地球流体電脳ライブラリ

ごくらく DCL

地球流体電脳倶楽部

1995 年 6 月 9 日 (DCL-5.0)

目 次

はじめに 1

1 まずはホップ 2
1.1 基本操作 . . . 2
1.2 基本概念 (1): 出力装置のオープンとクローズ . . . 4
1.3 USGRPH: 自動スケーリング折れ線図 . . . 4

2 つぎのステップ 5
2.1 USGRPH の分解 . . . 5
2.2 基本概念 (2): 折れ線 (軌跡図) . . . 6
2.3 基本概念 (3): マーカー列 (分布図) . . . 8

3 そしてジャンプ 10
3.1 基本概念 (4): 正規化変換 . . . 10
3.2 異なる大きさの図形 . . . 11
3.3 対数座標 . . . 13

4 おまかせ一次元図 15
4.1 基本概念 (5): 内部変数 「未定義値」 . . . 15
4.2 f(iΔx) . . . 16
4.3 g(jΔy) . . . 17

5 おまかせ二次元図 18
5.1 等高線図 . . . 18
5.2 トーン付き等高線図 . . . 19
5.3 ベクトル場 . . . 21
5.4 トーン付き等高線とベクトル場の重ね書き . . . 22

6 レイアウトしよう 24
6.1 フレームの分割 . . . 24
6.2 描画領域の変更 . . . 26

付録 27

謝辞 30

はじめに

地球流体電脳ライブラリ (DCL) は, 地球流体電脳倶楽部が長年にわたって開発・整備してきたものです. 我々は, このライブラリが多くの人に使われ, 地球流体力学の発展に寄与することを願っています.

DCL は機能によっていくつかの部分に分かれていますが, 「ごくらく DCL」ではグラフィクス部分だけを解説しています. DCL グラフィクスではサブルーチンを数行呼ぶだけで, 折れ線図や等高線図などさまざまな作図が簡単にできます. ここでは特に初心者を念頭に置いて, サンプルプログラム (FORTRAN) と実行結果のグラフを中心に, とりあえずどうすればイメージした図が出力できるかをわかりやすく解説しましょう. まず, 第 1 章で基本操作を解説し, つぎの第 2・3 章で, 描画の基本概念を説明していきます. さらに, 第 4~6 章では, 「おまかせ」で一次元データや二次元データを描いたり, 図の割りつけをするパッケージを紹介します. 描きたいグラフを思い浮かべて, 該当するところの解説を読むだけで, すぐにお望みのグラフが得られることでしょう. サブルーチンの一覧表を付録につけます. ライブラリの全体像をつかんだり, 個々の説明ページを見つけるのに便利かと思います.

「ごくらく DCL」の内容をマスターした頃には, きっといろいろな不満が湧いていることでしょう. 文字や記号を書き込みたい, 折れ線のパターンが気に入らない, 座標軸のスタイルを変えたい, 欠損値処理をしたい, 色とりどりの図にしたい, ... DCL グラフィクスはこのような要望にも応えられるパッケージの数々を用意しています. 「ごくらく DCL」は卒業して DCL グラフィクスを基本からじっくりと学びたい方は, 姉妹編「らくらく DCL」を御覧下さい。また、それぞれのパッケージのマニュアルを通読してみることもたいへん有意義なことでしょう.

第 1 章

まずはホップ

DCL グラフィクスの基本的な操作と概念を説明し, 具体的にどうすれば図形出力が得られるかを概観します. ここでは, UNIX システムで DCL が標準的にインストールされていることを前提としています. そうでない環境で DCL を学び始めようという方は, まず, 基本操作のどこが違うか身近な先達に直接尋ねてみて下さい. ここで紹介する FORTRAN プログラム自体は, どんな環境でも同じように動くようになっています.

1.1　基本操作

データ解析でも数値計算でも一刻も早く計算結果が見たいものですが, そんな時, DCL を用いるとわずか数行でデータをグラフ化できます. 最初の例題として, リサジューの図形を描いてみましょう. FORTRAN プログラムは, 次の `HOP` です.

```
1         PROGRAM HOP
2
3         PARAMETER( NMAX=400 )
4         REAL X(NMAX), Y(NMAX)
5
6   *-- リサジューの図 ----
7         DT = 2.*3.14159 / (NMAX-1)
8
9         DO 10 N=1,NMAX
10          T = DT*(N-1)
11          X(N) = 1.E 2*SIN(4.*T)
12          Y(N) = 1.E-3*COS(5.*T) + 6.
13      10 CONTINUE
14
15  *-- グラフ ----
16        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
17        CALL SGPWSN
18        READ (*,*) IWS
19
20        CALL GROPN( IWS )
21        CALL GRFRM
22
23        CALL USSTTL( 'X-TITLE', 'x-unit', 'Y-TITLE', 'y-unit' )
24        CALL USGRPH( NMAX, X, Y )
25
26        CALL GRCLS
27
28        END
```

UNIX システムで DCL が標準的にインストールされている場合には,

```
% dclfrt -o hop hop.f
```

によって `hop` という実行ファイルが作られます. そこで,

```
% hop
```

といれると,

```
WORKSTATION ID (I) ? ;
1:X, 2:PS, 3:Tek ;
```

ときいてきます. プログラムの 17 行めでサブルーチン SGPWSN を呼んだので, このように今の環境で利用可能な図形出力装置のリストが書き出されます.

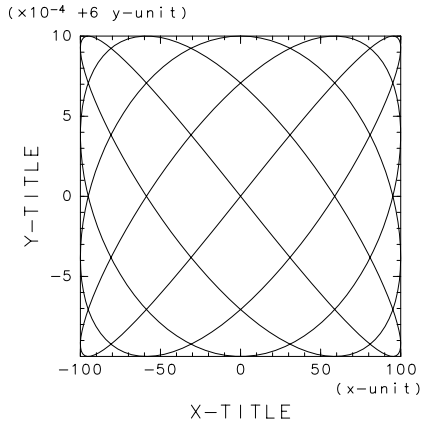
この場合, 3 つの出力先が可能です. X ウインドウシステムが起動されている状態で 1 を入力すると, ウインドウがひとつ現れます. マウスクリックでウインドウの位置を確定すると, 描画がはじまり下のようなグラフが得られます. このとき, 次の警告メッセージが出ると思いますが, 特に気にする必要はありません. 図形表示の終了はまたマウスクリックで行ないます.

```
*** WARNING (STSWTR) *** WORKSTATION VIEWPORT WAS MODIFIED.
```

一方, 2 (PS) を指定すると, カレントディレクトリに dcl.ps というポストスクリプトファイルができます. そこで,

```
% lpr dcl.ps
```

と入力すれば, ポストスクリプトプリンタに結果が出力されます. また, 3 (Tek) を指定するとテクトロ端末で描画ができます.



hop.f: frame1

1.2　基本概念 (1): 出力装置のオープンとクローズ

プログラム HOP で, DCL グラフィクスの基本的な構成を説明しておきましょう. まず, 7 行めから 13 行めまででデータを作り, 実質的には 20 行めから 26 行めまでで図形を描いています.

まず, 出力装置のオープンとクローズに関連する 3 つのサブルーチンについて説明しましょう. 例えば 1 冊のノートにグラフを描くことを思い浮かべると, 次のような手順に相当します. ノートを出してきて机の上に置き (GROPN), 新しいページを開いて (GRFRM), そして, 何かグラフを描きます. 最後は, 後片付け (GRCLS) をして出来上がりです. この GRFRM と GRCLS のあいだでさまざまな作画ルーチンを呼ぶことにより, 多種多様な図形が描けるようになります.

- GROPN(IWS) [出力装置のオープン]

  IWS　(整数型)　各装置の番号. IWS>0: 横長画面, IWS<0: 縦長画面 (90 度回転).
  図形を描くために, ディスプレイやプリンタなど図形出力装置の準備をする.

- GRFRM [フレームの設定]

  新しい作画領域を設定する.
  もう一度呼ぶと, いわゆる「改ページ」操作になる.

- GRCLS [出力装置のクローズ]

  描画を終了する時の操作.
  図形処理の最後に呼ぶ.

1.3　USGRPH: 自動スケーリング折れ線図

プログラム例 HOP で具体的にグラフを描いているのは, GRFRM に続く 2 つのサブルーチンです. USSTTL ルーチンで座標軸のタイトルとなる文字列を指定し, USGRPH で折れ線グラフを描いています. データがはみ出さないように x 座標と y 座標の範囲を定め, 各座標軸を描き, 座標軸のラベル (数字) とタイトル・単位をつけます. ここでラベルが長くならないように, ファクター (y 軸の括弧内の $\times 10^{-4}$) やオフセット (同じく $+6$) が自動的に用いられます. そして最後に, 与えたデータ列を実線で結びます.

- USSTTL(CXTTL,CXUNIT,CYTTL,CYUNIT) [座標軸のタイトル・単位の指定]

  CXTTL　(文字型)　x 座標軸のタイトル.
  CXUNIT　(文字型)　x 座標軸の単位.
  CYTTL　(文字型)　y 座標軸のタイトル.
  CYUNIT　(文字型)　y 座標軸の単位.

- USGRPH(N,X,Y) [自動スケーリングを行って折れ線グラフを描く]

  N　(整数型)　データ数.
  X, Y　(実数型)　折れ線の (x,y) 座標値を与える配列.

第 2 章

つぎのステップ

図形を構成する基本要素は, 折れ線とマーカー列です. (x,y) 座標値の配列とそのデータ数を与えて, 折れ線で結んだり, マーカー列で表示します. いわゆる, 軌跡図や分散図の基本です. 線やマーカーにはそれぞれいくつかの属性があり, これらを変更すると多種多様な線やマーカーが描けます. 前章のおまかせルーチン USGRPH をいくつかのサブルーチンに分解すると, このような変更が容易に行なえます.

グラフ描画の基本要素として, これら二つ以外にも, 文字列を描いたり, 多角形領域を塗りつぶしたりする機能がありますが, ここでは詳しく述べません. 気になる人は 「らくらく DCL」を参考にして下さい.

第 2.1節では 「正規化変換」に関する記述が出てきますが, ちゃんとした説明は次章で行ないます. ここで登場する USPFIT と GRSTRF の二つのサブルーチンは, とりあえず 「おまじない」と思っておいてもかまいません.

2.1　USGRPH の分解

第 1.1節のプログラム HOP で呼んだ USGRPH サブルーチンがどのようなルーチンで構成されているか, 見ておきましょう. 実は,

```
CALL USGRPH( NMAX, X, Y )
```

というサブルーチン ・コールは, 次の 5 つのサブルーチンを順に呼ぶことと同じなのです.

```
CALL USSPNT( NMAX, X, Y )
CALL USPFIT
CALL GRSTRF
CALL USDAXS
CALL UULIN( NMAX, X, Y )
```

データを自動的にスケーリングするためには, まず, 描きたいデータすべてのなかから最大値と最小値を見つける必要があります. サブルーチン USSPNT がこれを行ないます. つぎの USPFIT では, これらのデータの最大値 ・最小値を切りの良い数値に丸めて作画範囲を決め, ほかの 「正規化変換」のパラメータも 「おまかせ」で決めます. そして, GRSTRF ルーチンで 「正規化変換」を確定します. ここで出てきた 「正規化変換」については, 第 3.1節で詳しく説明します. いまは, USPFIT と GRSTRF の二つのサブルーチンで, これらの作業を行なっているとだけ認識しておいて下さい.

次の USDAXS ルーチンは 「おまかせ」で座標軸を描くルーチンです. まさに 「おまかせ」ですから, 引数はありません. そして, UULIN ルーチンで折れ線を描いているのです.

- USSPNT(N,X,Y) [作画範囲に含めたいデータの指定]

  N　(整数型)　データ数.
  X, Y　(実数型)　(x,y) 座標値を与える配列.

- USDAXS [「おまかせ」で座標軸を描く]

  座標軸の目盛りは 4 辺とも描かれるが, ラベルやタイトルは左側と下側だけに描かれる. ラベルが長くなる場合には, 自動的にファクターやオフセットが用いられ, 括弧内に示される. 座標軸のタイトルと単位は, USSTTL で設定すれば, それらも描かれる.

- USPFIT [「おまかせ」で正規化変換のパラメータを決める]

  第 3.1節参照.

- GRSTRF [正規化変換の確定]

  第 3.1節参照.

2.2　基本概念 (2): 折れ線 (軌跡図)

次のプログラム STEP1 では, UULIN サブルーチンを用いて 4 種類の折れ線を一枚の図に重ね書きしてみましょう. まず, USSPNT ルーチンを 4 回呼んで X と Y0, Y1, Y2, Y3 のデータのなかから x と y の最大値と最小値を見つけ,「おまかせ」で正規化変換を確定して, 座標軸も描きます.

さて, 折れ線の描画ですが, 折れ線には, 線種と線の太さの 2 つの属性があります. UUSLNT と UUSLNI のサブルーチンでこれらの属性を変更でき, UULIN ルーチンで折れ線を描きます. 属性を陽に変更しなければ, それぞれの初期値 (デフォルト) がそのまま使われます.

このプログラム例では, まず初期値のまま (X,Y0) の折れ線を描き, 次に少し線を太くして, 破線で (X,Y1) を, 点線で (X,Y2) を, そして 1 点鎖線で (X,Y3) を描いています.

- UULIN(N,X,Y) [折れ線を描く]

  N　(整数型)　データ数.
  X, Y　(実数型)　折れ線を結ぶ点の (x,y) 座標値を与える配列.

- UUSLNT(ITYPE) [折れ線の線種の設定]

  ITYPE　(整数型)　折れ線の線種. 1:実線, 2:破線, 3:点線, 4:1 点鎖線. (初期値は 1)

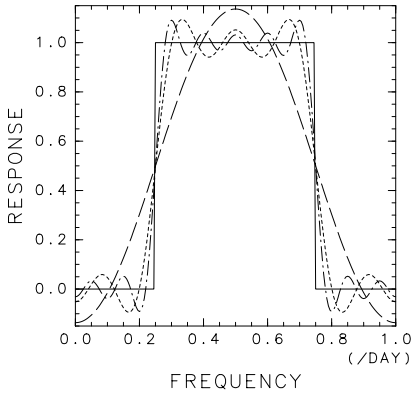
- UUSLNI(INDEX) [折れ線の太さの設定]

  INDEX　(整数型)　折れ線の太さ. 1 から大きくなるにつれて太くなる. (初期値は 1)

これらは, 個々のルーチンで属性を決めてから UULIN で折れ線を描く,「根回し型」のルーチン群です. これに対して, ひとつのサブルーチンで属性も同時に指定して折れ線を描く,「上意下達型」のルーチン UULINZ も用意されています. 好みの方を使って下さい.

- UULINZ(N,X,Y,ITYPE,INDEX) [「上意下達型」で折れ線を描く]

  パラメータは上と同じ.



step1.f: frame1

```
1         PROGRAM STEP1
2
3         PARAMETER( NMAX=201, IMAX=5 )
4         REAL X(NMAX), Y0(NMAX), Y1(NMAX), Y2(NMAX), Y3(NMAX), A(IMAX)
5
6   *-- データ ----
7         PI = 3.14159
8         DO 10 I=1,IMAX
9           II = 2*I - 1
10          A(I) = (-1)**I *2./(II*PI)
11    10 CONTINUE
12
13        DO 20 N=1,NMAX
14          X(N)  = 1.*(N-1)/(NMAX-1)
15          T     = 2.*PI*X(N)
16          IF(T.LT.PI/2. .OR. T.GE.PI*3./2.) THEN
17            Y0(N) = 0.
18          ELSE
19            Y0(N) = 1.
20          END IF
21          Y1(N) = 0.5 + A(1)*COS(T)
22          Y2(N) = 0.5
23          Y3(N) = 0.5
24          DO 30 I=1,IMAX
25            II = 2*I - 1
26            IF(I .LE. 3) Y2(N) = Y2(N) + A(I)*COS(II*T)
27            Y3(N) = Y3(N) + A(I)*COS(II*T)
28    30    CONTINUE
29    20 CONTINUE
30
31  *-- グラフ ----
32        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
33        CALL SGPWSN
34        READ (*,*) IWS
35
36        CALL GROPN( IWS )
37        CALL GRFRM
38
39        CALL USSPNT( NMAX, X, Y0 )
40        CALL USSPNT( NMAX, X, Y1 )
41        CALL USSPNT( NMAX, X, Y2 )
42        CALL USSPNT( NMAX, X, Y3 )
43        CALL USPFIT
44        CALL GRSTRF
45
46        CALL USSTTL( 'FREQUENCY', '/DAY', 'RESPONSE', '' )
47        CALL USDAXS
48
49        CALL UULIN( NMAX, X, Y0 )
50
51        CALL UUSLNT( 2 )
52        CALL UUSLNI( 3 )
53        CALL UULIN( NMAX, X, Y1 )
```

```
54
55          CALL UUSLNT( 3 )
56          CALL UULIN( NMAX, X, Y2 )
57
58          CALL UUSLNT( 4 )
59          CALL UULIN( NMAX, X, Y3 )
60
61          CALL GRCLS
62
63          END
```

2.3 基本概念 (3): マーカー列 (分布図)

前節で紹介した UULIN ルーチンのかわりに UUMRK を用いると, データ列をマーカー列で表現でき, いわゆる分布図が描けます. 次のプログラム STEP2 では, 乱数列をもとに (X,Y) のデータをつくり, 分布図を描きます. 8 行めと 10 行めの関数 RNGU3(ISEED) は, DCL の中の 「その他の基本関数パッケージ」にある, 一様乱数を生成する関数のひとつです. このように, DCL にはグラフィクス以外にもさまざまなパッケージが用意されていることを心に留めておいて下さい. USSPNT でデータの範囲を求め, USPFIT と GRSTRF で正規化変換を確定し, USDAXS で座標軸を描き, UUMRK を使ってマーカー列を描いています.

ところで, マーカーには, マーカーの種類, 描く線の太さ, マーカーの大きさの 3 つの属性があります. UUSMKT, UUSMKI, UUSMKS の各サブルーチンでこれらの属性を変更できます. これらを呼んで思い通りのマーカーが選べたら, UUMRK ルーチンでマーカー列を描くことになります. また, 同様に 「上意下達型」のルーチン UUMRKZ もあります.

このプログラム例では, 太い線で初期値の 1.5 倍の大きさのマーカーを選び, マーカーの種類は, 最初の 25 個を初期値の '·' で, 次からの 25 個ずつをそれぞれ,' +', '∗', '○' で描いています.

- UUMRK(N,X,Y) [マーカー列を描く]

N	(整数型)	データ数.
X, Y	(実数型)	マーカーを打つ点の (x,y) 座標値を与える配列.

- UUSMKT(ITYPE) [マーカーの種類の設定]

ITYPE	(整数型)	マーカーの種類. 1:'·', 2:'+', 3:'∗', 4:'○', など, フォントテーブル (付録) の DCL 文字番号に対応する文字 ・記号を描く. (初期値は 1)

- UUSMKI(INDEX) [マーカーを描く線の太さの設定]

INDEX	(整数型)	マーカーを描く線の太さ. 1 から順に太くなる. (初期値は 1)

- UUSMKS(RSIZE) [マーカーの大きさの設定]

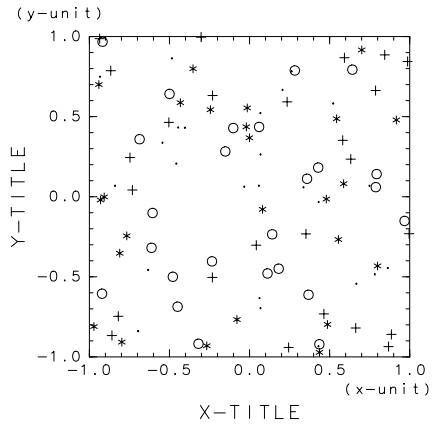
RSIZE	(実数型)	マーカーの大きさ. V-座標系での値で与える. (初期値は 0.01)

- UUMRKZ(N,X,Y,ITYPE,INDEX,RSIZE) [「上意下達型」でマーカー列を描く]

  パラメータは上と同じ.

ここで, マーカーの大きさで出てきた 『V-座標系」について説明しておきましょう. 実際に作画できる領域は出力装置によって異なりますが, DCL ではそれらに最大内接する正方形を考えて, 「描画領域」とします. この 1 辺の長さが 1 になるように規格化した $[0,1]\times[0,1]$ の座標系を V-座標系と呼びます.

```
1         PROGRAM STEP2
2
3         PARAMETER( NMAX=100 )
4         REAL X(NMAX), Y(NMAX)
5
6   *-- データ ----
7         ISEED = 1
8         X(1) = 2.*(RNGU3(ISEED)-0.5)
9         DO 10 N=2,NMAX
10          X(N)   = 2.*(RNGU3(ISEED)-0.5)
11          Y(N-1) = X(N)
12     10 CONTINUE
13        Y(NMAX) = X(1)
14
15  *-- グラフ ----
16        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
17        CALL SGPWSN
18        READ (*,*) IWS
19
20        CALL GROPN( IWS )
21        CALL GRFRM
22
23        CALL USSPNT( NMAX, X, Y )
24        CALL USPFIT
25        CALL GRSTRF
26
27        CALL USSTTL( 'X-TITLE', 'x-unit', 'Y-TITLE', 'y-unit' )
28        CALL USDAXS
29
30        CALL UUSMKI( 5 )
31        CALL UUSMKS( 0.015 )
32        CALL UUMRK( NMAX/4, X( 1), Y( 1) )
33
34        CALL UUSMKT( 2 )
35        CALL UUMRK( NMAX/4, X(26), Y(26) )
36
37        CALL UUSMKT( 3 )
38        CALL UUMRK( NMAX/4, X(51), Y(51) )
39
40        CALL UUSMKT( 4 )
41        CALL UUMRK( NMAX/4, X(76), Y(76) )
42
43        CALL GRCLS
44
45        END
```



step2.f: frame1

第 3 章

そしてジャンプ

これまでとにかく簡単に 2 次元平面内の軌跡図や分布図を描く方法を紹介してきましたが, この章ではもう少しグラフィクスの基本的な概念を理解して, 応用力をつけることにしましょう.

まず, 第 2.1節で宿題となっていた,　「正規化変換」に関して説明します. これらを理解すると, 描画領域内の任意の場所に自分で図の縦横比を決めて作画したり, 対数座標のグラフを描いたりできるようになります.

3.1　基本概念 (4): 正規化変換

グラフ用紙に何かのデータをプロットする時のことを思い浮かべてください. まず, 直角一様座標か対数座標かのグラフ用紙を用意し, これからプロットしようとするデータと, グラフ用紙の目盛りの数をにらんで「一目盛りいくらにしよう」と考えるはずです. このようにして実際のデータの数値とグラフ用紙の目盛りとを対応づけるわけですが, このような操作を DCL では「正規化変換」と呼びます.

前章で使った次のサブルーチン・コールを,「おまかせ」でなく自分で陽に指定することを考えましょう.

```
CALL USSPNT( NMAX, X, Y )
CALL USPFIT
CALL GRSTRF
```

は, 第 1 章のプログラム HOP の場合, 次と同じになります.

```
CALL GRSWND( -100., 100., 5.999, 6.001 )
CALL GRSVPT( 0.2, 0.8, 0.2, 0.8 )
CALL GRSTRN( 1 )
CALL GRSTRF
```

ユーザーの使っている座標系でグラフに描きたい範囲を「ウインドウ」と呼びますが, 上の「おまかせ」では USSPNT ルーチンで X と Y の最小値・最大値を求め, USPFIT で切りの良い値にしてウインドウを設定しています. USSPNT はウインドウ情報を指定する代わりに, これからプロットしたいデータそのものを与えて, これらがすべてウインドウ内に納まるようにするものです. この例のデータでは (UXMIN,UXMAX,UYMIN,UYMAX) = (-100.,100.,5.999,6.001) ですから, この範囲でウインドウを指定するには, GRSWND ルーチンでこれらの値を陽に与えます. 第 1 章の軌跡図 (3ページ) で, 軌跡のまわりに少し余白を与えようと思うと, これらの範囲を大きめにとれば良いことになります (次節のプログラム JUMP1 参照).

次に, このウインドウを V-座標系 (実際に作画できる領域に最大内接する正方形で $[0,1] \times [0,1]$ で規格化された描画領域.　第 2.3 節参照) のどの範囲に対応させるかを考えて, これを「ビューポート」とし

ます. ビューポートとは, V-座標系で通常座標軸が描かれる矩形の領域のことです.　「おまかせ」では (VXMIN,VXMAX,VYMIN,VYMAX) = (0.2,0.8,0.2,0.8) の範囲をビューポートとしますが, ここではGRSVPT ルーチンでこれらの値を陽に与えます.

これで, ウインドウとビューポートの四隅は対応させることが出来ましたが, ウインドウ内の各点をビューポート内の点に対応させる必要があります. 線形に対応させるか, 対数をとって対応させるかなどの任意性がありますから, 具体的に変換関数を決めなければなりません.　「おまかせ」では直角一様座標ですから, GRSTRN ルーチンで変換関数番号を 1 と指定します.

このように設定されたパラメータの値は, 変換関数を確定するルーチン GRSTRF ルーチンを呼ぶことで有効になります. GRSWND などで値を設定しただけでは何も変わらず, GRSTRF が呼ばれてはじめて正規化変換が具体的に決められるのです.

- GRSWND(UXMIN,UXMAX,UYMIN,UYMAX) [ウインドウの設定]

  UXMIN, UXMAX　(実数型)　ウインドウの x 座標の最小値と最大値.
  UYMIN, UYMAX　(実数型)　ウインドウの y 座標の最小値と最大値.

- GRSVPT(VXMIN,VXMAX,VYMIN,VYMAX) [ビューポートの設定]

  VXMIN, VXMAX　(実数型)　ビューポートの x 座標の最小値と最大値.(初期値は 0.2 と 0.8)
  VYMIN, VYMAX　(実数型)　ビューポートの y 座標の最小値と最大値.(初期値は 0.2 と 0.8)

- GRSTRN(ITR) [正規化変換の変換関数番号の設定]

  ITR　(整数型)　変換関数番号. 1:直角一様座標, 2:片対数 (y軸) 座標, 3:片対数 (x 軸) 座標, 4:両対数座標. (初期値は 1)

- GRSTRF [正規化変換の確定]

  ウインドウ, ビューポート, および変換関数を設定したあとで, このルーチンを呼び, 正規化変換を確定する.

USSPNT ルーチンを使ってウインドウを決めたり, 初期値を使うことでビューポートや変換関数番号の設定を省略する場合には, USPFIT ルーチンを呼んで正規化変換の設定をおまかせすることになります. 一方, GRSWND, GRSVPT, GRSTRN の 3 つを自前で呼んでこれらを設定した場合には, あと何も必要ありません. いずれの場合にも, GRSTRF ルーチンを呼んで正規化変換を確定します.

- USPFIT [「おまかせ」で正規化変換のパラメータを決める]

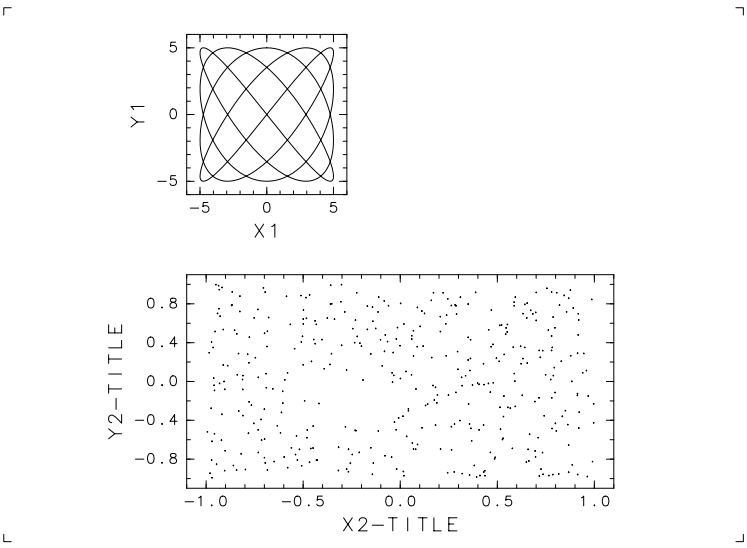
  USSPNT で設定されたデータをもとにウインドウを決め, それ以外のパラメータも陽に指定しなければそれぞれの初期値を使って, 正規化変換を設定する.

3.2　異なる大きさの図形

大きい図形の隣に小さな図形を並べる時には, GRSVPT を使って陽にビューポートを設定することになります. 次のプログラム JUMP1 を御覧下さい. まず, ビューポートを (0.15,0.45,0.65,0.95) として, 描画範囲の左上に正方形の図を描きます. 内容は, 第 1.1節のプログラム HOP と同様のリサジュー図です. 次に, 43 行めで GRFIG ルーチンを呼んで 2 番めの図を描くために必要な初期化をします. GRFRM と違って, 次のフレームには移りません. そして, ビューポートを (0.15,0.95,0.1,0.5) として下の図を描きます. こうすれば, 縦横比が 1:2 の長方形の図となります. 内容は, 第 2.3節の分布図と同様のものです.

- GRFIG [新しい図を描くために必要な初期化をする]

  一つのフレーム内に複数の図を描くとき, 2 つ目以降の図を描き始める前に, そのつど, このルーチンを呼ぶ.



jump1.f: frame1

```
1         PROGRAM JUMP1
2
3         PARAMETER( NMAX=400 )
4         REAL X(NMAX), Y(NMAX)
5
6     *-- データ 1 ----
7         DT = 2.*3.14159 / (NMAX-1)
8
9         DO 10 N=1,NMAX
10          T = DT*(N-1)
11          X(N) = 5.*SIN(4.*T)
12          Y(N) = 5.*COS(5.*T)
13      10 CONTINUE
14
15    *-- グラフ 1 ----
16        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
17        CALL SGPWSN
18        READ (*,*) IWS
19
20        CALL GROPN( IWS )
21        CALL GRFRM
22
23        CALL GRSWND(  -6.,   6.,  -6.,   6. )
24        CALL GRSVPT( 0.15, 0.45, 0.65, 0.95 )
25        CALL GRSTRN( 1 )
26        CALL GRSTRF
27
28        CALL USSTTL( 'X1', '', 'Y1', '' )
29        CALL USDAXS
30
31        CALL UULIN( NMAX, X, Y )
32
33    *-- データ 2 ----
34        ISEED = 1
35        X(1) = 2.*(RNGU3(ISEED)-0.5)
36        DO 20 N=2,NMAX
37          X(N)   = 2.*(RNGU3(ISEED)-0.5)
38          Y(N-1) = X(N)
39      20 CONTINUE
40        Y(NMAX) = X(1)
41
42    *-- グラフ 2 ----
43        CALL GRFIG
44
45        CALL GRSWND(  -1.1,  1.1, -1.1, 1.1 )
```

```
46          CALL GRSVPT(  0.15, 0.95,  0.1, 0.5 )
47          CALL GRSTRN( 1 )
48          CALL GRSTRF
49
50          CALL USSTTL( 'X2-TITLE', '', 'Y2-TITLE', '' )
51          CALL USDAXS
52
53          CALL UUMRK( NMAX, X, Y )
54
55          CALL GRCLS
56
57          END
58
```

3.3 対数座標

次のプログラム JUMP2 は両対数座標のグラフを描く例です. GRSTRN ルーチンで正規化変換の変換関数番号を 4 に指定します.

対数座標軸の描画も USDAXS ルーチンにおまかせ下さい. ここで, USSTTL ルーチンで指定する文字列ですが, '|', '"' などの制御文字を使うと上付や下付の添字も描けます.

- 添字モードの制御文字

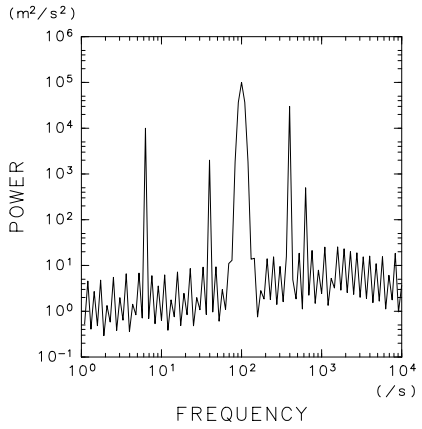
  '|' 上付添字の始まり

  '_' 下付添字の始まり

  '"' 上付または下付添字の終わり

  注意: 文字列が添字で終る場合でも, 最後に必ず '"' を入れて通常の文字モードに戻すこと.

```
1           PROGRAM JUMP2
2
3           PARAMETER ( NMAX=100 )
4           REAL X(NMAX), Y(NMAX)
5
6     *-- データ ----
7           R  = 0.2
8           R0 = 0.0
9           DO 10 I=1,NMAX
10            R  = 3.6*R*(1.-R)
11            R0 = R0 + R*4 - 2.58
12            X2 = (I-50)**2
13            REXP = 4.*I/NMAX
14            X(I) = 10**REXP
15            Y(I) = 1.E5*EXP(-X2) + 10.**R0
16       10 CONTINUE
17          Y(20) = 1.E4
18          Y(40) = 2.E3
19          Y(65) = 3.E4
20          Y(70) = 5.E2
21
22    *-- グラフ ----
23          WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
24          CALL SGPWSN
25          READ (*,*) IWS
26
27          CALL GROPN( IWS )
28          CALL GRFRM
29
30          CALL GRSWND( 1.E0, 1.E4, 1.E-1, 1.E6 )
31          CALL GRSVPT(  0.2,  0.8,   0.2,  0.8 )
32          CALL GRSTRN( 4 )
33          CALL GRSTRF
34
35          CALL USSTTL( 'FREQUENCY', '/s', 'POWER', 'm|2"/s|2"' )
36          CALL USDAXS
37
38          CALL UULIN( NMAX, X, Y )
39
40          CALL GRCLS
41
42          END
```



jump2.f: frame1

第 4 章

おまかせ一次元図

第 2.2節のプログラム STEP1 では, x 座標値の配列を宣言して等間隔に値を代入し, UULIN で $f(x)$ 型の一次元図を描きましたが, これはちょっと大げさな気もします. USGRPH や UULIN などのサブルーチンで　朱定義値」をうまく使うと, x または y 方向の座標値が等間隔な $f(x)$ や $g(y)$ 型のグラフを簡単に作図できます.

4.1　基本概念 (5): 内部変数　朱定義値」

DCL では xxpGET, xxpSET 型の内部変数管理ルーチンが多く使われています (いままで一度もお目にかかっていませんが). xx は通常パッケージの先頭 2 文字で, p は変数の型によって, I(整数型), R(実数型), L(論理型), C(文字型) のうちのひとつになります. 内部変数はそれぞれのパッケージ毎に数多くあり,　朱定義値」'RUNDEF' もそのような実数型内部変数のひとつで, SYSLIB パッケージで管理されています. 一般に, 内部変数は, あらかじめシステムが用意した値　朝期値」をデフォルトで保持しています. 'RUNDEF' の場合, この値を GLRGET ルーチンによって参照し, GLRSET ルーチンによって変更することがでます.

- GLRGET(CP,RPARA) [DCL 全体で使用する実数型内部変数を参照する]

CP	(文字型)	内部変数の名前.
RPARA	(実数型)	内部変数の値.

- GLRSET(CP,RPARA) [DCL 全体で使用する実数型内部変数を変更する]

  パラメータは上と同じ.

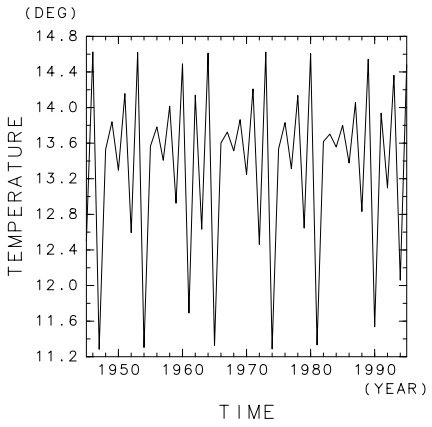
ところで, 'RUNDEF' は　ユーザーが陽に指定していない」ことを表す内部変数で, その値 RUNDEF を　朱定義値」と呼びます. これまで, おまかせサブルーチンの説明で　ら ○ が陽に指定されていなければ」というくだりがいくつかでてきましたが, これは　ら ○ が RUNDEF に等しい時には」ということだったのです. この RUNDEF を USGRPH ルーチンなどの引数に用いると, x または y の座標値を　おまかせ」にすることが出来ます.

この RUNDEF の値は初期値として $-999.$ が与えられていますが, この値が不都合な時 (例えば, -1000 から -999 までのグラフを描く場合など) は GRFRM ルーチンを呼ぶ前に CALL GLRSET('RUNDEF', -999999.) などとして未定義値の値を変更しておきます. GRFRM ルーチンでこの RUNDEF を使っているからです.

4.2 f(i∆x)

次のプログラム例 U1D1 では, x 方向には $[1945, 1995]$ の範囲で等間隔に点をとり, y 方向には配列 Y で与えられた座標値を結んで折れ線を描きます. 25 行めの USGRPH ルーチンで X を指定するかわりに RUNDEF を指定する (つまり, X は定義されていないと宣言する) と, USGRPH は x 座標値がウインドウの幅いっぱいに等間隔にならんでいるものと解釈してグラフを描きます. このとき, USGRPH が呼ばれる前に x 方向のウインドウは決まっていないといけませんから, GRSWND ルーチンで x 方向だけを陽に与えています. ここでも, y 方向は未定義にして, 正規化変換の確定は USGRPH におまかせします. なお, RUNDEF の値は, 13 行めの GLRGET ルーチンで参照しています.

```
1         PROGRAM U1D1
2
3         PARAMETER( NMAX=51, XMIN=1945, XMAX=1995 )
4         REAL Y(NMAX)
5
6   *-- データ ----
7         Y0 = 0.5
8         DO 10 N=1,NMAX
9           Y(N) = 5.*Y0 + 10.
10          Y0   = 3.7*Y0*(1.-Y0)
11    10 CONTINUE
12
13        CALL GLRGET( 'RUNDEF', RUNDEF )
14
15  *-- グラフ ----
16        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
17        CALL SGPWSN
18        READ (*,*) IWS
19
20        CALL GROPN( IWS )
21        CALL GRFRM
22
23        CALL GRSWND( XMIN, XMAX, RUNDEF, RUNDEF )
24        CALL USSTTL( 'TIME', 'YEAR', 'TEMPERATURE', 'DEG' )
25        CALL USGRPH( NMAX, RUNDEF, Y )
26
27        CALL GRCLS
28
29        END
```

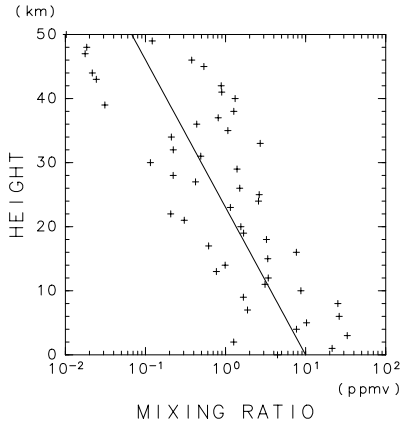


u1d1.f: frame1

4.3 g(jΔy)

前節と同様に, y 方向に座標値が等間隔な場合にも, Y を指定するかわりに RUNDEF を指定します. 次のプログラム U1D2 では, USSPNT, UUMRK, UULIN の各ルーチンで Y を未定義値としています. ここでは USSPNT ルーチンを使ってウインドウを決め, ビューポートの設定は初期値に頼りますので, GRSTRF ルーチンの前に USPFIT を呼んでいることを再確認しておきましょう. なお, 変換関数番号を 3 として, 片対数座標にしています.

```
1         PROGRAM U1D2
2
3         PARAMETER( NMAX=50, YMIN=0., YMAX=50. )
4         REAL X1(0:NMAX), X2(0:NMAX)
5
6   *-- データ ----
7         ISEED = 1
8         DO 10 N=0,NMAX
9           Y = YMIN + (YMAX-YMIN)*N/NMAX
10          X1(N) = 10.*(EXP(-Y/20))**2 * EXP((RNGU3(ISEED)-0.5)*2)**2
11          X2(N) = 10.*(EXP(-Y/20))**2
12      10 CONTINUE
13
14        CALL GLRGET( 'RUNDEF', RUNDEF )
15
16  *-- グラフ ----
17        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
18        CALL SGPWSN
19        READ (*,*) IWS
20
21        CALL GROPN( IWS )
22        CALL GRFRM
23
24        CALL GRSWND( RUNDEF, RUNDEF, YMIN, YMAX )
25        CALL USSPNT( NMAX+1, X1, RUNDEF )
26        CALL USSPNT( NMAX+1, X2, RUNDEF )
27        CALL GRSTRN( 3 )
28        CALL USPFIT
29        CALL GRSTRF
30
31        CALL USSTTL( 'MIXING RATIO', 'ppmv', 'HEIGHT', 'km' )
32        CALL USDAXS
33
34        CALL UUSMKT( 2 )
35        CALL UUMRK( NMAX+1, X1, RUNDEF )
36
37        CALL UULIN( NMAX+1, X2, RUNDEF )
38
39        CALL GRCLS
40
41        END
```



u1d2.f: frame1

第 5 章

おまかせ二次元図

この章では, 2 次元データ $A(x,y)$ をグラフ化します. A がスカラーなら同じ値のところを結ぶ等高線図 (コンター ・プロット) になり, A が 2 次元のベクトルならば矢印を用いた 「流れ図」となります. これらの場合, 作画範囲 (x,y) は自分で把握できているので, ウインドウは自分で陽に設定し, コンターや矢印を 「おまかせ」で描くことにします.

5.1 等高線図

格子点で与えられた 2 次元のスカラーデータを手早く等高線図で描きたいというときには, サブルーチン UDCNTR を呼びます (U2D1). UDCNTR ルーチンは等高線を描くだけですから, まず正規化変換を設定します. ウインドウは GRSWND ルーチンで [TMIN,TMAX]×[ZMIN,ZMAX]($[0,5]\times[20,50]$) と陽に設定しますが, ビューポート ($[0.2,0.8]\times[0.2,0.8]$) と変換関数番号 (1: 直角一様座標) は括弧内の初期値を用いるので, USPFIT ルーチンを呼び, さらに GRSTRF ルーチンでこれらを確定しています.

次に, USSTTL ルーチンで座標軸タイトルの情報を与え, USDAXS ルーチンでおまかせの座標軸を描きます.

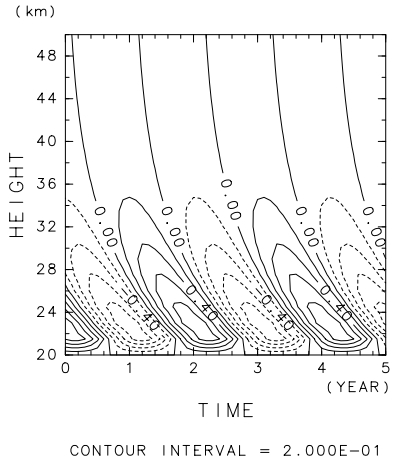
そして, 最後に UDCNTR ルーチンを呼んで等高線を描きます. 現在設定されているウインドウいっぱいに等間隔の格子点が設定されて, コンタリングが行なわれます. つまり, 座標軸の目盛に関係なく, U(1,1) が左下隅, U(NT,NZ) が右上隅にくるように等間隔の格子点座標が設定されます. コンターレベルは自動的に決定され, 図の下にそのコンター間隔が表示されます. UDCNTR の引数の 2 番目以降で配列の寸法を指定しますが, 第 1 次元寸法を 2 度指定するのは, 配列の一部分だけを作画できるようにするためです. この例のようにデータ全部を描く場合は, 2 番目の引数と 3 番目の引数を同じ (NT) にします.

- UDCNTR(Z,MX,NX,NY) [2 次元等高線図を描く]

Z	(実数型)	MX×NY の 2 次元配列. 作画には NX×NY の部分を使う.
MX	(整数型)	配列 Z の第 1 次元整合寸法.
NX	(整数型)	作画に使う配列 Z の第 1 次元寸法.
NY	(整数型)	作画に使う配列 Z の第 2 次元寸法.

```
1         PROGRAM U2D1
2
3         PARAMETER( NT=51, NZ=21 )
4         PARAMETER( TMIN=0, TMAX=5, ZMIN=20, ZMAX=50 )
5         PARAMETER( DT=(TMAX-TMIN)/(NT-1), DZ=(ZMAX-ZMIN)/(NZ-1) )
```

```
6         REAL U(NT,NZ)
7
8   *-- データ ----
9         DO 10 J=1,NZ
10          Z  = (J-1)*DZ
11          UZ = EXP(-0.2*Z)*(Z**0.5)
12          DO 20 I=1,NT
13            T = (I-1)*DT - 2.*EXP(-0.1*Z)
14            U(I,J) = UZ*SIN(3.*T)
15     20   CONTINUE
16     10 CONTINUE
17
18  *-- グラフ ----
19        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
20        CALL SGPWSN
21        READ (*,*) IWS
22
23        CALL GROPN( IWS )
24        CALL GRFRM
25
26        CALL GRSWND( TMIN, TMAX, ZMIN, ZMAX )
27        CALL USPFIT
28        CALL GRSTRF
29
30        CALL USSTTL( 'TIME', 'YEAR', 'HEIGHT', 'km' )
31        CALL USDAXS
32
33        CALL UDCNTR( U, NT, NT, NZ )
34
35        CALL GRCLS
36
37        END
```



u2d1.f: frame1

5.2　トーン付き等高線図

サブルーチン UETONE を呼ぶだけで, とりあえず負の領域にトーンをつけることができます. 次のプログラム例 U2D2 は, U2D1 の結果に加えて, 負の領域に斜線のハッチをつけます. ここで, 24 行めで SGLSET ルーチンを呼んで, トーンに関する論理型内部変数 'LSOFTF' を .TRUE. に設定しています. これを .FALSE.(初期値) で描くと, 出力装置によっては先に描かれた図形が消えてしまうことがあります. 'LSOFTF' の設定を変えたり, UETONE を呼ぶ順序を USDAXS の前にしたりして, 結果がどのようになるか確かめてみましょう. また, この例では, ビューポートを陽に設定して長方形にしている点にも注意して下さい.

- UETONE(Z,MX,NX,NY) [2 次元等値線図の塗り分けを行なう]

Z　(実数型)　MX×NY の 2 次元配列. 作画には NX×NY の部分を使う.

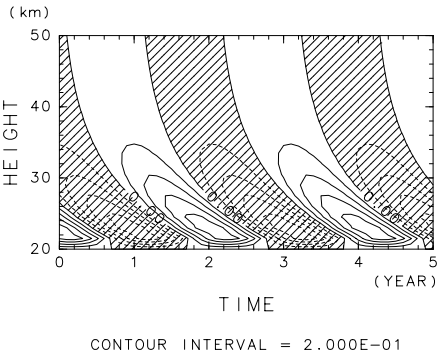
MX　(整数型)　配列 Z の第 1 次元整合寸法.

NX　(整数型)　作画に使う配列 Z の第 1 次元寸法.

NY　(整数型)　作画に使う配列 Z の第 2 次元寸法.

塗り分けの設定を陽に行なわなければ, 負の領域に斜線を引く. この塗り分けレベルとパターンの設定は第 5.4節を参照のこと.

```
      PROGRAM U2D2

      PARAMETER( NT=51, NZ=21 )
      PARAMETER( TMIN=0, TMAX=5, ZMIN=20, ZMAX=50 )
      PARAMETER( DT=(TMAX-TMIN)/(NT-1), DZ=(ZMAX-ZMIN)/(NZ-1) )
      REAL U(NT,NZ)

*-- データ ----
      DO 10 J=1,NZ
        Z  = (J-1)*DZ
        UZ = EXP(-0.2*Z)*(Z**0.5)
        DO 20 I=1,NT
          T = (I-1)*DT - 2.*EXP(-0.1*Z)
          U(I,J) = UZ*SIN(3.*T)
   20   CONTINUE
   10 CONTINUE

*-- グラフ ----
      WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
      CALL SGPWSN
      READ (*,*) IWS

      CALL GROPN( IWS )
      CALL SGLSET( 'LSOFTF', .TRUE. )
      CALL GRFRM

      CALL GRSWND( TMIN, TMAX, ZMIN, ZMAX )
      CALL GRSVPT(  0.2,  0.9,  0.4,  0.8 )
      CALL USPFIT
      CALL GRSTRF

      CALL USSTTL( 'TIME', 'YEAR', 'HEIGHT', 'km' )
      CALL USDAXS

      CALL UDCNTR( U, NT, NT, NZ )
      CALL UETONE( U, NT, NT, NZ )

      CALL GRCLS

      END
```



u2d2.f: frame1

5.3　ベクトル場

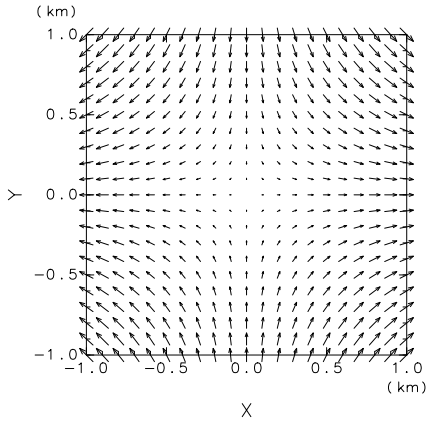
今度は, 2 次元のベクトル場を手早く矢印で描きたいというときの例題です. 次のプログラム U2D3 は簡単な変形場を描くものですが, サブルーチン UGVECT 1 つを呼ぶだけで十分です. 前節の等高線図の場合と同様に, おまかせの座標軸を描画したあとで UGVECT ルーチンを呼んでベクトル場を描いています.

この例でも, 等間隔の格子点を設定して, それぞれの格子点でのベクトルを矢印で表現します. おまかせ描画のときには, 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ます. この場合, x 成分と y 成分のスケーリングファクターは同じになっていて, 図の下部マージンにはその値が表示されています.

- UGVECT(U,MU,V,MV,NX,NY) [2 次元ベクトル場を描く]

U	(実数型)	ベクトルの x 成分を与える MU×NY の 2 次元配列. 作画には NX×NY の部分を使う.
MU	(整数型)	配列 U の第 1 次元整合寸法.
V	(実数型)	ベクトルの y 成分を与える MV×NY の 2 次元配列. 作画には NX×NY の部分を使う.
MV	(整数型)	配列 V の第 1 次元整合寸法.
NX	(整数型)	作画に使う配列 U, V の第 1 次元寸法.
NY	(整数型)	作画に使う配列 U, V の第 2 次元寸法.

```
1         PROGRAM U2D3
2
3         PARAMETER( NX=21, NY=21 )
4         PARAMETER( XMIN=-1, XMAX=1, YMIN=-1, YMAX=1 )
5         PARAMETER( DX=(XMAX-XMIN)/(NX-1), DY=(YMAX-YMIN)/(NY-1) )
6         REAL U(NX,NY), V(NX,NY)
7
8   *-- データ ----
9         DO 10 J=1,NY
10        DO 10 I=1,NX
11          X = XMIN + (I-1)*DX
12          Y = YMIN + (J-1)*DY
13          U(I,J) =   X
14          V(I,J) = - Y
15     10 CONTINUE
16
17  *-- グラフ ----
18        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
19        CALL SGPWSN
20        READ (*,*) IWS
21
22        CALL GROPN( IWS )
23        CALL GRFRM
24
25        CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
26        CALL USPFIT
27        CALL GRSTRF
28
29        CALL USSTTL( 'X', 'km', 'Y', 'km' )
30        CALL USDAXS
31
32        CALL UGVECT( U, NX, V, NX, NX, NY )
33
34        CALL GRCLS
35
36        END
```



u2d3.f: frame1

5.4　トーン付き等高線とベクトル場の重ね書き

最後にトーン付の等高線図とベクトル場を重ね書きした例を示します (U2D4). UGVECT, UDCNTR, UETONE のサブルーチンを順に呼んでいます. ここでは, UESTLV ルーチンでトーンをつけるレベルとパターンを指定しています.

- UESTLV(TLEV1,TLEV2,IPAT) [トーンを塗り分けるレベルとパターンの設定]

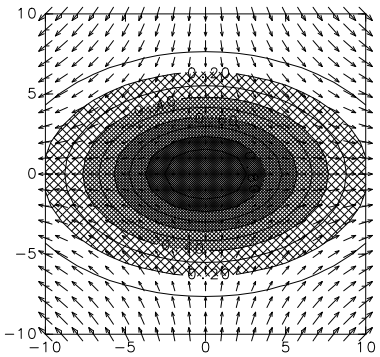
  TLEV1　(実数型)　塗り分けるレベルの下限値.
  TLEV2　(実数型)　塗り分けるレベルの上限値.
  IPAT　(整数型)　トーンパターン番号.

- トーンパターン番号 [3 桁の整数 ijk]

  i　パターンの種類. 0:ドット, 1:横線, 2:斜線 (右上がり), 3:縦線, 4:斜線 (左上がり), 5:格子 (縦横), 6:格子 (斜め).
  j　ドットの大きさや斜線の太さ. 0 から 5 まで値が大きくなるにつれて, ドットは大きく, 線は太くなる.
  k　ドットや斜線の密度. 0 から 5 まで値が大きくなるにつれて, 密度が高くなる.

```
1         PROGRAM U2D4
2
3         PARAMETER( NX=21, NY=21 )
4         PARAMETER( XMIN=-10, XMAX=10, YMIN=-10, YMAX=10 )
5         PARAMETER( KMAX=5, PMIN=0, PMAX=1 )
6         REAL U(NX,NY), V(NX,NY), P(NX,NY)
7
8         DO 10 J=1,NY
9         DO 10 I=1,NX
10          X = XMIN + (XMAX-XMIN)*(I-1)/(NX-1)
11          Y = YMIN + (YMAX-YMIN)*(J-1)/(NY-1)
12          U(I,J) =  X
13          V(I,J) = -Y
14          P(I,J) = EXP( -X**2/64 -Y**2/25 )
15       10 CONTINUE
16
17        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
```

```
18          CALL SGPWSN
19          READ (*,*) IWS
20
21          CALL GROPN( IWS )
22          CALL SGLSET( 'LSOFTF', .TRUE. )
23          CALL GRFRM
24
25          CALL GRSWND( XMIN, XMAX, YMIN, YMAX )
26          CALL USPFIT
27          CALL GRSTRF
28
29          CALL USDAXS
30          CALL UGVECT( U, NX, V, NX, NX, NY )
31          CALL UDCNTR( P, NX, NX, NY )
32
33          DP = (PMAX-PMIN)/KMAX
34          DO 20 K=1,KMAX
35            TLEV1 = (K-1)*DP
36            TLEV2 = TLEV1 + DP
37            IPAT  = 600 + K - 1
38            CALL UESTLV( TLEV1, TLEV2, IPAT )
39        20 CONTINUE
40
41          CALL UETONE( P, NX, NX, NY )
42
43          CALL GRCLS
44
45          END
```



u2d4.f: frame1

第 6 章

レイアウトしよう

レイアウトのパッケージを用いると, ひとつの用紙に何個もの図形を並べたり, 描画領域をそれぞれの出力装置のもつ作画可能な領域全体に広げたりすることが簡単にできます.

6.1　フレームの分割

同じ様な図形を沢山並べたい時, GROPN のあとで SLDIV ルーチンを呼ぶと, 第 1 レベルめ (用紙全体) のフレームが分割され, 次のレベルのフレームが定義されます. 分割されたフレームをあたかも 1 枚の紙のように扱って, GRFRM の実行により, 次のフレームに自動的に移っていきます.

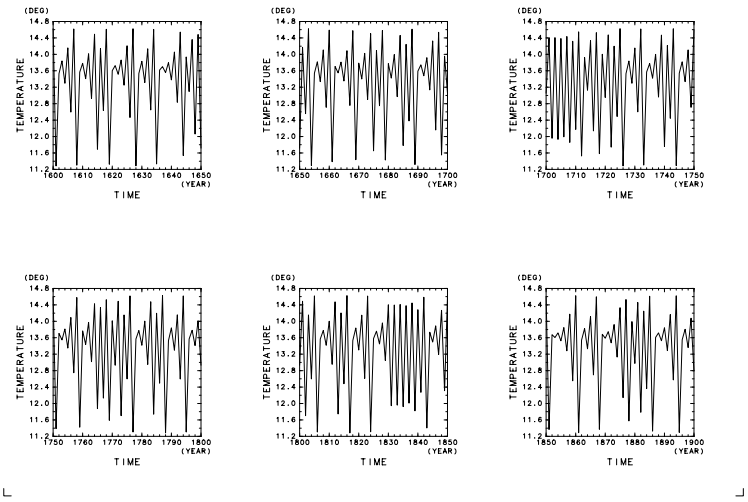
次のプログラム LAY1 では, DO 20 のループの中で, 普通に改ページをしながら描画するのと同じように, GRFRM と USGRPH を呼んで折れ線グラフを描いています. 21 行めの SLDIV ルーチンでは $3 \times 2 = 6$ 分割していますから, 7 番めの図は自動的に次のページに移っています. また, 第 4.2節の出力結果と比較して, 文字の大きさが分割されたフレームの大きさに応じて小さくなっていることにも注意して下さい.

- SLDIV(CFORM,IX,IY) [フレームの分割を行ない, 次のレベルのフレームを定義する]

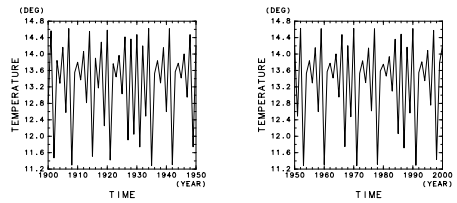
  CFORM　(文字型)　フレームを順に割り付けていく方向. 'T':縦方向, 'Y':横方向.
  IX, IY　(整数型)　x 方向, y 方向の分割数.

```
1         PROGRAM LAY1
2
3         PARAMETER( NMAX=401, JMAX=51 )
4         REAL Y(NMAX), YY(JMAX)
5
6   *-- データ ----
7         Y0 = 0.5
8         DO 10 N=1,NMAX
9           Y(N) = 5.*Y0 + 10.
10          Y0   = 3.7*Y0*(1.-Y0)
11      10 CONTINUE
12
13        CALL GLRGET( 'RUNDEF', RUNDEF )
14
15  *-- グラフ ----
16        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
17        CALL SGPWSN
18        READ (*,*) IWS
19
20        CALL GROPN( IWS )
21        CALL SLDIV( 'Y', 3, 2 )
22
23        DO 20 I=1,8
24          CALL GRFRM
```

```
25
26          IJMIN = (I-1)*(JMAX-1) + 1
27          IJ = IJMIN
28          DO 30 J=1,JMAX
29            IJ = IJ + 1
30            YY(J) = Y(IJ)
31        30  CONTINUE
32          XMIN = IJMIN + 1599
33          XMAX =  XMIN + JMAX - 1
34
35          CALL GRSWND( XMIN, XMAX, RUNDEF, RUNDEF )
36          CALL USSTTL( 'TIME', 'YEAR', 'TEMPERATURE', 'DEG' )
37          CALL USGRPH( JMAX, RUNDEF, YY )
38        20 CONTINUE
39
40          CALL GRCLS
41
42          END
```



lay1.f: page1

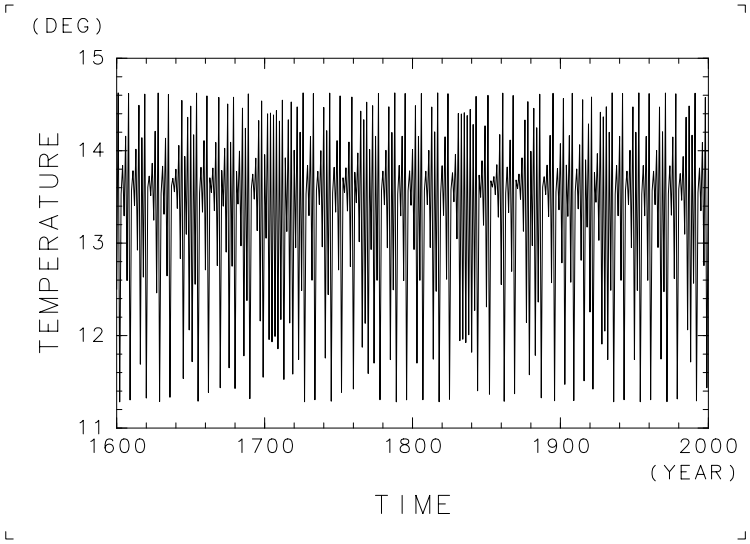


lay1.f: page2

6.2　描画領域の変更

前章までの例では, 用紙の形は長方形なのに, そこに最大内接する正方形の領域をとって $[0,1] \times [0,1]$ の V-座標系を考え, その中だけに図を描いてきました. これに対して, 紙一杯に図を描く例が次のプログラム LAY2 です. 物理的な描画範囲一杯に作画したい時は, SGLSET ルーチンを用いてレイアウトに関する論理型内部変数 'LFULL' を .TRUE. にします. 長辺が V-座標系で $[0,1]$ となりますから, ここではビューポートを (0.15,0.95,0.15,0.65) として物理的な作画範囲に納まるようにとっています. この範囲は出力装置によって異なりますので, プログラムによっては出力する際にエラーを起こす可能性もあります. 御注意下さい.

```
1         PROGRAM LAY2
2
3         PARAMETER( NMAX=401, XMIN=1600, XMAX=2000 )
4         REAL X(NMAX), Y(NMAX)
5
6   *-- データ ----
7         Y0 = 0.5
8         DO 10 N=1,NMAX
9           X(N) = XMIN + (XMAX-XMIN)*(N-1)/(NMAX-1)
10          Y(N) = 5.*Y0 + 10.
11          Y0   = 3.7*Y0*(1.-Y0)
12     10 CONTINUE
13
14  *-- グラフ ----
15        WRITE(*,*) ' WORKSTATION ID (I)  ? ;'
16        CALL SGPWSN
17        READ (*,*) IWS
18
19        CALL GROPN( IWS )
20        CALL SGLSET( 'LFULL', .TRUE. )
21        CALL GRFRM
22
23        CALL GRSWND( XMIN, XMAX,  11.0, 15.0 )
24        CALL GRSVPT( 0.15, 0.95,  0.15, 0.65 )
25        CALL USPFIT
26        CALL GRSTRF
27
28        CALL USSTTL( 'TIME', 'YEAR', 'TEMPERATURE', 'DEG' )
29        CALL USDAXS
30
31        CALL UULIN( NMAX, X, Y )
32
33        CALL GRCLS
34
35        END
```



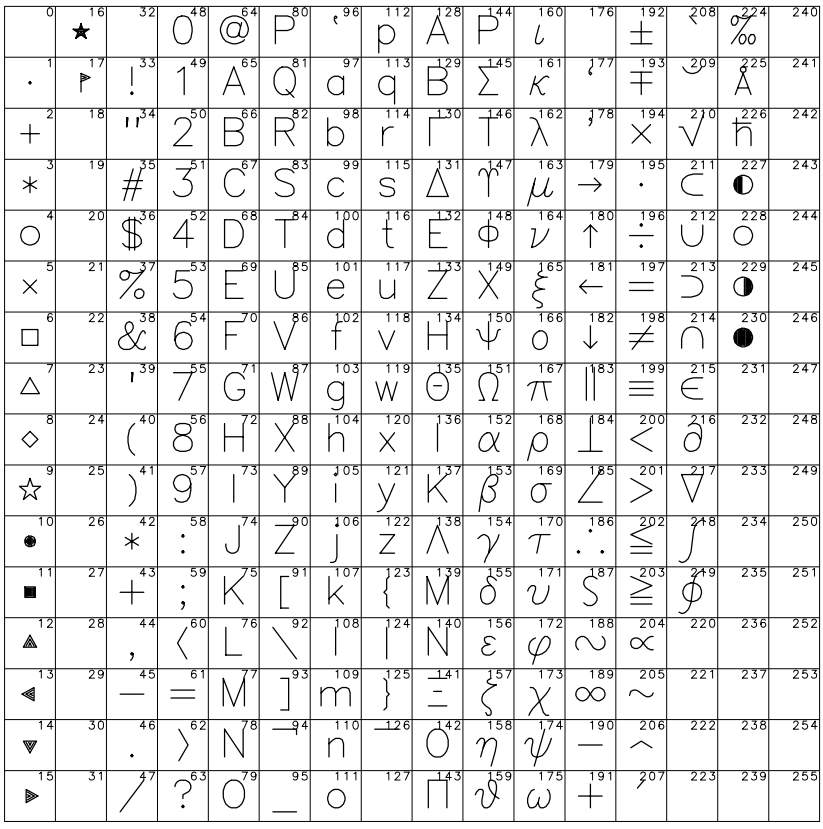
lay2.f: frame1

付録

フォントテーブル

次のテーブルは, フォント番号 1 のフォントテーブルです.

FONT NO. = 1



sgfont.f: frame1

サブルーチン索引

MATH1(数学処理下位パッケージ)

SYSLIB(内部変数管理, メッセージ出力)

サブルーチン	機能	解説ページ
GLRGET(CP,RPARA)	DCL 全体で使用する実数型内部変数を参照する	15
GLRSET(CP,RPARA)	DCL 全体で使用する実数型内部変数を変更する	15

GRPH1(図形処理下位パッケージ)

SLPACK(レイアウトルーチン)

サブルーチン	機能	解説ページ
SLDIV(CFORM,IX,IY)	フレームの分割, 次レベルのフレームの定義	24

GRPH2 図形処理上位パッケージ

GRPACK(上位コントロールルーチン)

サブルーチン	機能	解説ページ
GROPN(IWS)	出力装置のオープン	4
GRFRM	フレームの設定	4
GRFIG	次の図の初期化	12
GRCLS	出力装置のクローズ	4
GRSWND(UXMIN,UXMAX,UYMIN,UYMAX)	ウインドウの設定	11
GRSVPT(VXMIN,VXMAX,VYMIN,VYMAX)	ビューポートの設定	11
GRSTRN(ITR)	正規化変換の変換関数番号の設定	11
GRSTRF	正規化変換の確定	6, 11

UUPACK(1 次元グラフルーチン)

サブルーチン	機能	解説ページ
UULIN(N,X,Y)	折れ線を描く	6
UUSLNT(ITYPE)	折れ線の線種の設定	6
UUSLNI(INDEX)	折れ線の太さの設定	6
UULINZ(N,X,Y,ITYPE,INDEX)	「上意下達型」で折れ線を描く	6
UUMRK(N,X,Y)	マーカー列を描く	8
UUSMKT(ITYPE)	マーカーの種類の設定	8
UUSMKI(INDEX)	マーカーを描く線の太さの設定	8
UUSMKS(RSIZE)	マーカーの大きさの設定	8
UUMRKZ(N,X,Y,ITYPE,INDEX,RSIZE)	「上意下達型」でマーカー列を描く	8

USPACK(自動スケーリングルーチン)

サブルーチン	機能	解説ページ
USGRPH(N,X,Y)	自動スケーリングで折れ線グラフを描く	4
USSTTL(CXTTL,CXUNIT,CYTTL,CYUNIT)	座標軸のタイトル ・単位の指定	4
USSPNT(N,X,Y)	作画範囲に含めたいデータの指定	6
USPFIT	「おまかせ」で正規化変換を設定	6, 11
USDAXS	「おまかせ」で座標軸を描く	6

UDPACK, UEPACK, UGPACK(2 次元量のグラフ表示ルーチン)

サブルーチン	機能	解説ページ
UDCNTR(Z,MX,NX,NY)	2 次元等高線図を描く	18
UETONE(Z,MX,NX,NY)	2 次元等値線図を塗り分ける	19
UESTLV(TLEV1,TLEV2,IPAT)	トーンレベルとパターンの設定	22
UGVECT(U,MU,V,MV,NX,NY)	2 次元ベクトル場を描く	21

謝辞

資源	**dcl-5**
プログラム製作	石岡圭一, 酒井敏, 佐藤薫, 塩谷雅人, 沼口敦
プログラム作成協力	石渡正樹, 竹広真一, 中島健介, 西憲敬, 林祥介, 保坂征宏, 余田成男
デバッグ協力	石田明生, 石渡正樹, 市川香, 沖大幹, 沖理子, 桜井隆博, 佐藤正樹, 竹広真一, 中島健介, 沼口敦, 根田昌典, 野村真佐子, 藤尾伸三, 保坂征宏
文章	酒井敏, 塩谷雅人, 余田成男
文章協力	石渡正樹, 佐藤薫, 佐藤正樹, 竹広真一, 中島健介, 西憲敬, 沼口敦, 林祥介, 保坂征宏
データソース	NCAR [フォント情報, 地図情報]
参考資源	Layout [二木徹], XYGRAPH [佐藤亨], Numerical Recipes [Willam H. Press et al.], NCARG [NCAR], FFTPACK [Paul Swarztrauber]

版権

本資源の版権は地球流体電脳倶楽部に属する. 資源の利用にあたっては地球流体電脳倶楽部の定める規定にしたがっていただきたい. 原則として, 教育的目的の場合には自由に使用・改変して良いものとしている. 地球流体電脳倶楽部が定める規定の詳細は地球流体電脳倶楽部サーバー

```
dennou.gaia.h.kyoto-u.ac.jp
```

上の

```
~ftp/saloon/dennou/To.Users
```

というファイルに収められているので参照されたい. (このファイルは電脳ライブラリを収めたファイルシステムのトップディレクトリにも置かれている.) 上記のアクセスができない場合は, 電子メールにて地球流体電脳倶楽部管理グループ

```
dennou_admin@gaia.h.kyoto-u.ac.jp
```

へ連絡されたい.

引用について

「地球流体電脳ライブラリ」の著作者は「地球流体電脳倶楽部」, その英語名は 'GFD-DENNOU Library', 'GFD-DENNOU Club' である. 地球流体電脳ライブラリを用いて作成された著作物等には, 英語では例えば "The figures were produced by GFD-DENNOU Library." 日本語では例えば「図の作成には地球流体電脳ライブラリをもちいた.」のように引用していただければ幸いである.