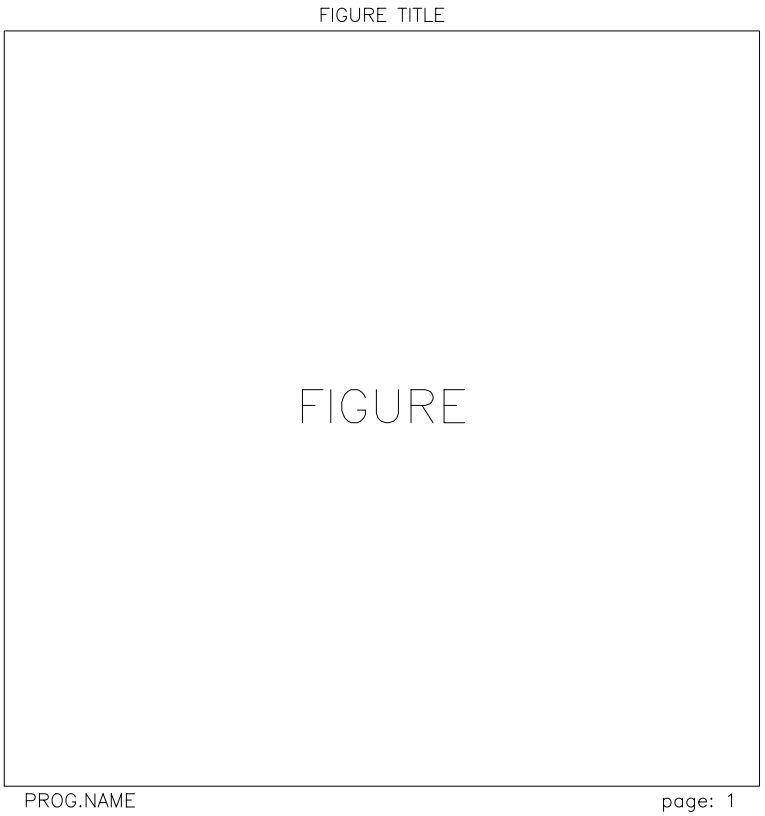
このように, 同じ図形を規則的に並べるのは SLPACK を使うと非常に簡単にできる. しかし, 大きな図形の横に小さな図形を並べるというようなことは, SLPACK を使うより SGSVPT で ビューポートの設定をしなおした方がよいだろう.

なお, SLDIV は 2 回まで呼ぶことができて, 分割されたフレームをさらにもう一度分割することが可能である. SLPACK では 最初の SLDIV で分割される前の大きなフレームを第 1 レベル, SLDIV で 1 度分割された小さなフレームを第 2 レベル, さらに SLDIV で細かく分割されたフレームを第 3 レベル のフレームと呼ぶ.

```
1   *-----------------------------------------------------------------------
2         PROGRAM SLPK01
3         CHARACTER*7  CTXT
4         WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
5         CALL SGPWSN
6         READ(*,*) IWS
7         CALL SGOPN(-ABS(IWS))
8         CALL SLMGN(0.1, 0.1, 0.05, 0.05)    ! <-- 第 1 レベルのマージン
9         CALL SLDIV('S', 3, 5)               ! <-- フレーム分割 (第 2 レベル)
10        CALL SLMGN(0.05, 0.05, 0.05, 0.05)  ! <-- 第 2 レベルのマージン
11        CALL SLRAT(1., 1.)                  ! <-- 第 2 レベルの縦横比を指定
12        CALL SGSTXI(2)
13        CALL SGSTXS(0.1)
14        DO 100 I=1, 15
15          CALL SGFRM                        ! <--
16          CALL SLPWWR(1)                    ! <    この中は普通に 1 ページ
17          WRITE(CTXT, ''(''FRAME'',I2.2)'') I ! <    描画するのと同じ
18          CALL SGTXV(0.5, 0.5, CTXT)        ! <--
19    100 CONTINUE
20        CALL SGCLS
21        END
```

2.3 マージンにタイトル等を書く

沢山の図を出力すると, 何の図だったかわからなくなってしまう. そんな時, SLPACK の機能を使って, タイトル, その図を出力したプログラム名, 使用したデータ名等をマージンに書き込んでおくと, あとの整理が楽になる.



slpk02.f:page1

タイトルなどを書くには SLSTTL (SL Set TiTLe) で SGFRM の前に書きたい文字列を指定しておく. 文字列は, 5 つまで番号をつけて指定でき, それぞれに書くべき位置が設定できる. この番号を指定することで, 例えば右下の文字列だけをページごとに変える, ということが可能になる. (SLSTTL は, SLPACK のルーチンの中で唯一例外的に SGFRM を呼んだあとでも呼ぶことができる.)

タイトルなどの文字が書かれるのは第 1 レベルのマージンに対してだけである. フレームを分割してもタイトルなどは, 1 ページに 1 つしか書かれない. したがって, タイトル等を書くときにはあらかじめ第 1 レベルのマージンをとっておかなければならない.

ここで注意すべきことは, マージンは各レベルごとの最大作画領域に対する比率で指定されるのに対して, 文字の大きさの単位は第 1 レベルにおける最大作画領域の長辺を 1 とするような単位となることである. たとえば, ここで描いたタイトルと次の作画例におけるタイトルの大きさが違うことに注意していただきたい. すな

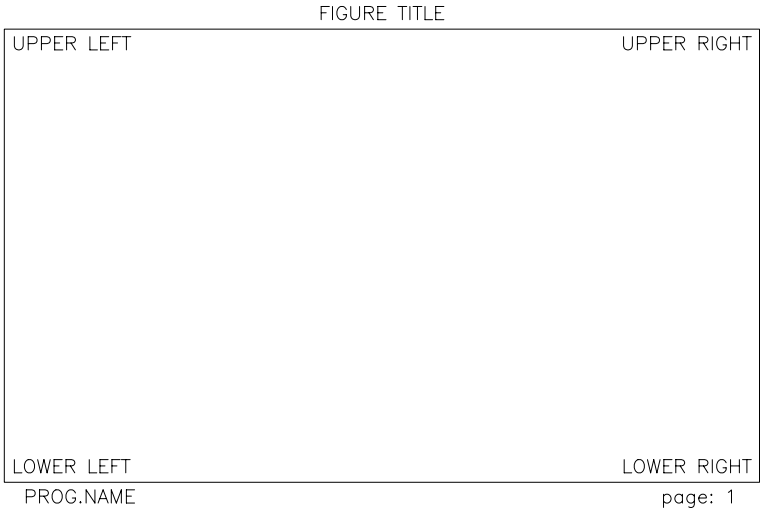
わち，slpk02.f の例では陽に SLRAT を用いて正方形の最大作画領域を長方形に内接するように指定しているため，タイトル文字の大きさは正方形の一辺が 1 となるような単位で描かれている．いっぽう，sgpk03.f では長方形の画面をいっぱいに使うように指定しているため，長方形領域の長辺を 1 とするような単位でタイトル文字が描かれている．

なお, SLSTTL で指定できる文字列には, #PAGE, #TIME, #DATE という予約語があって, それらの予約語はそれぞれ, ページ数, 時刻, 日付に変換される．

```
1   *---------------------------------------------------------------------
2         PROGRAM SLPK02
3         WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
4         CALL SGPWSN
5         READ(*,*) IWS
6         CALL SGOPN(IWS)
7         CALL SLMGN(0., 0., 0.05, 0.05)                        ! <-- マージン設定
8         CALL SLRAT(1., 1.)                                    ! <-- 縦横比設定
9         CALL SLSTTL('FIGURE TITLE',  'T',  0., -1., 0.02, 1) ! <-+
10        CALL SLSTTL('page:#PAGE',    'B',  1.,  1., 0.02, 2) ! <-+ タイトル設定
11        CALL SLSTTL('PROG.NAME',     'B', -1.,  1., 0.02, 3) ! <-+
12        CALL SGFRM                          ! <-- ここで実際にタイトルが書かれる
13        CALL SLPWWR(1)
14        CALL SGTXZV(0.5, 0.5, 'FIGURE', 0.05, 0, 0, 1)
15        CALL SGCLS
16        END
```

2.4 紙を一杯に使う

用紙の形は普通長方形なので, そこに正方形の領域をとって, その中だけに図を書くのはもったいない. 紙一杯に図を描きたい, という時も SLPACK を活用して欲しい.



slpk03.f:page1

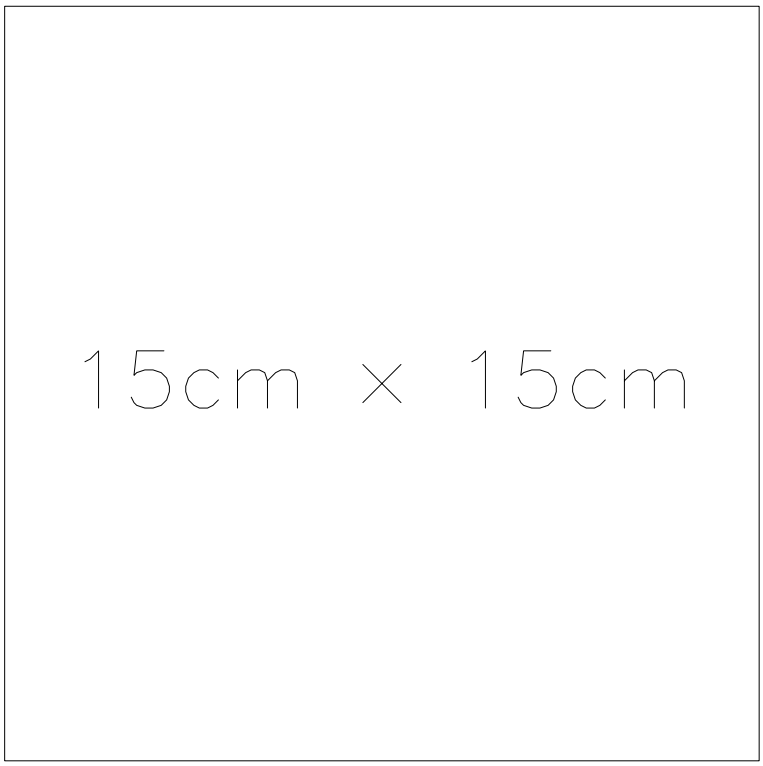
紙を一杯に使う機能そのものは SGPACK の基本的な機能で, SGLSET を用い内部変数 'LFULL' を .TRUE. にすれば, 物理的な描画範囲一杯に描画することができる. しかしその場合, 物理的な描画範囲はデバイスによって違うので, 異なるデバイスに出力する際にエラーを起こして出力できなくなる可能性がある. そこで, SGLSET で'LFULL' を.TRUE. にしたときは, SLRAT を使って「私は 1×0.6 の領域に図を描きたいのだ」と宣言しておくことをお勧めする. SLRAT で縦横比が指定されると, 以後, この領域が最大内接するように描画領域を設定するので, 1×0.6 の比率を持つデバイスには一杯の図が描けるし, そうでないデバイスに出力してもエラーは起こさない.

この例では, 作画領域の上下に 0.08 のマージンをとってタイトル等のスペースを確保した後 1×0.6 の領域を宣言しているので, 実際には $0.6 +$ マージン幅 の長さが縦方向に確保されることになる.

```
1  *---------------------------------------------------------------------
2        PROGRAM SLPK03
3        WRITE(*,*) 'WORKSTATION ID ? '
4        CALL SGPWSN
5        READ (*,*) IWS
6        CALL SGOPN(IWS)
7        CALL SGLSET('LFULL', .TRUE.)
8        CALL SLMGN(0., 0., 0.08, 0.08)
9        CALL SLRAT(1., 0.6)                              ! <-- 縦横比を指定
10       CALL SLSTTL('FIGURE TITLE',  'T',  0., -1., 0.02, 1)
11       CALL SLSTTL('page:#PAGE',    'B',  1.,  1., 0.02, 2)
12       CALL SLSTTL('PROG.NAME',     'B', -1.,  1., 0.02, 3)
13       CALL SGFRM
14       CALL SLPWWR(1)
15       CALL SGTXZV(0.01, 0.02, 'LOWER LEFT',  0.02, 0, -1, 3)
16       CALL SGTXZV(0.01, 0.58, 'UPPER LEFT',  0.02, 0, -1, 3)
17       CALL SGTXZV(0.99, 0.02, 'LOWER RIGHT', 0.02, 0,  1, 3)
18       CALL SGTXZV(0.99, 0.58, 'UPPER RIGHT', 0.02, 0,  1, 3)
19       CALL SGCLS
20       END
```

2.5 cm 単位で長さを指定する

SLPACK では基本的に最大描画領域に対する比率で, マージンなどをとるようになっているが, 目的によっては絶対的な長さ (例えば 10cm) を指定したい場合もある.



slpk04.f:page1

SLFORM (SL FORM) または SLSIZE (SL SIZE) で, 自分が描きたい描画範囲の大きさを具体的に指定しておけば, その描画範囲が物理的に描画できる限り, 異なったデバイスでも同じ大きさの図が出力できる. SLFORM は描画範囲を cm 単位で, また, SLSIZE は A4, B5 等の大きさを指定するものである. (通常 A4 の用紙の最大描画範囲は, 用紙そのものの大きさよりも小さいので, A4 の紙に SLSIZE で A4 を指定するとエラーとなる.)

同じ A4 サイズに出力するデバイスでも, その最大描画領域は微妙に異なっている. 自分が使う可能性のあるデバイスの最大描画範囲をあらかじめ調べておいて, その最小値を SLFORM で常に指定するようにすれば, 異なったデバイスで出力した図をもとに OHP を作成しても, 同じスケールで重ねることができる.

ただし, 絶対的な大きさを指定するということは, 汎用性を損ねるリスクと背中合わせであることを認識しておいて頂きたい. また, コンソールディスプレイなど, 物理的な大きさがはっきりしないデバイスに対しては, 適当な大きさが仮定されている.

```
*-----------------------------------------------------------------------
      PROGRAM SLPK04
      REAL X(5), Y(5)
      CHARACTER USGI*3
      DATA X / -7.5, 7.5, 7.5, -7.5, -7.5/
      DATA Y / -7.5,-7.5, 7.5,  7.5, -7.5/
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN(IWS)
      XMAXF = 26.
      YMAXF = 18.5
      CALL SLFORM(XMAXF, YMAXF)        ! <-- 鐃緒申鐃緒申的鐃緒申鐃術誌申鐃緒申鐃順き鐃緒申鐃緒申鐃緒申鐃緒申
      CALL SLRAT(XMAXF, YMAXF)
      CALL SGLSET('LFULL', .TRUE.)
      CALL SGFRM
      CALL SGSWND(-XMAXF/2., XMAXF/2., -YMAXF/2., YMAXF/2.)
      CALL SGSTRF
      CALL SGPLU(5, X, Y)
      CALL SGTXZU(0., 0., '15cm '//USGI(194)//' 15cm', 0.05, 0, 0, 3)
      CALL SGCLS
      END
```

第3章　高度なコントロール

3.1　概要

GRPACK は GRPH2 の各種パッケージを使って新しい図を描くとき, これらのパッケージを初期化したり, 変換関数に関するパラメータを (実行時オプションの介入を許して) 設定するものである.

GRPACK は以下のように, 基本的に GRPH1 の「コントロール」および「正規化変換」に関するルーチンに相当するものを用意している (対応する GRPH1/SGPACK のルーチン名を括弧内に示す).

GROPN (SGOPN)	出力装置のオープン.
GRFRM (SGFRM)	フレームの設定.
GRFIG	新たな図の設定.
GRCLS (SGCLS)	出力装置のクローズ.
GRSVPT (SGSVPT)	ビューポートの設定.
GRSWND (SGSWND)	ウインドウの設定.
GRSSIM (SGSSIM)	相似変換の設定.
GRSMPL (SGSMPL)	地図投影の極の設定.
GRSTRN (SGSTRN)	変換関数の番号による設定.

SGOPN などは電脳ライブラリの「下克上禁止令」により GRPH2 のルーチンを初期化できない. しかし, GRPH2 のサブパッケージの中には, 改ページ等のタイミングで初期化が必要なものがある (UZPACK, USPACK 等; 初期化動作としては, GRPH2 でのサブパッケージが自動的に決める可能性のあるパラメタに, 「未定義」の値を代入したりする). GRPACK はこれらのサブパッケージを初期化するとともに, 内部で対応する SGPACK のルーチンを呼ぶ. したがって, GRPACK を使って改ページなどをすれば, そのたびに GRPH2 のライブラリの初期化ルーチンを呼ぶ必要はない.

さらに GRPACK の各ルーチンが設定するパラメータについては, 実行時オプションの介入が可能である. たとえば SGOPN に相当する GROPN というルーチンを使うと, 指定するワークステーション番号は実行時オプションによって変更できる.

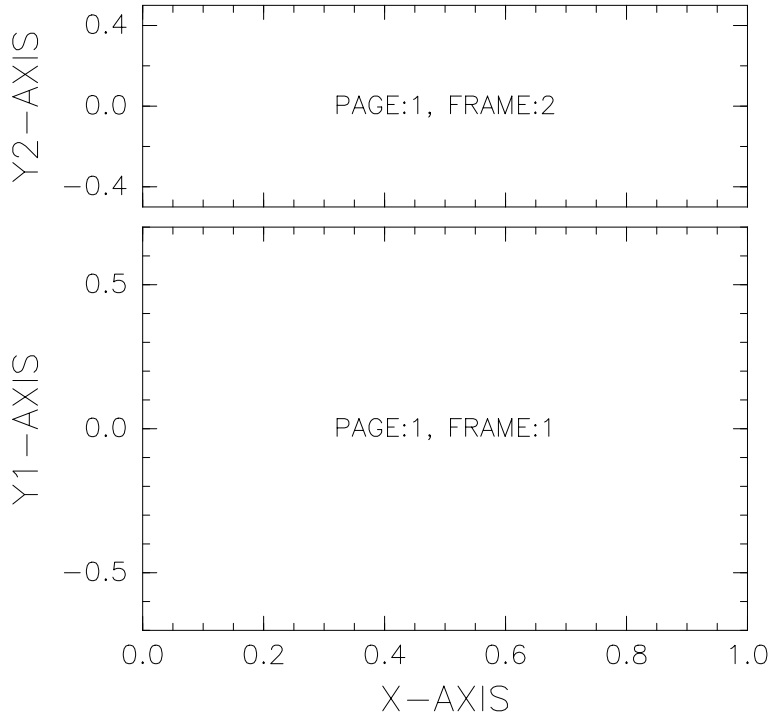
また, GRPACK は GRPH1 より高機能な GRPH2 のライブラリを使うことを前提にしているので, GROPN の中で SGLSET を呼んで内部変数'LCNTL' を .TRUE. とし, 上つき下つきなどの制御文字を有効にしている ('LCNTL' の省略値は .FALSE. である).

3.2 使用例

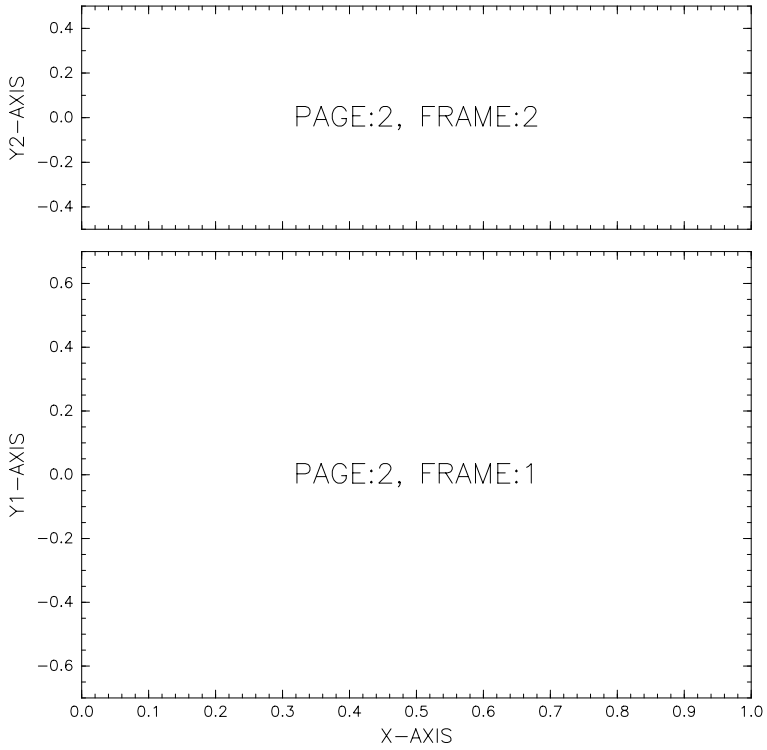
以下の例では, まず, GROPN, GRFRM, GRFIG などコントロールルーチンの役割を認識して欲しい. さらに, 実行時オプションの指定が可能な環境では (たとえば, コマンドライン引数によって実行時オプションの指定が可能な環境では),

```
> grpk01 -sg:iws=-1 -sg:ifont=2 -sg:vxmin=0.5
```

などとして, その効果を試してもらいたい.



grpk01.f:page1



grpk01.f:page2

```
*-----------------------------------------------------------------------
      PROGRAM GRPK01
      CALL SWCSTX('FNAME','GRPK01')
      CALL SWLSTX('LSEP',.TRUE.)
*     WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
*     CALL SGPWSN
*     READ (*,*) IWS
      IWS=2
      CALL GROPN(IWS)
      CALL GRFRM
      CALL GRSWND(0.0, 1.0, -0.7, 0.7)
      CALL GRSVPT(0.2, 0.8,  0.2, 0.6)
      CALL GRSTRN(1)
      CALL GRSTRF
      CALL USSTTL('X-AXIS', ' ', 'Y1-AXIS', ' ')
      CALL USDAXS
      CALL SGTXZU(0.5, 0.0, 'PAGE:1, FRAME:1', 0.02, 0, 0, 3)
      CALL GRFIG
      CALL GRSWND(0.0, 1.0, -0.5, 0.5)
      CALL GRSVPT(0.2, 0.8, 0.62, 0.82)
      CALL GRSTRN(1)
      CALL GRSTRF
      CALL UZLSET('LABELXB', .FALSE.)
      CALL USSTTL('X-AXIS', ' ', 'Y2-AXIS', ' ')
      CALL USDAXS
      CALL SGTXZU(0.5, 0.0, 'PAGE:1, FRAME:2', 0.02, 0, 0, 3)
      CALL UZLSET('LABELXB', .TRUE.)
      CALL GRFRM
      CALL UZFACT(0.5)
      CALL GRSWND(0.0, 1.0, -0.7, 0.7)
      CALL GRSVPT(0.2, 0.8,  0.2, 0.6)
      CALL GRSTRN(1)
      CALL GRSTRF
      CALL USSTTL('X-AXIS', ' ', 'Y1-AXIS', ' ')
      CALL USDAXS
      CALL SGTXZU(0.5, 0.0, 'PAGE:2, FRAME:1', 0.02, 0, 0, 3)
      CALL GRFIG
      CALL GRSWND(0.0, 1.0, -0.5, 0.5)
      CALL GRSVPT(0.2, 0.8, 0.62, 0.82)
      CALL GRSTRN(1)
      CALL GRSTRF
      CALL UZLSET('LABELXB', .FALSE.)
      CALL USSTTL('X-AXIS', ' ', 'Y2-AXIS', ' ')
      CALL USDAXS
      CALL SGTXZU(0.5, 0.0, 'PAGE:2, FRAME:2', 0.02, 0, 0, 3)
      CALL UZLSET('LABELXB', .TRUE.)
      CALL GRCLS
      END
```

第4章　座標軸の描画

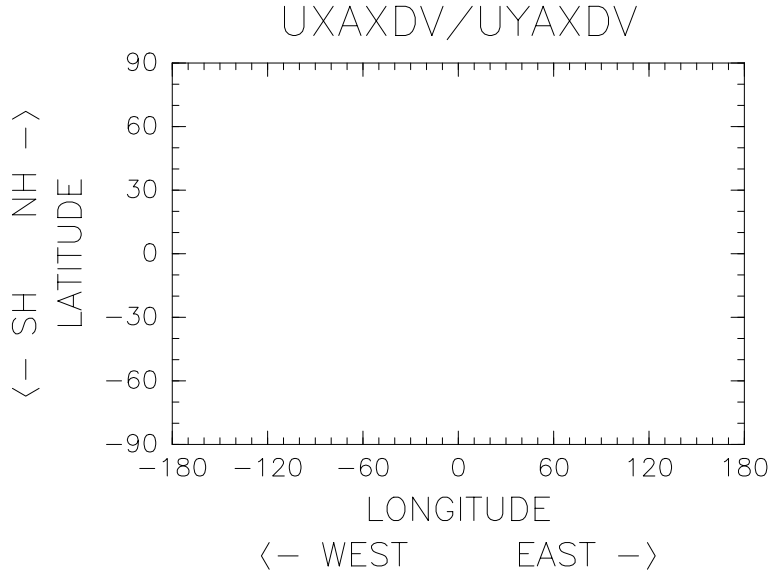
4.1　概要

ここでは U[XYZ]PACK/ULPACK/UCPACK の基本的な機能を, いくつかの簡単なプログラム例で示す. これらのデモプログラムは dcl-x.x/demo/grph2/uxyzpk の中にあるので, 参考にしていただきたい. 以下の例からも, 多種多様な座標軸が描けることはわかっていただけると思うが, これらは序の口でまだまだ凝った座標軸を描くことができる.

なお, 以下のデモプログラムでは, GRPACK を用いて初期化や改ページの操作をおこなっているので, 初期化ルーチン `UZINIT` は呼ばなくてもよいようになっている.

4.2　簡単な座標軸

まず, 最も簡単な例を示そう. X 軸 (左右方向) · Y 軸 (上下方向) とも線形座標軸 (均等な目盛り) を描く. それぞれの軸には目盛りとラベル (目盛りのところにつける数字), タイトル (軸の説明) をつける. さらに図のタイトルもつける.



uxyz01.f:page1

座標軸関連のパッケージを使用するときに必ずおこなわなければならないことは, ウインドウとビューポートを適切に設定することである. ウインドウとはユーザーの定義する座標系 (U 座標系: ここでは経度と緯度の座標系) において定義される矩形領域である; ビューポートとは仮想直角座標系 (V 座標系: ふつうは最大 $[0.0, 1.0] \times [0.0, 1.0]$ の閉じた領域に限定される仮想的な座標系) において定義される矩形領域である. ウインドウとビューポートの対応関係は, サブルーチン SGSWND, SGSVPT, SGSTRN (「GRPH1」マニュアル, 正規化変換の項参照) を用いて設定し, SGSTRF によって確定する (Ver.4 までは, これらを SGSNTR でおこなっていた). この例では, X 方向について, 経度 –180 ° から +180 ° までを描画可能な範囲の 0.2 から 0.8 (左端が 0.0, 右端が 1.0) へ対応させている; また Y 方向について, 緯度 –90 ° から +90 ° までを 0.3 から 0.7 (下端が 0.0, 上端が 1.0) へ対応させている. (なおここでの描画可能な範囲は, コーナーマークで示される長方形領域に最大内接するような正方形領域となっていることに注意していただきたい.)

座標軸の作画を行なうにあたって最も基本的なことは, 1 本の座標軸 (1 本の軸とそれに付ける目盛りおよびラベルによって構成される) を描くために作画ルーチンを 1 回呼ぶということである. この例では, 上下左右あわせて 4 回作画ルーチン (X 軸は UXAXDV, Y 軸は UYAXDV) を呼んでいる. 最初の引数 'B', 'T', 'L', 'R' によって, それぞれ, 下, 上, 左, 右側の軸を描画することを指定する. 座標軸を描くこれらの場所は, ウインドウとビューポートを設定した矩形領域のちょうど境界線上にある. 次の 2 つの引数は, 短い目盛りと長い目盛りをどんな間隔で打つか指定する. 単位は U 座標系の値 (ここでは, 経度, 緯度の値) である. ラベルは長い目盛りのところにだけ描かれる.

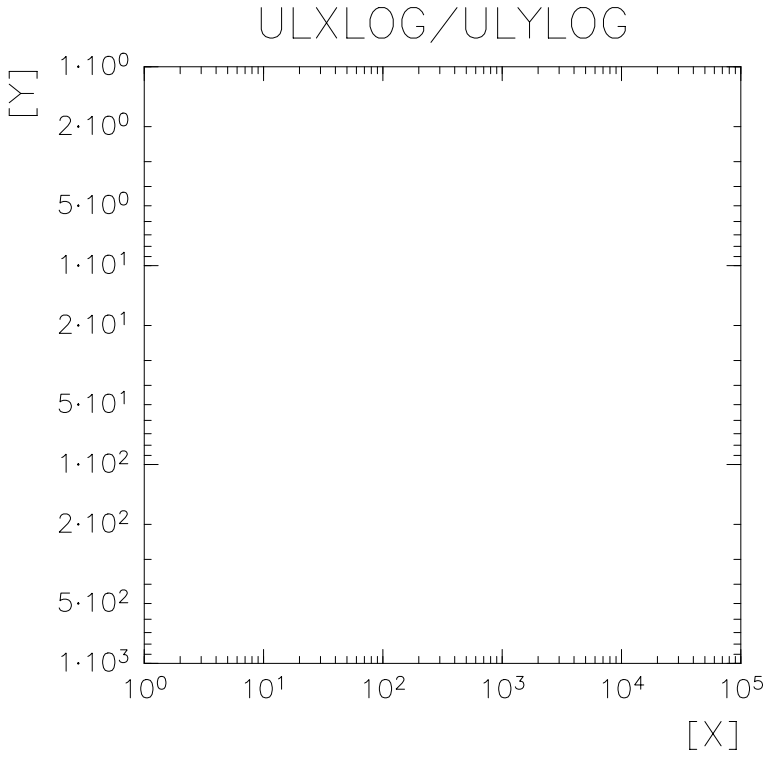
軸につけるタイトルおよび図のタイトルも同じように, それぞれの軸についてタイトル描画ルーチンを呼ぶ. 小さい文字で描きたいときは UXSTTL, UYSTTL を, 大きい文字で描きたいときは UXMTTL, UYMTTL を用いる. 最初の引数によって, タイトルをつける軸の場所を指定する. 2 番目の引数はタイトルとして描く文字列を指定する. 最後の引数はタイトルを描く位置を–1 から +1 までの実数値で指定する. たとえば X 軸については, –1: 左寄せ, 0: センタリング, +1: 右寄せとなる; また Y 軸については, –1: 下寄せ, 0: センタリング, +1: 上寄せとなる. この例の X 軸 (下側), Y 軸 (左側) のように, 同じ軸に対して 2 回以上タイトル描画ルーチンを呼んだときは, 重ならないようにだんだんと外側の方へずらしてタイトルが描かれる. これによって, それぞれの軸に複数のタイトルをつけることができる.

言われるままに作画してみて, 「はて, タイトルやラベルの大きさ, 目盛の長さなどは一体どのように決まっているのだろか?」と, ほとんどのユーザーは不思議に思っていることであろう. じつはこれらの属性は, UZPACK のなかにあるサブルーチン UZ*p*GET/UZ*p*SET の管理する内部変数を参照して決められている. 実際, ユーザーがタイトルの大きさやその描く向きなどを指定することは可能であるが, ここではそのような立ち入った例は示さない. ふつうの用途には, パッケージがあらかじめ用意している内部変数の値にしたがっていても, 充分に満足できる座標軸が作画可能なはずである.

```
*---------------------------------------------------------------------
      PROGRAM UXYZ01
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL SGSWND( -180.0, +180.0, -90.0, +90.0 )
      CALL SGSVPT( 0.2, 0.8, 0.3, 0.7 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      CALL UXAXDV( 'B', 10.0, 60.0 )
      CALL UXAXDV( 'T', 10.0, 60.0 )
      CALL UXSTTL( 'B', 'LONGITUDE', 0.0 )
      CALL UXSTTL( 'B', '<- WEST      EAST ->', 0.0 )
      CALL UYAXDV( 'L', 10.0, 30.0 )
      CALL UYAXDV( 'R', 10.0, 30.0 )
      CALL UYSTTL( 'L', 'LATITUDE', 0.0 )
      CALL UYSTTL( 'L', '<- SH    NH ->', 0.0 )
      CALL UXMTTL( 'T', 'UXAXDV/UYAXDV', 0.0 )
      CALL GRCLS
      END
```

4.3 対数座標軸

次は対数座標軸の例である. 対数座標軸の描画は ULPACK が担当している.



uxyz02.f:page1

対数座標軸を描くには, まず SGSTRN によって対数変換をあらわす変換関数番号を設定しなければならない. すなわち SGSTRN の引数として 2 (Y 軸のみ対数変換), 3 (X 軸のみ対数変換), 4 (X, Y 軸とも対数変換) のどれかを指定する.

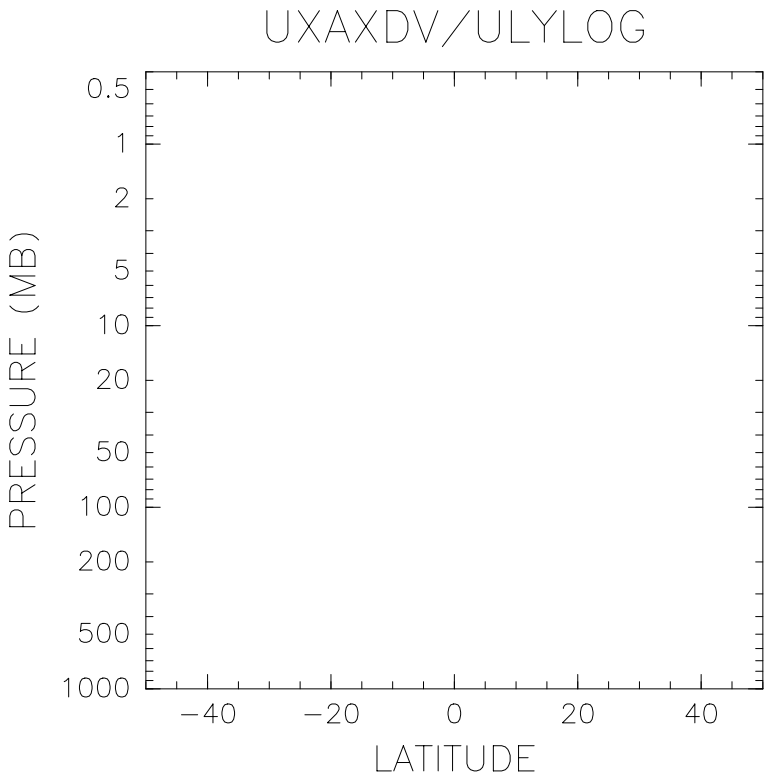
対数座標軸を描くサブルーチンは ULXLOG, ULYLOG である. ここでも, 最初の引数は座標軸を描く場所を指定する. 2 つめの引数は, 1 桁の範囲に描くラベルの数であり, X 軸については 1 となっていて, 10^n のところにのみラベルが描かれている. また Y 軸については 3 となっていて, 10^n 以外の 2×10^n, 5×10^n にもラベルが描かれている. 3 つめの引数は, 1 桁の範囲に描く目盛りの数を表している. ここでは 1 から 9 すべての目盛を打つように, 9 を指定している.

また X 軸と Y 軸につけるタイトルは, それぞれ右寄せ (1.0), 上寄せ (1.0) して描いた.

```
1  *---------------------------------------------------------------------
2        PROGRAM UXYZ02
3        WRITE(*,*) ' WORKSTATION ID (I) ? ;'
4        CALL SGPWSN
5        READ(*,*) IWS
6        CALL GROPN( IWS )
7        CALL GRFRM
8        CALL SGSWND( 1.0E0, 1.0E5, 1.0E3, 1.0E0 )
9        CALL SGSVPT( 0.2, 0.8, 0.2, 0.8 )
10       CALL SGSTRN( 4 )
11       CALL SGSTRF
12       CALL ULXLOG( 'B', 1, 9 )
13       CALL ULXLOG( 'T', 1, 9 )
14       CALL UXSTTL( 'B', '[X]', 1.0 )
15       CALL ULYLOG( 'L', 3, 9 )
16       CALL ULYLOG( 'R', 3, 9 )
17       CALL UYSTTL( 'L', '[Y]', 1.0 )
18       CALL UXMTTL( 'T', 'ULXLOG/ULYLOG', 0.0 )
19       CALL GRCLS
20       END
```

4.4　中途半端なウインドウ

ウインドウの境界値は必ずしも切りのよい値でなくてもよい.



uxyz03.f:page1

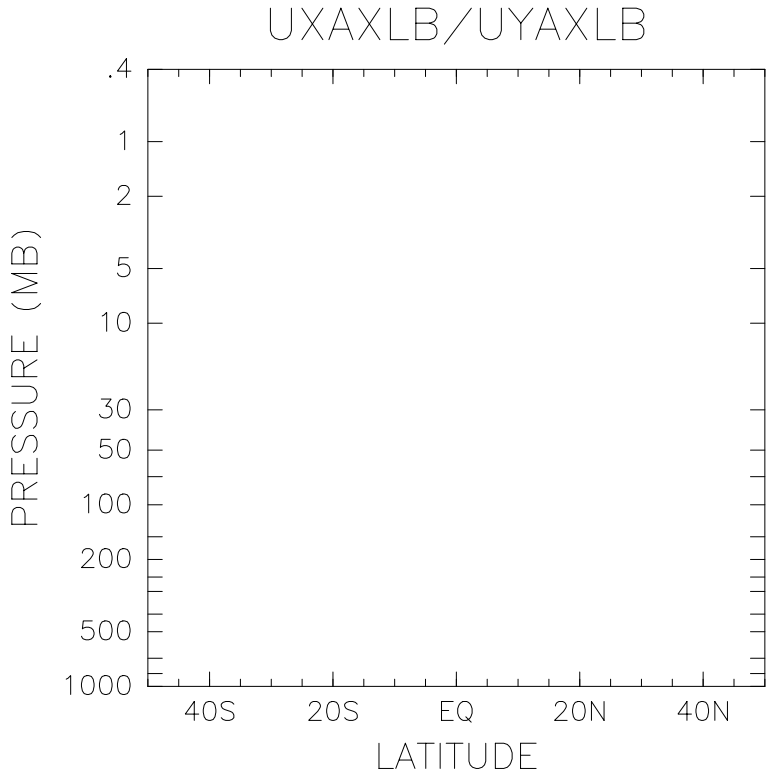
この例では, ウインドウを, X 軸については–50 から +50 の範囲で, Y 軸については 1000 から 0.4 の範囲で設定している. このとき UXAXDV は, きざみ値の整数倍となるところに目盛とラベルを描く. また ULYLOG は, 10^n から 10^{n+1} を 1 サイクルとして目盛とラベルを描く. このように, 目盛やラベルは切りのよいサイクルを単位として作画がおこなわれる.

なお, この例では, ULYLOG を用いて対数座標軸を描画する際, UL*p*GET/UL*p*SET の管理する内部変数'IYTYPE' を変更してラベルの書式を変えている. このように, 対数軸に付けるラベルの書式は何種類も用意されている.

```
1   *---------------------------------------------------------------------
2         PROGRAM UXYZ03
3         WRITE(*,*) ' WORKSTATION ID (I) ? ;'
4         CALL SGPWSN
5         READ(*,*) IWS
6         CALL GROPN( IWS )
7         CALL GRFRM
8         CALL SGSWND( -50.0, +50.0, 1.0E3, 0.4 )
9         CALL SGSVPT( 0.2, 0.8, 0.2, 0.8 )
10        CALL SGSTRN( 2 )
11        CALL SGSTRF
12        CALL UXAXDV( 'B', 5.0, 20.0 )
13        CALL UXAXDV( 'T', 5.0, 20.0 )
14        CALL UXSTTL( 'B', 'LATITUDE', 0.0 )
15        CALL ULISET( 'IYTYPE', 3 )
16        CALL ULYLOG( 'L', 3, 9 )
17        CALL ULYLOG( 'R', 3, 9 )
18        CALL UYSTTL( 'L', 'PRESSURE (MB)', 0.0 )
19        CALL UXMTTL( 'T', 'UXAXDV/ULYLOG', 0.0 )
20        CALL GRCLS
21        END
```

4.5 注文の多い目盛りうち

好みの間隔で目盛りをうち, 好みのところにラベルを描く. しかも, ラベルには数字だけでなく文字列も使いたい. そのような要求にもこのパッケージは対応する.



uxyz04.f:page1

スケーリングは uxyz03.f の例と同じで, 緯度のラベルを文字列で表現したい, また圧力の目盛はデータが存在するレベルだけに打ちたい, というときには `UXAXLB`, `UYAXLB` を用いるとよい. 最初の引数はおなじみの, 場所を指定する引数である. 2, 3 番目の引数には, 小さい目盛りをうつ場所に関する情報を指定する. 具体的には, 2 番目の引数として, 小さい目盛りをうつ場所の U 座標系での値 (この例では緯度) を実数型配列として与える. 3 番目はその配列の大きさである. 4 番目から 7 番目までは, 同様にして, 大きい目盛りについての情報を与える; 目盛りをうつ場所を指定する配列の次に, ラベルを指定する文字型配列, ラベルの文字数, および配列の大きさに関する情報が付け加わる.

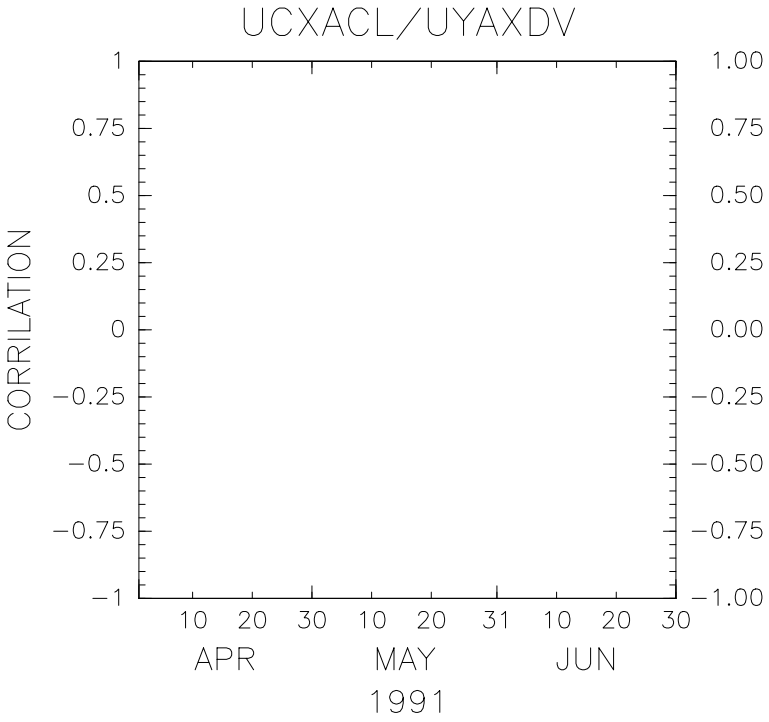
なお, ラベルとして与える文字列の有効な長さは, 後方のブランクを無視して数える. つまり, X 軸の`'EQ'` のように, DATA 文では後方に 2 文字のブランクを含んで文字列を与えても, 先行する有効な 2 文字のみがラベルの作画対象となり, 2 文字分がセンタリングされて作画される. Y 軸に関しても同様に, 文字列`'.4'` は後方のブランクを無視し右寄せして描かれる.

この例の UYAXLB のように, 小さい目盛りを描きたくなかったら, 小さい目盛りの配列を与えるべきところに適当な変数名を書いておいて配列の大きさを 0 とすればよい.

```
*---------------------------------------------------------------------
      PROGRAM UXYZ04
      PARAMETER ( NX1=21, NX2= 5 )
      PARAMETER ( NY1= 0, NY2=18 )
      REAL      RX1(NX1), RX2(NX2), RY2(NY2)
      CHARACTER CX2(NX2)*4, CY2(NY2)*4
      DATA      RX1/-50,-45,-40,-35,-30,-25,-20,-15,-10, -5,  0,
     +               5, 10, 15, 20, 25, 30, 35, 40, 45, 50/
      DATA      RX2/  -40 ,  -20 ,    0 ,   20 ,   40 /
      DATA      CX2/'40S ','20S ','EQ  ','20N ','40N '/
      DATA      RY2/ 1000 , 850  , 700  , 500  , 400  , 300  ,
     +              250  , 200  , 150  , 100  , 70   , 50   ,
     +              30   , 10   , 5    , 2    , 1    , 0.4  /
      DATA      CY2/'1000','    ','    ','500 ','    ','    ',
     +             '    ','200 ','    ','100 ','    ','50  ',
     +             '30  ','10  ','5   ','2   ','1   ','.4  '/
      WRITE(*,*) ' WORKSTATION ID (I) ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL SGSWND( -50.0, +50.0, 1.0E3, 0.4 )
      CALL SGSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL SGSTRN( 2 )
      CALL SGSTRF
      CALL UXAXLB( 'B', RX1, NX1, RX2, CX2, 4, NX2 )
      CALL UXAXLB( 'T', RX1, NX1, RX2, CX2, 4, NX2 )
      CALL UXSTTL( 'B', 'LATITUDE', 0.0 )
      CALL UYAXLB( 'L', DUMMY, NY1, RY2, CY2, 4, NY2 )
      CALL UYAXLB( 'R', DUMMY, NY1, RY2, CY2, 4, NY2 )
      CALL UYSTTL( 'L', 'PRESSURE (MB)', 0.0 )
      CALL UXMTTL( 'T', 'UXAXLB/UYAXLB', 0.0 )
      CALL GRCLS
      END
```

4.6　これはうれしい日付軸

データ解析をした人なら, 日付軸をつけることの面倒くささはご承知であろう. UCPACK を用いると, 閏年まで考慮した完ぺきな日付軸を作画することができる.



uxyz05.f:page1

サブルーチン UCXCAL, UCYCAL が日付に関する座標軸を描く. 最初の引数は, これまで通り, 座標軸を描く場所を指定する. 2 つ目の引数は起日, つまり座標軸を描きはじめる最初の日であるが, どのように与えたらよいかはこの例から明かであろう. 最後の引数で, 何日間分を描くかを指定する. なお日付軸を描くときは, 日数を単位としてウインドウが設定してなければならない. また座標軸の作画は, U 座標系において 0 に相当する位置からおこなわれる. たとえばこの例では, X 方向のウインドウが $[0.0, 90.0]$ として設定してあり, 0 に対応する日付が 1991 年 4 月 1 日であるから, グラフの左端は 1991 年 4 月 1 日の 90 日後の 1991 年 6 月 30 日となる.

なおこの例では, 右側の座標軸にもラベルを描くようにしてみた. これまで, 下と上の軸, 左と右の軸をセットにして描きながら, それぞれ下の軸と左の軸しかラベルが描かれなかった. これは UZ*p*GET/UZ*p*SET の管理する内部変数によって, ラベルを描くか描かないかが制御されていたためである. ここでは, 右側の座標軸に関して'LABELYR' という内部変数を.TRUE. とすることによって, 右側の座標軸についてもラベルを描くよう指定した.

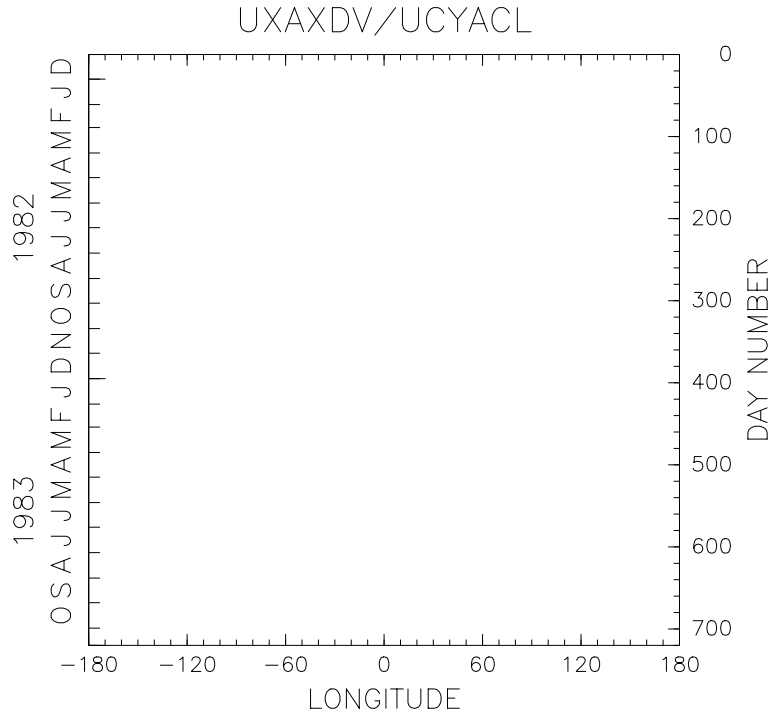
また, つけるラベルのフォーマットを変更することができることも示してみた. X 軸については UXSFMT によって, Y 軸については UYSFMT によって, いわゆる FORTRAN でいうところの文字書式仕様を設定してや

ればよい. 何も設定しない場合 (左の軸) と'(F6.2)' を指定した場合 (右の軸) の違いを見ていただきたい. 何も設定していないときには,「有効数字 3 ケタで文字化し, 小数点以下に続く後方の 0 は取り除く, さらに小数点が最下位ならそれも取り除く」ようなフォーマットが, 内部的に決定されて適用される.

```
*-----------------------------------------------------------------------
      PROGRAM UXYZ05
      PARAMETER ( ID0=19910401, ND=90 )
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      RND=ND
      CALL SGSWND( 0.0, RND, -1.0, +1.0 )
      CALL SGSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      CALL UCXACL( 'B', ID0, ND )
      CALL UCXACL( 'T', ID0, ND )
      CALL UYAXDV( 'L', 0.05, 0.25 )
      CALL UZLSET( 'LABELYR', .TRUE. )
      CALL UYSFMT( '(F6.2)' )
      CALL UYAXDV( 'R', 0.05, 0.25 )
      CALL UYSTTL( 'L', 'CORRILATION', 0.0 )
      CALL UXMTTL( 'T', 'UCXACL/UYAXDV', 0.0 )
      CALL GRCLS
      END
```

4.7 日付軸の例–その 2

日付軸はもちろん Y 方向についても描くことができる.



uxyz06.f:page1

この例からわかるように, 日数のラベルが描けないような場合は, 適当に判断して月と年の目盛だけを描いている. なお, UCXACL/UCYACL の下位ルーチンを呼ぶことによって, 日, 月, 年の座標軸を別々に描くこともできる.

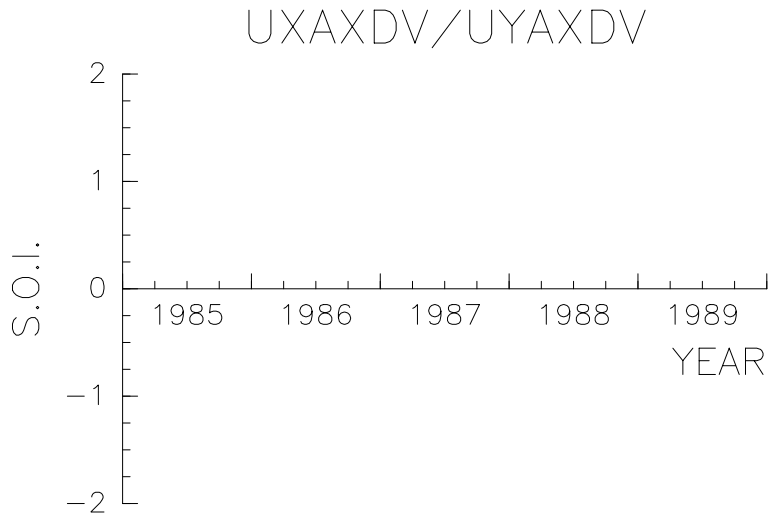
またここでは, uxyz05.f の知識を生かして, 右側の軸は UYAXDV によってラベル付きの異なるタイプの座標軸を描いてみた. さらに UZFACT を呼んで, 文字や目盛のサイズを全体的に小さく (この例では 0.8 倍) して座標軸を描いている.

UZFACT は現在設定されている値を何倍かするルーチンなので, 複数枚の図を描くとき GRFRM を呼ぶたびに UZFACT を呼ぶと, 文字や目盛のサイズがどんどん小さく (あるいは大きく) なるので注意すること.

```
*-----------------------------------------------------------------------
      PROGRAM UXYZ06
      PARAMETER ( ID0=19811201, ND=720 )
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( IWS )
      CALL UZFACT( 0.8 )
      CALL GRFRM
      RND=ND
      CALL SGSWND( -180.0, 180.0, RND, 0.0 )
      CALL SGSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      CALL UXAXDV( 'B', 10.0, 60.0 )
      CALL UXAXDV( 'T', 10.0, 60.0 )
      CALL UXSTTL( 'B', 'LONGITUDE', 0.0 )
      CALL UCYACL( 'L', ID0, ND )
      CALL UZLSET( 'LABELYR', .TRUE. )
      CALL UYAXDV( 'R', 20.0, 100.0 )
      CALL UYSTTL( 'R', 'DAY NUMBER', 0.0 )
      CALL UXMTTL( 'T', 'UXAXDV/UCYACL', 0.0 )
      CALL GRCLS
      END
```

4.8 好みの場所に軸を描く

座標軸は 'B', 'T', 'L', 'R' で指定する以外のところにも描くことができる.



uxyz07.f:page1

座標軸をウインドウ (またはビューポート) の境界線以外のところへ, たとえば, この例のようにグラフのまん中に描きたいときは, 場所をあらわす引数として'U' を指定すれば, それがユーザー指定の座標軸として扱われる. この例では UXAXDV で'U' を指定し, 座標軸を描いている. 具体的にどこに軸を描くかは, UZRSET を用いて, X 軸については内部変数'UYUSER' の値を, Y 軸については内部変数'UXUSER' の値を指定してやればよい. たとえば, この例の X 軸の場合, 'UYUSER' を 0.0 と指定することによって, U 座標系でみた Y 座標の値が 0.0 のところに X 軸が描かれる. タイトルについても, 場所を示す引数を'U' としてやることによって, ユーザー指定の座標軸にタイトルを描くことができる.

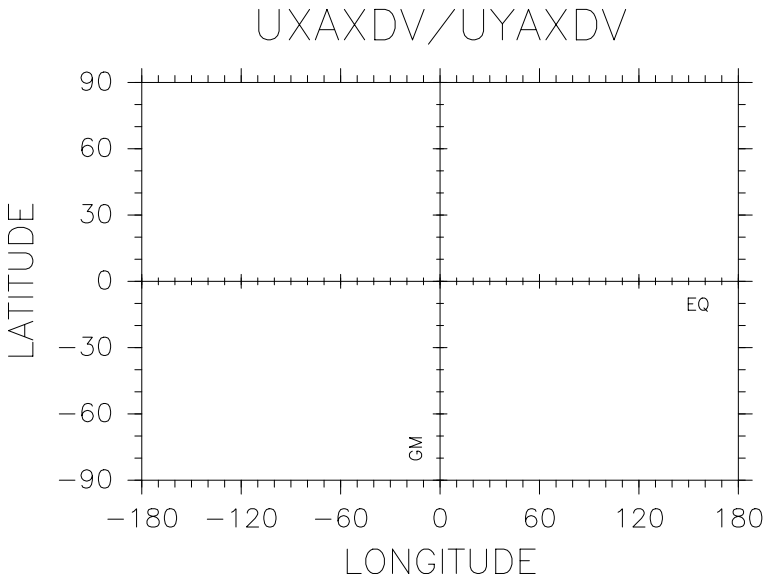
またこの例では, 目盛と目盛の間にラベルを描いてみた. これは UZ*p*GET/UZ*p*SET が管理する内部変数'LBTWN' を.TRUE. とすることによって実現できる. ただし注意すべきことは, ほかの座標軸を描くときにもこの設定が影響を及ぼすので, ラベルを間に描き終ったら.FALSE. に戻しておく必要がある.

さらにこの例では, UXSFMT によってフォーマットを'(I4)' と陽に指定している. これは, 省略時におけるフォーマットが有効数字 3 桁で数値を表現しようとするためであって, このように 4 桁目までちゃんと表現する必要があるときには, ユーザーがフォーマットを指定しなければならない.

```
1   *---------------------------------------------------------------------
2         PROGRAM UXYZ07
3         WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
4         CALL SGPWSN
5         READ(*,*) IWS
6         CALL GROPN( IWS )
7         CALL GRFRM
8         CALL SGSWND( 1985.0, 1990.0, -2.0, +2.0 )
9         CALL SGSVPT( 0.2, 0.8, 0.3, 0.7 )
10        CALL SGSTRN( 1 )
11        CALL SGSTRF
12        CALL UZRSET( 'UYUSER', 0.0 )
13        CALL UZLSET( 'LBTWN', .TRUE. )
14        CALL UXSFMT( '(I4)' )
15        CALL UXAXDV( 'U', 0.25, 1.0 )
16        CALL UZLSET( 'LBTWN', .FALSE. )
17        CALL UXSTTL( 'U', 'YEAR', +1.0 )
18        CALL UYAXDV( 'L', 0.25, 1.0 )
19        CALL UYSTTL( 'L', 'S.O.I.', 0.0 )
20        CALL UXMTTL( 'T', 'UXAXDV/UYAXDV', 0.0 )
21        CALL GRCLS
22        END
```

4.9 目盛を外向きにつける

私は, 目盛を外向きに打つのが好きだという人もいることだろう.



uxyz08.f:page1

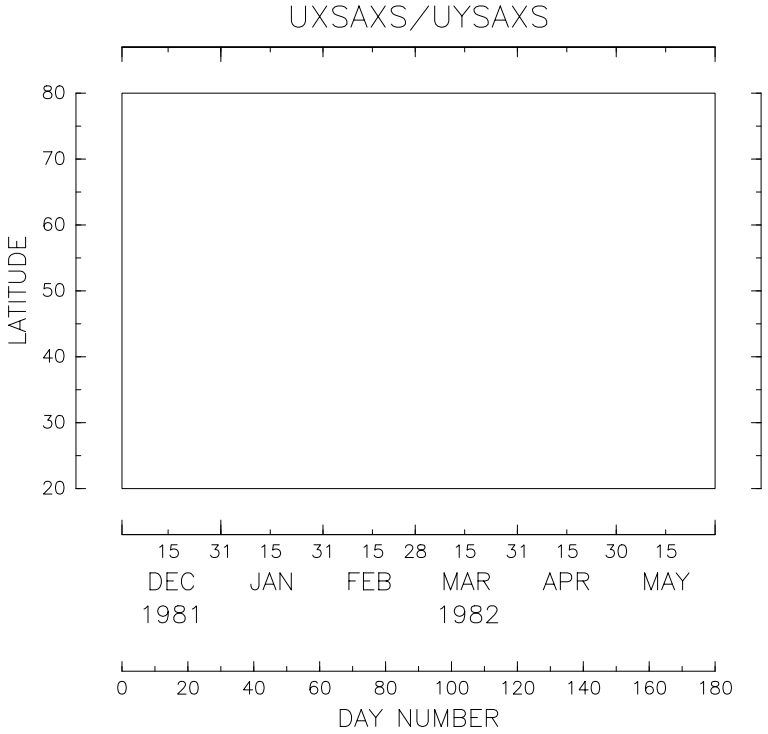
目盛を外向きにつけるときは, UZ*p*GET/UZ*p*SET が管理する内部変数'INNER' を–1 に設定してやればよい. ここではユーザーの指定した場所に座標軸を描きながら目盛を外向きにつける例を示す. まず, UZRSET で内部変数'UXUSER', 'UYUSER' の値をそれぞれ 0.0, 0.0 としている. また, UZLSET で内部変数'LABELXU', 'LABELYU' をそれぞれ.FALSE. にして, ユーザー指定の軸についてはラベルを描かないようにしている.

ユーザー指定の座標軸については 2 度書きすることで軸の両側に目盛を打つようにしている. また UZFACT を呼んで描くタイトルを小さくしている.

```
*---------------------------------------------------------------------
      PROGRAM UXYZ08
      PARAMETER ( X1=-180, X2=+180, DX1=10, DX2=60 )
      PARAMETER ( Y1= -90, Y2= +90, DY1=10, DY2=30 )
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL SGSWND( X1, X2, Y1, Y2 )
      CALL SGSVPT( 0.2, 0.8, 0.3, 0.7 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      CALL UZISET( 'INNER', -1 )
      CALL UZRSET( 'UXUSER', 0.0 )
      CALL UZRSET( 'UYUSER', 0.0 )
      CALL UZLSET( 'LABELXU', .FALSE. )
      CALL UZLSET( 'LABELYU', .FALSE. )
      CALL UXAXDV( 'B', DX1, DX2 )
      CALL UXAXDV( 'T', DX1, DX2 )
      CALL UXSTTL( 'B', 'LONGITUDE', 0.0 )
      CALL UYAXDV( 'L', DY1, DY2 )
      CALL UYAXDV( 'R', DY1, DY2 )
      CALL UYSTTL( 'L', 'LATITUDE', 0.0 )
      CALL UXMTTL( 'T', 'UXAXDV/UYAXDV', 0.0 )
      CALL UZFACT( 0.5 )
      CALL UXAXDV( 'U', DX1, DX2 )
      CALL UXSTTL( 'U', 'EQ', +0.9 )
      CALL UYAXDV( 'U', DY1, DY2 )
      CALL UYSTTL( 'U', 'GM', -0.9 )
      CALL UZISET( 'INNER', +1 )
      CALL UXAXDV( 'U', DX1, DX2 )
      CALL UYAXDV( 'U', DY1, DY2 )
      CALL GRCLS
      END
```

4.10 同じ側に何本もの軸を

ウインドウ (あるいはビューポート) の境界線から少し離して座標軸を描きたい, あるいは同じ側に 2 本以上の座標軸を描きたいということもあろう.



uxyz09.f:page1

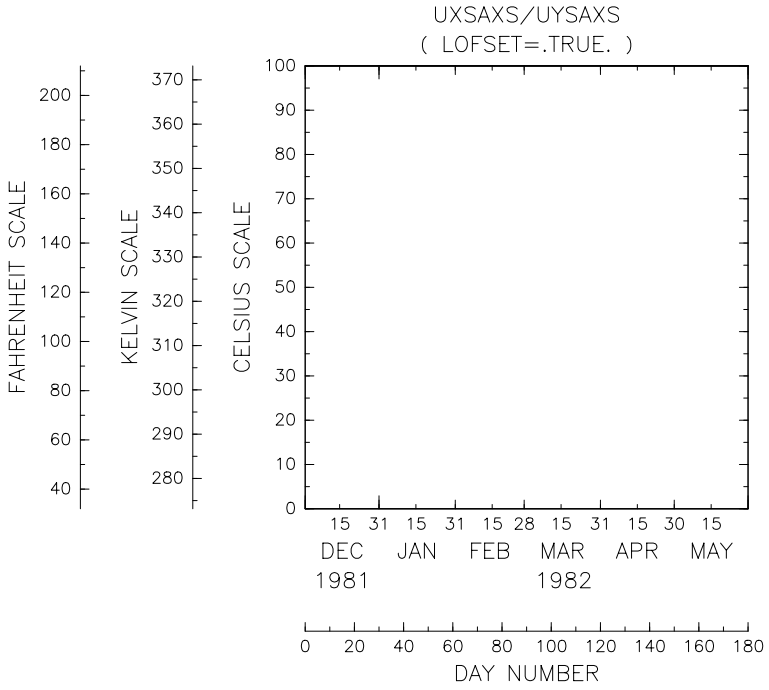
軸を外側へずらしたい, またはすでに描いた軸の外側にもう一本軸を描きたいというときは, X 軸については UXSAXS, Y 軸については UYSAXS を呼べばよい. 1 つだけある引数は, 場所を指定するおなじみの引数である. これらのルーチンを呼ぶと, 次の軸は適度に (内側の軸と重ならない程度に) 外側に描かれる. これを何回も使えば, 簡単に複数の軸を一つの側に描かせることができる.

この例では, 軸が外側へ移ってしまったために, 本来ならグラフの枠は描かれない. しかし, サブルーチン SLPVPR が呼ばれているために, ビューポート, つまりグラフの外側の枠が描かれている.

```
*-----------------------------------------------------------------------
      PROGRAM UXYZ09
      PARAMETER ( ID0=19811201, ND=180, RND=ND )
      PARAMETER ( RLAT1=20, RLAT2=80, DLAT1=5, DLAT2=10 )
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( IWS )
      CALL UZFACT( 0.7 )
      CALL GRFRM
      CALL SGSWND( 0.0, RND, RLAT1, RLAT2 )
      CALL SGSVPT( 0.2, 0.8, 0.4, 0.8 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      CALL SLPVPR( 1 )
      CALL UXSAXS( 'B' )
      CALL UCXACL( 'B', ID0, ND )
      CALL UXSAXS( 'B' )
      CALL UXAXDV( 'B', 10.0, 20.0 )
      CALL UXSTTL( 'B', 'DAY NUMBER', 0.0 )
      CALL UXSAXS( 'T' )
      CALL UCXACL( 'T', ID0, ND )
      CALL UYSAXS( 'L' )
      CALL UYAXDV( 'L', DLAT1, DLAT2 )
      CALL UYSAXS( 'R' )
      CALL UYAXDV( 'R', DLAT1, DLAT2 )
      CALL UYSTTL( 'L', 'LATITUDE', 0.0 )
      CALL UXMTTL( 'T', 'UXSAXS/UYSAXS', 0.0 )
      CALL GRCLS
      END
```

4.11 スケーリングを変えずに別の目盛りを

1 つの側に複数の軸が描けるのはわかったけど, そのためにいちいちウインドウを設定し直すのはめんどくさそう, と思ったあなたに最後の例を示そう.



uxyz10.f:page1

この例では, Y 座標としてセ氏のデータを与えたいのだが, 換算の便のためにカ氏やケルビンの目盛りもあわせて描いている. スケーリングに影響を与えず, 目盛りだけを変えて描きたいときは, オフセット機能を用いる. そのためにまず UZLSET で内部変数'LOFFSET' を.TRUE. にしておく. 必要なところで, 内部変数 'YOFFSET', 'YFACT' (X 軸については'XOFFSET', 'XFACT') を設定すると, 目盛りの位置が平行移動し, 目盛りの間隔が変わる.

```
*-----------------------------------------------------------------------
      PROGRAM UXYZ10
      PARAMETER ( ID0=19811201, ND=180, RND=ND )
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( IWS )
      CALL UZFACT( 0.7 )
      CALL UZLSET( 'LOFFSET',.TRUE.)
      CALL GRFRM
      CALL SGSWND( 0.0, RND, 0.0, 100.0 )
      CALL SGSVPT( 0.4, 0.9, 0.3, 0.8 )
      CALL SGSTRN( 1 )
      CALL SGSTRF
      CALL SLPVPR( 1 )
      CALL UCXACL( 'B', ID0, ND )
      CALL UXSAXS( 'B' )
      CALL UXAXDV( 'B', 10.0, 20.0 )
      CALL UXSTTL( 'B', 'DAY NUMBER', 0.0 )
      CALL UCXACL( 'T', ID0, ND )
      CALL UYAXDV( 'L', 5.0, 10.0 )
      CALL UYAXDV( 'R', 5.0, 10.0 )
      CALL UYSTTL( 'L', 'CELSIUS SCALE', 0.0 )
      CALL UYSAXS( 'L' )
      CALL UZRSET( 'YOFFSET', 273.15 )
      CALL UZRSET( 'YFACT  ', 1.0 )
      CALL UYAXDV( 'L', 5.0, 10.0 )
      CALL UYSTTL( 'L', 'KELVIN SCALE', 0.0 )
      CALL UYSAXS( 'L' )
      CALL UZRSET( 'YOFFSET', 32.0 )
      CALL UZRSET( 'YFACT  ', 1.8 )
      CALL UYAXDV( 'L', 10.0, 20.0 )
      CALL UYSTTL( 'L', 'FAHRENHEIT SCALE', 0.0 )
      CALL UXSTTL( 'T', '( LOFSET=.TRUE. )', 0.0 )
      CALL UXSTTL( 'T', 'UXSAXS/UYSAXS', 0.0 )
      CALL GRCLS
      END
```

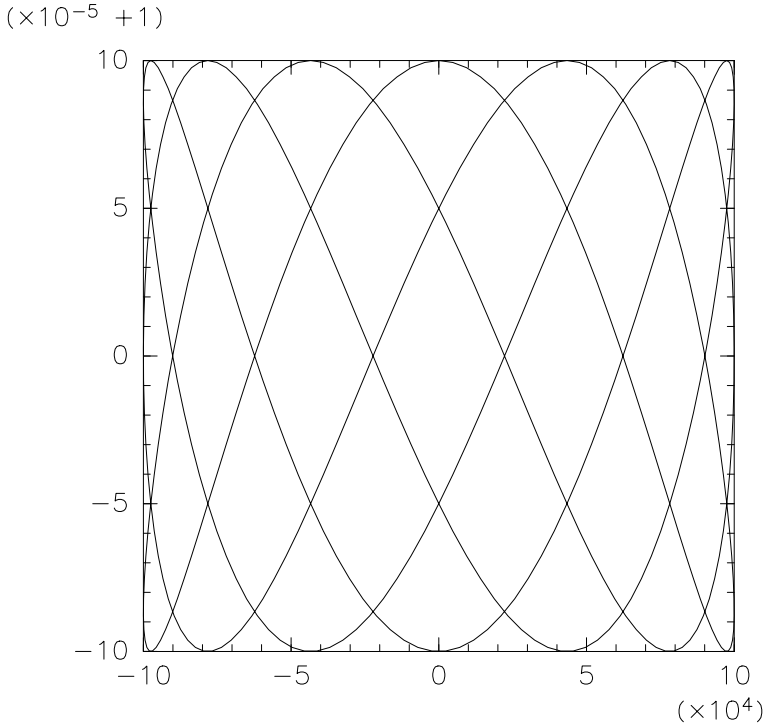
第5章　1次元量の表示

5.1　概要

USPACK の主な機能を使ったプログラム例を以下に示す．　これらのデモプログラムは dcl-x.x/demo/grph2/uspack の中にあるので, 参考にしていただきたい．いくつかのプログラムに関しては, その実行結果として描かれるグラフの顔つきが例示したものと異なる場合があることを注意しておこう．例題として用いた関数形のカオス的な振舞いのために，丸め誤差が大きな結果の差を生むからである．このことは, 関数値の計算精度を単精度・倍精度と変えてみることによって容易に確かめられる．

5.2　とりあえずグラフを描く

一刻も早く計算結果のグラフが見たい!! そんな時 USGRPH は, たったの 1 行 (GROPN 等を含めて 4 行) でデータをグラフ化してくれる．



uspk01.f:page1

しかも, uspk01.f の様に意地の悪いデータが与えられても, Y 軸に 1.00005 等という不細工なラベルを付けて, ラベルが描画範囲を越えてしまう様なことはない. USGRPH は与えられたデータから適当な目盛間隔やラベルの間隔を求めて座標軸を描くが, この時のラベルの文字数が大きすぎる場合には, この例のようにファクター値やオフセット値を適当に選んで軸の端に表示し, ラベルが適当な文字数におさまるようにする. この最大文字数は US*p*GET/US*p*SET が管理する内部変数 'MXDGTX'/'MXDGTY' により設定できる.

この例では, GRPH1/GRPH2 を使う時のお作法である変換関数の設定に関するルーチン群が見あたらない. というのは, USGRPH がその内部で, 与えられたデータから適当な変換関数を設定しているからである. (具体的に, 内部でおこなわれている動作に関しては, 次節以降でおりにふれ説明する.)

なお, 添字には少し小さな字を使うので, 分解能の低いワークステーションではこの添字が識別できないことがあるが, そのような時には GROPN の後に

```
CALL SGLSET('LCNTL', .FALSE.)
```

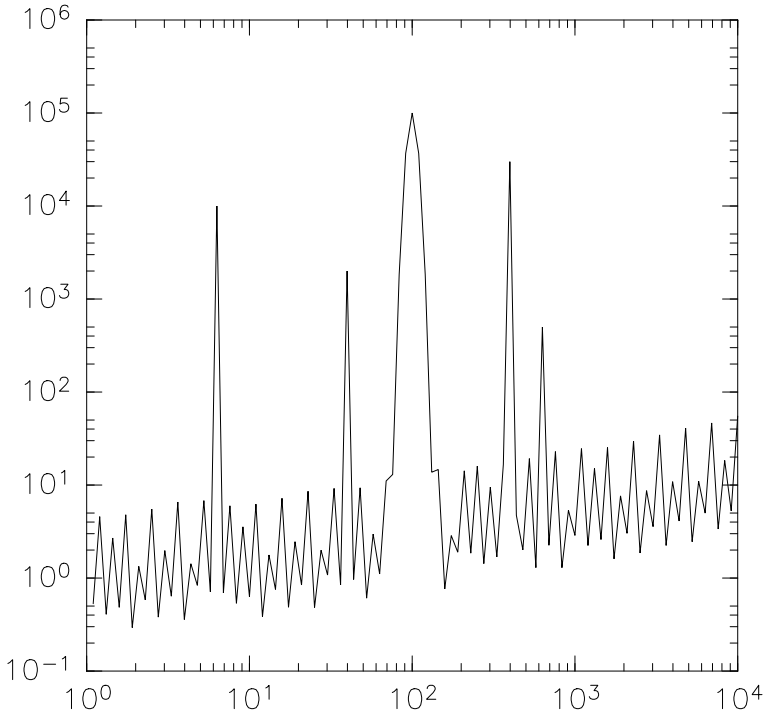
を入れると, 添字を使わない表現になる. この内部変数 'LCNTL' は, SGTXU などで上つき下つきなどの制御コードを有効にするかどうかを指定するものであるが, USGRPH はこの内部変数 'LCNTL' を調べて適切な表現をする. なお, 'LCNTL' の標準値は.FALSE. であるが, GROPN の内部で.TRUE. に設定されている.

また, データの変化がデータ自身の大きさに比べて小さすぎて, 計算誤差程度しかない時は, これを定数と見なし, この定数と 0 を含む様にスケーリングする. (この計算誤差は, GL*p*GET/GL*p*SET が管理する内部変数 'REPSL' をもとに判断する.) さらに, データが全て 0 の場合には最大値・最小値をそれぞれ −1, 1 とする.

```
1   *---------------------------------------------------------------------
2         PROGRAM USPK01
3         PARAMETER (N=400)
4         REAL X(N), Y(N)
5   *--------------------------- DATA DEFINITION --------------------------
6         DT = 3.14159 / (N-1)
7         A = 1.E5
8         B = 1.E-4
9         C = 1.
10        DO 100 I=1, N
11          T = DT*(I-1)
12          X(I) = A*SIN(6.*T)
13          Y(I) = B*COS(14.*T) + C
14    100 CONTINUE
15  *------------------------------ GRAPH ----------------------------------
16        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
17        CALL SGPWSN
18        READ (*,*) IWS
19        CALL GROPN(IWS)
20        CALL GRFRM
21        CALL USGRPH(N, X, Y)
22        CALL GRCLS
23  *---------------------------------------------------------------------
24        END
```

5.3　対数座標のグラフを描く

すぐにグラフ化してくれるのはよいけれど, 私のデータは対数座標で見ないとよくわからない. というスペクトル解析屋さんの要望にも USGRPH は対応する.



uspk02.f:page1

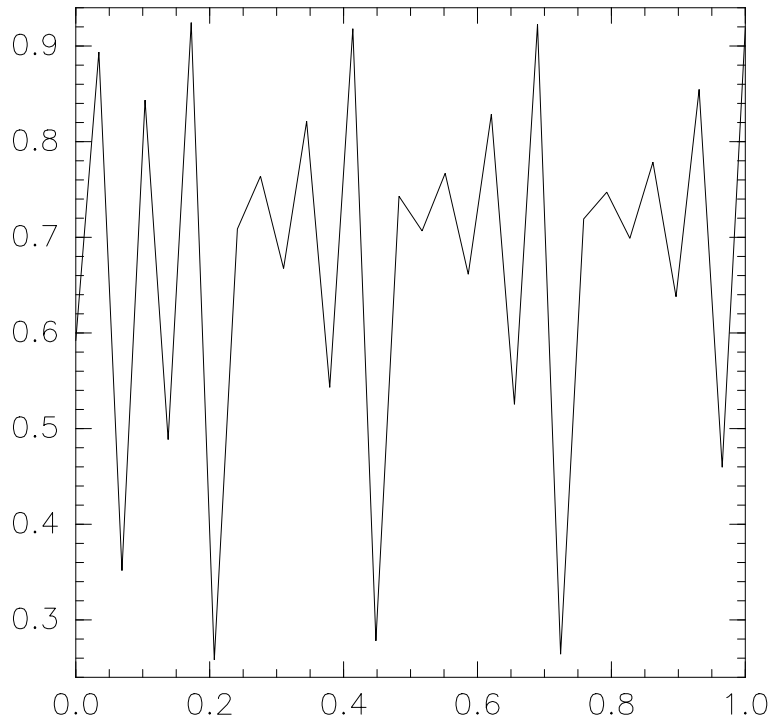
対数座標のグラフを描くためには, USGRPH を呼ぶ前に GRSTRN を 1 行追加して変換関数番号を指定すればよい (変換関数番号については「GRPH1」のマニュアルを参照されたい). ここでは, 変換関数番号として 4 を指定し, 両対数のグラフを作画している. もちろん, この番号を適当に選べば一様座標 (線形座標) との組み合わせも可能である.

この例のように, GRSTRN によって変換関数番号だけを指定して, 変換関数を確定するルーチン GRSTRF (あるいは SGSTRF) を呼ばないのは奇異な感じがするかも知れない. USGRPH のようなお任せルーチンでは, 変換関数を確定させるために必要なパラメータの設定を USPFIT がおこない, そのうえで変換関数の確定を GRSTRF がおこなっている. USPFIT は, ユーザーが指定しなかった変換関数に関する情報を適当に決めてくれるという機能を持っている. 具体的には, GRFRM (あるいは GRFIG) が呼ばれたタイミングで, 変換関数に関する内部変数はすべて「不定」に設定される. このあと, ユーザーが意識的に指定した変換関数に関する情報はそれが有効となり, USPFIT は「不定」のままになっている内部変数を適当に決める. (USPFIT は座標変換に関するパラメータばかりでなく, 目盛の間隔などに関するパラメータをも決めている.)

```
*---------------------------------------------------------------------
      PROGRAM USPK02
      PARAMETER (N=100)
      DOUBLE PRECISION A, R
      REAL X(N), Y(N)
*---------------------------- DATA DEFINITION --------------------------
      R = 0.2D0
      A = 3.6D0
      R0 = 0.
      DO 100 I=1, N
        R = A*R*(1.D0-R)
        R0 = R0 + R*4 - 2.58
        X2 = (I-50)**2
        REXP = 4.*I/N
        X(I) = 10**REXP
        Y(I) = 1.E5*EXP(-X2) + 10.**R0
  100 CONTINUE
      Y(20) = 1.E4
      Y(40) = 2.E3
      Y(65) = 3.E4
      Y(70) = 5.E2
*------------------------------ GRAPH ----------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
      CALL GRSTRN(4)
      CALL USGRPH(N, X, Y)
      CALL GRCLS
*---------------------------------------------------------------------
      END
```

5.4　1 変数のグラフを描く

2 次元のグラフと言っても片側の座標値は等間隔なので, そのためにわざわざ配列を宣言して座標値を代入するのはするのはめんどくさい, ということがある. そんなときには, 等間隔な座標軸に関するウインドウ情報を指定した上で, USGRPH に与える引数のうちその座標軸方向ものを「不定」にしてやればよい.



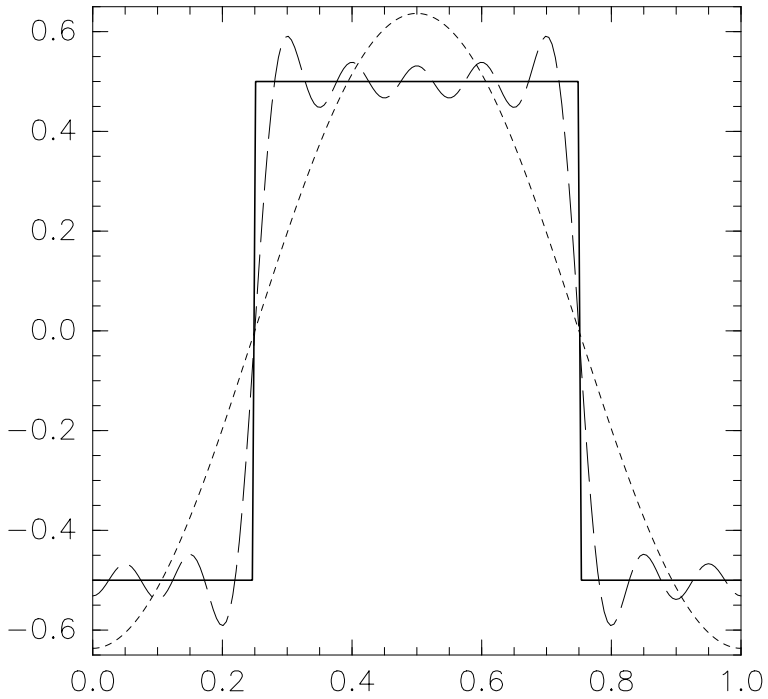
uspk03.f:page1

この例では USPFIT のパラメータ補完機能を使う. まず, GRSWND で等間隔とみなす X 軸方向のウインドウ情報を指定する (このとき, Y 軸方向は実際のデータにもとづいてスケーリングしたいので「不定」としておく). そのうえで USGRPH の X 軸方向の引数を不定としてやることによって, たとえばこの例では, X 方向には [0.0, 1.0] の範囲で等間隔に, Y 方向には配列 Y で与えられた座標値を折れ線で描く.

```
*-----------------------------------------------------------------------
      PROGRAM USPK03
      PARAMETER (N=30)
      DOUBLE PRECISION A, R
      REAL Y(N)
*---------------------------- DATA DEFINITION --------------------------
      R = 0.2D0
      A = 3.7D0
      DO 100 I=1, N
        R = A*R*(1.D0-R)
        Y(I) = R
  100 CONTINUE
*------------------------------ GRAPH ----------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GLRGET('RUNDEF',RUNDEF)
      CALL GROPN(IWS)
      CALL GRFRM
      CALL GRSWND(0.0, 1.0, RUNDEF, RUNDEF)
      CALL USGRPH(N, RUNDEF, Y)
      CALL GRCLS
*-----------------------------------------------------------------------
      END
```

5.5　複数のデータを 1 つのグラフに描く

とにかく計算はできるようになったのでデータはどんどん出てくるが, 一つのグラフに一つのデータでは紙がいくらあっても足りない. かといって, USGRPH の後で SGPLU あるいは UULIN (これは片方の座標軸の点列情報を等間隔とみなして折れ線を描く機能を持った, 基本的には SGPLU と同じような動作をする作画ルーチンである) 等を使って折れ線だけ追加すると, 座標軸からはみ出してしまう可能性もある.



uspk04.f/uspk05.f:page1

uspk04.f はこのような場合のプログラム例である. USSPNT は, X, Y 軸のスケーリング範囲に含めるべきデータを指定して作画範囲を決める. このルーチンは USGRPH を呼ぶ前に何度でも呼ぶことができる. (USSPNT は X あるいは Y 軸方向のどちらかを「不定」として片側のみのスケーリングもできる.) USGRPH はこのルーチンで指定したデータを含むようにスケーリングを行うので, USGRPH の後で SGPLU あるいは UULIN を使って折れ線を追加しても, 座標軸からはみ出すことはない.

また, USGRPH では折れ線のラインインデックスやラインタイプを直接指定することはできないが, USGRPH は単に UULIN を呼んでいるだけなので, あらかじめ UUSLNT/UUSLNI でこれらを指定しておけば, USGRPH で描かれる折れ線の属性を変えることができる.

ところで, uspk04.f で描かれる 3 本の線はどれも同じ様な折れ線でありながら, 1 本は USGRPH で, あとの 2 本は UULINZ で描く, というのは何となく不平等な感じで気持ちが悪い. そんな時, USGRPH をばらして使ってやれば気持ちのよいプログラムが書ける. uspk05.f は uspk04.f と全く同じ結果を得るプログラムである.

USDAXS は, 現在設定されている正規化変換に対して, とにかく適当な座標軸 (デフォルト座標軸) を描くものである. ここで描かれる座標軸の位置は USCSET の内部変数'CXSIDE', 'CYSIDE' で指定され, 初期値はそれぞれ'BT', 'LR' である. したがって, USDAXS 一行で 4 本の座標軸が描かれることになる.

USDAXS を使うと, データを与える部分と折れ線を描く部分が完全に独立して書けるので, よりきれいなプログラムとなり, DO ループで複数の折れ線を書くときなどは便利である. また, この書き方では折れ線を描かずに, SGPMU あるいは UUMRK 等でマーカー列を描くことも自由にできる. ただし, USDAXS は単に座標軸を描くだけのルーチンなので, その前に USPFIT によって変換関数に必要なパラメータの設定, および GRSTRF によって変換関数の確定をしておく必要がある.

以降の例題ではもっぱら USGRPH を用いているが, uspk05.f の様にこれを USSPNT, USPFIT, GRSTRF, USDAXS および UULIN に分解しても全く同じ結果が得られる. (事実, USGRPH もこれらの 4 行で構成されている.)

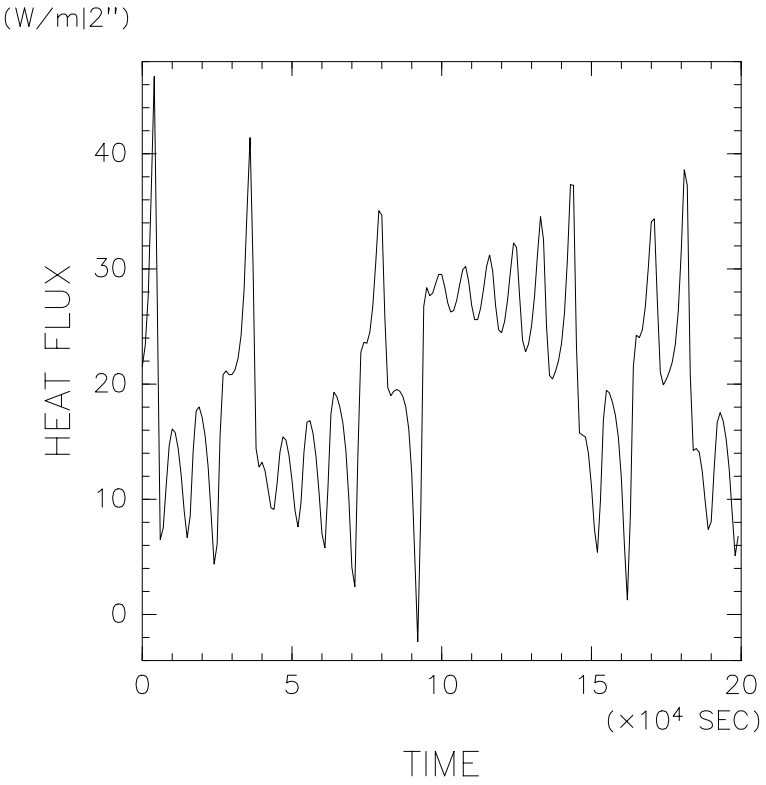
なお, USSPNT で設定された値は, USGRPH または USDAXS を実行した際にリセットされる.

```
*-----------------------------------------------------------------------
      PROGRAM USPK04
      PARAMETER(N=200, M=5)
      REAL X(N), Y0(N), Y1(N), Y2(N), A(M)
*-----------------------------------------------------------------------
      DT = 1./(N-1)
      PI = 3.14159
      DO 50 J=1, M
        JJ = J*2-1
        A(J) = (-1)**J *2./(JJ*PI)
 50   CONTINUE
      DO 100 I=1, N
        T    = DT*(I-1)*2*PI
        X(I) = DT*(I-1)
        Y2(I) = 0.
        DO 150 J=1, M
          JJ = J*2-1
          YY = A(J)*COS(JJ*T)
          Y2(I)  = Y2(I) + YY
 150    CONTINUE
        Y1(I) = A(1)*COS(T)
        IF(T.LT.PI/2. .OR. T.GE.PI*3./2.) THEN
          Y0(I) = -0.5
        ELSE
          Y0(I) = 0.5
        ENDIF
 100  CONTINUE
*-----------------------------------------------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
      CALL USSPNT(N, X, Y1)
      CALL USSPNT(N, X, Y2)
      CALL UUSLNI(5)
      CALL USGRPH(N, X, Y0)
      CALL UULINZ(N, X, Y1, 3, 1)
      CALL UULINZ(N, X, Y2, 2, 2)
      CALL GRCLS
      END
```

```
*---------------------------------------------------------------------
      PROGRAM USPK05
      PARAMETER(N=200, M=5)
      REAL X(N), Y0(N), Y1(N), Y2(N), A(M)
*---------------------------------------------------------------------
      DT = 1./(N-1)
      PI = 3.14159
      DO 50 J=1, M
        JJ = J*2-1
        A(J) = (-1)**J *2./(JJ*PI)
  50  CONTINUE
      DO 100 I=1, N
        T    = DT*(I-1)*2*PI
        X(I) = DT*(I-1)
        Y2(I) = 0.
        DO 150 J=1, M
          JJ = J*2-1
          YY = A(J)*COS(JJ*T)
          Y2(I)  = Y2(I) + YY
  150   CONTINUE
        Y1(I) = A(1)*COS(T)
        IF(T.LT.PI/2. .OR. T.GE.PI*3./2.) THEN
          Y0(I) = -0.5
        ELSE
          Y0(I) = 0.5
        ENDIF
  100 CONTINUE
*---------------------------------------------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
      CALL USSPNT(N, X, Y0)
      CALL USSPNT(N, X, Y1)
      CALL USSPNT(N, X, Y2)
      CALL USPFIT
      CALL GRSTRF
      CALL USDAXS
      CALL UULINZ(N, X, Y0, 1, 5)
      CALL UULINZ(N, X, Y1, 3, 1)
      CALL UULINZ(N, X, Y2, 2, 2)
      CALL GRCLS
      END
```

5.6　タイトルを描く

とにかくグラフは描けるようになった. しかし, 座標軸にタイトルがないと何のグラフだったかわからなくなってしまう.



uspk06.f:page1

USSTTL を使えば, X 軸と Y 軸のタイトルや単位を描くことができる. これらは, 以下に述べるように USCGET/USCSET が管理する内部変数を設定してやることによっても指定できる. また, USSTTL や USCSET を使わずに, USGRPH を呼んだ後で UXPACK/UYPACK のタイトル作画ルーチンを使って適当なタイトルを付けてもよい.

USGRPH で座標軸にタイトルを付けるには, USGRPH の前に USCSET を使って, 内部変数'CXTTL', 'CYTTL' を設定しておけばよい (タイトルは 80 文字まで設定できる).

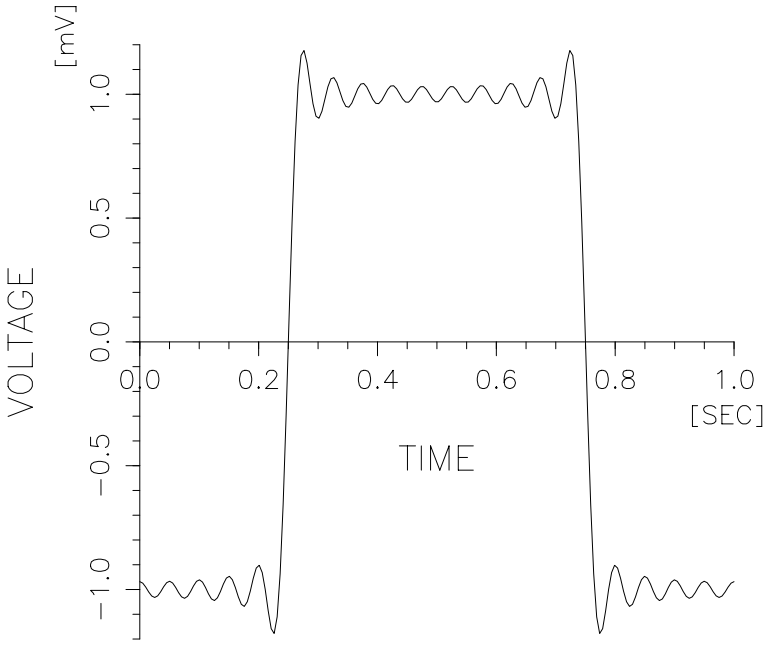
また, USGRPH ではファクター値やオフセット値を描くために USXSUB/USYSUB というサブラベル (座標軸の端に付け足すラベル) を描くためのルーチンを用意したので, ついでに座標軸の単位などもこのサブラベルに含めることができるようにしてある. これも, タイトルと同様に USCSET で'CXUNIT', 'CYUNIT' を設定すればよい. ここで指定した文字列は, ファクター値やオフセット値がある場合これらの後に書かれる.

なお, タイトル等は GRFRM, GRFIG により初期化される. 同じタイトルで沢山の図を描きたい時には USISET で'IRESET' を 0 にすれば, タイトルの設定は 1 度で済む.

```
*---------------------------------------------------------------------
      PROGRAM USPK06
      PARAMETER(N=200)
      DOUBLE PRECISION X, Y, Z, DX, DY, DZ, DT, S, R, B
      REAL T(N), A(N)
      DATA    X,    Y,    Z,     S,     R,     B,     DT /
     #     0.D0, 1.D0, 1.D0, 10.D0, 26.D0, 2.6D0, 0.01D0 /
*---------------------------------------------------------------------
      DO 10 I=1, N
        DO 20 J=1, 8
          DX = -S*X + S*Y
          DY = -X*Z + R*X - Y
          DZ =  X*Y - B*Z
          X = X + DX*DT
          Y = Y + DY*DT
          Z = Z + DZ*DT
   20   CONTINUE
        T(I) = (I-1)*1000
        A(I) = Y + 20.
   10 CONTINUE
*---------------------------------------------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
      CALL USSTTL('TIME', 'SEC', 'HEAT FLUX', 'W/m|2"')
      CALL USGRPH(N, T, A)
      CALL GRCLS
      END
```

5.7　座標軸のスタイルを変える

とにかくデータは描けるようになったが, いつも真四角な箱型の座標軸ではつまらない. と, 人間だんだん欲が出てくることを USPACK はちゃんと見通している. というより, USPACK は座標軸の細かな設定に関しては, かなりの部分で UXPACK/UYPACK に「よきに計らえ」と任せているので, 細かなことはこれらをコントロールする UZPACK にお願いすれば, いろいろと面倒を見てくれる. 裏を返せば, 細かいことまで思い通りにしたければ USPACK を呼ぶ前に, UZPACK にちゃんと根回しをしておかなければならない.



uspk07.f:page1

まず, 座標軸を描く位置は USPACK が管理しているので, USCGET/USCSET の内部変数'CXSIDE'(X 軸) と'CYSIDE'(Y 軸) を指定する. 'CXSIDE', 'CYSIDE' は長さ 2 の文字型変数で, 'CXSIDE' は'T'(Top), 'B'(Bottom), 'U'(User) のうち 2 文字まで, 'CYSIDE' は'R'(Right), 'L'(Left), 'U'(User) のうち 2 文字まで指定でき, 初期値はそれぞれ'BT', 'LR' である. これらの文字は 1 文字づつ座標軸ルーチン (UXAXDV/UYAXDV) に渡される. 後はこれらのルーチンの仕事である.

uspk07.f の様に'CXSIDE' に'U' を指定した場合には, UXPACK に「ユーザー定義の位置」を通知しておかなければ UXPACK が困ってしまう. そこで UZRSET で内部変数'UYUSER' を 0. にして, X 軸を Y=0 の位置に描くように指定する. この時, USGRPH は Y 軸のスケーリングに関して, X 軸を描く位置 (Y=0) を含めてスケーリングするので, 仮にデータが 0 を含まなくても座標軸がとんでもないところに描かれることはない.

その他, uspk07.f の中では UZISET で 'IROTLYL' と'ICENTL' を変更して, Y 軸のラベルを軸に平行にし, 'INNER' を −1 にして目盛を外向きにしている.

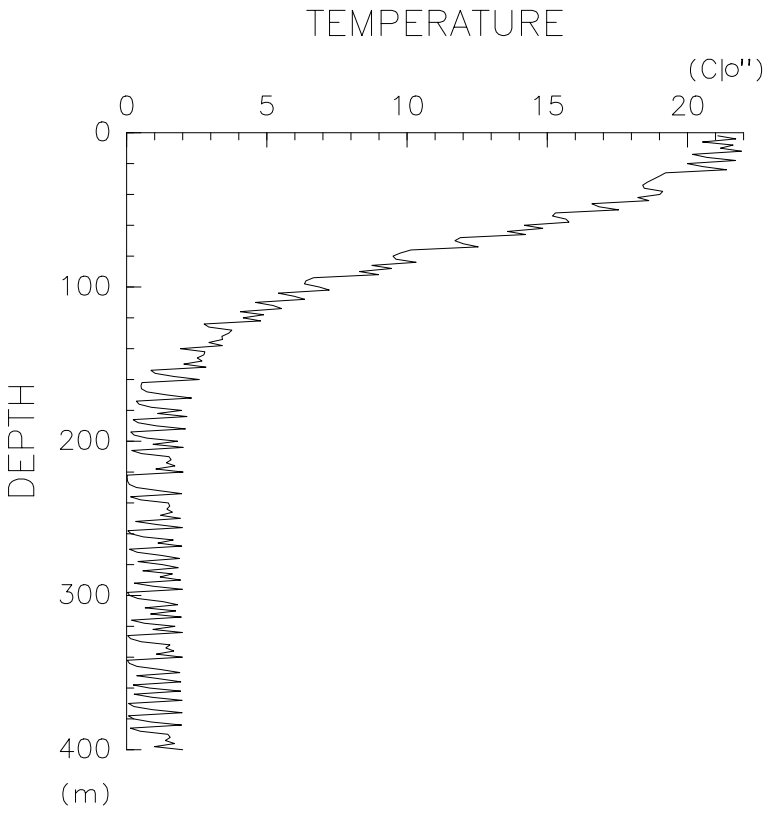
なお, 座標軸のタイトルとサブラベルに関しては USPACK が管理しており, 'CXSIDE', 'CYSIDE' に指定された最初の文字の位置にこれらを描くことになっている. (サブラベルの向きは UZISET の'IROTL*zs*' に従う) また, サブラベルの文字を囲む括弧が気に入らない場合には, USCSET で内部変数'CBLKT' を指定することによって変更することができる. この変数に空白を指定すると括弧は描かない.

これら US*p*SET/USCSET 及び UZ*p*SET で設定された座標軸のスタイルに関するパラメタは, 再設定されるまで有効である. タイトルとスケーリングに関するパラメタは GRFRM または GRFIG によって初期化される. 詳しい内部変数名およびリセットされるタイミングは, 「GRPH2」マニュアルの USPACK の章を参照のこと.

```
*-----------------------------------------------------------------------
      PROGRAM USPK07
      PARAMETER(N=200, M=10)
      REAL X(N), Y(N), A(M)
*-----------------------------------------------------------------------
      DT = 1./(N-1)
      PI = 3.14159
      DO 50 J=1, M
        JJ = J*2-1
        A(J) = (-1)**J *2./(JJ*PI)
 50   CONTINUE
      DO 100 I=1, N
        T    = DT*(I-1)*2*PI
        X(I) = DT*(I-1)
        YY = 0.
        DO 150 J=1, M
          JJ = J*2-1
          YD = A(J)*COS(JJ*T)
          YY  = YY + YD
 150    CONTINUE
        Y(I) = YY*2.
 100  CONTINUE
*-----------------------------------------------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
*           --- X AXIS ---
      CALL UZRSET('UYUSER'  , 0.)
      CALL USCSET('CXSIDE'  , 'U')
*           --- Y AXIS ---
      CALL UZISET('IROTLYL' , 1)
      CALL UZISET('ICENTYL' , 0)
      CALL USCSET('CYSIDE'  , 'L')
*           --- etc. ---
      CALL UZISET('INNER' , -1)
      CALL USCSET('CBLKT' , '[]')
      CALL USSTTL('TIME', 'SEC', 'VOLTAGE', 'mV')
      CALL USGRPH(N, X, Y)
      CALL GRCLS
      END
```

5.8　座標軸を逆さまにする

これまでの X 軸は右向き, Y 軸は上向きを正としてきたが, USLSET で内部変数'LXINV', 'LYINV' を .TRUE. にするとそれぞれの軸の方向を逆にすることができる. これは, 常に世の中を逆さまに見ている人々のための機能である.



uspk08.f:page1

しかし, この機能を使うにはちょっと注意が必要である. 問題は, この例のように Y 軸を下向きにした場合, X 軸をグラフの上側に描くことから生じる. X 軸をグラフの上側に描くには uspk07.f で説明したように

```
CALL USCSET('CXSIDE', 'T')
```

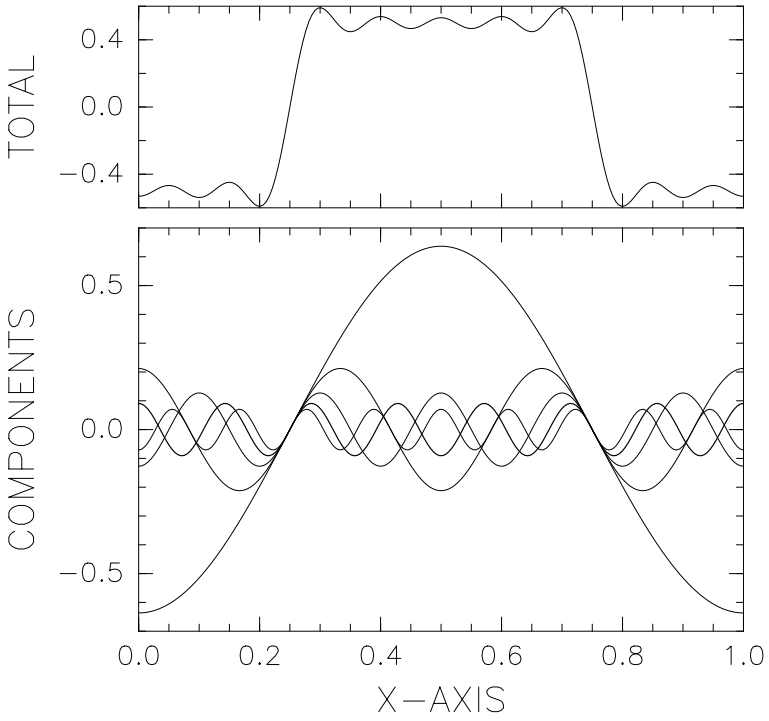
とすればよいように思えるが, これだけでは X 軸そのもの (線分と目盛) は描いても, ラベルやタイトルなどが一切出力されない. これは, 座標軸とラベルを描くルーチン UXAXDV/UYAXDV が UZPACK の内部変数 'LABEL*zs*' を参照して, 場所に応じてラベルを描くかどうか判断しており, 初期状態ではグラフの上側と右側にラベルを描かないようになっているためである. (これまで箱型の座標軸を描いたときに, 上側や右側の軸にはラベルやタイトルがついていなかったことを思い出して欲しい.) ラベルに関してはほとんど UZPACK の管理下にあるので, 当然 USPACK が描くラベルもこの内部変数に自動的に従うことになる. これに対して, サブラベルとタイトルは USPACK の管理下にあり, USCSET の内部変数'CXSIDE', 'CYSIDE' で指定される 1 文字目の場所にサブラベル (単位等) とタイトルを描くことになっている. しかしながら, ラベルがないところ

にサブラベルやタイトルを描いてもしかたがないので, USPACK は UZPACK の管理情報を盗み見して, 内部変数'LABEL*zs*' が.FALSE. のところにはこれらを描かないことにしている. 要するに, ラベルやタイトルなど座標軸に付属する文字はすべて'LABEL*zs*' によって制御されているので, グラフの上側や右側に座標軸とタイトルを出力するためには, あらかじめ UZLSET で'LABEL*zs*' を.TRUE. にしておけばよい.

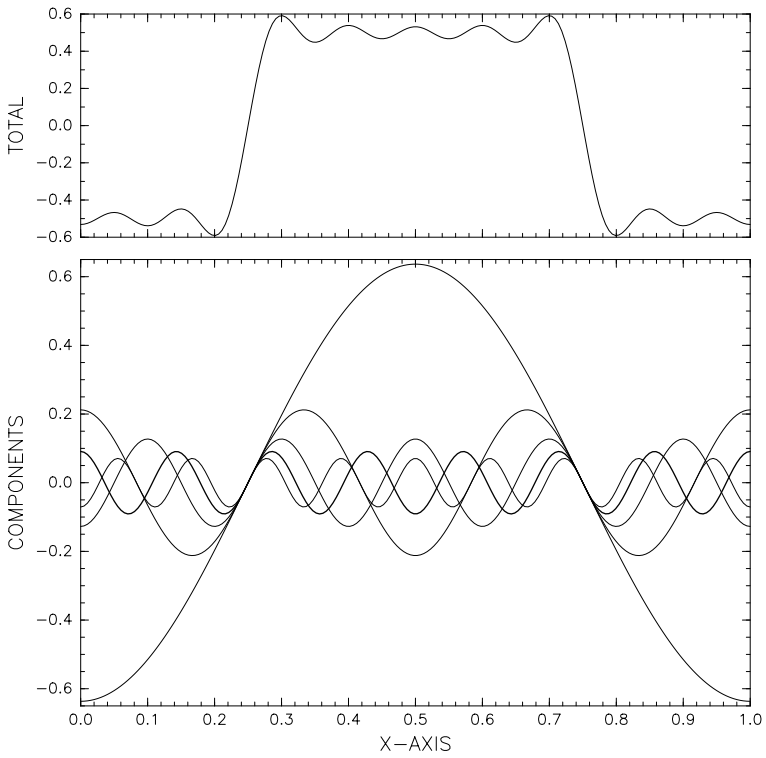
```
*---------------------------------------------------------------------
      PROGRAM USPK08
      PARAMETER(N=200)
      CHARACTER USGI*3
      REAL T(N), Z(N)
      DOUBLE PRECISION R, A
      R = 0.2D0
      A = 4.0D0
      DO 10 I=1, N
        R = A*R*(1.D0-R)
        Z2  = (FLOAT(I-5)/40.)**2
        T(I) = 20.*EXP(-Z2) + R*2.
        Z(I) = I*2
   10 CONTINUE
*---------------------------------------------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
*          --- OMAJINAI ---
      CALL USLSET('LYINV'   , .TRUE.)
      CALL UZLSET('LABELXT' , .TRUE.)
      CALL USCSET('CYSPOS'  , 'B')
      CALL USCSET('CXSIDE'  , 'T')
      CALL USCSET('CYSIDE'  , 'L')
      CALL USSTTL('TEMPERATURE', 'C|'//USGI(4)//'"', 'DEPTH', 'm')
      CALL USGRPH(N, T, Z)
      CALL GRCLS
      END
```

5.9　グラフの大きさ, 位置を変える

USGRPH は特に指定しなければ画面の縁から適当なマージンをとってビューポートを設定するが, 1 ページに複数のグラフを並べて描きたいときにはこれを変更する必要がある.



uspk09.f:page1



uspk09.f:page2

まず, ビューポートを変更するには USGRPH の前に, GRSVPT でビューポートの座標を指定する. この例では 1 ページの下側に 1 枚目のグラフを描く. 2 枚目のグラフを描く手順も同様であるが, 2 枚目のグラフを描く前に GRFIG で UZPACK 等の初期化をしなければならない. また, この例では X 軸のラベルは 1 枚目のグラフと同じなので, UZLSET で'LABELXB' を .FALSE. にして X 軸のラベルを抑制している. この例では, 上下のグラフは同じ大きさであるが, 異なる大きさのグラフを描いてもラベルの文字の大きさが変わることはない.

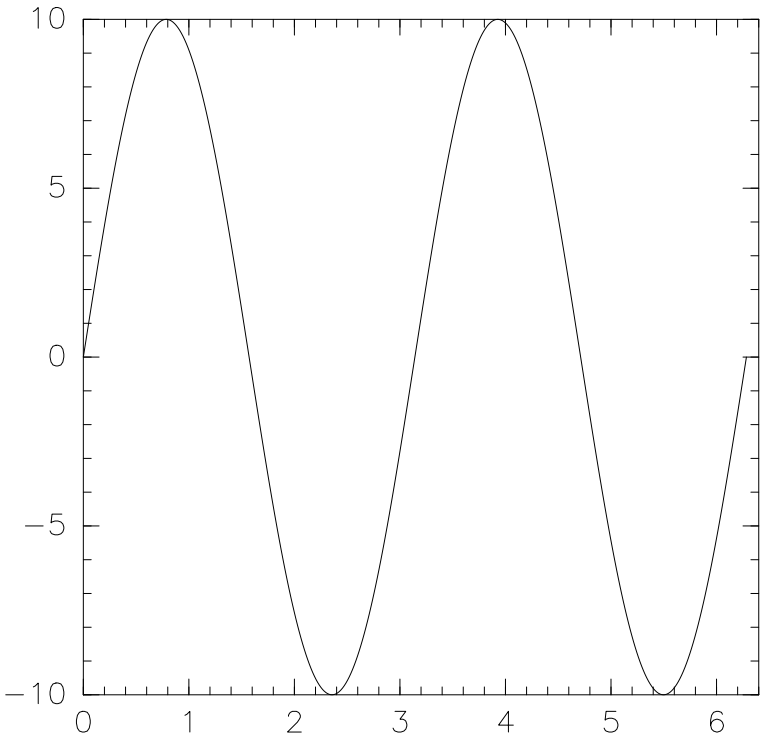
改ページをして次のページにグラフを描くには, GRFRM を呼ぶ. GRFRM は GRFIG で行われる処理を全て行うので GRFIG をあらためて呼ぶ必要はない. 複数のグラフを 1 ページに収めるとラベルの文字が大きすぎることがあるが, そのような時には, UZFACT で文字の大きさを変えることができる. その場合, 文字の大きさに応じて目盛やラベルの間隔も変わる. (USGRPH では, 文字の大きさを基準にしてスケーリングを行っている)

なお, GRSVPT でビューポートを設定しないときには, 文字幅に内部変数'RMRGN' をかけた幅のマージンをとるので, 文字幅によってビューポートの大きさも変わる. 初期状態では, 上下左右に 0.2 のマージンをとる.

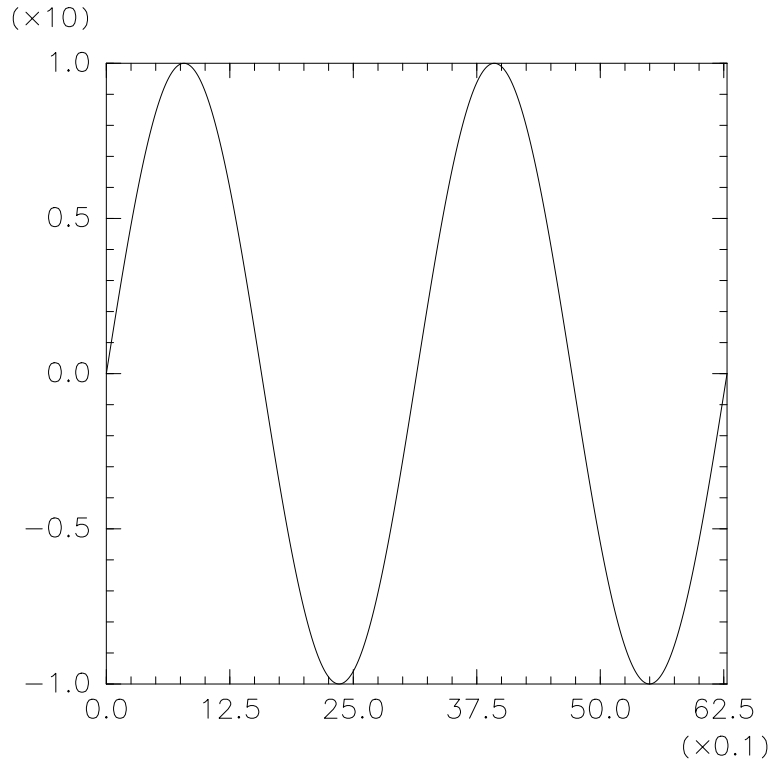
```
*-----------------------------------------------------------------------
      PROGRAM USPK09
      PARAMETER(N=200, M=5)
      REAL X(N), Y(N), YC(N,M), A(M)
      DT = 1./(N-1)
      PI = 3.14159
      DO 50 J=1, M
        JJ = J*2-1
        A(J) = (-1)**J *2./(JJ*PI)
 50   CONTINUE
      DO 100 I=1, N
        T    = DT*(I-1)*2*PI
        X(I) = DT*(I-1)
        Y(I) = 0.
        DO 150 J=1, M
          JJ = J*2-1
          YC(I,J) = A(J)*COS(JJ*T)
          Y(I)  = Y(I) + YC(I,J)
 150    CONTINUE
 100  CONTINUE
*-------------------------- 1ST PAGE ---------------------------------
      CALL SWCSTX('FNAME','USPK09')
      CALL SWLSTX('LSEP',.TRUE.)
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GLRGET('RUNDEF',RUNDEF)
      CALL GROPN(IWS)
      CALL GRFRM
      CALL GRSVPT(0.2, 0.8, 0.2, 0.6)
      CALL USSPNT(N*M, RUNDEF, YC)
      CALL USSTTL('X-AXIS', ' ', 'COMPONENTS', ' ')
      CALL USGRPH(N, X, YC)
      DO 200 J=2,M
        IP = MOD(J-1,4) + 1
        CALL UULINZ(N, X, YC(1,J), 1, IP)
 200  CONTINUE
*     --- NEW FIG ---
      CALL GRFIG
      CALL GRSVPT(0.2, 0.8, 0.62, 0.82)
      CALL UZLSET('LABELXB', .FALSE.)
      CALL USSTTL('X-AXIS', ' ', 'TOTAL', ' ')
      CALL USGRPH(N, X, Y)
*-------------------------- 2ND PAGE ---------------------------------
      CALL GRFRM
      CALL GRSVPT(0.2, 0.8, 0.2, 0.6)
      CALL UZFACT(0.5)
      CALL UZLSET('LABELXB', .TRUE.)
      CALL USSPNT(N*M, RUNDEF, YC)
      CALL USSTTL('X-AXIS', ' ', 'COMPONENTS', ' ')
      CALL USGRPH(N, X, YC)
      DO 300 J=2,M
        IP = MOD(J-1,4) + 1
        CALL UULINZ(N, X, YC(1,J), 1, IP)
 300  CONTINUE
*     --- NEW FIG ---
      CALL GRFIG
      CALL GRSVPT(0.2, 0.8, 0.62, 0.82)
      CALL UZLSET('LABELXB', .FALSE.)
      CALL USSTTL('X-AXIS', ' ', 'TOTAL', ' ')
      CALL USGRPH(N, X, Y)
      CALL GRCLS
      END
```

5.10　スケーリングを制限する

USGRPH によるスケーリングが気に入らないとき, 特定のパラメータを強制的に固定することも可能である.



uspk10.f:page1



uspk10.f:page2

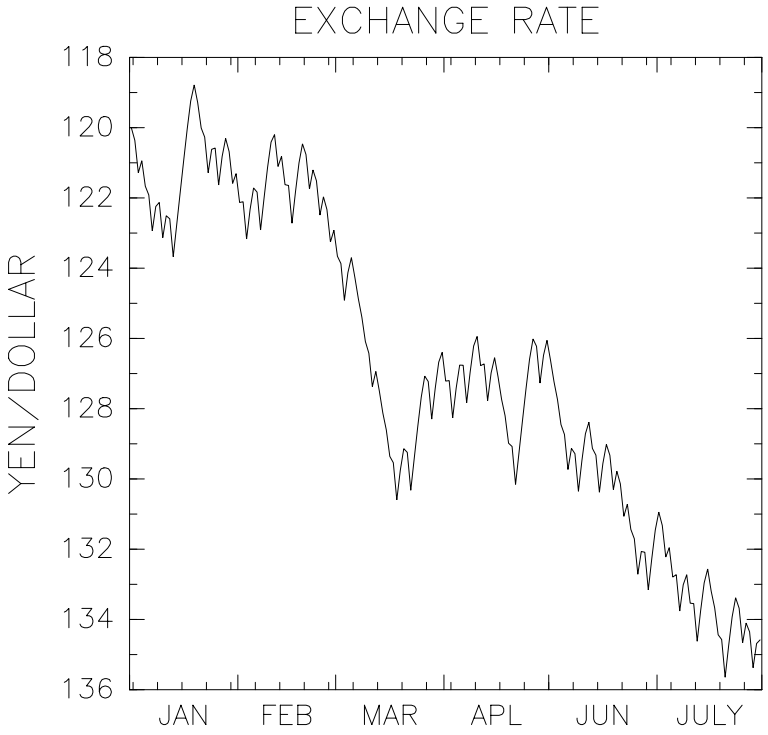
例えばデータの X 座標の範囲が中途半端な時に, 何も指定しなければ USGRPH は 1 ページ目のように座標軸の両端に「きりのよい値」が来るようにスケーリングしてしまい, 折れ線が「しりきれとんぼ」になってしまう.

このようなスケーリングで具合が悪いときには, まず GRSWND でウインドウを指定し (この例では, X 座標のみ陽に指定し, Y 座標「不定」を示す値を指定して USPACK がデータから判断するようにしている), さらには USRSET で目盛/ラベルの間隔だけでなく, ラベルのファクターまで制御することができる. この例の 2 ページ目では X 軸に関するパラメータ全てを陽に指定して自動スケーリング機能を完全に抑制している.

```
*-----------------------------------------------------------------------
      PROGRAM USPK10
      PARAMETER (N=400)
      REAL X(N), Y(N)
      PI = 3.14159
      DT = 2*PI/(N-1)
      DO 100 I=1, N
        T = DT*(I-1)
        X(I) = T
        Y(I) = 10.*SIN(2.*T)
  100 CONTINUE
*--------------------------- 1ST PAGE ---------------------------------
      CALL SWCSTX('FNAME','USPK10')
      CALL SWLSTX('LSEP',.TRUE.)
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
      CALL USGRPH(N, X, Y)
*--------------------------- 2ND PAGE ---------------------------------
      CALL GLRGET('RUNDEF', RUNDEF)
      CALL GRFRM
      CALL GRSWND(0., 2.*PI, RUNDEF, RUNDEF)
      CALL USRSET('DXL' , 1.25)
      CALL USRSET('DXT' , 0.25)
      CALL USRSET('XFAC', 0.1)
      CALL USRSET('YFAC', 10.)
      CALL USGRPH(N, X, Y)
*-----------------------------------------------------------------------
      CALL GRCLS
      END
```

5.11 Y 軸だけ USPACK を使う

Y 軸のデータ範囲などは自動的に決めて欲しいけど, X 軸に関しては自分の好みの座標軸を描きたい. そのようなときには, 'CXSIDE' をブランクにし USGRPH が軸そのものを描かないように指定しておいて, 後でユーザーが好みの座標軸を UXPACK 等を使って描くこともできるが, それよりも USPACK の基本レベルのサブルーチンを使う方がわかりやすい.



uspk11.f:page1

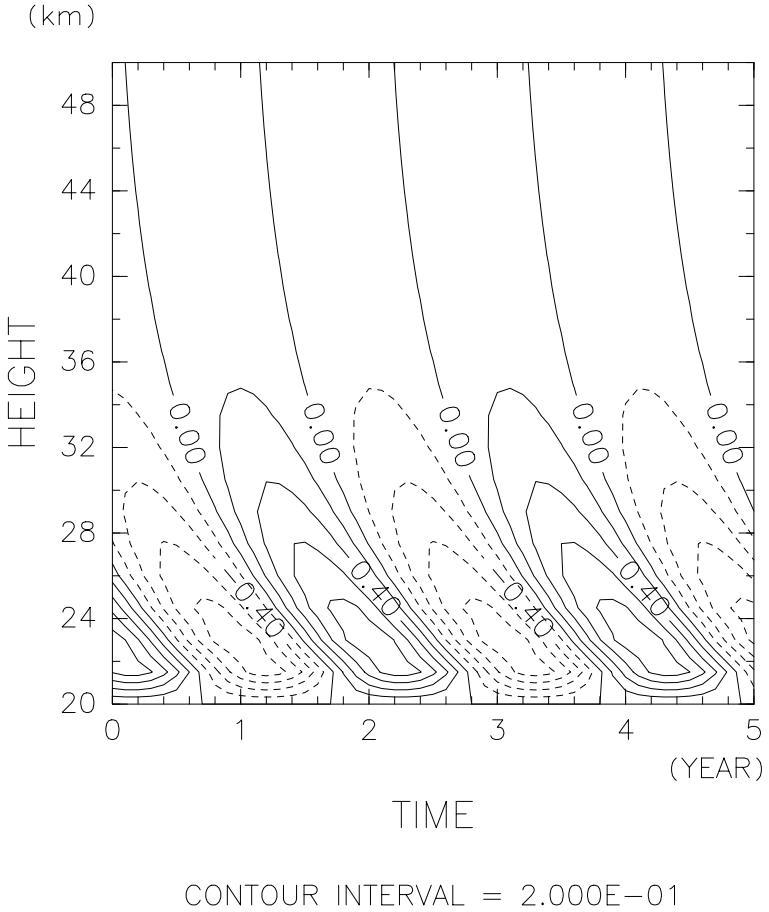
この例は Y 軸は USPACK を使い, X 軸に関しては UXPACK を使って座標軸を描いた例である.

Y 軸に関しては, どのような値になるかわからないので, GRSWND で X 軸方向のウインドウ情報のみを指定し, Y 軸方向は「不定」とする. 次に, X 座標に関する引数を「不定」とし USSPNT によってスケーリングをおこなう. そのうえで, USPFIT で変換関数に関するパラメータを設定し, GRSTRF によって変換関数を確定したのちに Y 軸を USYAXS で描いている. これは USGRPH 等, 上位のおまかせルーチンで行っている作業と同じである.

```
*---------------------------------------------------------------------
      PROGRAM USPK11
      PARAMETER(N=181, N1=26, N2=7, NC=4)
      REAL T(N), Y(N), UX1(N1), UX2(N2)
      DOUBLE PRECISION R, A
      CHARACTER*(NC) CH(N2)
      DATA CH  /'JAN', 'FEB', 'MAR', 'APL', 'JUN', 'JULY', '   '/
      DATA UX2 /   0.,   31.,   59.,   90.,  120.,   151.,  181./
      R = 0.2D0
      A = 4.D0
      Y(1) = 120.
      T(1) = 0.5
      DO 10 I=2, N
        R = A*R*(1.D0-R)
        Y(I) = Y(I-1) + (R-0.46)*2
        T(I) = I - 0.5
   10 CONTINUE
      DO 20 I=1, N1
        UX1(I) = (I-1)*7 + 1
   20 CONTINUE
*---------------------------------------------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GLRGET('RUNDEF',RUNDEF)
      CALL USLSET('LYINV',.TRUE.)
      CALL GROPN(IWS)
      CALL GRFRM
      CALL GRSWND(0., 181., RUNDEF, RUNDEF)
      CALL GRSVPT(0.2, 0.8, 0.2, 0.8)
      CALL GRSTRN(1)
      CALL USSPNT(N, RUNDEF, Y)
      CALL USPFIT
      CALL GRSTRF
*------------------------------ Y-AXIS --------------------------------
      CALL USYAXS('L')
      CALL USYAXS('R')
      CALL UYSTTL('L', 'YEN/DOLLAR', 0.)
*------------------------------ X-AXIS --------------------------------
      CALL UZLSET('LBTWN', .TRUE.)
      CALL UXAXLB('B', UX1, N1, UX2, CH, NC, N2)
      CALL UXAXLB('T', UX1, N1, UX2, CH, NC, N2)
      CALL UXSTTL('T', 'EXCHANGE RATE', 0.)
*-------------------------------- LINE --------------------------------
      CALL SGPLU(N, T, Y)
      CALL GRCLS
      END
```

5.12　座標軸ユーティリティーとして使う

USPACK の仕事のうち, 最も大切なのは目盛間隔とラベル間隔を決めることである. これまでも述べてきたように, これは USPFIT が受け持っている. 適当なスケーリングを自分でおこなうのは結構面倒なことなので, 等高線図のまわりに座標軸を描く時のように, たとえ最大値・最小値が始めからわかっているときでも, USPACK を使う価値は充分にある.



uspk12.f:page1

この例では, GRSWND などによって変換関数に関する情報を指定し, GRSTRF で変換関数を確定させた上で USDAXS によって座標軸を描いている.

タイトルに関しては, USCSET または USSTTL で指定されていれば USDAXS が描くが, UXSTTL 等を使って描いてもよい.

```
*---------------------------------------------------------------------
      PROGRAM USPK12
      PARAMETER(NT=51, NZ=21, ZMIN=20., ZMAX=50., TMAX=5.)
      PARAMETER(DZ=(ZMAX-ZMIN)/(NZ-1), DT=TMAX/(NT-1))
      REAL U(NT, NZ)
      DO 20 J=1, NZ
        Z  = DZ*(J-1)
        UZ = EXP(-0.2*Z)*(Z**0.5)
        DO 10 I=1, NT
          T  = DT*(I-1) - 2*EXP(-0.1*Z)
          U(I,J) = UZ*SIN(3.*T)
   10   CONTINUE
   20 CONTINUE
*---------------------------------------------------------------------
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
      CALL GRSWND(0., TMAX, ZMIN, ZMAX)
      CALL GRSVPT(0.2, 0.8, 0.2, 0.8)
      CALL GRSTRN(1)
      CALL GRSTRF
      CALL USSTTL('TIME', 'YEAR', 'HEIGHT', 'km')
      CALL USDAXS
      CALL UDCNTR(U, NT, NT, NZ)
      CALL GRCLS
      END
```

5.13　こんなことも

- **X 軸と Y 軸のラベル間隔をあわせる.**
  USGRPH では原則として X 軸と Y 軸を独立にスケーリングしており, しかも, 座標軸とラベルのなす角 (直角か平行か) によって, スケーリングの方法が若干異なる. そのため, 同じ様な値をもつデータを与えても, X 軸と Y 軸ではラベル間隔が異なることがある. そのような場合に, 内部変数 'LMATCH' を .TRUE. にすると, ラベルの向きに関わらず, 同じスケーリング方法を取るので, ラベル間隔が一致する確率が高くなる.

- **サブラベルとタイトルの衝突を防ぐ.**
  Y 軸の単位やファクター等を描くサブラベルが長くなると, X 軸のラベルやタイトルを描く領域に侵入するようになり, この後に UXSTTL 等により X 軸の上にタイトルを描こうとすると, X 軸のタイトルと Y 軸のサブラベルが衝突することもある. これを避ける手段が 2 つ用意されている.

  1 つは内部変数'LPRTCT' を .TRUE. にして, サブラベルが他方の軸の領域に侵入するのを防ぐ. この場合, Y 軸のサブラベルは'MXDGTSX' を無視して左に伸びていくのでワークステーションウインドウを飛び出してしまう可能性もある.

  もう 1 つは X 軸のタイトルを描く前に USXOFF を呼んでタイトルの OFFSET をずらす. こちらの方が安全かつ確実な方法である.

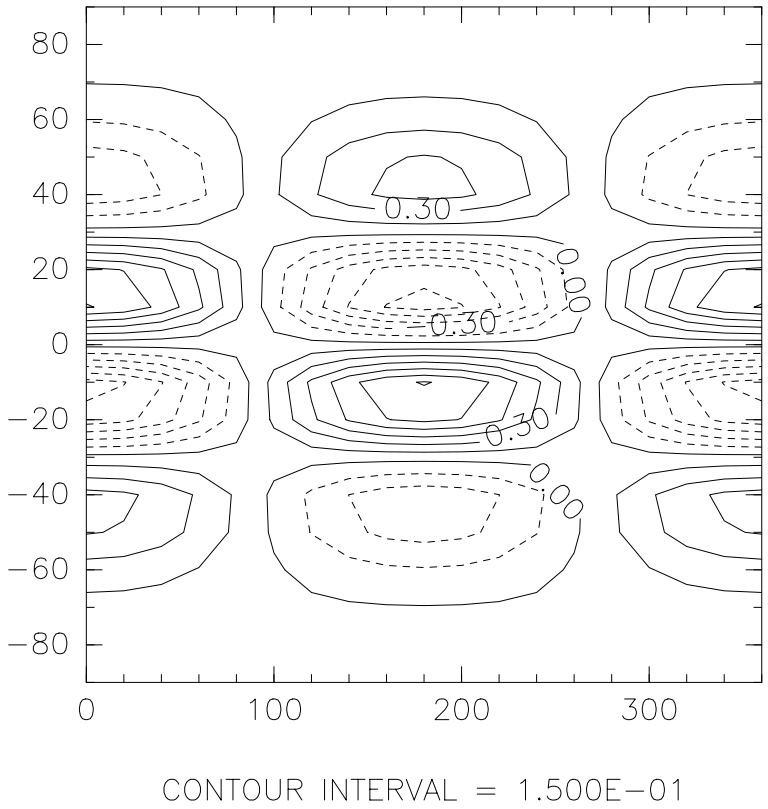
第6章　2次元量の表示

6.1　概要

ここでは UDPACK/UEPACK/UGPACK の基本的な機能を, いくつかの簡単なプログラム例で示す. これらの デモプログラムは dcl-x.x/demo/grph2/udegpk の中にあるので, 参考にしていただきたい.
なお, UDPACK, UEPACK は U 座標系で作画しているので, 後で述べる地図投影にも対応できるが, UGPACK は V 座標系で作画しているため地図投影には対応していない.

6.2　等高線図のクイックルック

2 次元データを手早くコンタリングしたいというときには, サブルーチン UDCNTR 1 つを呼ぶだけでよい.



u2df01.f:page1

UDCNTR は等高線図を描くだけで, 枠を描いたりはしてくれないから, この例ではまず, 正規化変換を設定し (GRSWND, GRSVPT, GRSTRN, GRSTRF), USPACK の中のルーチン USDAXS を使っておまかせの座標軸を描画したのちに, UDCNTR を呼んで等高線図を描いている. UDCNTR の引数は, 最初が作画しようとする場を与える 2 次元配列, 2 番目が実際に宣言した配列の第 1 次元寸法, 3, 4 番目がそれぞれ, 作画に使う配列の第 1 次元および第 2 次元寸法である. この例のように, 宣言した配列をフルに用いて作画するときは, 2 番目と 3 番目の引数は同じになる. 第 1 次元目の寸法について, なぜこのように 2 つの引数を指定する必要があるのかは u2df03.f の例で明らかになる.

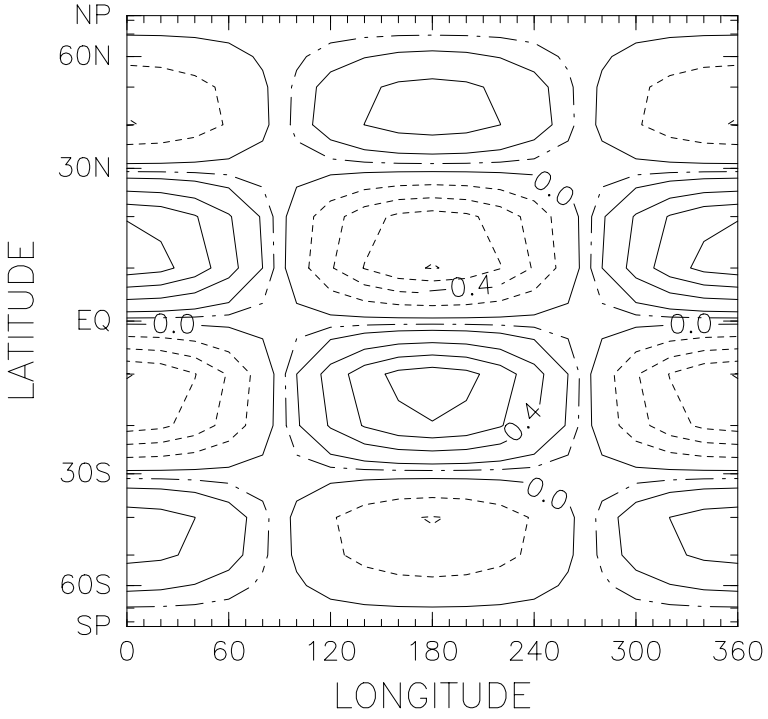
UWPACK によって, 格子点座標に関する情報を設定していないときは, 現在設定されているウインドウいっぱいに等間隔の格子点を設定してコンタリングをおこなう. つまり, 座標軸の目盛に関係なく, たとえばこの例では, P(1,1) が左下隅, P(NX,NY) が右上隅にくるような等間隔の格子点座標が設定される. ハッチングやベクトル場の表示に関する UETONE, UGVECT についても, UWPACK を用いていないときは同様にして格子点座標の設定がおこなわれる.

また, UDCNTR だけを呼ぶ, このいちばん簡単な使用例では, コンターレベルも自動的に決定される. さらに図の下には, そのコンター間隔が表示される.

```
*-----------------------------------------------------------------------
      PROGRAM U2DF01
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=0, XMAX=360, YMIN=-90, YMAX=90 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      REAL      P(NX,NY)
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL USDAXS
      CALL UDCNTR( P, NX, NX, NY )
      CALL GRCLS
      END
```

6.3　格子点が不等間隔の場合

UDCNTR を呼ぶ前に UWPACK のサブルーチン UWSGXA (X 方向)/ UWSGYA (Y 方向) を呼んで格子点の座標値をあらかじめ指定しておけば, 不等間隔な格子点座標系でのコンタリングもできる.



u2df02.f:page1

この例では, u2df01.f と同じ 2 次元配列を用い, ただし Y 方向にはサイン緯度 (–1 から +1 の値を取る) の座標系を設定してコンタリングをおこなっている. まず, UYAXLB を用いて文字列のラベルを指定しながら不等間隔の座標軸を描く. つぎに, この Y 方向については UWSGYA を用いて不等間隔の格子点を設定している. このとき, 格子点座標は U 座標系の値で指定する. X 方向については, UWSGXB を用い最大値と最小値および格子点数を指定することによって, 格子点座標に関する情報を設定している. ただし, 格子点をウインドウいっぱいに設定するようにしているだけなので, UWSGXB を呼ばなくても結果は同じである.

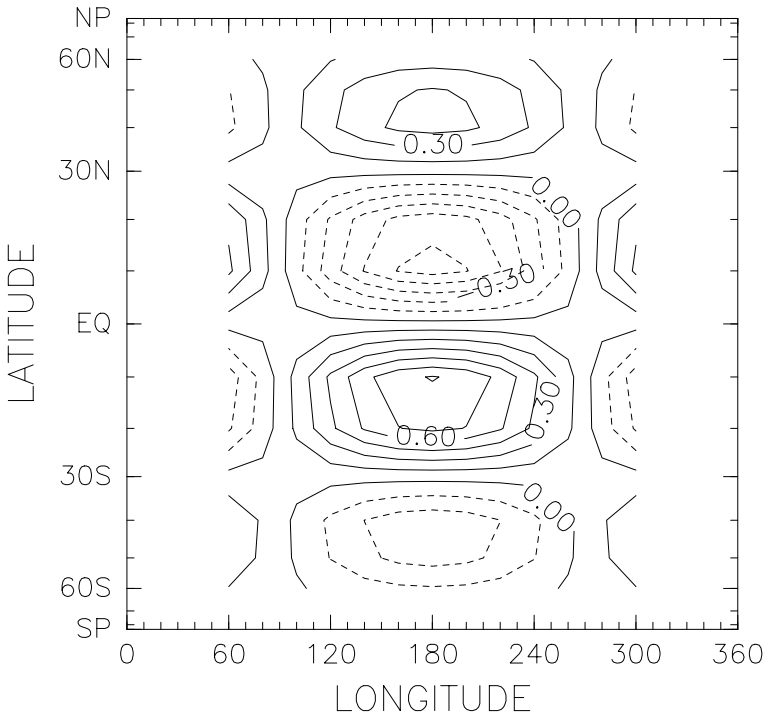
また, UDGCLB を呼ぶことによってコンター間隔を陽に指定して, コンターレベルを生成するすることができる. さらに, コンターラインにつける数値ラベルのフォーマットは UDSFMT で指定できる. コンターラインにつけるラベルは, コンターレベルを決定するルーチンの中で生成されるので, UDSFMT はコンターラインを生成するルーチン (UDGCLA や UDGCLB) の前に呼ばなければならない. またここでは, UDGCLB で等間隔のコンターレベルを生成したあとで, コンターレベルの値が 0.1 のコンターを UDSCLV によって 1 本だけ補助的に設定している. UDSCLV は新しくコンターレベルを設定するだけでなく, それまでに設定したコンターレベルについ

てその属性を変更することもできる. なお, コンターレベルは UDGCLA を用い, 最小値・最大値およびきざみ値を指定することによって生成することもできる.

```
*---------------------------------------------------------------------
      PROGRAM U2DF02
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=0, XMAX=360, DX1=10, DX2=60 )
      PARAMETER ( YMIN=-90, YMAX=90, MY=7, NC=3 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      REAL      P(NX, NY), UY1(NY), UY2(MY)
      CHARACTER CH(MY)*(NC)
      DATA      CH/ 'SP ', '60S', '30S', 'EQ ', '30N', '60N', 'NP ' /
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      DO 30 J = 1, NY
        UY1(J) = SIN( ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD )
   30 CONTINUE
      DO 40 J = 1, MY
        UY2(J) = SIN( ( YMIN + (YMAX-YMIN) * (J-1) / (MY-1) ) * DRAD )
   40 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, UY1(1), UY1(NY) )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL UXAXDV( 'B', DX1, DX2 )
      CALL UXAXDV( 'T', DX1, DX2 )
      CALL UXSTTL( 'B', 'LONGITUDE', 0.0 )
      CALL UYAXLB( 'L', UY1, NY, UY2, CH, NC, MY )
      CALL UYAXLB( 'R', UY1, NY, UY2, CH, NC, MY )
      CALL UYSTTL( 'L', 'LATITUDE', 0.0 )
      CALL UWSGXB( XMIN, XMAX, NX )
      CALL UWSGYA( UY1, NY )
      CALL UDSFMT( '(F6.1)' )
      CALL UDGCLB( P, NX, NX, NY , 0.2 )
      CALL UDSCLV( 0.1, 1, 4, ' ', 0.01 )
      CALL UDCNTR( P, NX, NX, NY )
      CALL GRCLS
      END
```

6.4　配列の一部分を描く

配列の一部分だけを作画するにはどうしたらよいのだろうか?



u2df03.f:page1

u2df02.f と同じ格子点座標で同じ格子点値を持った配列を用い, 経度 60° から 300°, 緯度–60° から +60° の部分だけを描くことを考えてみよう. u2df02.f と異なる点は, 5, 6 番目のパラメータ文で描く範囲を決めるいくつかの定数を定義していることと, UWSGXB, UWSGYB, UDCNTR の引数の指定方法が少し違うことである. (コンターレベルの設定に関する 3 つのサブルーチンは u2df02.f から除いた.)

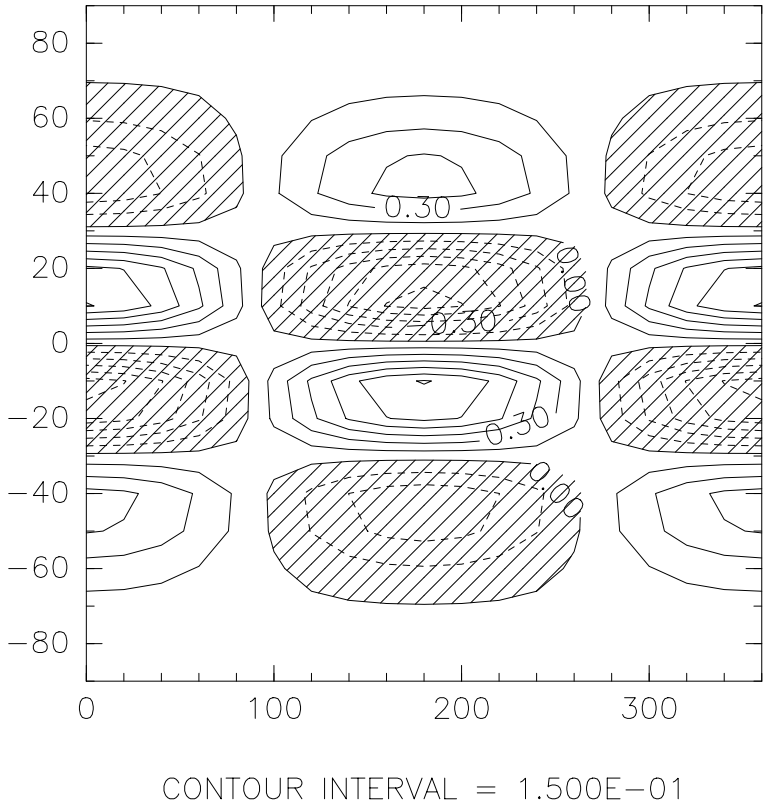
配列の一部分を作画するためにはまず, UWPACK を用い, 自分が描きたいと考えている範囲で格子点座標系を設定しなければならない. この例では, 格子点座標系を, X 方向については UWSGXB を使って経度 60° から 300° まで 20° おきの 13 点と設定し, Y 方向については UWSGYA を使って–60° から +60° に対応するサイン緯度で不等間隔に 13 点設定している. こうした上で, UDCNTR にわたす最初の引数としては, この矩形領域の左下 $(60°, \sin(-60°))$ に対応する配列要素 (ここではその添字が (IXZ1, IYZ1)) を陽に書き, 2 番目には実際に宣言した配列の第 1 次元寸法, 3, 4 番目には実際の作画に使う配列の第 1 次元および第 2 次元寸法を与えることによって配列の一部分だけを作画することができる.

この辺の事情を正しく理解するためには, FORTRAN が記憶領域上で配列要素をどのように順序づけているのかを知る必要がある. (FORTRAN の文法書などを参照のこと.)

```
*---------------------------------------------------------------------
      PROGRAM U2DF03
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=0, XMAX=360, DX1=10, DX2=60 )
      PARAMETER ( YMIN=-90, YMAX=90, MY=7, NC=3 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      PARAMETER ( XMNZ=60, XMXZ=300, NXZ=13, IXZ1=4 )
      PARAMETER ( IYZ1=4, NYZ=13 )
      REAL      P(NX, NY), UY1(NY), UY2(MY)
      CHARACTER CH(MY)*(NC)
      DATA      CH/ 'SP ', '60S', '30S', 'EQ ', '30N', '60N', 'NP ' /
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      DO 30 J = 1, NY
        UY1(J) = SIN( ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD )
   30 CONTINUE
      DO 40 J = 1, MY
        UY2(J) = SIN( ( YMIN + (YMAX-YMIN) * (J-1) / (MY-1) ) * DRAD )
   40 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, UY1(1), UY1(NY) )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL UXAXDV( 'B', DX1, DX2 )
      CALL UXAXDV( 'T', DX1, DX2 )
      CALL UXSTTL( 'B', 'LONGITUDE', 0.0 )
      CALL UYAXLB( 'L', UY1, NY, UY2, CH, NC, MY )
      CALL UYAXLB( 'R', UY1, NY, UY2, CH, NC, MY )
      CALL UYSTTL( 'L', 'LATITUDE', 0.0 )
      CALL UWSGXB( XMNZ, XMXZ, NXZ )
      CALL UWSGYA( UY1(IYZ1), NYZ )
      CALL UDCNTR( P(IXZ1,IYZ1), NX, NXZ, NYZ )
      CALL GRCLS
      END
```

6.5　とりあえず負の領域にトーンをはる

サブルーチン UETONE 1 つを呼ぶだけで, とりあえず負の領域にトーンをはることができる.



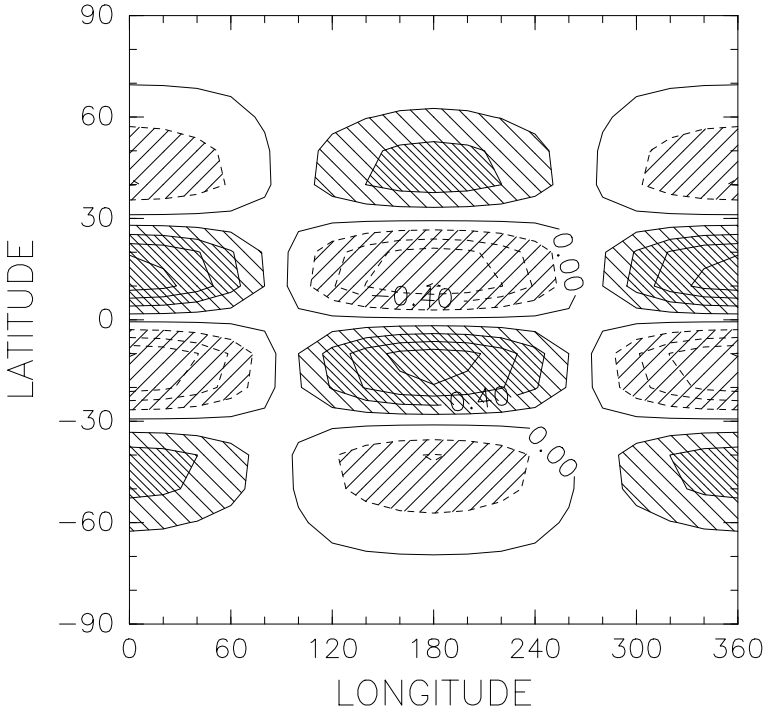
u2df04.f:page1

この例は, u2df01.f に UETONE の 1 行を付け加えただけのものであるが, これによって, 現在設定されているウインドウいっぱいに等間隔の格子点を設定し, 負の領域に斜線のハッチをつけることができる.

```
*---------------------------------------------------------------------
      PROGRAM U2DF04
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=0, XMAX=360, YMIN=-90, YMAX=90 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      REAL      P(NX,NY)
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL SGLSET( 'LSOFTF', .TRUE. )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL USDAXS
      CALL UDCNTR( P, NX, NX, NY )
      CALL UETONE( P, NX, NX, NY )
      CALL GRCLS
      END
```

6.6　トーンパターンを指定する

u2df04.f の例のように, ただ負の領域に斜線のハッチをつけるだけではあまりに芸がない. もちろん UEPACK では, トーンをつけるレベルやパターンを指定することができる.



u2df05.f:page1

この例も, u2df04.f と同じようにコンターとの重ね書きをおこなっているが, プログラムの構成上大きく異なる点がある. それは, SGLSET (「GRPH1」のマニュアル参照) で内部変数'LSOFTF' を .FALSE. として, ハードフィルを指定しているために, 座標軸の描画やコンタリングをおこなうよりも前に UETONE を呼んでいることである. UEPACK のハッチングは「GRPH1」のトーンプリミティブを用いている. トーンプリミティブは出力装置の能力に応じてハードフィルとソフトフィルとを切替えることができるが, 出力装置によってはハードフィルによる塗りわけによって先に描かれた図形が消えてしまうことがある. u2df04.f の例では, 内部変数 'LSOFTF' を .TRUE. とし, ソフトフィルを用いていたので, そのような不都合が起こらなかった. しかしこの例では, ハードフィルを指定したので, UETONE を最初に呼んでいるわけである.

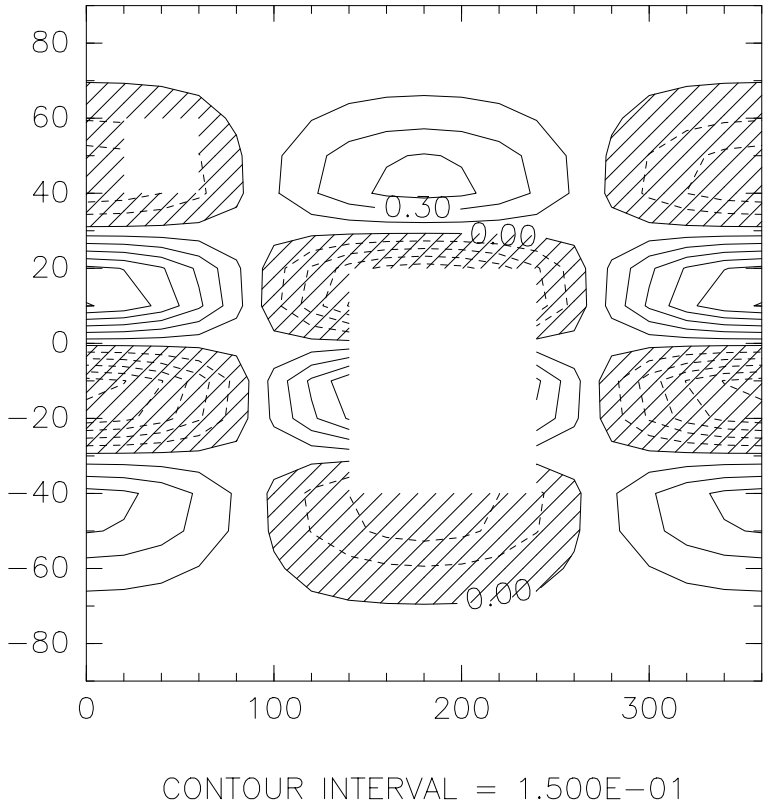
トーンの指定は, 必要なレベルの分だけ UESTLV を呼ぶことによっておこなう. (UESTLN というルーチンを用いれば, 複数のレベルを配列で指定することもできる; また UDGCLA, UDGCLB と同様のトーンレベル生成ルーチン UEGTLA, UEGTLB もある). 最初の 2 つの引数によってぬりわけるレベルの下限値と上限値を指定し, 最後の引数でトーンパターン番号を指定する. トーンパターンのリストは「GRPH1」マニュアルに付されている

ので参照されたい. この例の UESTLV による第 1 レベル目の設定のように, GL*p*GET/GL*p*SET が管理する内部変数'RMISS' を下限値として使えば 2 つ目の引数で与える値以下の領域をぬりわけることができる. また第 3 レベル目のように, 'RMISS' を上限値として使えば最初の引数以上の部分をぬりわけることができる. またもちろん, 第 2, 3 レベル目の設定のように, あるレベルの上限値と他のレベルの下限値を等しくして, 連続的なぬりわけも可能である.

```
*---------------------------------------------------------------------
      PROGRAM U2DF05
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=  0, XMAX=360, DX1=20, DX2=60 )
      PARAMETER ( YMIN=-90, YMAX=+90, DY1=10, DY2=30 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05, DP=0.2 )
      REAL      P(NX,NY)
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GLRGET( 'RMISS', RMISS )
      CALL SGLSET( 'LSOFTF', .FALSE. )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL UESTLV( RMISS,  -DP, 201 )
      CALL UESTLV(    DP, DP*2, 401 )
      CALL UESTLV( DP*2, RMISS, 402 )
      CALL UETONE( P, NX, NX, NY )
      CALL UXAXDV( 'B', DX1, DX2 )
      CALL UXAXDV( 'T', DX1, DX2 )
      CALL UXSTTL( 'B', 'LONGITUDE', 0.0 )
      CALL UYAXDV( 'L', DY1, DY2 )
      CALL UYAXDV( 'R', DY1, DY2 )
      CALL UYSTTL( 'L', 'LATITUDE', 0.0 )
      CALL UDGCLB( P, NX, NX, NY, DP )
      CALL UDCNTR( P, NX, NX, NY )
      CALL GRCLS
      END
```

6.7　欠損値格子点の取り扱い

格子点の値が欠損値のときでも, 欠損値処理の指定をすることによって, UDPACK や UEPACK は適切な作画をおこなう.



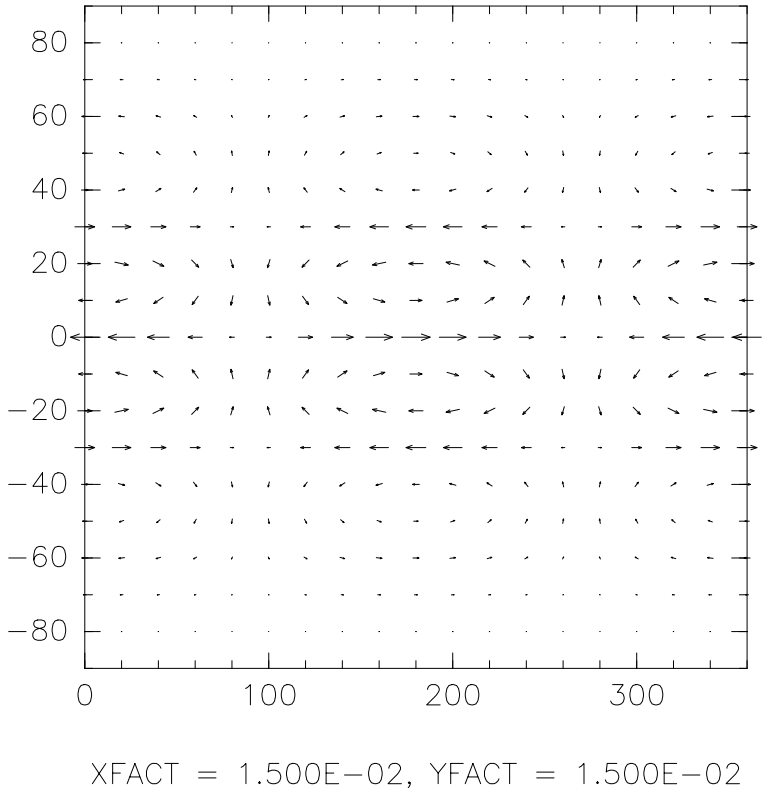
u2df06.f:page1

この例では, 基本的には u2df04.f と同じプログラム構成で, ただし格子点値の一部分を欠損値とし, 欠損値処理の指定をおこなって作画している. 欠損値処理の指定は, GLLSET によって内部変数'LMISS' を .TRUE. とすることによっておこなう. P(3,15) (経度 40°, 緯度 50° に対応) を欠損値とした例からもわかるように, 欠損値のまわりの格子点を境界として等値線やトーンが描かれない.

```
*---------------------------------------------------------------------
      PROGRAM U2DF06
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=0, XMAX=360, YMIN=-90, YMAX=90 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      REAL      P(NX,NY)
      CALL GLRGET( 'RMISS', RMISS )
      CALL GLLSET( 'LMISS', .TRUE. )
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          IF ((I.EQ.3 .AND. J.EQ.15) .OR.
     +      ((9.LE.I .AND. I.LE.12) .AND. (7.LE.J .AND. J.LE.11))) THEN
            P(I,J)=RMISS
          ELSE
            P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
          END IF
   10   CONTINUE
   20 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL SGLSET( 'LSOFTF', .TRUE.)
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL USDAXS
      CALL UDCNTR( P, NX, NX, NY )
      CALL UETONE( P, NX, NX, NY )
      CALL GRCLS
      END
```

6.8　ベクトル場の表示

2 次元格子点上で与えられたベクトルデータのクイックルックには UGVECT を用いればよい.



u2df07.f:page1

この例では, 基本的には ud2f01.f と同様なプログラム構成で (ただしベクトル場の生成をおこなった上で), UGVECT を呼んでいる. 格子点座標に関する情報を設定していないときには, これまでの UDCNTR や UETONE と同じように, 現在設定されているウインドウいっぱいに等間隔の格子点を設定する.

ベクトルの各成分は V 座標系 (ビューポートの座標系) における単位であたえる. ただしこの例のようなおまかせ描画のときは, ベクトルの長さが格子点間隔を越えないようにスケーリングファクターが決定され, それを乗じてベクトルが描かれる. これは, UG*p*GET/UG*p*SET の管理する内部変数'LNRMAL' が初期値としては .TRUE. (自動的にスケーリングファクターを決める) となっているためである. さらに, X と Y のスケーリングファクターを同じにするかどうかを決める内部変数'LEQRAT' が, 初期値として .TRUE. となっているので, X と Y のスケーリングファクターは同じになっている. 図の下部マージンには用いられたスケーリングファクターが表示される.

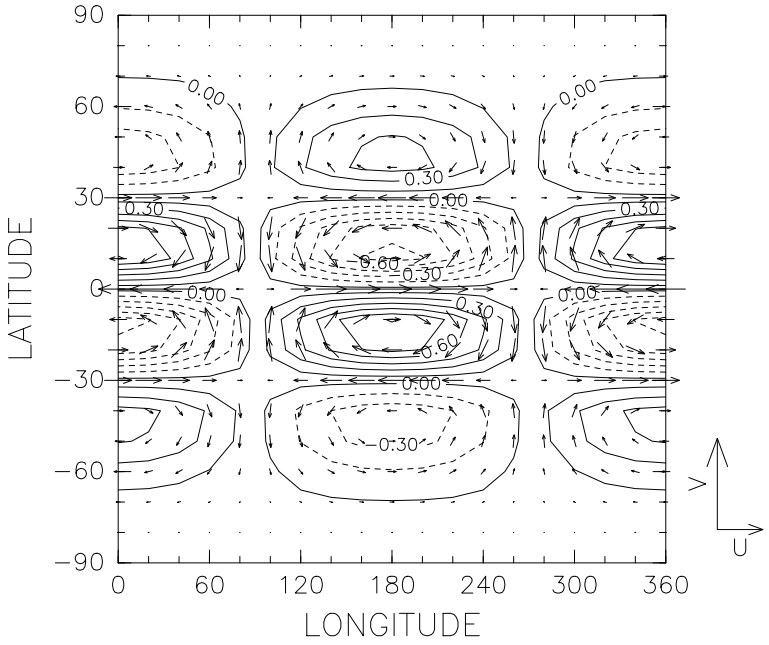
これまでに説明した UDPACK や UEPACK と同様に, UGPACK においても欠損値処理が可能である. ここでは, Y 方向の下端と上端は欠損値となっているので, GLLSET によって内部変数'LMISS' を .TRUE. とし欠損値処理をおこなうよう指示している.

(流れの場は, 単に u2df01.f で用いた場の差分から作ったものでコリオリのファクターなどは取り込まれていませんので悪しからず.)

```
*---------------------------------------------------------------------
      PROGRAM U2DF07
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=0, XMAX=360, YMIN=-90, YMAX=90 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      REAL      P(NX,NY), U(NX,NY), V(NX,NY)
      EXTERNAL  IMOD
      CALL GLRGET( 'RMISS', RMISS )
      CALL GLLSET( 'LMISS', .TRUE. )
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      DO 40 J = 1, NY
        DO 30 I = 1, NX
          IF (J.EQ.1 .OR. J.EQ.NY) THEN
            U(I,J)=RMISS
            V(I,J)=RMISS
          ELSE
            U(I,J) = P(I,J-1) - P(I,J+1)
            V(I,J) = P(IMOD(I,NX-1)+1,J) - P(IMOD(I-2,NX-1)+1,J)
          END IF
   30   CONTINUE
   40 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL USDAXS
      CALL UGVECT( U, NX, V, NX, NX, NY )
      CALL GRCLS
      END
```

6.9　等高線とベクトル場の重ね書き

最後に等高線図とベクトル場を重ね書きした例を示す.



u2df08.f:page1

u2df07.f の説明のところでも述べたが, ベクトルの各成分の単位は V 座標系 (仮想直角座標系) における単位で表現する. ただし実際には, V 座標系の単位に変換した配列を与える必要はなく, スケーリングファクターを与えてやればよい. このスケーリングファクターは, 内部変数'LNRMAL' が.TRUE.(初期値) なら内部的に決定される; .FALSE. なら内部変数'XFACT1', 'YFACT1' を参照する. したがって, この例のようにスケーリングファクターを陽に指定する場合は, 'LNRMAL' を.FALSE. として'XFACT1', 'YFACT1' を設定してやればよい. ここでは, XFACT1=0.025, YFACT1=0.05 であるから, たとえば, ベクトル (U,V)=(4,1) は, V 座標系単位で (0.1, 0.05) の大きさのベクトルとして表示される. つまり, スケーリングファクターとは, 「次元量」として与えられたベクトルの成分を V 座標系の単位で表現される矢印の成分に変換するための比例係数なのである.

またこの例では, 内部変数'LUNIT' を.TRUE. としてユニットベクトルを描いている (.FALSE. なら描かない; 初期値は.FALSE.). 表示するユニットベクトルの大きさは V 座標系の単位で内部変数'VXUNIT', 'VYUNIT' によって設定する. もしも 「次元量」で指定したいならば内部変数'UXUNIT', 'UYUNIT' を設定することでユニットベクトルを描くこともできる.

図の右下に表示されるユニットベクトルには, UGSUT でタイトルをつけることができる. なお, 図の下部に表示されるユニットベクトルの大きさ XUNIT, YUNIT は「次元量」である. ユニットベクトルの大きさを V 座標系の単位で指定した場合はそれぞれスケーリングファクターで割った量が表示される. この例では, VXUNIT=0.05, VYUNIT=0.1 としてあるから, XUNIT=2, YUNIT=2 が表示されている.

UGPACK が図の下に書くメッセージについては, ユニットベクトルを描かないときにはスケーリングファクターが表示される; ユニットベクトルを描くときにはユニットベクトルの大きさが表示される. ユニットベクトルを描くときにも, 内部変数 'LUMSG' を .FALSE. とすることによってスケーリングファクターの表示に切替えることもできる. メッセージを何も描かせたくないときには内部変数 'LMSG' を .FALSE. とすればよい.

なお, コンターにつけるラベルの文字高や, 下部マージンに書くメッセージの文字高を, UDPACK については 'RSIZEL', 'RSIZET' を設定して, また, UGPACK については'RSIZET' を設定して小さめに描画している.

```
*-----------------------------------------------------------------------
      PROGRAM U2DF08
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=  0, XMAX=360, DX1=20, DX2=60 )
      PARAMETER ( YMIN=-90, YMAX=+90, DY1=10, DY2=30 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      REAL      P(NX,NY), U(NX,NY), V(NX,NY)
      EXTERNAL  IMOD
      CALL GLRGET( 'RMISS', RMISS )
      CALL GLLSET( 'LMISS', .TRUE. )
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      DO 40 J = 1, NY
        DO 30 I = 1, NX
          IF (J.EQ.1 .OR. J.EQ.NY) THEN
            U(I,J)=RMISS
            V(I,J)=RMISS
          ELSE
            U(I,J) = P(I,J-1) - P(I,J+1)
            V(I,J) = P(IMOD(I,NX-1)+1,J) - P(IMOD(I-2,NX-1)+1,J)
          END IF
   30   CONTINUE
   40 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
      CALL GRSTRN( 1 )
      CALL GRSTRF
      CALL UXAXDV( 'B', DX1, DX2 )
      CALL UXAXDV( 'T', DX1, DX2 )
      CALL UXSTTL( 'B', 'LONGITUDE', 0.0 )
      CALL UYAXDV( 'L', DY1, DY2 )
      CALL UYAXDV( 'R', DY1, DY2 )
      CALL UYSTTL( 'L', 'LATITUDE', 0.0 )
      CALL UDRSET( 'RSIZEL', 0.015 )
      CALL UDRSET( 'RSIZET', 0.015 )
      CALL UDCNTR( P, NX, NX, NY )
      CALL UGRSET( 'RSIZET', 0.015 )
      CALL UGLSET( 'LNRMAL', .FALSE. )
      CALL UGRSET( 'XFACT1', 0.025 )
      CALL UGRSET( 'YFACT1', 0.050 )
      CALL UGLSET( 'LUNIT', .TRUE. )
      CALL UGSUT( 'X', 'U' )
      CALL UGSUT( 'Y', 'V' )
```

```
55          CALL UGRSET( 'VXUNIT', 0.05 )
56          CALL UGRSET( 'VYUNIT', 0.10 )
57          CALL UGVECT( U, NX, V, NX, NX, NY )
58          CALL GRCLS
59          END
```

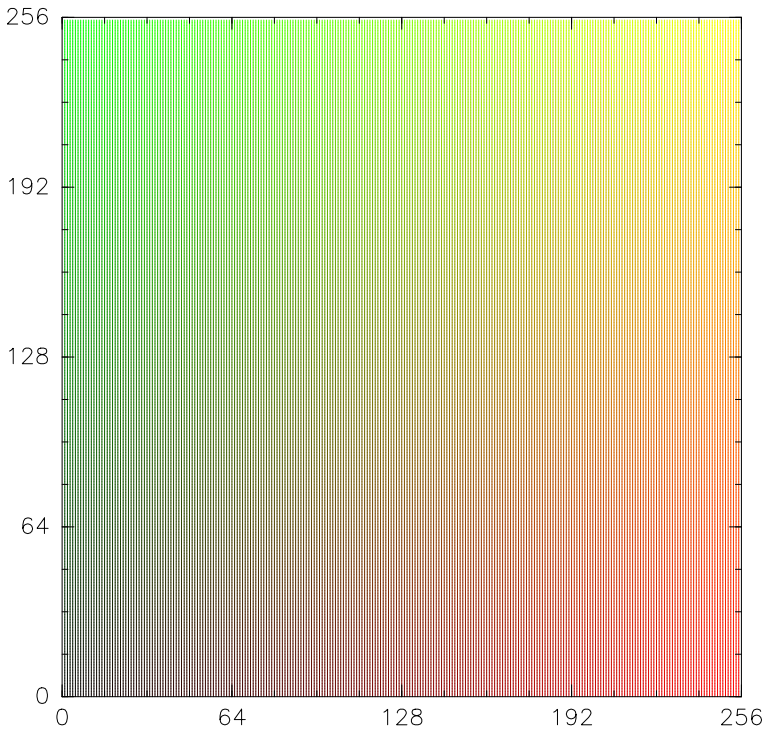
第7章　フルカラー色塗り

7.1　概要

ここでは UIPACKPACK の基本的な機能を, いくつかの簡単なプログラム例で示す. これらの デモプログラムは dcl-x.x/demo/grph2/uipack の中にあるので, 参考にしていただきたい.
なお, UIPACK は U 座標系で作画しているので, 後で述べる地図投影にも対応できる.

7.2　1 つめのサンプル

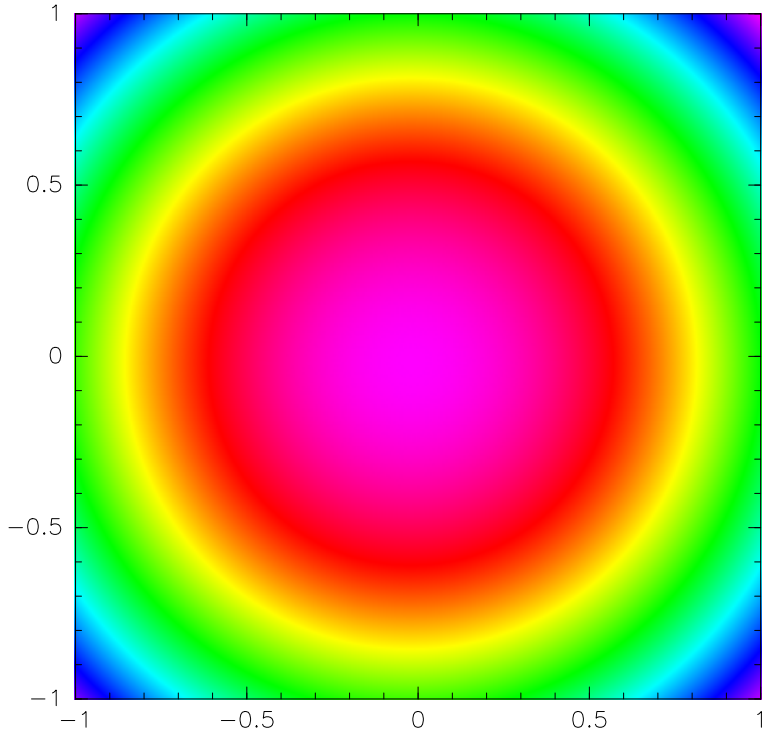
```
1         PROGRAM UIPK01
2         REAL A(2), B(2)
3         CALL SWISET('WINDOW_HEIGHT', 300)
4         CALL SWISET('WINDOW_WIDTH', 300)
5         WRITE(*,*) ' WORKSTATION ID (I) ? ;'
6         CALL SGPWSN
7         READ(*,*) IWS
8         CALL GROPN(IWS)
9         CALL GRFRM
10        CALL GRSWND(0., 256., 0., 256.)
11        CALL GRSVPT(.1, .9, .1, .9)
12        CALL GRSTRN(1)
13        CALL GRSTRF
14        DO I = 1, 255
15           A(1) = 1. * I
16           A(2) = 1. * I
17           DO J = 1, 255
18              B(1) = 1. * (J - 1)
19              B(2) = 1. * J
20              CALL SGPLXU(2, A, B, 1, 1, ISGRGB(I,J,0))
21           END DO
22        END DO
23        CALL UXAXDV('T', 16., 64.)
24        CALL UXAXDV('B', 16., 64.)
25        CALL UYAXDV('L', 16., 64.)
26        CALL UYAXDV('R', 16., 64.)
27        CALL GRCLS
28        END
```



uipk01.f:page1

7.3　2 つめのサンプル

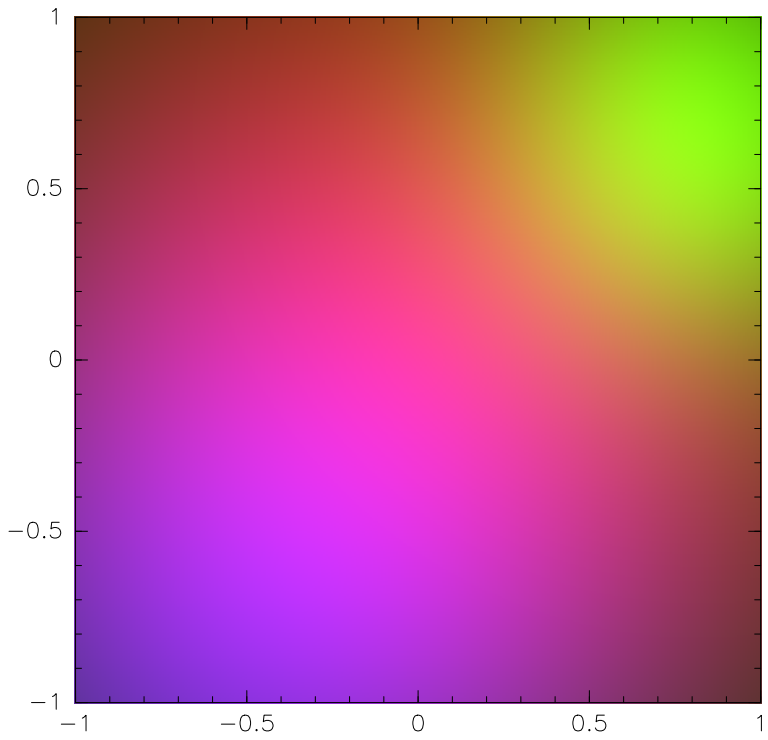
```
1         PROGRAM SAMPLE02
2         PARAMETER (NX = 50, NY = 50)
3         REAL Z(NX,NY)
4         DO I = 1, NX
5            DO J = 1, NY
6               Z(I,J) = - (I - NX / 2.)**2 - (J - NY / 2.)**2
7            END DO
8         END DO
9         CALL SWISET('WINDOW_HEIGHT', 300)
10        CALL SWISET('WINDOW_WIDTH', 300)
11        WRITE(*,*) ' WORKSTATION ID (I) ? ;'
12        CALL SGPWSN
13        READ(*,*) IWS
14        CALL GROPN(IWS)
15        CALL GRFRM
16        CALL GRSWND(-1., 1., -1., 1.)
17        CALL GRSVPT(.1, .9, .1, .9)
18        CALL GRSTRN(1)
19        CALL GRSTRF
20        CALL UIPDAT(Z, NX, NX, NY)
21        CALL UXAXDV('T', .1, .5)
22        CALL UXAXDV('B', .1, .5)
23        CALL UYAXDV('L', .1, .5)
24        CALL UYAXDV('R', .1, .5)
25        CALL GRCLS
26        END
```



uipk02.f:page1

7.4　3 つめのサンプル

```
      program UIPK03
  PARAMETER (NX=100, NY=100, LEVEL=256)
      REAL R(NX,NY),B(NX,NY),G(NX,NY)
      REAL RC(NX,NY),BC(NX,NY),GC(NX,NY)
      REAL CL(LEVEL)
      INTEGER IR(LEVEL),IG(LEVEL),IB(LEVEL)
      INTEGER I
      INTEGER IMAGE(300)
  do i=1,level
CL(i) = 1.0*(i-1)/(level-1)
call UIFPAC(level-i,0,0,ir(i))
call UIFPAC(0,level-i,0,ig(i))
call UIFPAC(0,0,level-i,ib(i))
  end do
  do i=1,nx
do j=1,ny
r(i,j) = exp(-((i-nx/2. )**2.  + (j-ny/2.)**2. ) / 5000.)
g(i,j) = exp(-((i-nx/1.1)**2.  + (j-ny/1.2)**2.) / 1000.)
b(i,j) = exp(-((i-ny/3. )**2.  + (j-ny/5.)**2. ) / 3000.)
end do
      end do
      CALL RNORML(R,RC,NX,NY,0.0,1.0)
      CALL RNORML(G,GC,NX,NY,0.2,1.0)
      CALL RNORML(B,BC,NX,NY,0.0,1.0)
      CALL SWISET('WINDOW_HEIGHT', 300)
      CALL SWISET('WINDOW_WIDTH', 300)
      WRITE(*,*) ' WORKSTATION ID (I) ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN(IWS)
      CALL GRFRM
      CALL GRSWND( -1., 1., -1., 1.)
      CALL GRSVPT(.1, .9, .1, .9)
      CALL GRSTRN(1)
      CALL GRSTRF
      call UISCMP(ig(1),ig(level),ib(1),ib(level))
      call prcopn('DclPaintGrid3')
      if(nu1 /= nv1 .or. nu2 /= nv2) then
        call msgdmp('E', 'DclPaintGrid2',
     *       'Size of arrays are not consistent.')
      end if
      call sgqvpt(vxmin, vxmax, vymin, vymax)
      call stfpr2(vxmin, vymin, rx, ry)
      call stfwtr(rx, ry, wx, wy)
      call swfint(wx, wy, ix1, iy1)
      call stfpr2(vxmax, vymax, rx, ry)
      call stfwtr(rx, ry, wx, wy)
      call swfint(wx, wy, ix2, iy2)
      call uipd3z(rc, gc, bc, nx, nx, ny, image, iwidth)
      call prccls('DclPaintGrid3')
      CALL UXAXDV('T', .1, .5)
      CALL UXAXDV('B', .1, .5)
      CALL UYAXDV('L', .1, .5)
      CALL UYAXDV('R', .1, .5)
      CALL GRCLS
      end program
```



uipk03.f:page1

第8章　地図投影

8.1　概要

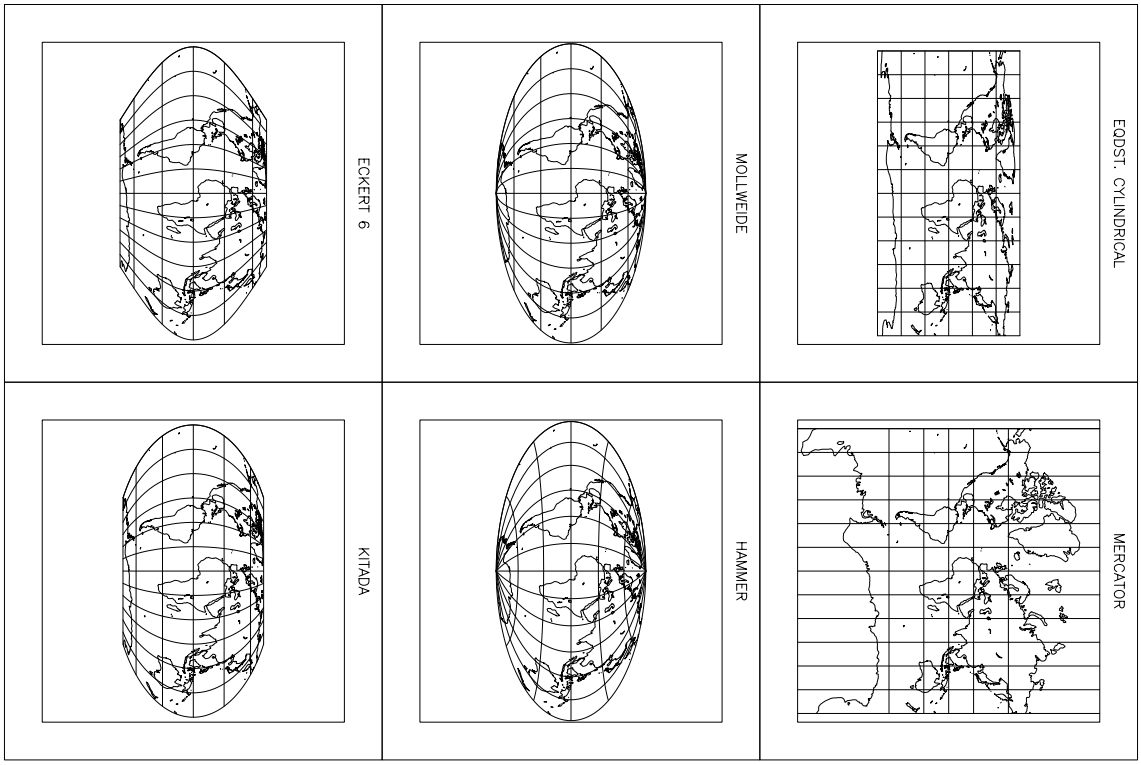
ここでは地図投影変換と関連して, UMPACK を用い海岸線や緯度･経度線を描いたり, 同時に等高線図やトーンによる塗りわけ，ベクトル場を描く例を示す．これらのデモプログラムは dcl-x.x/demo/grph2/umpack の中にあるので, 参考にしていただきたい.

地図投影変換に関する詳しい記述は,「GRPH1」マニュアルを参照されたい; また UMPACK に関しては「GRPH2」マニュアルを参照されたい.

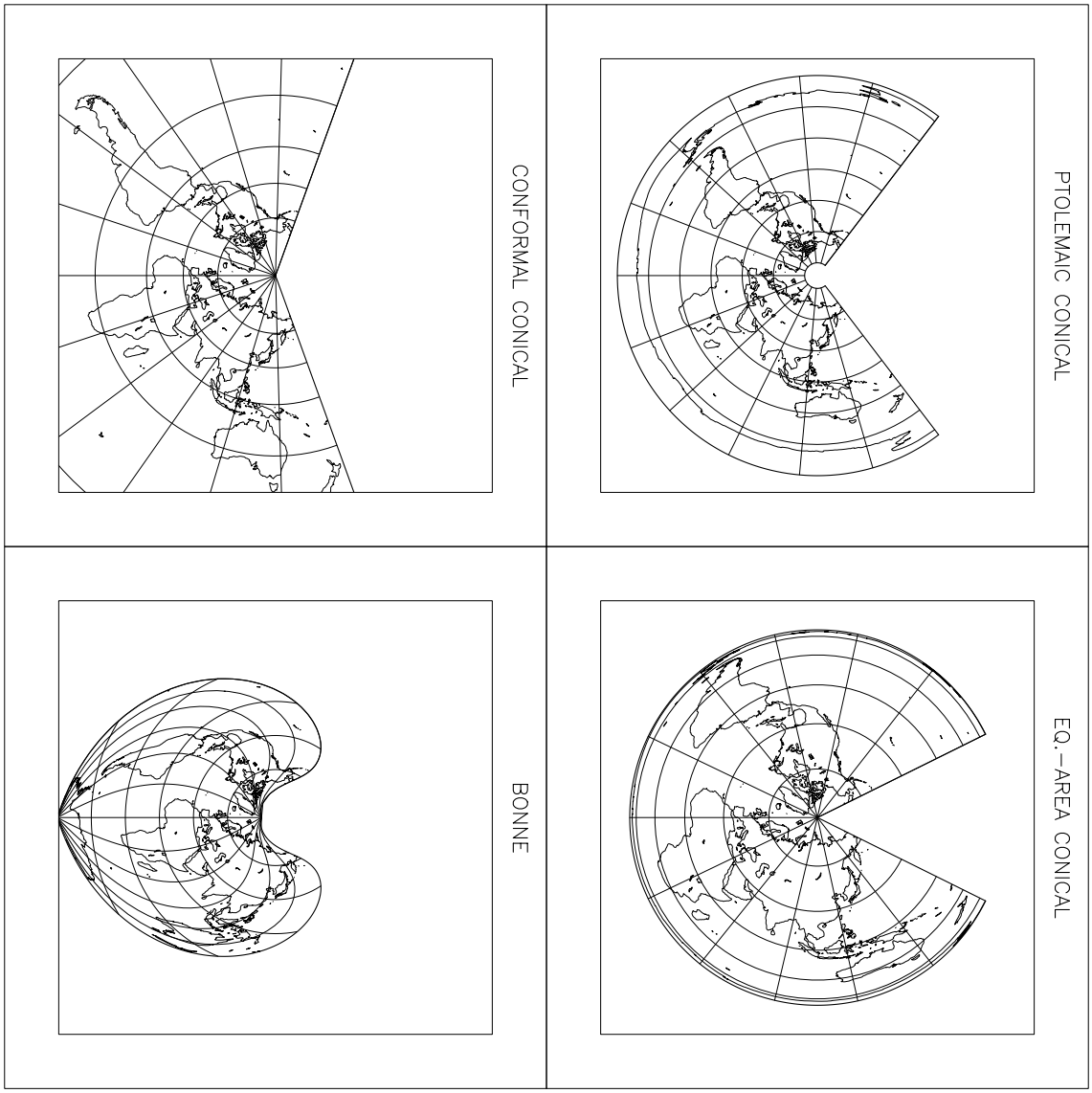
8.2　いろいろな地図投影法

GRPH1 で扱える座標系としては，直角直線座標系，曲線座標系のほかに 14 種類の地図投影座標系がある．(標準ライブラリの ENV2 が正しく移植されていれば，コマンド `dcltrf` によってサポートされている座標系の番号，略号，名称を知ることができる．) U 座標を 2 次元 V 座標に変換するこれらの正規化変換を設定することによって，地図座標系での描画が可能となる.

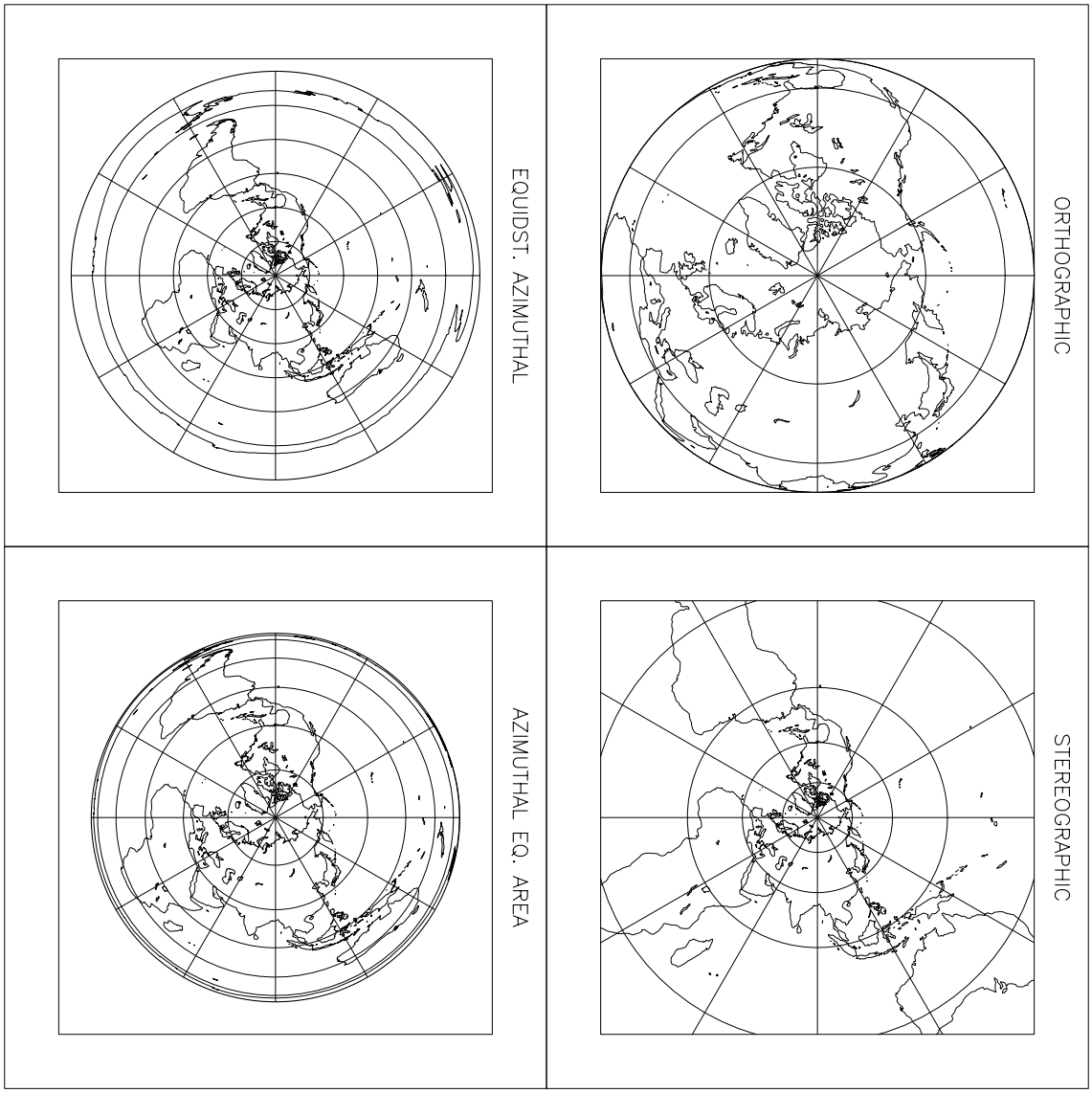
次ページ以降の 3 ページには，選択可能な地図投影変換を用いて，全球を表示した例を示す．変換関数番号 10 から 33 までの 14 種類の投影法で海岸線情報と緯度・経度線が描かれている．第 1 ページ目では円筒図法の地図投影法を, 第 2 ページ目では円錐図法の地図投影法を, 第 3 ページ目では方位図法の地図投影法を用いて作画した.



umpk01.f:page1



umpk01.f:page2



umpk01.f:page3

地図投影変換を設定するために必要なパラメータとしては，**ビューポート**, **相似変換パラメータ**，**地図座標回転パラメータ**がある．これらは SGSVPT, SGSSIM, SGSMPL によって設定する．

ビューポートは直角直線座標系の設定でも用いられていたように, V 座標系上においてユーザーが見たい矩形領域である．ビューポートは, 矩形領域の左下と右上のすみの V 座標値を指定する．この例でもおこなっているように, SG*p*GET/SG*p*SET が管理する内部変数'LCLIP' を .TRUE. としてクリッピングをするように指定すると, この枠からはみ出した部分は描画されない．

相似変換パラメータは拡大縮小と原点移動に関する指定をおこなう． 地図投影座標系の関数は MATH1/MAPLIB の関数によって定義されている． GRPH1 では, これらの関数が返す値に, SIMFAC (SGSSIM の最初の引数) を掛けて V 座標系の値に変換し, 原点をビューポートの中心から (VXOFF,VYOFF) (SGSSIM の 2, 3 番めの引数) だけ平行移動した位置に設定する．

地図座標回転パラメータは地図投影をする前におこなう緯度・経度座標の回転を指定する．(PLX,PLY) (SGSMPL の最初の 2 つの引数) は投影座標の極をおく経度・緯度を U 座標系で指定し, PLROT (SGSMPL の最後の引数) は投影座標の極の回りの回転角を指定する．くわしくは「GRPH2 マニュアル」を参照のこと．

このほかに，円錐図法による投影をおこなう場合は**標準緯度**を設定しなければならない (これは SG*p*GET/SG*p*SET の管理する内部変数'STLAT1', 'STLAT2' を設定する)．また，ウインドウに関する情報は，地図投影変換そのものでは用いられないが，GRPH2 で参照されることがある．

変換関数番号が 30 (正射図法) かどうかで条件判断している部分では，この変換関数を用いて地球を投影したとき，裏と表の両面を書かないようにするために地図投影におけるクリッピング境界を他の投影法と変えている．ふつうの投影法だと経度 [–180,+180], 緯度 [–90,+90] を地図投影におけるクリッピング境界としておけばよいが正射図法では片面だけを書かせるようにするために，緯度に関するクリッピング境界を [0.0,+90] とする必要がある．

これらのパラメータの指定は, 地図投影に関する十分な知識がないと必ずしも容易ではないかも知れない．次節では, 必要なパラメータのみ指定し, ほかは適当に決めてくれるルーチン UMPFIT を使用した例を示す．

SGTRNL は変換関数番号から変換関数名を得るルーチンである．また, UMPMAP は各種地図情報を描く; UMPGLB は緯度･経度線および地図の輪郭線を描く．なお, UM*p*GET/UM*p*SET の管理する内部変数'LGRIDMN' を .FALSE. として minor な緯度・経度線は描かないように指定した; また major な緯度・経度線のラインインデクスを 1 とした (くわしくは，「GRPH2」マニュアルの UMPACK の章を参照されたい). UMPMAP が描く地図情報としては世界の海岸線 ('coast_world') を指定している．ほかに, UMPMAP で指定できる地図情報としては, 日本の海岸線 ('coast_japan'), 世界の国境 ('border_world'), 日本の県境 ('pref_japan') などがある．

```
*---------------------------------------------------------------------
      PROGRAM UMPK01
      PARAMETER (NP=14)
      INTEGER   NTR(NP)
      REAL      FCT(NP)
      CHARACTER CTTL*32
      DATA NTR /   10,   11,   12,   13,   14,   15,
     +             20,   21,   22,   23,   30,   31,   32,   33/
      DATA FCT / 0.12, 0.12, 0.14, 0.14, 0.14, 0.14,
     +           0.11, 0.16, 0.12, 0.12, 0.40, 0.12, 0.12, 0.17/
      CALL SWCSTX('FNAME','UMPK01')
      CALL SWLSTX('LSEP',.TRUE.)
      WRITE(*,*) ' WORKSTATION IS (I) ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL SGOPN( -ABS(IWS) )
      CALL SLRAT( 2.0, 3.0 )
      CALL SLDIV( 'Y', 2, 3 )
      CALL SGRSET( 'STLAT1', 45.0 )
      CALL SGRSET( 'STLAT2', 30.0 )
      CALL UMLSET( 'LGRIDMN', .FALSE. )
      CALL UMISET( 'INDEXMJ', 1 )
      DO 10 I=1,NP
        CALL SGFRM
        CALL SGSSIM( FCT(I), 0.0, 0.0 )
        CALL SGSMPL( 0.0, 90.0, 0.0 )
        CALL SGSVPT( 0.1, 0.9, 0.1, 0.9 )
        IF ( NTR(I).EQ.30 ) THEN
          CALL SGSTXY( -180.0, 180.0,   0.0, 90.0 )
        ELSE
          CALL SGSTXY( -180.0, 180.0, -90.0, 90.0 )
        END IF
        CALL SGSTRN( NTR(I) )
        CALL SGSTRF
        CALL SGLSET( 'LCLIP', .TRUE. )
        CALL SLPWWR( 1 )
        CALL SLPVPR( 1 )
        CALL SGTRNL( NTR(I), CTTL )
        CALL SGTXZR( 0.5, 0.95, CTTL, 0.03, 0, 0, 3 )
        CALL UMPMAP( 'coast_world' )
        CALL UMPGLB
        IF ( NTR(I).EQ.23 ) THEN
          CALL SGFRM
          CALL SGFRM
        END IF
   10 CONTINUE
      CALL SGCLS
      END
```

8.3 変換パラメータの自動決定

umpk01.f では相似変換パラメータや地図座標回転パラメータを陽に指定したが, これらのパラメータを適切に決めるためには地図投影に関する知識が必要である.

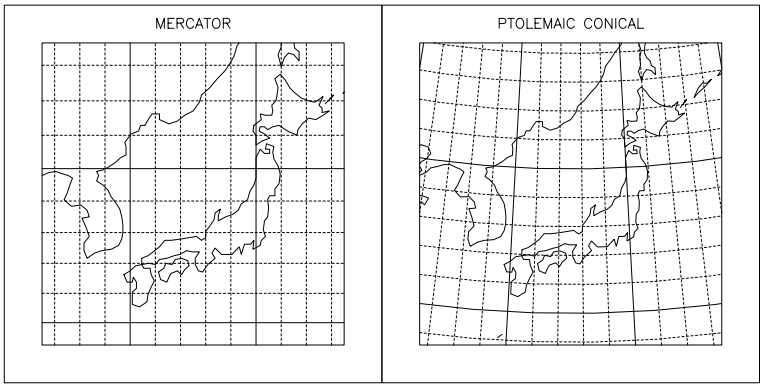
たいがいの場合は, ビューポートいっぱいに全球を描くか, ある地域を描くかのどちらかであろう. これら 2 つの用途を念頭において, UMPFIT は地図投影に必要なパラメータを自動的に決定してくれる. UMPFIT は GRPACK と組み合わせて用いる. つまり, GRPACK によって必要なパラメータを指定し, 変換関数確定の直前に UMPFIT を呼ぶことによって, ユーザーが指定しなかったパラメータを適当に補った上で, GRSTRF を呼んで変換関数を確定する.

umpk02.f は基本的には umpk01.f と同じような描画をおこなうプログラムである. ただしここでは, ビューポートと変換関数番号を指定しただけで, 相似変換パラメータと地図座標回転パラメータは指定していない. ウインドウ情報が指定されていないと, UMPFIT は全球を描くべく相似変換パラメータなどを決定して変換関数を確定する. (なお, 次の章ではウインドウを指定して, 一部地域を描く例を示す.)

```
1  *---------------------------------------------------------------------
2        PROGRAM UMPK02
3        PARAMETER (NP=14)
4        INTEGER   NTR(NP)
5        CHARACTER CTTL*32
6        DATA NTR /   10,   11,   12,   13,   14,   15,
7       +             20,   21,   22,   23,   30,   31,   32,   33/
8        WRITE(*,*) ' WORKSTATION IS (I) ? ;'
9        CALL SGPWSN
10       READ(*,*) IWS
11       CALL GROPN( -ABS(IWS) )
12       CALL SLRAT( 2.0, 3.0 )
13       CALL SLDIV( 'Y', 2, 3 )
14       CALL UMISET( 'INDEXMJ', 1 )
15       CALL UMISET( 'ITYPEMN', 1 )
16       DO 10 I=1,NP
17         CALL GRFRM
18 *       CALL GRSMPL( 0.0, 90.0, 0.0 )
19         CALL GRSVPT( 0.1, 0.9, 0.1, 0.9 )
20         CALL GRSTRN( NTR(I) )
21         CALL UMPFIT
22         CALL GRSTRF
23         CALL SGLSET( 'LCLIP', .TRUE. )
24         CALL SLPWWR( 1 )
25         CALL SLPVPR( 1 )
26         CALL SGTRNL( NTR(I), CTTL )
27         CALL SGTXZR( 0.5, 0.95, CTTL, 0.03, 0, 0, 3 )
28         CALL UMPMAP( 'coast_world' )
29         CALL UMPGLB
30         IF ( NTR(I).EQ.23 ) THEN
31           CALL GRFRM
32           CALL GRFRM
33         END IF
34     10 CONTINUE
35       CALL GRCLS
36       END
```

8.4 地域モード

UMPFIT と GRPACK を組み合わせて用いたとき, 緯度・経度に関するウインドウ情報が指定されていると, その領域をほぼ含むような一部地域を描画するような地図投影変換が設定される.



umpk03.f:page1

umpk03.f ではウインドウ情報を指定し, UMPFIT を呼んでいる. (この例ではメルカトールとコニカル 2 つの地図変換についてのみ示した.) こうすると, ウインドウとして指定した緯度・経度の範囲がビューポート内にほぼ収まるように変換関数が設定される.

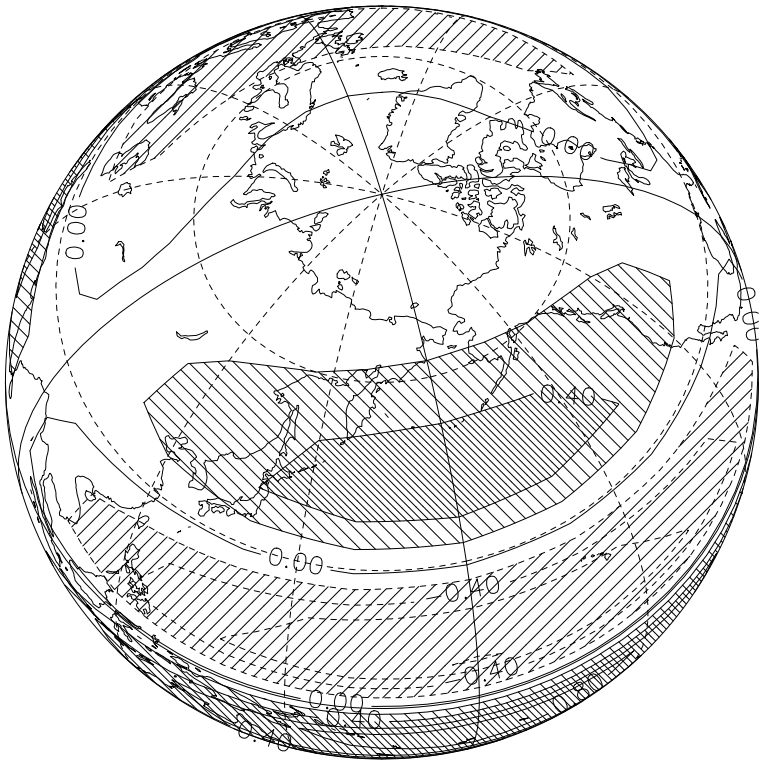
この例では, ISGTRC によって変換関数の省略名から変換関数番号を得ている. また major な緯度・経度線と minor な緯度・経度線の間隔をそれぞれ 10 度, 2 度としている.

またここでは, UMPMAP が描く地図情報としては coast_japan が利用可能であるが, 作画量が多いので coast_world を用いた.

```
*-----------------------------------------------------------------------
      PROGRAM UMPK03
      PARAMETER (NP=2)
      CHARACTER CTTL*32, CTR(NP)*3
      EXTERNAL  ISGTRC
      DATA CTR /'MER','CON'/
      WRITE(*,*) ' WORKSTATION IS (I) ? ;'
      CALL SGPWSN
      READ(*,*) IWS
      CALL GROPN( ABS(IWS) )
      CALL SLRAT( 2.0, 1.0 )
      CALL SLDIV( 'Y', 2, 1 )
      DO 10 I=1,NP
        CALL GRFRM
        CALL GRSWND( 123.0, 147.0, 30.0, 46.0)
        CALL GRSVPT( 0.1, 0.9, 0.1, 0.9 )
        CALL GRSTRN( ISGTRC(CTR(I)) )
        CALL UMPFIT
        CALL GRSTRF
        CALL SGLSET( 'LCLIP', .TRUE. )
        CALL SLPWWR( 1 )
        CALL SLPVPR( 1 )
        CALL SGTRNL( ISGTRC(CTR(I)), CTTL )
        CALL SGTXZR( 0.5, 0.95, CTTL, 0.03, 0, 0, 3 )
*       CALL UMPMAP( 'coast_japan' )
        CALL UMPMAP( 'coast_world' )
        CALL UMPGLB
   10 CONTINUE
      CALL GRCLS
      END
```

8.5 等高線図

等高線図作画パッケージ UDPACK とトーンによる塗りわけパッケージ UEPACK は U 座標系で作画するので, 地図投影変換を指定していても利用可能である.



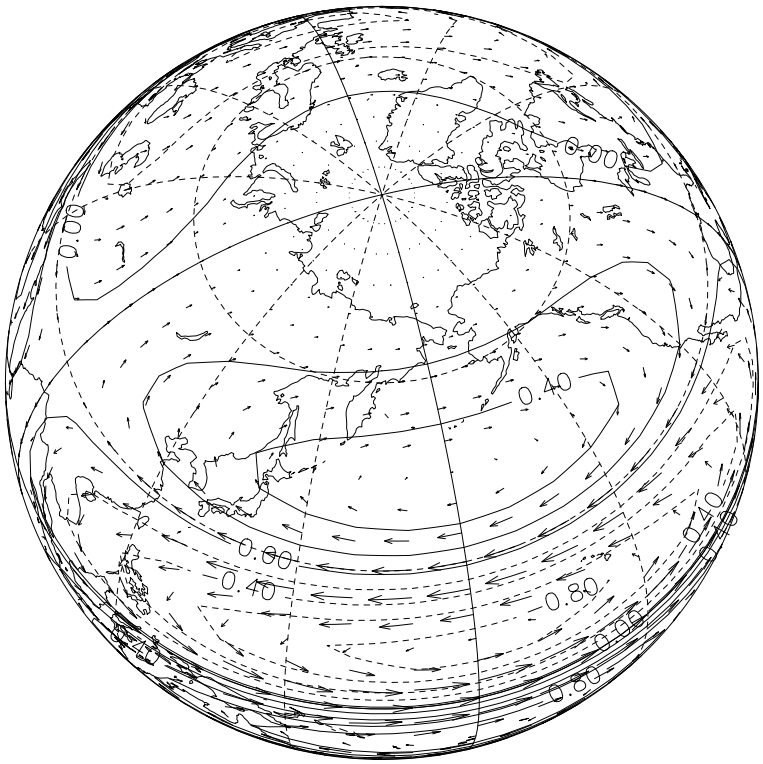
umpk04.f:page1

umpk04.f の基本構造は, 第 6 章のデモプログラム u2df05.f と同じである. ただここでは, 変換関数として地図投影変換 (変換関数番号 30:正射図法) を選んでいる. u2df05.f と同じ等高線とハッチが地図上で作画されているのがわかる.

```
*---------------------------------------------------------------------
      PROGRAM UMPK04
      PARAMETER ( NX=19, NY=19 )
      PARAMETER ( XMIN=  0, XMAX=360, YMIN=-90, YMAX=+90 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05, DP=0.2 )
      REAL      P(NX,NY)
      DO 20 J = 1, NY
        DO 10 I = 1, NX
          ALON = ( XMIN + (XMAX-XMIN) * (I-1) / (NX-1) ) * DRAD
          ALAT = ( YMIN + (YMAX-YMIN) * (J-1) / (NY-1) ) * DRAD
          SLAT = SIN(ALAT)
          P(I,J) = COS(ALON) * (1-SLAT**2) * SIN(2*PI*SLAT) + DZ
   10   CONTINUE
   20 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GLRGET( 'RMISS', RMISS )
      CALL SGLSET( 'LSOFTF', .FALSE. )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.1, 0.9, 0.1, 0.9 )
      CALL GRSSIM( 0.4, 0.0, 0.0 )
      CALL GRSMPL( 165.0, 60.0, 0.0 )
      CALL GRSTXY( -180.0, 180.0, 0.0, 90.0 )
      CALL GRSTRN( 30 )
      CALL GRSTRF
      CALL SGLSET( 'LCLIP', .TRUE. )
      CALL UESTLV( RMISS,  -DP, 201 )
      CALL UESTLV(    DP, DP*2, 401 )
      CALL UESTLV( DP*2, RMISS, 402 )
      CALL UETONE( P, NX, NX, NY )
      CALL UDGCLB( P, NX, NX, NY, DP )
      CALL UDCNTR( P, NX, NX, NY )
      CALL UMPMAP( 'coast_world' )
      CALL UMPGLB
      CALL GRCLS
      END
```

8.6　ベクトル場

ベクトル場作画パッケージ UGPACK は V 座標系上で作画するので, 残念ながら, 現在のところ地図投影には対応していない. しかしながら, アローサブプリミティブを使って, 緯度・経度の単位を持った矢印を地図上に描くことは簡単にできる. したがって, DO ループをもちい, 次元を持ったベクトルを連続的に描けば, 地図上にベクトル場を描くことも容易に実現できる.



umpk05.f:page1

umpk05.f のプログラムの骨組みは, 基本的に第 6 章のデモプログラム u2df08.f と同じである. ベクトルの各成分を緯度・経度の単位でスケーリングしておけば, U 座標系での矢印作画ルーチン `SGLAZU` を用いて, 地図上でのベクトル場の表現が可能になる.

```
*---------------------------------------------------------------------
      PROGRAM UMPK05
      PARAMETER ( NX=37, NY=37 )
      PARAMETER ( XMIN=  0, XMAX=360, YMIN=-90, YMAX=+90 )
      PARAMETER ( PI=3.141592, DRAD=PI/180, DZ=0.05 )
      PARAMETER ( FACT=10 )
      REAL      P(NX,NY), U(NX,NY), V(NX,NY), ALON(NX), ALAT(NY)
      EXTERNAL  IMOD
      CALL GLRGET( 'RMISS', RMISS )
      CALL GLLSET( 'LMISS', .TRUE. )
      DO 10 I = 1, NX
        ALON(I) = XMIN + (XMAX-XMIN) * (I-1) / (NX-1)
   10 CONTINUE
      DO 20 J = 1, NY
        ALAT(J) = YMIN + (YMAX-YMIN) * (J-1) / (NY-1)
   20 CONTINUE
      DO 40 J = 1, NY
        DO 30 I = 1, NX
          SLAT = SIN(ALAT(J)*DRAD)
          P(I,J) = COS(ALON(I)*DRAD) * (1-SLAT**2) * SIN(2*PI*SLAT)
     +             + DZ
   30   CONTINUE
   40 CONTINUE
      DO 60 J = 1, NY
        DO 50 I = 1, NX
          IF (J.EQ.1 .OR. J.EQ.NY) THEN
            U(I,J) = RMISS
            V(I,J) = RMISS
          ELSE
            U(I,J) = ( P(I,J-1) - P(I,J+1) ) * FACT
            V(I,J) = ( P(IMOD(I,NX-1)+1,J) - P(IMOD(I-2,NX-1)+1,J) )
     +             * FACT
          END IF
   50   CONTINUE
   60 CONTINUE
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS
      CALL GROPN( IWS )
      CALL GRFRM
      CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
      CALL GRSVPT( 0.1, 0.9, 0.1, 0.9 )
      CALL GRSSIM( 0.4, 0.0, 0.0 )
      CALL GRSMPL( 165.0, 60.0, 0.0 )
      CALL GRSTXY( -180.0, 180.0, 0.0, 90.0 )
      CALL GRSTRN( 30 )
      CALL GRSTRF
      CALL SGLSET( 'LCLIP', .TRUE. )
      CALL UMPMAP( 'coast_world' )
      CALL UMPGLB
      CALL UDCNTR( P, NX, NX, NY )
      DO 80 J=1,NY
        DO 70 I=1,NX
          IF (.NOT.(U(I,J).EQ.RMISS .OR. V(I,J).EQ.RMISS)) THEN
            CALL SGLAZU(ALON(I),ALAT(J),ALON(I)+U(I,J),ALAT(J)+V(I,J),
     +           1, 3 )
          END IF
   70   CONTINUE
   80 CONTINUE
      CALL GRCLS
      END
```

8.7 湖や地上の塗りつぶし (後で修正が必要)

使い方は簡単で, これまで

call umpmap('coast_world')

としていた個所において

call umfmap('coast_world') call umpmap('coast_world')

とするのみである. 塗り潰しのパターンを変更したい場合には

call umiset('IPATLAND', 2999) ! 赤のべた塗り
とすればよい.
デモプログラム (カラーマップの取り換えも同時に実行しているもの) が
demo/grph2/umpack/test09.f
にあります.