目 次

1　はじめに

学生や研究者だけでなくほとんどの人は自分の考えを他の人に伝えるとき、写真・図形・グラフ・文字などの情報を使用する。一般に、これらの情報は一枚の紙に描かれていることもあるし、パンフレットのように綴じられた何枚かの紙にレイアウトされていることもある。最近では、コンピュータネットワークのホームページであったり、パワーポイントのようなプレゼンテーションソフトウェアで書かれたコンピュータ画面の一ページであったりする。形は異なってもこれらの表現形態に共通な点は、他人に自分の考えを端的に伝えるために二次元矩形領域に写真・図形・グラフ・文字を適切に配置していることである。

このような情報のレイアウトを目的としてアドビ社によって作られた言語がポストスクリプト (PostScript) である。ポストスクリプト言語は現在市販されている多くのコンピュータ用プリンタに採用されており、いくつかのコンピュータの画面を記述する言語としても採用されている。

科学の多くの分野でグラフという概念は重要な位置を占めている。数学を頻繁に使用する理工学の分野はもちろんのこと、ほとんど数学を使わない分野においても、いくつかの量の間の関係を調べたり、因果関係を調べるとき、グラフを使用する。いろいろな現象には数多くの要素が互いに関係していて複雑な相関関係を持っている場合が多いが、科学はそのもつれあった関係という糸を解きほぐそうとする努力そのものでもある。そのような場合、人間の思考に一番適合しているのが２次元平面グラフである。このような２次元グラフを描くのにもポストスクリプトは適している。今では、コンピュータを用いていくつかの数値の間の関係を調べるときには、表計算ソフトウェアが使われており、それらの表計算ソフトウェアには簡単にグラフを描く機能が備わっている。ポストスクリプト言語はこれらの表計算ソフトウェアが描くグラフに比べてどのように優れているのか。その解答はこの解説書を読んで自分で見つけていただきたい。ただし、表計算ソフトウェアとポストスクリプト言語は相反するものではない。

科学技術の分野では計算のためにフォートラン (Fortran) 言語やＣ言語が使われる。これらの言語は手続き形の言語と呼ばれ、人間が考える手順に従ってコンピュータが計算を行うようにコンピュータに指示をするための言語である。ポストスクリプト言語も同様に手続き形の言語である。しかし、フォートラン言語やＣ言語は変数を多用して手順を記述するのに対して、ポストスクリプト言語はスタックと呼ばれる方法を使用する。この方法は料理をするときにお皿を積み重ねていった後、積み重ねた順と逆の順番にお皿を使用する方法によく似ている。一世代前のコンピュータでよく用いられたアゼンブラー言語を経験した人たちにはなじみ深いものであり、今でも電卓を使うときの方法にもよく似ている。この方法は逆ポーランド形式と呼ばれるものである。逆ポーランド形式言語とフォートラン言語やＣ言語とはその考える手順がずいぶんと異なるため、一方の言語に慣れた人にとっては他方が非常に取っつきにくく感じられる。

この解説書はフォートラン言語やＣ言語を使う人がポストスクリプト言語を用いてプレゼンテーション用のページを記述するための手段を説明するものである。ここで提供される道具 (サブルーチンや関数) を用いれば、ポストスクリプト言語に精通することなく、ポストスクリプト言語の長所のみを使うことができる。また、ここで提供される道具は各自で改変、改良が可能であり、いくらでも変形することができる長所がある。ただし、そのためにはポストスクリプト言語の習得が必要となってくる。

2　ポストスクリプトの概要と命令

ポストスクリプト言語はもともとコンピュータ用の言語として開発されたものをアドビ社が買い取ってレーザープリンタ用の言語として完成したものである。レーザープリンタについては現在もまだプリンタ製造各社が独自の言語を使ってプリントするように設定している。たとえば、キャノンは LIPS という仕様、エプソンなどは ESC/P という仕様などを用いている。今後の流れとして、ポストスクリプトによるプリンタ制御が増えるものと予想されている。

ポストスクリプト言語は図形・画像・文字などを含む‘ページ’を記述できる言語であることから、レーザープリンタだけでなく、コンピュータ画面へ出力する‘ページ’を記述するのにも適した言語である。実際、サン社やネクスト社から発売されたコンピュータにおける画面出力には標準としてポストスクリプト言語が採用された。現在でもマイクロソフト社の OS（オペレーションシステム）である Windows 98/NT/2000 上で動くソフトウェアやアップル社のコンピュータ上で動くソフトウェアには画面出力とプリンタ出力をポストスクリプト言語で記述しているものが数多く存在する。

ポストスクリプト言語は普通のコンピュータ言語と趣きを異にしている。その最大の特徴は逆ポーランド記述と呼ばれる記述の仕方にある。この記述法は電卓を使い慣れた人になじみの深いものである。古くから用いられてきたコンピュー

タでの図形やグラフ作成のための道具にはカルコンプ社のプロットルーチンや GKS、PHIGS などと呼ばれるものがあるが、これらと比較して数多くの点でポストスクリプト言語は図形およびグラフ作成という面で優れた特徴がある。第１に線の太さや種類が簡単にきめ細かく設定できる。第２にコンピュータで出力したとは思えないほど美しい文字を入れてレタリングできる。その他にも再帰呼び出しや多くの優れた面を持っているがそれらは少し専門的になる。

3　ポストスクリプトの使い方の基本例

ここではポストスクリプト言語の使い方について具体例を用いて説明を行う。ポストスクリプトでは、逆ポーランド記述を用いる。逆ポーランド記述の方法を説明するために、ポストスクリプト言語を用いて簡単な計算を行う。例えば、$1 \times 2 + (3 + 4)/7$ をポストスクリプト言語を用いて表現すると次のようになる。

プログラム 1

```
1 2 mul 3 4 add 7 div add
```

このポストスクリプトによる表現は、“1 と 2 をかけたものと 3 と 4 を加えて 7 で割ったものを加える” という日本語による表現の語順に一致している。しかし、ポストスクリプト言語は科学技術計算によく用いられるプログラミング言語である C 言語やフォートラン (Fortran) 言語とは、記述法が全く異なる言語であることがよくわかる。この解説書は、科学技術計算で良く使われる、C 言語やフォートラン (Fortran) 言語を用い、図形やグラフを描くのに、それらの言語からポストスクリプト言語を呼び出して使う方法について説明する。

ポストスクリプトでは、仮想的なページであるカレントページに描画を行う。プログラム開始時点では、カレントページは空の状態である。このカレントページ上にカレントパスで直線や曲線などのパスを定義する。この定義では、まだ紙面に直線や曲線などのパスで定義した図形は描画されていない。このパスをカレントページに後述の stroke オペレータや fill オペレータを用いて、線を描いたり、パス内を塗りつぶしたりして使用する。ポストスクリプト言語では、さまざまな線や文字を描くのに座標を用いる。特に指定しなければ座標は、図 1 に示すように紙 (A4: 21.0cm × 29.7cm) の左下を原点にし、座標の単位は、ポイント (pt) であり 1 ポイントが 1/72 インチ（1 インチ ∼ 2.5cm）の大きさである。ただし、単位としてはセンチやインチなどを定義して使うことも可能である。

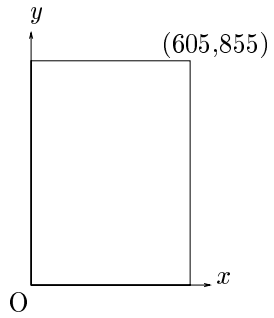

図 1: ポストスクリプトの座標．単位はポイント (pt)．

3.1　線と正方形

直線

最も簡単で基本的な例として 2 点間を直線で結ぶ例を考える。(72,144) から (216,144)(単位は pt) まで直線を引くプログラムは次のようになる。

プログラム 2

```
1: %!
2: newpath
3: 72 144 moveto
4: 216 144 lineto
5: stroke
6: showpage
```

図 2: 直線.

出力結果を図 2 に示す。このプログラムを一行づつ説明する。1 行目の ”%! ”は、これ以降がポストスクリプト言語で記述されていることを示しており、表示プログラム (Ghostscript) やプリンタはポストスクリプト翻訳モードに入る。2 行目に newpath オペレータが呼び出されている。このオペレータはカレントパスの内容を空にし、新しいパスを開始することを宣言している。3 行目と 4 行目の数字が座標を表している。moveto オペレータで、その指定された座標にペンをおき、lineto オペレータの指定する座標まで線を引くことになる。つまりこの線は、用紙の左下から x 方向に 72pt(=1 インチ)、y 方向に 144pt(=2 インチ) の点から x の正の方向に 144pt(=2 インチ) の線を引くことになる。4 行目にある stroke オペレータは、作成したパスをカレントページ上に描画する働きをする。これにより、作成したパスが目に見える線になる。最後の showpage により、カレントページを表示する。

点線

プログラム 2 に 1 行加えると点線を書くことができる。

プログラム 3

```
1: %!
2: newpath
3: [10 5] 0 setdash
4: 72 144 moveto
5: 216 144 lineto
6: stroke
7: showpage
```


図 3: [10-5] 点線. 10 ポイントの長さの線を引き，5 ポイントあけて再び 10 ポイントの長さの線を引く.

出力結果は図 3 に示すようになる。オペレータ setdash は今後描く線が点線であることを示す。[ ] 内の数字で点線の間隔を設定し、[ ] の右横の数字で点線をどこから書き始めるかを決めるのである。この例では、幅 10(pt) の線分を引き、5 の隙間をあける点線で、0 進んだところから点線を書き始める。同じ間隔の点線で、書き始めの位置を変えたものを並べると、「書き始め」の意味が理解できるだろう (図 4)。

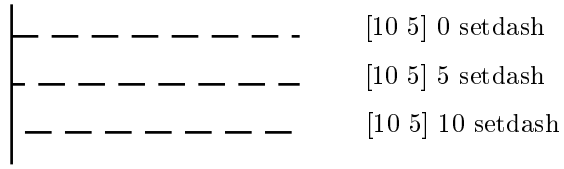

図 4: 書き始めを変えたときの点線.

[ ] 内の数字は 2 つでなくてもよく、[ ] 内に並べられた数字を左から順に線、隙間、線、隙間、... とみなして点線を描く。例えば、上の例 [10 5] の代わりに [15 3 3 3] を用いると次のようになる (図 5)。
このように [ ] 内の数字の並びを考えれば、さまざまな点線を書くことが可能である。特に [ ] 0 setdash とするとこれは実線となる。

正方形

次に、上の例を応用して正方形を書くと以下のようになる (図 6)。

図 5: 15-3-3-3 点線.

プログラム 4

```
1: %!
2: newpath
3: 72 144 moveto
4: 72 504 lineto
5: 432 504 lineto
6: 432 144 lineto
7: 72 144 lineto
8: stroke
9: showpage
```

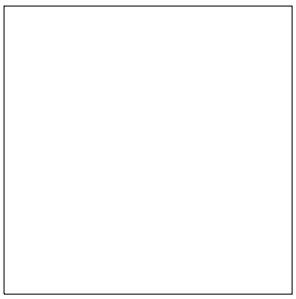

図 6: 正方形.

このプログラムは、正方形を左下から時計まわりに描いている。ここで、始点と終点である左下を拡大したものを図 7 に示す。

図 7: 正方形の拡大図.

図 7 より正方形の左下が欠けていることがわかる。これは線が太さを持っているために起こる。この問題は次のように closepath オペレータを用いて書き換えると解消される。

プログラム 5

```
1: %!
2: newpath
3: 72 144 moveto
4: 72 504 lineto
5: 432 504 lineto
6: 432 144 lineto
```

```
 7: closepath
 8: 12 setlinewidth
 9: stroke
10: showpage
```

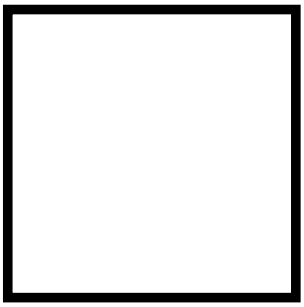

図 8: 正確な正方形.

プログラム 5 の出力結果を図 8 に示す。オペレータ closepath は、始点（moveto オペレータ）と終点（最後の lineto オペレータ）を直線で結ぶ役割をする。そのため、不完全な正方形のプログラムにある最後の lineto オペレータがいらなくなる。closepath オペレータの次に新しく setlinewidth オペレータが呼び出されている。このオペレータは線の太さを決めるもので、この場合は 12/72 インチの太さで線を引くように命令している。

正方形の内部を塗りつぶすこともできる。次のプログラム 6 が正方形内部を塗りつぶす例である。

プログラム 6

```
1: %!
2: newpath
3: 72 144 moveto
4: 72 504 lineto
5: 432 504 lineto
6: 432 144 lineto
7: closepath
8: fill
9: showpage
```

ここでは、パスをストロークするかわりに fill オペレータが呼び出されている。このオペレータはカレントパスの内部をインクで塗りつぶす働きをするので内部が黒く塗りつぶされた。色は黒だけでなく濃さを変えることもできる。

プログラム 7

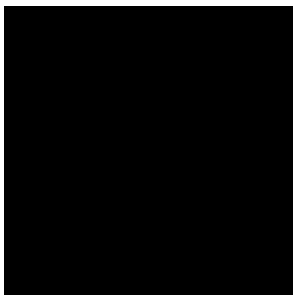

図 9: 塗りつぶされた正方形.

```
 1: %!
 2: newpath
 3: 72 144 moveto
 4: 72 504 lineto
 5: 432 504 lineto
 6: 432 144 lineto
 7: closepath
 8: 0.7 setgray
 9: fill
10: showpage
```

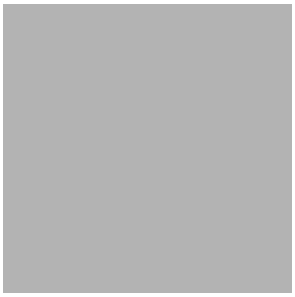

図 10: 灰色の正方形.

オペレータ setgray で灰色の濃さを決定することができる。この例では、濃さを 0.7 としており、0 が黒に、1 が白に対応する。

3.2 曲線と円

円弧を使ってつくる曲線と円について説明する。円弧の曲率中心点の座標が $(144, 144)$ であり、半径が 216 である円弧を描くには次のプログラム 9 のようになる。出力結果を図 11 に示す。

プログラム 8

```
1: %!
2: newpath
3: 144 144 216 0 90 arc
4: 15 setlinewidth
5: stroke
6: showpage
```

プログラムの 3 行目の arc オペレータが円弧を描く働きをする。その arc オペレータの前に 5 つの数字が並んでいる。左から円弧の曲率中心点の座標 (x,y)、曲率半径、x 軸の正方向から反時計回りに測った開始点と終点の角度である。円は、角度の開始点と終点を 0 °から 360 °にすると描くことができる。

また、arc オペレータの代わりに、arcn オペレータを使うと図 12 が描ける。この図は、曲率中心点を (288,288) にとり、半径 144 の円弧である。arc オペレータは、角度の開始点と終点を反時計回りに結ぶのに対して、arcn オペレータは時計回りに結ぶ。

プログラム 9

```
1: %!
```

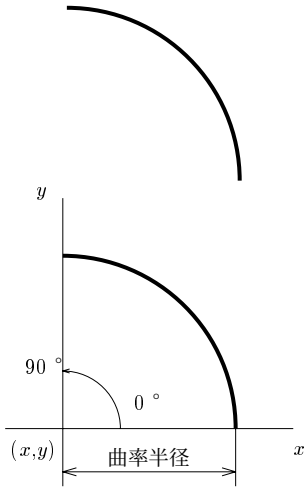

図 11: 円弧.

```
2: newpath
3: 288 288 144 0 90 arcn
4: 15 setlinewidth
5: stroke
6: showpage
```

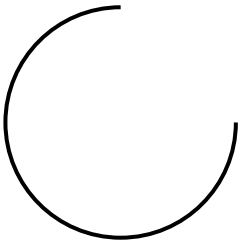

図 12: 円弧（逆回転）.

4　ポストスクリプトの応用例

ここまでは、ポストスクリプトの基本的な使い方を説明してきた。ここまでの知識だけでも頭の中で考えた図を描くことができる。しかし、少し複雑な図を描くことになるとプログラムが長くなって、読みにくくなったり、修正が困難になったりする。ここで紹介する例は、これらの問題を解決することを目的としている。

4.1　手続き

例として、上で取り扱った一辺 5 インチの正方形を描くプログラムの書き直しをしてみる。プログラム 5 の例では、(72,144) の点から時計回りに直線を引いていき、最後に closepath オペレータで正方形を閉じている。ここでは、つぎのように書き直す。

プログラム 10

```
1: %!
2: newpath
3: 72 144 moveto
4:  0 360 rlineto
5:  360 0 rlineto
6:  0 -360 rlineto
7: closepath
8: stroke
9: showpage
```

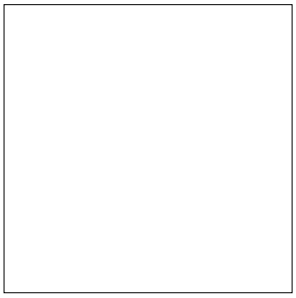

図 13: 正方形.

このようにすると図 13 を得ることができる。ここで新しくでてきた rline オペレータについて説明すると、これは相対的な移動量を与えるオペレータである。この例をとってみると (72,144) から y 方向に 360 進み、その点から x 方向に 360 進み、さらにその点から y 方向に-360 進んで最後に closepath オペレータで正方形を閉じている。

プログラム 10 を ”手続き ”を使用して図を描く例を示す。この ”手続き ”を使っていままでと同じ正方形を描くプログラムは次のようになる。

プログラム 11

```
1: %!
2: /box
3: {0 360 rlineto
4:  360 0 rlineto
5:  0 -360 rlineto
6: closepath } def
```

```
 7: newpath
 8: 72 144 moveto box
 9: stroke
10: showpage
```

ここでは手続きの名前を box とした。このプログラムは、まず手続きを定義し、8 行目に movoto オペレータを呼び出した後、手続き box を呼び出している。このプログラム 12 を少し変えて 3 つの正方形を描いてみると次のようになる。

プログラム 12

```
 1: %!
 2: /box
 3: {0 360 rlineto
 4:  360 0 rlineto
 5:  0 -360 rlineto
 6: closepath } def
 7: newpath
 8: 72 144 moveto box
 9: 0.0 setgray
10: fill
11: newpath
12: 108 180 moveto box
13: 0.5 setgray
14: fill
15: newpath
16: 144 216 moveto box
17: 0.9 setgray
18: fill
19: showpage
```

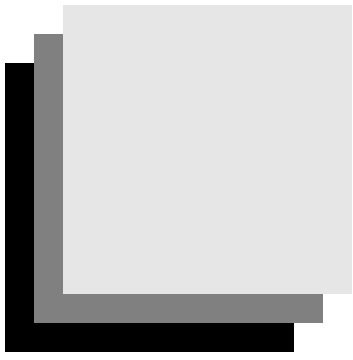

図 14: 3 つの正方形.

プログラム 12 の出力結果を図 14 に示す。まず初めに先程使用した手続き box を定義している。次に始点を (72,144) に moveto オペレータで持っていき手続き box を呼び出し、setgray オペレータで色を決定し、fill で真っ黒な正方形を描く。次に新たな始点を (108,180) に moveto オペレータで持っていった後、灰色の正方形を描く。さらに始点を (144,216) にもっていき同様に正方形を描く。このように手続きを使えば非常に簡単に複数の図を描くことができる。また、正方形の大きさを変えたいときは手続き box を変更するだけで 3 つともの正方形の大きさを変えることができる。このようにプログラムの修正も非常に簡単にできるというメリットもある。

4.2 原点の平行移動

これまでは原点を紙左下にとっていた。この原点の位置を移動するオペレータが translate オペレータである。下の例は、translate オペレータを使った例で、3 つの正方形を描くプログラムである。このプログラムは、正方形の左下が原点となるように描き、原点を移動することで、3 つの正方形を描いている。

プログラム 13

```
 1: %!
 2: /box
 3: {0 360 rlineto
 4:  360 0 rlineto
 5:  0 -360 rlineto
 6: closepath } def
 7: newpath
 8: 72 72 translate
 9: 0 0 moveto box
10: 0.0 setgray
11: fill
12: newpath
13: 72 72 translate
14: 0 0 moveto box
15: 0.5 setgray
16: fill
17: newpath
18: 72 72 translate
19: 0 0 moveto box
20: 0.9 setgray
21: fill
22: showpage
```

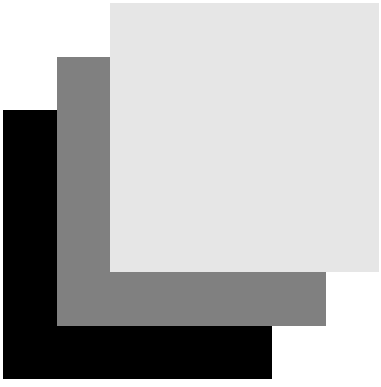

図 15: 3 つの正方形.

プログラム 13 の 8、13、18 行目に translate オペレータがある。8 行目の translate オペレータで原点を (72,72) 移動して、正方形を描き 13 行目の translate オペレータでさらに原点を (72,72) 移動して、18 行目の translate オペレータでまた原点を移動している。注意すべき点は、このように数回、原点を移動するとどこに原点があるのかわかりにくくなる。できあがった図を平行移動したりするときに便利である。

4.3 座標系の回転

座標を回転すること考える。座標の回転は、rotate オペレータで行う。プログラム 15 がその例である。3 インチの正方形を描く手続き名を square とし、各正方形に対して 10° づつ傾けている。

プログラム 14

```
 1: %!
 2: /square
 3: {0 216 rlineto
 4:  216 0 rlineto
 5:  0 -216 rlineto
 6: closepath } def
 7: newpath
 8:  72 72 translate
 9:  72 72 moveto square
10:  0.0 setgray
11: fill
12: newpath
13: 10 rotate
14:  144 144 moveto square
15:  0.3 setgray
16: fill
17: newpath
18: 10 rotate
19:  216 216 moveto square
20:  0.6 setgray
21: fill
22: newpath
23: 10 rotate
24:  288 288 moveto square
25:  0.9 setgray
26: fill
27: showpage
```

プログラム 14 の 13、18、23 行目に rotate オペレータがある。11 行目までに黒の正方形を描き、13 行目の rotate オペレータで座標を 10° 傾け、(144,144) 点から正方形を描いている。さらに、18、23 行目の rotate オペレータでさらに 10° づつ傾けた座標で正方形を描き、8 行目の translate オペレータで全体を (72,72) 動かしている。

4.4 拡大・縮小

ここでは、図形を相似的に変形し、その大きさを変える方法について説明する。相似変換は scale オペレータを用いて変えることができる。例えば、

2.0 1.5 scale

と記せば、x 方向に 2 倍し、y 方向に 1.5 倍することを意味する。その応用例を下に示す。この例は、正方形を 4 つ描き、正方形それぞれの座標単位を変えて描いている。また、この例は、座標系の回転の例の rotate オペレータのところを scale オペレータに書き換えたものである。

プログラム 15

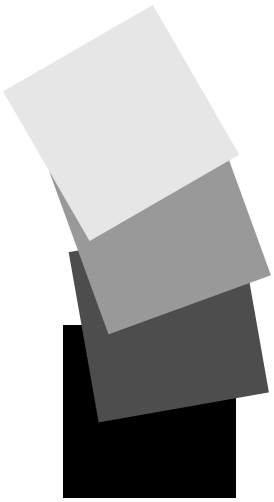

図 16: 4 つの正方形.

```
 1: %!
 2: /square
 3: {0 216 rlineto
 4:  216 0 rlineto
 5:  0 -216 rlineto
 6: closepath } def
 7: newpath
 8: 72 72 translate
 9:  72 72 moveto square
10:  0.0 setgray
11: fill
12: newpath
13: 1.5 1.0 scale
14:  144 144 moveto square
15:  0.3 setgray
16: fill
17: newpath
18: 1.0 1.5 scale
19:  216 216 moveto square
20:  0.6 setgray
21: fill
22: newpath
23: 1.2 1.2 scale
24:  288 288 moveto square
25:  0.9 setgray
26: fill
27: showpage
```

この例で scale オペレータが用いられているのは、13 行目、18 行目と 23 行目である。まず、7 行目から 11 行目で 3 インチの黒の正方形を描いている。2 つ目の四角形は 13 行目の scale オペレータで座標系の x 方向を 1.5 倍、y 方向を 1.0 倍して描いており、3 つ目の四角形のときに座標系の x 方向を 1.0 倍、y 方向を 1.5 倍することで 4.5(=3 $\times$ 1.5) インチの正方形を描いている。さらに座標系の x 方向、y 方向とも 1.2 倍し、5.4(=4.5 $\times$ 1.2) インチの正方形を描いている。

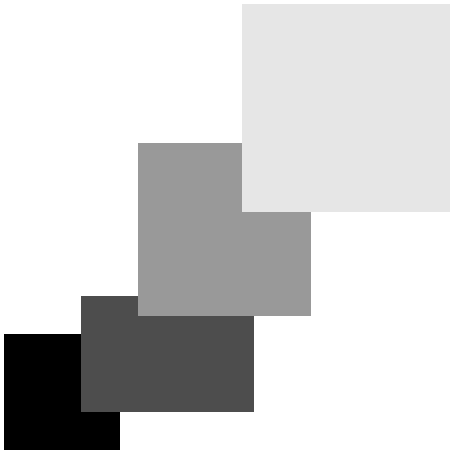

図 17: 4 つの四角形.

5　オペレータのまとめ

パス構築オペレータ

%! : ポストスクリプト言語での記述を始める。

newpath : カレントパスをクリアする。

x y **moveto** : カレントポイントを (x, y) に設定する。

x y **lineto** : (x, y) へ直線を引く。

closepath : 始点まで直線分を付加して、カレントパスを閉じる。

x y **rmoveto** : 相対的な moveto を行う。

x y **rlineto** : 相対的な lineto を行う。

x y r a b **arc** : 円弧の曲率中心点 (x, y)、曲率半径 r、角度 a °から b °までの円弧を反時計回りに描く。

x y r a b **arcn** : 円弧の曲率中心点 (x, y)、曲率半径 r、角度 a °から b °までの円弧を時計回りに描く。

グラフィック状態オペレータ

［c d］e **setdash** : 破線の設定。

ペイントオペレータ

stroke : カレントパスを描画する。

w **setlinewidth** : カレントの線幅 (w/72 インチ) に設定する。

fill : カレントパスをカレントカラーで塗りつぶす。

g **setgray** : 濃さを設定する。0 が白、1 が黒に対応する。

座標系オペレータ

x y **tranlate** : 原点を (x, y) に移動させる。

a **rotate** : 原点を中心に a °反時計回りに回転させる。

m n **scale** : 水平方向を m 倍、垂直方向を n 倍に拡大、縮小する。

出力オペレータ

showpage : カレントページを出力デバイスへ転送する。

6　フォートラン (Fortran) からの利用

工学や物理学の分野だけでなくいろいろな分野において現象を数値的に調べるときにはしばしばフォートラン (Fortran) や C 言語などの言語が用いられる。これらの言語を用いて解析した結果は主に数字として得られるが、数字の羅列では結果が意味するところを理解し難いときがある。解析結果を適切なグラフや表にすると、その結果の意味を容易に理解できることもある。現在では解析結果をグラフにするにはいろいろなツールが用意されているが、ポストスクリプト言語を使うと非常にきれいなグラフを自分の好みに合うように描画することが可能となる。しかし、ポストスクリプト言語を直接に使用するためにはポストスクリプト特有の逆ポーランド記述に従わねばならず、もう一つの言語を修得するというハードルを越える必要が生じる。このとき、フォートランや C 言語などからポストスクリプトの記述が可能であれば、新しい言語 (ポストスクリプト) を修得することなく、きれいなグラフを描くことができる。このセクション (節) ではフォートランからポストスクリプトを使用する方法について、次のセクションでは C 言語からポストスクリプトを使用する方法について説明する。

6.1　プログラムの作成

ポストスクリプトを用いて、フォートランで図形を描画するための基本的命令は psbasic.for に記述されている。この psbasic.for に記述されているサブルーチンを用いて、自分の好む図形を描く。プログラムの作成・編集には、WindowsNT に付属のメモ帳や秀丸など自分の好きなエディタを用いて編集を行う。プログラム編集後、プログラムファイルを保存する。ここでは、プログラムファイル名を main.for としておく。

6.2　プログラムのコンパイル・リンク操作

プログラム編集終了後、プログラム main.for をコンパイルし、psbasic.for とのリンク操作を行う。ここではコンピュータの D ドライブにディレクトリ \FORTRAN があり、このディレクトリに自分が作ったプログラム main.for とポストスクリプト基本命令 psbasic.for とがあるとする。

D : \FORTRAN ＞ f90 △ main.for △ psbasic.for

プログラムに間違いがなければ、実行ファイル main.exe が作成される。ここで、psbasic.for は、すでにシステムにインストールされているとする。もし、このディレクトリに psbasic.for がないときは、ホームページ

http://saffman.doshisha.ac.jp/

よりダウンロードする。

6.3　プログラムの実行

main.exe を実行することにより、計算が実行できる。グラフィック出力があるときには、temp1.ps という名の PostScript ファイルが作成される。

D : \FORTRAN ＞ main.exe

PostScript ファイル temp1.ps が作成される。

6.4　グラフの出力

Ghostscript あるいは Ghostview を起動し、上で作成されたファイル temp1.ps を読み込む。グラフの表示完了後、印刷をするときには、印刷用アイコンをクリックする。グラフィック出力用プログラム (psbasic.for) に対する基礎的なルーチンを以下に示す。

(1) グラフィック出力用プログラムに対するグラフィックルーチンの基本的仕様

```
PROGRAM △ MAIN
        :
call △ init
        :
        グラフィックルーチン（call 文形式で使用）
        :
call △ fin
        :
stop
end
```

(2) 画面構成

画面出力構成において、世界座標 (x, y) と機器座標 (X, Y) という考え方を導入する (後述のプログラム例の subroutine init の中の call xyworld で世界座標の設定を行い、call viewport で機器座標の設定を行う)。

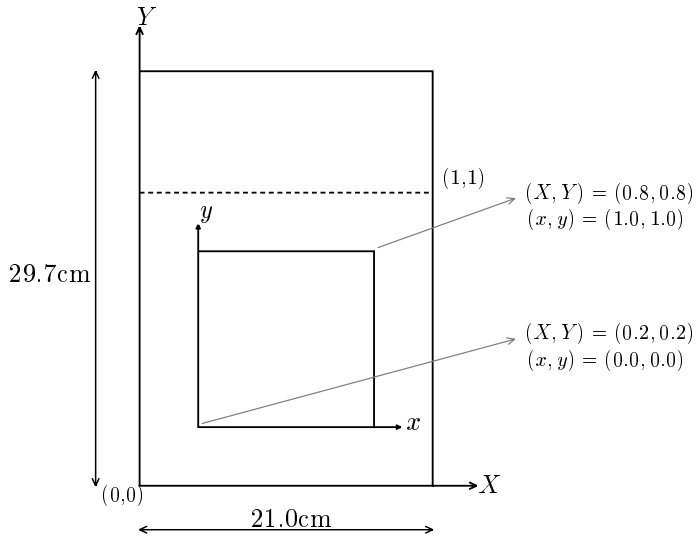

図 18: 世界座標 (x, y) と機器座標 (X, Y).

２次元グラフィックスにおいては普通は、世界座標 (x, y)→ 正規座標 (ξ, η)→ 機器座標 (X, Y) のようにマッピングを行う。ここで、マッピング: 世界座標 (x, y)→ 正規座標 (ξ, η) とは世界座標における左下の点が正規座標における左下の点に、世界座標における右上の点が正規座標における右上の点に移るように線形変換すなわち１次式で変換することである。ただし、ここでは簡単のために正規座標を省略して、直接にマッピング: 世界座標 (x, y)→ 機器座標 (X, Y) を行う。 これから描こうとする図は、世界座標 (x, y) で描く。すなわち、自分の好きなスケールや単位を選んで描くことができる。ここで紹介するプログラムにおいて、デフォルトでは世界座標は左下が $(0, 0)$ であり、右上が $(1, 0)$ となっている。これを変更するときはサブルーチン xyworld を用いる。図形を出力する機器は通常は紙またはコンピュータの画面であるが、ここでは、A4 の紙を想定する。A4 の紙は図 18 のように横が 21cm、縦が 29.7cm である。デフォルトでは A4 の紙の左下を $(0, 0)$、右上を $(1, 0)$ とする。ただし、デフォルトで viewport を用いて機器座標において、左下の座標を (0.2,0.2)、右上の座標を (0.8,0.8) とする正方形にマッピングしている。すなわち、すべてデフォルトの設定を用いると、世界座標において左下が $(0, 0)$ で右上が $(1, 0)$ となる図は機器座標において、左下の座標を $(0.2, 0.2)$ で右上の座標を $(0.8, 0.8)$ とする正方形いっぱいに描かれる。この設定を変更して、世界座標において左下が $(x1, y1)$ で右上が $(x2, y2)$ となる図を機器座標において、左下の座標が $(X1, Y1)$ で右上の座標が $(X2, Y2)$ である矩形領域に描くときには、call xyworld(x1, y1, x2, y2) のように世界座標変更のサブルーチンを用いて世界座標の設定を変更し、call viewport(X1, Y1, X2, Y2) のようにビューポートの変更を行う。

6.5 call 文形式で使用するグラフィックルーチン

付録 A のプログラム (psbasic.for) の各サブルーチンの説明を行う。サブルーチン名の最後に '1 'がついたものは、同じ名前のサブルーチンを簡略化したものである（例えば、line1 に対して line）。線の明るさ、太さや種類などを特にこだわらない場合はこちらを使うと容易に図を描くことができる。

init : ポストスクリプトの記述を始める. これ以降のポストスクリプトの記述はファイル 'temp1.ps' に書かれる. デフォルトの文字を 18 ポイントのタイムス (times-roman) とする. 長さの単位として cm が利用可能となる (デフォルトは cm となる). 用紙は A4(21.0[cm]×29.7[cm]) とする. 今後用紙の座標を機器座標と呼ぶことにし, 機器座標は左下を (0,0) 右上を (1, 1.4) とする. 物理座標 (世界座標) における (0,0)-(1,1) の正方形を機器座標の (0.2,0.2)- (0.8, 0.8) に対応づける. この対応付けを変更するときにはサブルーチン viewport と xyworld を用いる. デフォルトの線種として実線, 線の太さとして 1 (もっとも細い) を用いる. これらを変更するときは linety と linewidth を用いる.

呼び出し方: `call init`

viewport : 世界座標から切り取った図形をマッピングする機器座標の位置の設定.

呼び出し方: `call viewport(x1,y1,x2,y2)`

(x1,y1):機器座標での左下の座標,(x2,y2):機器座標での右上の座標.

xyworld : 世界座標において切り出す図形の位置の設定.

呼び出し方: `call xyworld(x1,y1,x2,y2)`

(x1,y1):用紙の左下の座標, (x2,y2):用紙が A4 のときは (x1,y1) を一つの角とする一 辺が 21.0[cm] の正方形の右上の座標.

fin : 基本的には、プログラムの最後に追加する.

呼び出し方: `call fin`

linety : 線のタイプを指定する.

呼び出し方: `call linety(it)`

it(整数): 線の太さ. it=1:実線,
it=2:点線, it=3:破線,
it=4:一点鎖線

linewidth : 線の太さを指定する.

呼び出し方: `call linewidth(w)`

w(実数): 線の太さ. w=1.0:細い,
w=2.0:標準, w=5.0:太い.

setgray : 色（白黒）を指定する.

呼び出し方: `call setgray(g)`

g(実数): 線の明るさ. g=1.0:白, g=0.0:黒

setrgb : 色（カラー）を指定する.

呼び出し方: `call setrgb(r,g,b)`

r,g,b(実数): 線の明るさ. r=1.0:赤, g=1.0:緑, b=1.0:青

newpath : カレントパスをクリアする.

呼び出し方: `call newpath`

closepath : 一本の折れ線の最初と最後をつなぐ.

呼び出し方: `call closepath`

stroke : カレントパスを描画する.

呼び出し方: `call stroke`

rotate : 原点を中心に回転させる.

呼び出し方: `call rotate(itheta)`

`itheta=0-360`(整数):反時計回りの回転角度.

scale : 縮尺率を指定する.

呼び出し方: `call scale(tx,ty)`

`tx=a`:x 方向を a 倍に, `ty=b`:y 方向を b 倍に拡大, 縮小する.

translate : 原点を移動する.

呼び出し方: `call translate(x,y)`

`x=a`:x 方向に a[cm] 移動, `y=b`:y 方向に b[cm] に移動.

clipon : 点 (`x1, y1`) を左下頂点, 点 (`x2, y2`) を右上頂点とする四角形の内側のパスのみ描画する.

呼び出し方: `call clipon(x1,y1,x2,y2)`

(`x1, y1`)(実数, 単位 [cm]): 長方形の左下頂点の座標.

(`x2, y2`)(実数, 単位 [cm]): 長方形の右上頂点の座標.

eoclipon : 点 (`x1, y1`) を左下頂点, 点 (`x2, y2`) を右上頂点とする四角形の外側のパスのみ描画する.

呼び出し方: `call clipon(x1,y1,x2,y2)`

(`x1, y1`)(実数, 単位 [cm]): 長方形の左下頂点の座標.

(`x2, y2`)(実数, 単位 [cm]): 長方形の右上頂点の座標.

clipoff : クリップの解除.

呼び出し方: `call clipoff`

plot : 現在の点から (`x,y`) までポインターを移動する.

呼び出し方: `call plot(x,y,ipen)`

(`x, y`)(実数, 単位 [cm]): 終点の座標.

`ipen`(整数): ペンの状態. `ipen=2`:線を描く (ペンを下ろす), `ipen=3`:線を描かない (ペンを上げる).

line : 2 点 (`x1, y1`) と (`x2,y2`) を直線で結ぶ.

呼び出し方: `call line(x1,y1,x2,y2,g,w,it)`

(`x1, y1`)(実数, 単位 [cm]): 始点の座標.

(`x2, y2`)(実数, 単位 [cm]): 終点の座標.

`g`(実数): 線の明るさ. `g=1.0`:白, `g=0.0`:黒

`w`(実数): 線の太さ. `w=1.0`:細い, `w=2.0`:標準, `w=5.0`:太い

`it`(整数): 線のタイプ. `it=1`:実線, `it=2`:点線, `it=3`:波線, `it=4`:一点鎖線.

line1 : 2 点 (`x1, y1`) と (`x2,y2`) を直線で結ぶ.

呼び出し方: `call line1(x1,y1,x2,y2)`

(`x1, y1`)(実数, 単位 [cm]): 始点の座標.

(`x2, y2`)(実数, 単位 [cm]): 終点の座標.

rect : 点 (`x1, y1`) を左下頂点, 点 (`x2, y2`) を右上頂点とする長方形を描く.

呼び出し方: `call rect(x1,y1,x2,y2,g,w,it)`

(`x1, y1`)(実数, 単位 [cm]): 長方形の左下頂点の座標.

(`x2, y2`)(実数, 単位 [cm]): 長方形の右上頂点の座標.

`g`(実数): 線の明るさ. `g=1.0`:白, `g=0.0`:黒. ただし, `g=0.0` 以外のときは内部が塗りつぶされるため外枠は描かれない.

`w`(実数): 線の太さ. `w=1.0`:細い, `w=2.0`:標準, `w=5.0`:太い

`it`(整数): 線のタイプ. `it=1`:実線, `it=2`:点線, `it=3`:波線, `it=4`:一点鎖線.

rect1 : 点 (x1, y1) を左下頂点, 点 (x2, y2) を右上頂点とする長方形を描く.

呼び出し方: `call rect1(x1,y1,x2,y2)`

(x1, y1)(実数, 単位 [cm]): 長方形の左下頂点の座標.

(x2, y2)(実数, 単位 [cm]): 長方形の右上頂点の座標.

rectfill1 : 点 (x1, y1) を左下頂点, 点 (x2, y2) を右上頂点とする長方形内を塗りつぶす.

呼び出し方: `call rectfill1(x1,y1,x2,y2)`

(x1, y1)(実数, 単位 [cm]): 長方形の左下頂点の座標.

(x2, y2)(実数, 単位 [cm]): 長方形の右上頂点の座標.

circ : 点 (x1, y1) を中心, 半径 r1 の円を描く.

呼び出し方: `call circ(x1,y1,r1,g,w,it)`

(x1, y1)(実数, 単位 [cm]): 円の中心の座標.

r1(実数, 単位 [cm]): 円の半径.

g(実数): 線の明るさ. g=1.0:白, g=0.0:黒. ただし, g=0.0 以外のときは内部が塗りつぶされるため外枠は描かれない.

w(実数): 線の太さ. w=1.0:細い, w=2.0:標準, w=5.0:太い

it(整数): 線のタイプ. it=1:実線, it=2:点線, it=3:波線, it=4:一点鎖線

circ1 : 点 (x1, y1) を中心, 半径 r1 の円を描く.

呼び出し方: `call circ1(x1,y1,r1)`

(x1, y1)(実数, 単位 [cm]): 円の中心の座標.

r1(実数, 単位 [cm]): 円の半径.

circfill1 : 点 (x1, y1) を中心, 半径 r1 の円内を塗りつぶす.

呼び出し方: `call circfill1(x1,y1,r1)`

(x1, y1)(実数, 単位 [cm]): 円の中心の座標.

r1(実数, 単位 [cm]): 円の半径.

ellipse : (x1, y1) を中心とし、x 方向の半径 rx、y 方向の半径 ry の楕円を描く.

呼び出し方: `call ellipse(x1,y1,rx,ry,g,w,it)`

(x1, y1): 楕円の中心,rx: x 方向の半径,ry: y 方向の半径 .

g(実数): 線の明るさ. g=1.0:白, g=0.0:黒.

w: 線の太さ.

it(整数): 線のタイプ. it=1:実線, it=2:点線, it=3:波線, it=4:一点鎖線.

ellipse1 : (x1, y1) を中心とし、x 方向の半径 rx、y 方向の半径 ry の楕円を描く.

呼び出し方: `call ellipse1(x1,y1,rx,ry)`

(x1, y1): 楕円の中心,rx: x 方向の半径,ry: y 方向の半径 .

ellipsefill1 : (x1, y1) を中心とし、x 方向の半径 rx、y 方向の半径 ry の楕円内を塗りつぶす.

呼び出し方: `call ellipse1(x1,y1,rx,ry)`

(x1, y1): 楕円の中心,rx: x 方向の半径,ry: y 方向の半径 .

arc : 点 (x1, y1) を中心, 半径 r1 の円弧を角 t1 から t2 まで描く.

呼び出し方: `call arc(x1,y1,r1,t1,t2,g,w,it)`

(x1, y1)(実数, 単位 [cm]): 円の中心の座標.

r1(実数, 単位 [cm]): 円の半径.

t1(実数, 単位 [°]): 円弧の始点角.

t2(実数, 単位 [°]): 円弧の終点角.

g(実数): 線の明るさ. g=1.0:白, g=0.0:黒. ただし, g=0.0 以外も内部が塗りつぶされない.

w(実数): 線の太さ. w=1.0:細い, w=2.0:標準, w=5.0:太い
it(整数): 線のタイプ. it=1:実線, it=2:点線, it=3:波線, it=4:一点鎖線.

arc1 : 点 (x1, y1) を中心, 半径 r1 の円弧を角 t1 から t2 まで描く.
呼び出し方: call circ(x1,y1,r1,t1,t2)
(x1, y1)(実数, 単位 [cm]): 円の中心の座標.
r1(実数, 単位 [cm]): 円の半径.
t1(実数, 単位 [°]): 円弧の始点角.
t2(実数, 単位 [°]): 円弧の終点角.

curv : 始点 (x1, y1) から終点 (x4, y4) までベジエ (Bézier) 曲線を描く. その曲線は始点 (x1, y1) において, 始点と点 (x2, y2) を結ぶ線分に接し, 終点において, 点 (x3, y3) と終点を結ぶ線分に接する.
呼び出し方: call curv(x1,y1,x2,y2,x3,y3,x4,y4,g,w,it)
(x1, y1)(実数, 単位 [cm]): 始点の座標.
(x2, y2)(実数, 単位 [cm]): 始点におけるコントロール点の座標.
(x3, y3)(実数, 単位 [cm]): 終点におけるコントロール点の座標.
(x4, y4)(実数, 単位 [cm]): 終点の座標.
g(実数): 線の明るさ. g=1.0:白, g=0.0:黒. ただし, g=0.0 以外も内部が塗りつぶされない.
w(実数): 線の太さ. w=1.0:細い, w=2.0:標準, w=5.0:太い
it(整数): 線のタイプ. it=1:実線, it=2:点線, it=3:波線, it=4:一点鎖線.

curv1 : 始点 (x1, y1) から終点 (x4, y4) までベジエ (Bézier) 曲線を描く. その曲線は始点 (x1, y1) において, 始点と点 (x2, y2) を結ぶ線分に接し, 終点において, 点 (x3, y3) と終点を結ぶ線分に接する.
呼び出し方: call curv1(x1,y1,x2,y2,x3,y3,x4,y4)
(x1, y1)(実数, 単位 [cm]): 始点の座標.
(x2, y2)(実数, 単位 [cm]): 始点におけるコントロール点の座標.
(x3, y3)(実数, 単位 [cm]): 終点におけるコントロール点の座標.
(x4, y4)(実数, 単位 [cm]): 終点の座標.

triangl : 点 (x1, y1) を左下頂点とする一辺 r1 の正三角形を描く.
呼び出し方: call triangl(x1,y1,r1,a1,g,w,it)
(x1, y1)(実数, 単位 [cm]): 正三角形の左下頂点の座標.
r1(実数, 単位 [cm]): 正三角形の一辺の長さ.
a1(実数, 単位 [°]): x 軸からの回転角度.
g(実数): 線の明るさ. g=1.0:白, g=0.0:黒, ただし, g=0.0 以外のときは内部が塗りつぶされるため外枠は描かれない.
w: 線の太さ.
it(整数): 線のタイプ. it=1:実線, it=2:点線, it=3:波線, it=4:一点鎖線.

triangl1 : 点 (x1, y1) を左下頂点とする一辺 r1 の正三角形を描く.
呼び出し方: call triangl(x1,y1,r1,a1)
(x1, y1)(実数, 単位 [cm]): 正三角形の左下頂点の座標.
r1(実数, 単位 [cm]): 正三角形の一辺の長さ.
a1(実数, 単位 [°]): x 軸からの回転角度.

trianglfill1 : 点 (x1, y1) を左下頂点とする一辺 r1 の正三角形内を塗りつぶす.
呼び出し方: call trianglfill1(x1,y1,r1,a1)
(x1, y1)(実数, 単位 [cm]): 正三角形の左下頂点の座標.
r1(実数, 単位 [cm]): 正三角形の一辺の長さ.
a1(実数, 単位 [°]): x 軸からの回転角度.

spline : (x1, y1) から (x_n, y_n) のデータをスプライン曲線で結ぶ.

呼び出し方: `call spline(x1,y1,x2,y2,x3,y3,x4,y4,ipart,g,w)`

ipart は次のようにする.

(x1, y1)-(x4, y4):ipart=-1,

(x2, y2)-(x5, y5):ipart=-1,

(x2, y2)-(x5, y5):ipart= 0,

:

(x_{n-4}, y_{n-4})-(x_{n-1}, y_{n-1}):ipart= 0,

(x_{n-3}, y_{n-3})-(x_n, y_n):ipart= 0,

(x_{n-3}, y_{n-3})-(x_n, y_n):ipart= 1,

g(実数): 線の明るさ. g=1.0:白, g=0.0:黒, ただし, g=0.0 以外のときは内部が塗りつぶされるため外枠は描かれない.

w: 線の太さ.

spline1 : (x1, y1) から (x_n, y_n) のデータをスプライン曲線で結ぶ.

呼び出し方: `call spline1(x1,y1,x2,y2,x3,y3,x4,y4,ipart)`

ipart は次のようにする.

(x1, y1)-(x4, y4):ipart=-1,

(x2, y2)-(x5, y5):ipart=-1,

(x2, y2)-(x5, y5):ipart= 0,

:

(x_{n-4}, y_{n-4})-(x_{n-1}, y_{n-1}):ipart= 0,

(x_{n-3}, y_{n-3})-(x_n, y_n):ipart= 0,

(x_{n-3}, y_{n-3})-(x_n, y_n):ipart= 1,

parabola : (x1,y1),(x2,y2),(x3,y3) を通る放物線を描く.

呼び出し方: `call parabola(x1,y1,x2,y2,x3,y3,g,w)`

(x1, y1),(x2, y2),(x3, y3):通過点の設定. g(実数): 線の明るさ. g=1.0:白, g=0.0:黒.

w: 線の太さ.

parabola1 : (x1,y1),(x2,y2),(x3,y3) を通る放物線を描く.

呼び出し方: `call parabola1(x1,y1,x2,y2,x3,y3)`

(x1, y1),(x2, y2),(x3, y3):通過点の設定.

arrow : (x1,y1) から (x2,y2) に線を引き,(x2,y2) 側が矢印となる.

呼び出し方: `call arrow(x1,y1,x2,y2,d,g,w)`

(x1, y1)(実数, 単位 [cm]): 始点の座標.

(x2, y2)(実数, 単位 [cm]): 終点の座標.

d(実数): 矢印の大きさ. g(実数): 線の明るさ. g=1.0:白, g=0.0:黒.

w: 線の太さ.

arrow1 : (x1,y1) から (x2,y2) に線を引き,(x2,y2) 側が矢印となる.

呼び出し方: `call arrow1(x1,y1,x2,y2,d)`

(x1, y1)(実数, 単位 [cm]): 始点の座標.

(x2, y2)(実数, 単位 [cm]): 終点の座標.

d(実数): 矢印の大きさ.

resist : (x1,y1) から (x2,y2) に電気回路の抵抗を描く.

呼び出し方: `call resist(x1,y1,x2,y2,d,w)`

(x1, y1)(実数, 単位 [cm]): 始点の座標.

(x2, y2)(実数, 単位 [cm]): 終点の座標.
d(実数): 抵抗記号の振幅の大きさ. w: 線の太さ.

battery : (x1,y1) から (x2,y2) に電気回路の電池を+極,-極の順に描く.
呼び出し方: `call battery(x1,y1,x2,y2,d,w)`
(x1, y1)(実数, 単位 [cm]): 始点の座標.
(x2, y2)(実数, 単位 [cm]): 終点の座標.
d(実数): 電池の大きさ. w: 線の太さ.

coil : (x1,y1) から (x2,y2) に電気回路のコイルを描く.
呼び出し方: `call coil(x1,y1,x2,y2,d,n,w)`
(x1, y1)(実数, 単位 [cm]): 始点の座標.
(x2, y2)(実数, 単位 [cm]): 終点の座標.
d(実数): コイルの大きさ. n(整数): コイルの巻き数.
w: 線の太さ.

text : 文字のタイプと大きさを指定する.
呼び出し方: `call text(x1,y1,n,'string')`
(x1, y1)(実数, 単位 [cm]): 最初の 1 文字の左下の座標.
n(整数): 文字数.
string(80 文字以内の文字): 出力する文字列.

textx : 文字のタイプと大きさを指定する. 特に x 軸のメモリを入れる際に有効.
呼び出し方: `call textx(x1,y1,n,'string')`
(x1, y1)(実数, 単位 [cm]): 文字列の真中が (x1, y1) の少し下側にくる.
n(整数): 文字数.
string(13 文字以内の文字): 出力する文字列.

texty : 文字のタイプと大きさを指定する. 特に y 軸のメモリを入れる際に有効.
呼び出し方: `call texty(x1,y1,n,'string')`
(x1, y1)(実数, 単位 [cm]): . 文字列の右端が (x1, y1) の少し左側にくる.
n(整数): 文字数.
string(13 文字以内の文字): 出力する文字列.

setchar : 文字のタイプと大きさを指定する.
呼び出し方: `call setchar(itype,ipoint)`
itype(整数): 文字のタイプ.
itype=1: Times-Roman, itype=2: Times-Bold, itype=3: Times-Italic, itype=4: Times-BoldItalic,
itype=5: Helvetica, itype=6: Helvetica-Bold, itype=7: Helvetica-Oblique, itype=8: Helvetica-BoldOblique,
itype=9: Courier, itype=10: Courier-Bold, itype=11: Courier-Oblique, itype=12: Courier-BoldOblique,
itype=13: Symbol
ipoint(実数, 単位 [ポイント,pt]): 文字の大きさ (1pt $\simeq$ 0.3515mm).

6.6 Fortran からの利用例

ここでは、§5.5 に紹介したグラフィックルーチンを用いて、各々のプログラムとその出力結果を紹介する。(各グラフィックルーチンの詳しい説明については §5.5 を参照。)

サンプル○-2 (samp ○ s.for) は、サブルーチン名の最後に '1 'がついたサブルーチン (例えば、line に対して line1) を用いて描いた例である。サンプル○ (samp ○.for) のプログラムと全く同じ図を描くが、各サブルーチンでの設定を簡略化したため、線の明るさや太さなどはほかのサブルーチンを使って設定を行っている。そのため、シンプルなプログラムとなって見やすいが、サンプル○ (samp ○.for) よりもプログラムが長くなる。

直線

直線を引くプログラムはつぎのようになる。このプログラム例では 3 本の直線を引いている。出力結果を図 19 に示す。

サンプル 1(samp1.for)

```
 1:C    Plotting program for general purpose using postscript
 2:      call init
 3:      call viewport(0.2,0.2,0.8,0.8)
 4:      call xyworld(0.0,0.0,1.0,1.0)
 5:      call linety(1)
 6:      call plot(0.1, 0.1, 3)
 7:      call plot(0.8, 0.1, 2)
 8:      call stroke
 9:      call linety(2)
10:      call plot(0.1, 0.2, 3)
11:      call plot(0.8, 0.2, 2)
12:      call stroke
13:      call linety(3)
14:      call plot(0.1, 0.3, 3)
15:      call plot(0.8, 0.8, 2)
16:      call stroke
17:      call fin
18:      stop
19:      end
```

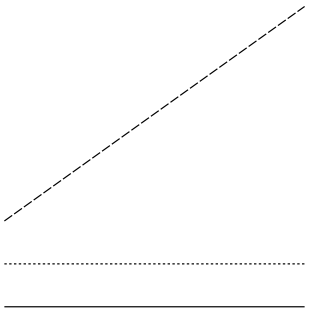

図 19: いろいろな直線.

まず、2 – 4 行目でポストスクリプトで図を書く準備をしている。2 行目でポストスクリプトの記述を始めることを宣言し、3 と 4 行目で機器座標と世界座標の設定を行っている。5 – 8 行目で図 19 の一番下の実線を引き、9 – 12 行目で図 19 の真中の点線を引く。そして、13 – 16 行目で図 19 の一番上の破線を引くプログラムとなっている。

円と四角形

円と四角形を描くプログラムとその出力結果は次のようになる。5 行目で線の太さを設定し、6 行目と 7 行目でそれぞれ円と四角形を描いている (図 20)。

サンプル 2(samp2.for)

```
 1:C     Plotting program for general purpose using postscript
 2:      call init
 3:      call viewport(0.2,0.2,0.8,0.8)
```

```
 4:       call xyworld(0.0,0.0,1.0,1.0)
 5:       call linewidth(2.0)
 6:       call circ(0.3, 0.3, 0.2, 0.0, 2.0, 1)
 7:       call rect(0.6, 0.1, 0.8, 0.5, 0.0, 1.0, 1)
 8:       call fin
 9:       stop
10:       end
```

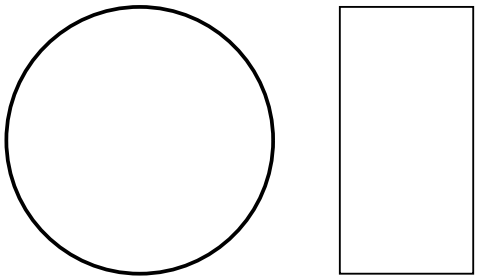

図 20: 円と四角形.

サンプル 2-2(samp2s.for)

```
 1:C      Plotting program for general purpose using postscript
 2:       call init
 3:       call viewport(0.2,0.2,0.8,0.8)
 4:       call xyworld(0.0,0.0,1.0,1.0)
 5:       call linewidth(2.0)
 6:       call circ1(0.3, 0.3, 0.2)
 7:       call linewidth(1.0)
 8:       call rect1(0.6, 0.1, 0.8, 0.5)
 9:       call fin
10:       stop
11:       end
```

文字と三角形

ポストスクリプトで文字を書くときは、下に示すプログラム例の 3 と 4 行目のサブルーチンを用いればよい。この例では文字のフォントには Times-Roman を用い、大きさは 10pt で書くように設定した。5 – 7 行目のサブルーチン triangl を用いて、同志社大学のマークを描いた (図 21)。

サンプル 3(samp3.for)

```
 1:        program main
 2:        call init
 3:        call setchar(1, 10)
 4:        call text(0.1,0.1,19,'Doshisha University')
 5:        call triangl(0.0,0.0,0.15,60.0,0.8,1,1)
 6:        call triangl(0.0,0.0,0.15,180.0,0.8,1,1)
 7:        call triangl(0.0,0.0,0.15,300.0,0.8,1,1)
 8:        call fin
 9:       stop
10:       end
```

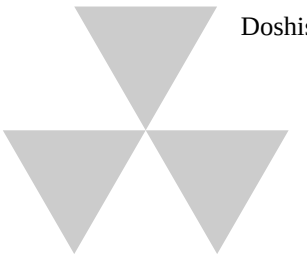

図 21: 文字と三角形.

サンプル 3-2(samp3s.for)

```
 1:       program main
 2:       call init
 3:       call setchar(1, 10)
 4:       call text(0.1,0.1,19,'Doshisha University')
 5:       call setgray(0.8)
 6:       call trianglfill1(0.0,0.0,0.15,60.0)
 7:       call trianglfill1(0.0,0.0,0.15,180.0)
 8:       call trianglfill1(0.0,0.0,0.15,300.0)
 9:       call fin
10:      stop
11:      end
```

軸と放物線

このプログラム例は、x 軸、y 軸と放物線を描くものである。5 行目で x 軸、6 行目 y 軸を描き、7 行目で放物線を描いている (図 22)。

サンプル 4(samp4.for)

```
 1:       program main
 2:       call init
 3:       call viewport(0.1, 0.1, 0.5, 0.5)
 4:       call xyworld(-1.0, -0.2, 1.0, 1.8)
 5:       call arrow(-1.0, 0.0, 1.0, 0.0, 0.05, 0.0, 1.5)
 6:       call arrow(0.0, -0.2, 0.0, 1.8, 0.05, 0.0, 1.5)
 7:       call parabola(-0.8, 1.5, 0.0, 0.0, 0.8, 1.5 , 0.0, 2.0)
 8:       call setchar(1, 20)
 9:       call text(0.95,-0.2, 1,'x')
10:       call text(-0.2, 1.75, 1,'y')
11:       call fin
12:       stop
13:       end
```

サンプル 4-2(samp4s.for)

```
 1:       program main
 2:       call init
 3:       call viewport(0.1, 0.1, 0.5, 0.5)
```

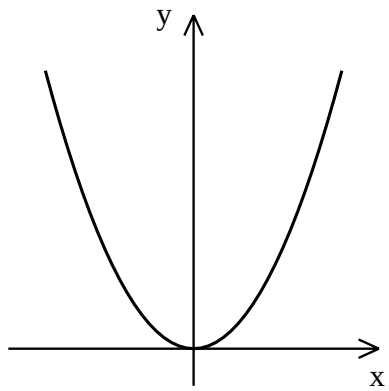

図 22: 軸と放物線.

```
 4:       call xyworld(-1.0, -0.2, 1.0, 1.8)
 5:       call arrow(-1.0, 0.0, 1.0, 0.0, 0.05, 0.0, 1.5)
 6:       call arrow(0.0, -0.2, 0.0, 1.8, 0.05, 0.0, 1.5)
 7:       call linewidth(2.0)
 8:       call parabola1(-0.8, 1.5, 0.0, 0.0, 0.8, 1.5)
 9:       call setchar(1, 20)
10:       call text(0.95,-0.2, 1,'x')
11:       call text(-0.2, 1.75, 1,'y')
12:       call fin
13:       stop
14:       end
```

方向場

下のプログラムでは、方向場を描くことができる。この例では、

$$\frac{dx}{dt} = 1, \quad \frac{dy}{dt} = 3xy$$

の方向場を描いている (図 23)。

サンプル 5(samp5.for)

```
1:C     Plotting program for general purpose using postscript
2:      call init
3:      call viewport(0.2,0.2,0.8,0.8)
4:      call xyworld(-0.2,-0.2,1.2,1.2)
5:      call line(0.5, 0.0, 0.5, 0.02, 0.0, 2.0, 1)
6:      call line(1.0, 0.0, 1.0, 0.02, 0.0, 2.0, 1)
7:      call line(0.0, 0.5, 0.02, 0.5, 0.0, 2.0, 1)
8:      call line(0.0, 1.0, 0.02, 1.0, 0.0, 2.0, 1)
9:      call arrow(-0.1, 0.0, 1.2, 0.0, 0.02, 0.0, 2.0)
10:     call arrow(0.0, -0.1, 0.0, 1.2, 0.02, 0.0, 2.0)
11:     call textx(1.2,-0.02,1,'x')
12:     call textx(0.5,-0.02,3,'0.5')
13:     call textx(1.0,-0.02,3,'1.0')
14:     call texty(-0.02,1.2,1,'y')
15:     call texty(0.0,0.5,3,'0.5')
```

```
16:       call texty(0.0,1.0,3,'1.0')
17:       call textx(-0.06,-0.02,1,'0')
18:       call plotd1
19:       call fin
20:       stop
21:       end
22:c
23:       subroutine plotd1
24:       common /ndata/n,nx,ny,ns
25:       common /xydata/x(100),y(100)
26:       write(1,*) ' 1.0 setlinewidth'
27:       write(1,*) ' 0.6 setgray'
28:       call linety(1)
29:       m=10
30:       do 50 ix=0,m
31:       do 50 iy=0,m
32:        x0=0.0+1.0*float(ix)/float(m)
33:        y0=0.0+1.0*float(iy)/float(m)
34:        dt=0.05
35:        dx=1.0
36:        dy=3.0*x0*y0
37:        ds=sqrt(dx**2+dy**2)
38:        if(ds.ne.0.0) then
39:        x1=x0+dx/ds*0.05
40:        y1=y0+dy/ds*0.05
41:        call arrow(x0,y0,x1,y1,0.01,0.6,1.0)
42:        end if
43:   50 continue
44:       return
45:       end
```

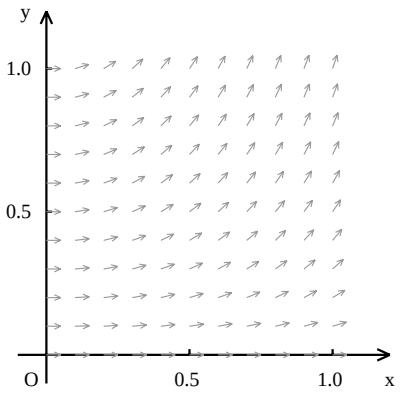

図 23: 方向場.

まず、メインプログラムについてみていく。5 – 8 行目でメモリ線を書いている。9、10 行目では軸を描き、11 – 17 行目でメモリを入れている。18 行目で方向場を描くサブルーチンを呼び出している。サブルーチン plotd1 においては、35 行目で上の左式の右辺を代入し、36 行目では右式の右辺を代入している。

サンプル 5-2(samp5s.for)

```
 1:C     Plotting program for general purpose using postscript
 2:      call init
 3:      call viewport(0.2,0.2,0.8,0.8)
 4:      call xyworld(-0.2,-0.2,1.2,1.2)
 5:      call linewidth(2.0)
 6:      call line1(0.5, 0.0, 0.5, 0.02)
 7:      call line1(1.0, 0.0, 1.0, 0.02)
 8:      call line1(0.0, 0.5, 0.02, 0.5)
 9:      call line1(0.0, 1.0, 0.02, 1.0)
10:      call arrow1(-0.1, 0.0, 1.2, 0.0, 0.02)
11:      call arrow1(0.0, -0.1, 0.0, 1.2, 0.02)
12:      call textx(1.2,-0.02,1,'x')
13:      call textx(0.5,-0.02,3,'0.5')
14:      call textx(1.0,-0.02,3,'1.0')
15:      call texty(-0.02,1.2,1,'y')
16:      call texty(0.0,0.5,3,'0.5')
17:      call texty(0.0,1.0,3,'1.0')
18:      call textx(-0.06,-0.02,1,'0')
19:      call plotd1
20:      call fin
21:      stop
22:      end
23:c
24:      subroutine plotd1
25:      common /ndata/n,nx,ny,ns
26:      common /xydata/x(100),y(100)
27:      write(1,*) ' 1.0 setlinewidth'
28:      write(1,*) ' 0.6 setgray'
29:      call linety(1)
30:      call linewidth(1.0)
31:      m=10
32:      do 50 ix=0,m
33:      do 50 iy=0,m
34:       x0=0.0+1.0*float(ix)/float(m)
35:       y0=0.0+1.0*float(iy)/float(m)
36:       dt=0.05
37:       dx=1.0
38:       dy=3.0*x0*y0
39:       ds=sqrt(dx**2+dy**2)
40:       if(ds.ne.0.0) then
41:       x1=x0+dx/ds*0.05
42:       y1=y0+dy/ds*0.05
43:       call arrow1(x0,y0,x1,y1,0.01)
44:       end if
45:   50 continue
46:      write(1,*) ' stroke '
```

```
47:      write(1,*) ' newpath'
48:      write(1,*) ' 0.4 setgray'
49:      write(1,*) ' 1.2 setlinewidth'
50:      call linety(1)
51:      n=40
52:      m=1
53:      do 10 ix=0,0
54:      do 10 iy=0,0
55:       x(0)=0.0
56:       y(0)=0.1
57:       dt=0.0128
58:       call plot(x(0),y(0),3)
59:        do 20 it=1,2*n
60:        x(it)= x(it-1)+(1.0)*(dt)
61:        y(it)= y(it-1)+(3.0*x(it-1)*y(it-1))*(dt)
62:        call plot(x(it),y(it),2)
63:   20 continue
64:   10 continue
65:      return
66:      end
```

回路図

このプログラムを用いれば簡単に回路図を描くことができる。その一例を下に示す。

サンプル 6(samp6.for)

```
 1:C     Plotting program for general purpose using postscript
 2:      call init
 3:      call viewport(0.2,0.2,0.8,0.8)
 4:      call xyworld(0.0,0.0,1.0,1.0)
 5:      call linewidth(2.0)
 6:      call line(0.0, 0.0, 0.1, 0.0, 0.0, 2.0, 1)
 7:      call coil(0.1, 0.0, 0.3, 0.0, 0.03, 6, 2.0)
 8:      call line(0.3, 0.0, 0.4, 0.0, 0.0, 2.0, 1)
 9:      call line(0.4, 0.0, 0.4, 0.3, 0.0, 2.0, 1)
10:      call line(0.4, 0.3, 0.3, 0.3, 0.0, 2.0, 1)
11:      call resist(0.3, 0.3, 0.1, 0.3, 2.0, 2.0)
12:      call line(0.1, 0.3, 0.0, 0.3, 0.0, 2.0, 1)
13:      call line(0.0, 0.3, 0.0, 0.2, 0.0, 2.0, 1)
14:      call battery(0.0, 0.2, 0.0, 0.1, 0.1, 2.0)
15:      call line(0.0, 0.1, 0.0, 0.0, 0.0, 2.0, 1)
16:      call fin
17:      stop
18:      end
```

このプログラム例の出力結果を図 24 に示す。このプログラムは、7 行目でコイル、11 行目で抵抗、そして 14 行目で電池を描いている。サブルーチン line でそれぞれをつないで回路にしている。

サンプル 6-2(samp6s.for)

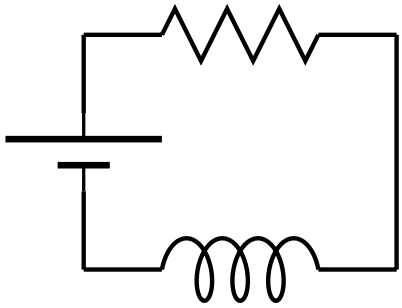

図 24: 回路図.

```
 1:C     Plotting program for general purpose using postscript
 2:      call init
 3:      call viewport(0.2,0.2,0.8,0.8)
 4:      call xyworld(0.0,0.0,1.0,1.0)
 5:      call linewidth(2.0)
 6:      call line1(0.0, 0.0, 0.1, 0.0)
 7:      call coil(0.1, 0.0, 0.3, 0.0, 0.03, 6, 2.0)
 8:      call line1(0.3, 0.0, 0.4, 0.0)
 9:      call line1(0.4, 0.0, 0.4, 0.3)
10:      call line1(0.4, 0.3, 0.3, 0.3)
11:      call resist(0.3, 0.3, 0.1, 0.3, 2.0, 2.0)
12:      call line1(0.1, 0.3, 0.0, 0.3)
13:      call line1(0.0, 0.3, 0.0, 0.2)
14:      call battery(0.0, 0.2, 0.0, 0.1, 0.1, 2.0)
15:      call line1(0.0, 0.1, 0.0, 0.0)
16:      call fin
17:      stop
18:      end
```

正弦関数

次のプログラムは $y = \sin x$ の関数を描いたものである。

サンプル 7(samp7.for)

```
 1:       program main
 2:       call init
 3:       call viewport(0.2, 0.2, 0.8, 0.8)
 4:       call xyworld(-1.2, -1.2, 1.2, 1.2)
 5:       call frame
 6:       call drawline
 7:       call fin
 8:      stop
 9:      end
10:c
11:      subroutine frame()
12:       call  arrow( -1.1, 0.0, 1.1, 0.0, 0.02, 0.0, 1.0)
13:       call  arrow( 0.0, -1.1, 0.0, 1.1, 0.02, 0.0, 1.0)
```

```
14:       call text(1.0,-0.1,1,'x')
15:       call text(-0.1,1.0,1,'y')
16:       call text(-0.1,-0.1,1,'0')
17:       n=10
18:       do 10 i=1, n
19:        x=0.1*i
20:        if(i/5*5.eq.i)  then
21:          dy=0.02
22:        else
23:          dy=0.01
24:        end if
25:        call line( -x, -dy, -x, dy, 0.0, 1.0, 1)
26:        call line( x, -dy, x, dy, 0.0, 1.0, 1)
27:   10  continue
28:       do 20 i=1, n
29:        y=0.1*i
30:        if(i/5*5.eq.i)  then
31:          dx=0.02
32:        else
33:          dx=0.01
34:        end if
35:        call line( -dx, -y, dx, -y, 0.0, 1.0, 1)
36:        call line( -dx, y, dx, y, 0.0, 1.0, 1)
37:   20  continue
38:      return
39:      end
40:      subroutine drawline
41:      parameter(nx= 100)
42:       x=-1.0
43:       y=0.0
44:       call linewidth(1.5)
45:       call plot( x, y, 3)
46:      do 10 i = -nx+1,nx
47:        t = float(i)/float(nx)
48:        x = t
49:        y = sin(t*3.14159)
50:        call  plot( x, y, 2)
51:   10 continue
52:      return
53:      end
```

サブルーチン frame で軸を描き、メモリを入れて、そしてメモリ線を引いている。次に、サブルーチン drawline で与えられた関数を描いている。49 行目に関数を与える。出力結果を図 25 に示す。

サンプル 7-2(samp7s.for)

```
 1:       program main
 2:       call init
```

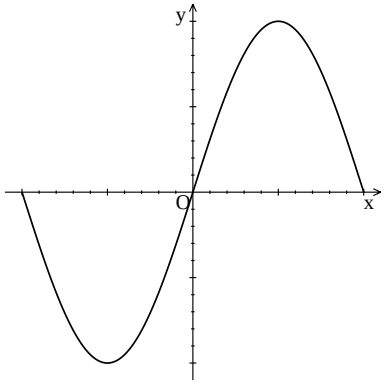

図 25: 正弦関数.

```
 3:       call viewport(0.2, 0.2, 0.8, 0.8)
 4:       call xyworld(-1.2, -1.2, 1.2, 1.2)
 5:       call frame
 6:       call drawline
 7:       call fin
 8:      stop
 9:      end
10:c
11:      subroutine frame()
12:       call  arrow1( -1.1, 0.0, 1.1, 0.0, 0.02)
13:       call  arrow1( 0.0, -1.1, 0.0, 1.1, 0.02)
14:       call text(1.0,-0.1,1,'x')
15:       call text(-0.1,1.0,1,'y')
16:       call text(-0.1,-0.1,1,'O')
17:       n=10
18:       do 10 i=1, n
19:        x=0.1*i
20:        if(i/5*5.eq.i)  then
21:          dy=0.02
22:        else
23:          dy=0.01
24:        end if
25:        call line1( -x, -dy, -x, dy)
26:        call line1(  x, -dy,  x, dy)
27:   10  continue
28:       do 20 i=1, n
29:        y=0.1*i
30:        if(i/5*5.eq.i)  then
31:          dx=0.02
32:        else
33:          dx=0.01
34:        end if
35:        call line1( -dx, -y, dx, -y)
36:        call line1( -dx,  y, dx,  y)
```

```
37:   20  continue
38:       return
39:       end
40:       subroutine drawline
41:       parameter(nx= 100)
42:        x=-1.0
43:        y=0.0
44:        call linewidth(1.5)
45:        call plot( x, y, 3)
46:       do 10 i = -nx+1,nx
47:         t = float(i)/float(nx)
48:         x = t
49:         y = sin(t*3.14159)
50:         call  plot( x, y, 2)
51:   10 continue
52:       return
53:       end
```

関数の作図 1

次のプログラムは $y = \exp -x \sin 4\pi x$ の関数を描いたものである。

サンプル 8(samp8.for)

```
 1: C     Plotting program for general purpose using postscript
 2:       call init
 3:       call viewport(0.2,0.2,0.8,0.6)
 4:       call xyworld(0.0,-1.0,1.0,1.0)
 5:       call frame
 6:       call plotd
 7:       call fin
 8:       stop
 9:       end
10:       function f(x)
11:        pi=atan(1.0)*4.0
12:        f=exp(-x)*sin(4*pi*x)
13:       return
14:       end
15:       subroutine plotd
16:       dimension x1(0:20),y1(0:20)
17:       n=20
18:       do 10 i=0,n
19:        x1(i)=float(i)/float(n)
20:        y1(i)=f(x1(i))
21:   10 continue
22:       call linewidth(1.5)
23:       call linety(1)
24:       call setgray(0.0)
25:       call plot(x1(0),y1(0),3)
26:       do 30 i=1,n-2
```

```
27:       call spline1(x1(i-1),y1(i-1),x1(i),y1(i),x1(i+1),y1(i+1),x1(i+2),
28:     &                                                    y1(i+2),-1)
29:   30 continue
30:      call spline1(x1(n-3),y1(n-3),x1(n-2),y1(n-2),x1(n-1),y1(n-1),
31:     &                                              x1(n),y1(n),0)
32:      call spline1(x1(n-3),y1(n-3),x1(n-2),y1(n-2),x1(n-1),y1(n-1),
33:     &                                              x1(n),y1(n),1)
34:      call stroke()
35:      continue
36:      return
37:      end
38:      subroutine frame
39:      call linewidth(1.0)
40:      call rect(0.0,-1.0,1.0,1.0,0.0,1.0,1)
41:      nx=2
42:      do 10 ix=1,nx*10-1
43:       x=float(ix)/float(nx*10)
44:       dy=1.0
45:       if(ix/10*10.eq.ix) then
46:         call setgray(0.5)
47:        else
48:         call setgray(0.8)
49:       end if
50:       call plot(x,-dy,3)
51:       call plot(x,dy,2)
52:       call stroke()
53:   10 continue
54:      ny=2
55:      do 20 iy=1,ny*10-1
56:       y=float(iy)/float(ny*10)*2.0-1.0
57:       dx=1.0
58:       if(iy/10*10.eq.iy) then
59:         call setgray(0.5)
60:        else
61:         call setgray(0.8)
62:       end if
63:       call plot(0.0,y,3)
64:       call plot(dx,y,2)
65:       call stroke()
66:   20 continue
67:      call setgray(0.0)
68:      call textx(0.75,0.0,1,'x')
69:      call textx(0.05,0.02,1,'0')
70:      call textx(0.55,0.02,3,'0.5')
71:      call textx(0.95,0.02,1,'1')
72:      call texty(0.0,0.5,1,'y')
73:      call texty(0.0,-1.0,2,'-1')
```

```
74:       call texty(0.0,0.0,1,'0')
75:       call texty(0.0,1.0,1,'1')
76:       return
77:       end
```

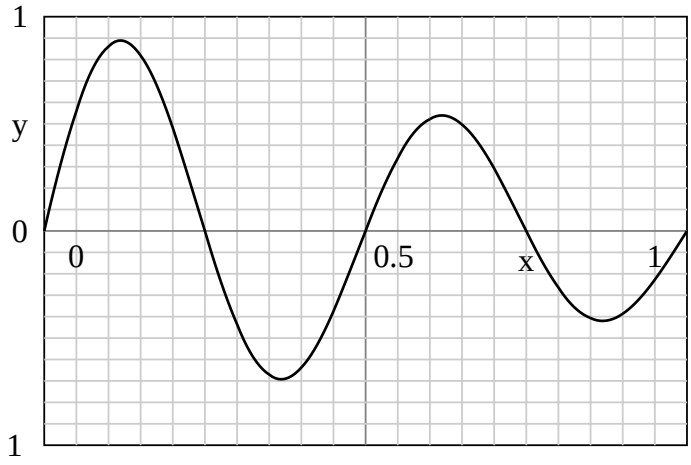

図 26: グラフ 1.

このプログラムでは、サブルーチン frame で軸を描き、メモリを入れて、そしてメモリ線を引いている。46 行目と 48 行目でメモリ線の色の設定を行っている。10 行目から 14 行目の Function で描きたい関数を与える。出力結果を図 26 に示す。

関数の作図 2

次のプログラムは関数 $y = x^3/(5.0 + x^2) + 10.0$ を両対数で出力する。

サンプル 9(samp9.for)

```
 1: C     Plotting program for general purpose using postscript
 2:       call init
 3:       call viewport(0.2,0.2,0.8,0.8)
 4:       call xyworld(0.0,0.0,4.0,4.0)
 5:       call frame
 6:       call plotd
 7:       call fin
 8:       stop
 9:       end
10:       function f(x)
11:        f=x**3/(5.0+x**2)+10.0
12:       return
13:       end
14:       subroutine plotd
15:       dimension x1(0:20),y1(0:20)
16:       n=20
17:       do 10 i=0,n
18:        x1(i)=1.0*(10.0)**(float(i)/float(n)*4.0)
19:        y1(i)=f(x1(i))
```

```
20:    10 continue
21:       do 20 i=0,n
22:        x1(i)=alog10(x1(i))
23:        y1(i)=alog10(y1(i))
24:    20 continue
25:       call linewidth(1.5)
26:       call linety(1)
27:       call setgray(0.0)
28:       call plot(x1(0),y1(0),3)
29:       do 30 i=1,n-2
30:        call spline1(x1(i-1),y1(i-1),x1(i),y1(i),x1(i+1),y1(i+1),x1(i+2),
31:      &                                                    y1(i+2),-1)
32:    30 continue
33:       call spline1(x1(i-3),y1(i-3),x1(i-1),y1(i-1),x1(i),y1(i),x1(i+1),
34:      &                                                     y1(i+1),0)
35:       call spline1(x1(i-3),y1(i-3),x1(i-1),y1(i-1),x1(i),y1(i),x1(i+1),
36:      &                                                     y1(i+1),1)
37:       call stroke()
38:       continue
39:       return
40:       end
41:       subroutine frame
42:       call linewidth(1.0)
43:       call rect(0.0,0.0,4.0,4.0,0.0,1.0,1)
44:       nx=4
45:       do 10 ix=0,nx-1
46:        call setgray(0.3)
47:        x=float(ix)
48:        dy=4.0
49:        if(ix.ne.0) then
50:         call plot(x,0.0,3)
51:         call plot(x,dy,2)
52:         call stroke()
53:        end if
54:       do 20 ixx=2,9
55:          call setgray(0.8)
56:          x=float(ix)+alog10(float(ixx))
57:          call plot(x,0.0,3)
58:          call plot(x,dy,2)
59:          call stroke()
60:    20  continue
61:    10 continue
62:       ny=4
63:       do 30 iy=0,ny-1
64:        call setgray(0.5)
65:        y=float(iy)
66:       dx=4.0
```

```
67:        if (iy.ne.0) then
68:         call plot(0.0,y,3)
69:         call plot(dx,y,2)
70:         call stroke()
71:       end if
72:       do 40 iyy=2,9
73:          call setgray(0.8)
74:          y=float(iy)+alog10(float(iyy))
75:          call plot(0.0,y,3)
76:          call plot(dx,y,2)
77:          call stroke()
78:    40  continue
79:    30 continue
80:       call setgray(0.0)
81:       call textx(3.5,0.0,1,'x')
82:       call textx(0.0,0.02,1,'1')
83:       call textx(1.0,0.02,2,'10')
84:       call textx(2.0,0.02,3,'100')
85:       call textx(3.0,0.02,4,'1000')
86:       call textx(4.0,0.02,5,'10000')
87:       call texty(0.0,3.5,1,'y')
88:       call texty(0.0,0.0,1,'1')
89:       call texty(0.0,1.0,2,'10')
90:       call texty(0.0,2.0,3,'100')
91:       call texty(0.0,3.0,4,'1000')
92:       call texty(0.0,4.0,5,'10000')
93:       return
94:       end
```

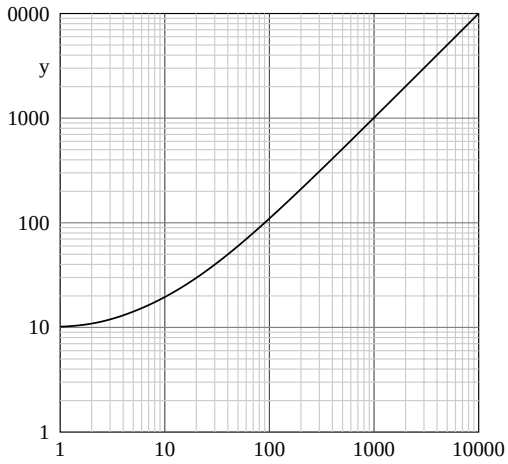

図 27: 両対数グラフ.

このプログラムでは、サブルーチン frame で軸を描き、メモリを入れて、そしてメモリ線を引いている。64 行目と 73 行目でメモリ線の色の設定を行っている。10 行目から 13 行目の Function で描きたい関数を与える。出力結果を図 27 に示す。

関数の作図 3

次のプログラムは関数 $y = \sin(2.0\pi x) \times \sin(\pi y)$ を等高線で出力する。

サンプル 10(samp10.for)

```
 1: c     Plotting routine of equi-streamline in colors
 2:       parameter (nx=200,ny=100)
 3:       common /par/dx,dy
 4:       call init
 5:       call viewport(0.2,0.2,0.8,0.5)
 6:       call xyworld(0.0,0.0,2.0,1.0)
 7:       call frame
 8:       call draw(1.0,-1.0,20)
 9:       call frame
10:      call fin
11:      stop
12:      end
13:c
14:      subroutine frame
15:      call linewidth(1.0)
16:      call rect(0.0,0.0,2.0,1.0,0.0,1.0,1)
17:      nx=4
18:      do 10 ix=0,2*nx-1
19:       call setgray(0.0)
20:       x=float(ix)/float(nx)*2.0
21:       dy=0.05
22:       call plot(x,0.0,3)
23:       call plot(x,dy,2)
24:       call stroke()
25:   10 continue
26:      ny=2
27:      do 20 iy=0,ny-1
28:       call setgray(0.0)
29:       y=float(iy)/float(ny)
30:       dx=0.05
31:       call plot(0.0,y,3)
32:       call plot(dx,y,2)
33:       call stroke()
34:   20 continue
35:      call textx(1.5,0.0,1,'x')
36:      call textx(0.0,0.02,1,'0')
37:      call textx(1.0,0.02,1,'1')
38:      call textx(2.0,0.02,1,'2')
39:      call texty(0.0,0.75,1,'y')
40:      call texty(0.0,0.0,1,'0')
41:      call texty(0.0,0.5,3,'0.5')
42:      call texty(0.0,1.0,1,'1')
43:      return
44:      end
```

```
45:       function f(x,y)
46:        pi=3.14159
47:        f=sin(2.0*pi*x)*sin(pi*y)
48:       return
49:       end
50:       subroutine draw(fmax,fmin,nline)
51:       parameter (nx=200,ny=100)
52:       dx=2.0/float(nx)
53:       dy=1.0/float(ny)
54:       do 10 iy=1,ny
55:        y=float(iy)/float(ny)
56:        do 10 ix=1,nx
57:         x=float(ix)/float(nx)
58:           color=(f(x,y)-fmin)/(fmax-fmin)
59:        call plot((ix-1)*dx,(iy-1)*dy,3)
60:        call plot(ix*dx,(iy-1)*dy,2)
61:        call plot(ix*dx,iy*dy,2)
62:        call plot((ix-1)*dx,iy*dy,2)
63:        write(1,*) ' closepath'
64:           r=color*0.8+0.2
65:           g=0.1
66:           b=(1.0-color)*0.8+0.2
67:           call setrgb(r,g,b)
68:          write(1,*) ' fill'
69:          write(1,*) ' stroke'
70:    10 continue
71:       return
72:       end
```

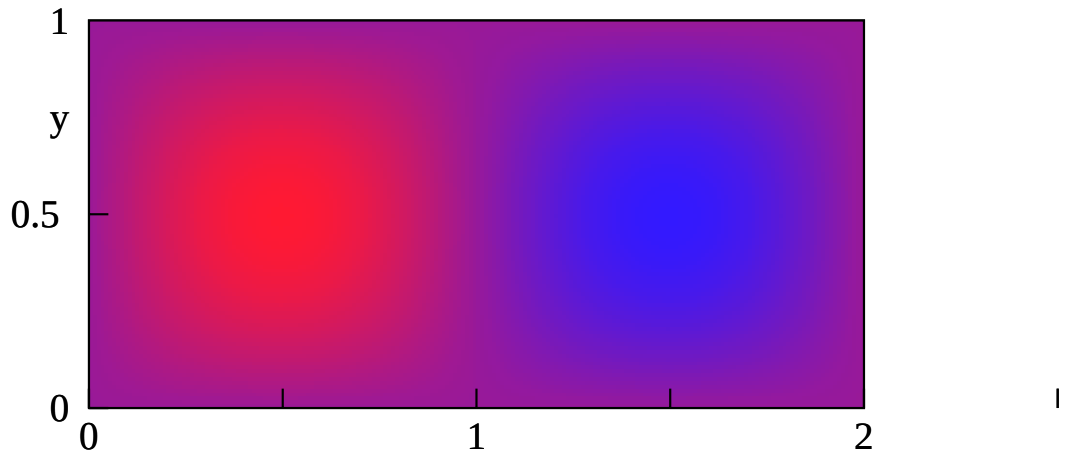

図 28: 等高線.

このプログラムでは、サブルーチン frame で軸を描き、メモリを入れて、そしてメモリ線を引いている。45 行目から 49 行目の Function で描きたい関数を与える。出力結果を図 28 に示す。

7　Cからの利用

前節ではフォートラン (Fortran) からポストスクリプト言語を呼び出すことにより図形やグラフを描く方法について説明を行った。C 言語をからポストスクリプト言語を呼び出す方法もフォートラン言語の場合とほぼ同様である。最近は手続き形の言語である C 言語よりも、オブジェクト形の言語である C++を使う方が一般ではあるが、ここでは初心者にも使いやすいようにあえて C 言語を用いる。

7.1　プログラムの作成

ポストスクリプトを用いて、C で図形を描画するための基本的命令は psbasic.cpp に記述されている。ファイル名の修飾子には cpp を用いてはいるが、この psbasic.cpp はすべて C 言語で書かれている。このファイルに記述されているサブルーチンを用いて、自分の好む図形を描く。プログラムの作成・編集には、WindowsNT に付属のメモ帳や秀丸など自分の好きなエディタを用いて編集を行う。プログラム編集後、プログラムファイルを保存する。ここでは、プログラムファイル名を main.cpp としておく。例についても C++の機能はほとんど使わないこととする。

7.2　プログラムのコンパイル・リンク操作

プログラム編集終了後、プログラム main.cpp をコンパイルし、psbasic.cpp とのリンク操作を行う。ここではコンピュータの D ドライブにディレクトリ \CC があり、このディレクトリに自分が作ったプログラム main.cpp とポストスクリプト基本命令 psbasic.cpp および ps.h とがあるとする。

D : \CC > cl △ main.cpp △ psbasic.cpp

プログラムに間違いがなければ、実行ファイル main.exe が作成される。ここで、psbasic.cpp および ps.h は、すでにシステムにインストールされているとする。もし、このディレクトリに psbasic.cpp および ps.h がないときは、ホームページ

http://saffman.doshisha.ac.jp/

よりダウンロードする。ps.h はポストスクリプトを呼び出す関数のヘッダーファイルである。

7.3　プログラムの実行

main.exe を実行することにより、計算が実行できる。グラフィック出力があるときには、temp1.ps という名の PostScript ファイルが作成される。

D : \CC > main.exe

PostScript ファイル temp1.ps が作成される。

7.4　グラフの出力

ghostscript か ghostview を起動し、上で作成されたファイル temp1.ps を読み込む。グラフの表示完了後、印刷をするときには、印刷用アイコンをクリックする。グラフィック出力用プログラム (psbasic.for) に対する基礎的なルーチンを以下に示す。

(1) グラフィック出力用プログラムに対するグラフィックルーチンの基本的仕様

```
void △ main()
{
        :
init();
        :
```

関数呼び出し形式で使用

```
        :
fin();
        :
}
```

(2) 画面構成

画面出力構成において、世界座標 (x, y) と機器座標 (X, Y) という考え方を導入する。(後述のプログラム例の subroutine init の中の call xyworld で世界座標の設定を行い、call viewport で機器座標の設定を行っている。)

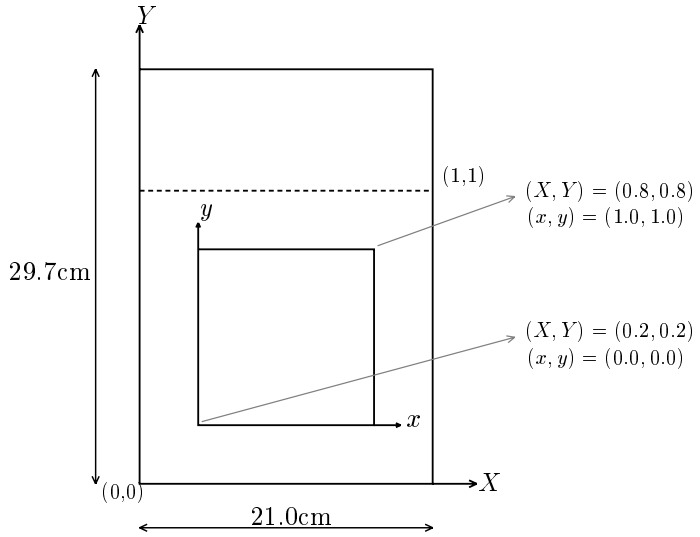

図 29: 世界座標 (x, y) と機器座標 (X, Y).

２次元グラフィックスにおいては普通は、世界座標 (x, y)→ 正規座標 (ξ, η)→ 機器座標 (X, Y) のようにマッピングを行う。ここで、マッピング: 世界座標 (x, y)→ 正規座標 (ξ, η) とは世界座標における左下の点が正規座標における左下の点に、世界座標における右上の点が正規座標における右上の点に移るように線形変換すなわち１次式で変換することである。ただし、ここでは簡単のために正規座標を省略して、直接にマッピング: 世界座標 (x, y)→ 機器座標 (X, Y) を行う。これから描こうとする図は、世界座標 (x, y) で描く。すなわち、自分の好きなスケールや単位を選んで描くことができる。ここで紹介するプログラムにおいて、デフォルトでは世界座標は左下が $(0, 0)$ であり、右上が $(1, 0)$ となっている。これを変更するときはサブルーチン xyworld を用いる。図形を出力する機器は通常は紙またはコンピュータの画面であるが、ここでは、A4 の紙を想定する。A4 の紙は図 18 のように横が 21cm、縦が 29.7cm である。デフォルトでは A4 の紙の左下を $(0, 0)$、右上を $(1, 0)$ とする。ただし、デフォルトで viewport を用いて機器座標において、左下の座標を (0.2,0.2)、右上の座標を (0.8,0.8) とする正方形にマッピングしている。すなわち、すべてデフォルトの設定を用いると、世界座標において左下が $(0, 0)$ で右上が $(1, 0)$ となる図は機器座標において、左下の座標を $(0.2, 0.2)$ で右上の座標を $(0.8, 0.8)$ とする正方形いっぱいに描かれる。この設定を変更して、世界座標において左下が $(x1, y1)$ で右上が $(x2, y2)$ となる図を機器座標において、左下の座標が $(X1, Y1)$ で右上の座標が $(X2, Y2)$ である矩形領域に描くときには、xyworld(x1, y1, x2, y2) のように世界座標変更のサブルーチンを用いて世界座標の設定を変更し、viewport(X1, Y1, X2, Y2) のようにビューポートの変更を行う。

7.5 C からの利用例

ここでは、§5.5 に紹介したグラフィックルーチンを用いて、各々のプログラムとその出力結果を紹介する。(各グラフィックルーチンの詳しい説明については §5.5 を参照)

サンプル○-2 (samp ○ s.cpp) は、サブルーチン名の最後に '1 'がついたサブルーチン (例えば、`line` に対して `line1`) を用いて描いた例である。サンプル○ (samp ○.cpp) のプログラムと全く同じ図を描くが、各サブルーチンでの設定を簡

略化したため、線の明るさや太さなどはほかのサブルーチンを使って設定を行っている。そのため、シンプルなプログラムとなって見やすいが、サンプル○ (samp ○.cpp) よりもプログラムが長くなる。

直線

直線を引くプログラムはつぎのようになる。このプログラム例では 3 本の直線を引いている。出力結果を図 30 に示す。

サンプル 1(samp1.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:      init();
 9:      viewport(0.2,0.2,0.8,0.8); xyworld(0.0,0.0,1.0,1.0);
10:      linety(1); plot(0.1, 0.1, 3); plot(0.8, 0.1, 2);  stroke();
11:      linety(2); plot(0.1, 0.2, 3); plot(0.8, 0.2, 2);  stroke();
12:      linety(3); plot(0.1, 0.3, 3); plot(0.8, 0.8, 2);  stroke();
13:      fin();
14: }
```

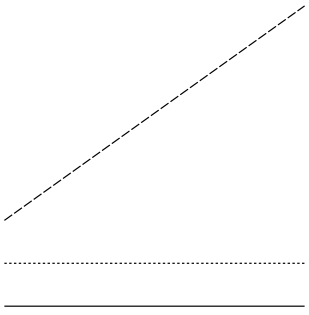

図 30: いろいろな直線.

まず、8、9 行目でポストスクリプトで図を書く準備をしている。8 行目でポストスクリプトの記述を始めることを宣言し、9 行目で機器座標と世界座標の設定を行っている。10 行目で図 19 の一番下の実線を引き、11 行目で図 19 の真中の点線を引く。そして、12 行目で図 19 の一番上の破線を引くプログラムとなっている。

円と四角形

円と四角形を描くプログラムとその出力結果は次のようになる。10 行目で線の種類と太さを設定し、11 行目と 12 行目でそれぞれ円と四角形を描いている (図 31)。

サンプル 2(samp2.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
```

```
 6: void main()
 7: {
 8:      init();
 9:      viewport(0.2,0.2,0.8,0.8); xyworld(0.0,0.0,1.0,1.0);
10:      linety(1); linewidth(2.0);
11:      circ(0.3, 0.3, 0.2, 0.0, 2.0, 1);
12:      rect(0.6, 0.1, 0.8, 0.5, 0.0, 1.0, 1);
13:      fin();
14: }
```

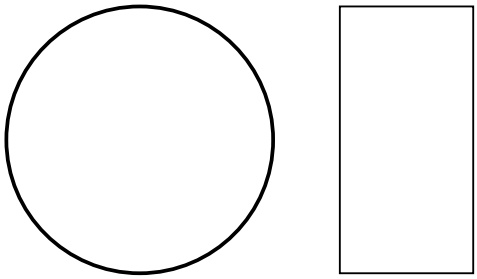

図 31: 円と四角形.

サンプル 2-2(samp2s.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:      init();
 9:      viewport(0.2,0.2,0.8,0.8); xyworld(0.0,0.0,1.0,1.0);
10:      linety(1); linewidth(2.0);
11:      circ1(0.3, 0.3, 0.2);
12:      linewidth(1.0);
13:      rect1(0.6, 0.1, 0.8, 0.5);
14:      fin();
15: }
```

文字と三角形

ポストスクリプトで文字を書くときは、下に示すプログラム例の 8 と 10 行目のサブルーチンを用いればよい。この例では文字のフォントには Times-Roman を用い、大きさは 10pt で書くように設定した。11 – 13 行目のサブルーチンで 3 つの三角形を用いて、同志社のマークを描いた (図 32)。

サンプル 3(samp3.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
```

```
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:      init();setchar(1,10);
 9:      viewport(0.2,0.2,0.8,0.8); xyworld(0.0,0.0,1.0,1.0);
10:      text(0.1,0.1,19,"Doshisha University");
11:      triangl(0.0,0.0,0.15,60.0,0.8,1.0,1);
12:      triangl(0.0,0.0,0.15,180.0,0.8,1.0,1);
13:      triangl(0.0,0.0,0.15,300.0,0.8,1.0,1);
14:      fin();
15: }
```

図 32: 文字と三角形.

サンプル 3-2(samp3s.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:      init();setchar(1,18);
 9:      viewport(0.2,0.2,0.8,0.8); xyworld(0.0,0.0,1.0,1.0);
10:      text(0.1,0.1,19,"Doshisha University");
11:      linewidth(1.0);setgray(0.8);
12:      trianglfill1(0.0,0.0,0.15,60.0);
13:      trianglfill1(0.0,0.0,0.15,180.0);
14:      trianglfill1(0.0,0.0,0.15,300.0);
15:      fin();
16: }
```

軸と放物線

このプログラム例は、x 軸、y 軸と放物線を描くものである。11 行目で x 軸、12 行目 y 軸を描き、13 行目で放物線を描いている (図 33)。

サンプル 4(samp4.cpp)

```
 1: #include <stdio.h>
```

```
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:      init();
 9:      viewport(0.1, 0.1, 0.5, 0.5);
10:      xyworld(-1.0, -0.2, 1.0, 1.8);
11:      arrow(-1.0, 0.0, 1.0, 0.0, 0.05, 0.0, 1.5);
12:      arrow(0.0, -0.2, 0.0, 1.8, 0.05, 0.0, 1.5);
13:      parabola(-0.8, 1.5, 0.0, 0.0, 0.8, 1.5 , 0.0, 2.0, 1);
14:      setchar(1, 20);
15:      text(0.95,-0.2, 1,"x"); text(-0.2, 1.75, 1,"y");
16:      fin();
17: }
```

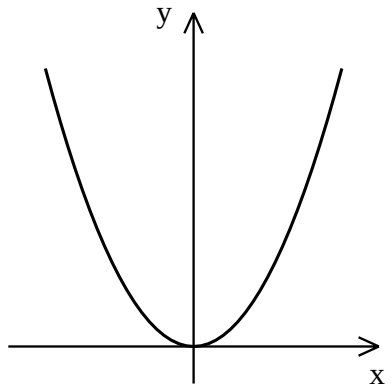

図 33: 軸と放物線.

サンプル 4-2(samp4s.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:      init();
 9:      viewport(0.1, 0.1, 0.5, 0.5);
10:      xyworld(-1.0, -0.2, 1.0, 1.8);
11:      linewidth(1.5);
12:      arrow1(-1.0, 0.0, 1.0, 0.0, 0.05);
13:      arrow1(0.0, -0.2, 0.0, 1.8, 0.05);
14:      parabola1(-0.8, 1.5, 0.0, 0.0, 0.8, 1.5);
15:      setchar(1, 20);
16:      text(0.95,-0.2, 1,"x"); text(-0.2, 1.75, 1,"y");
```

```
17:      fin();
18: }
```

方向場

下のプログラムでは、方向場を描くことができる。この例では、

$$\frac{dx}{dt} = 1, \quad \frac{dy}{dt} = 3xy$$

の方向場を描いている (図 34)。

サンプル 5(samp5.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void plotd1(void);
 7: void main()
 8: {
 9:      init();
10:      viewport(0.2, 0.2, 0.8, 0.8);
11:      xyworld(-0.2, -0.2, 1.2, 1.2);
12:      line(0.5, 0.0, 0.5, 0.02, 0.0, 2.0, 1);
13:      line(1.0, 0.0, 1.0, 0.02, 0.0, 2.0, 1);
14:      line(0.0, 0.5, 0.02, 0.5, 0.0, 2.0, 1);
15:      line(0.0, 1.0, 0.02, 1.0, 0.0, 2.0, 1);
16:      arrow(-0.1, 0.0, 1.2, 0.0, 0.02, 0.0, 2.0);
17:      arrow(0.0, -0.1, 0.0, 1.2, 0.02, 0.0, 2.0);
18:      textx(1.2,-0.02,1,"x");
19:      textx(0.5,-0.02,3,"0.5");
20:      textx(1.0,-0.02,3,"1.0");
21:      texty(-0.02,1.2,1,"y");
22:      texty(0.0,0.5,3,"0.5");
23:      texty(0.0,1.0,3,"1.0");
24:      textx(-0.06,-0.02,1,"0");
25:      plotd1();
26:      fin();
27: }
28: void  plotd1(void)
29: {
30:       int n,m,ix,iy,it;double dt,dx,dy,ds,x0,y0,x1,y1;
31:       linewidth(1.0); setgray(0.6); linety(1);
32:       m=10;
33:       for (ix=0;ix<=m;ix++){
34:        for (iy=0; iy<=m;iy++){
35:         x0=0.0+1.0*double(ix)/double(m); y0=0.0+1.0*float(iy)/float(m);
36:         dt=0.05; dx=1.0; dy=3.0*x0*y0; ds=sqrt(dx*dx+dy*dy);
37:         if(ds!=0.0) {
```

```
38:           x1=x0+dx/ds*0.05; y1=y0+dy/ds*0.05; arrow(x0,y0,x1,y1,0.01,0.6,1.039: );
40:            }
41:         }
42:       }
43:        stroke(); newpath(); setgray(0.4); linewidth(1.2); linety(1);
44:     n=40; m=1;
45:      for(ix=0;ix<=0;ix++){
46:      for(iy=0;iy<=0;iy++){
47:         x0=0.0; y0=0.1;
48:         dt=0.0128;
49:         plot(x0,y0,3);
50:          for(it=1;it<=2*n;it++){
51:           x1= x0+1.0*dt;y1= y0+3.0*x0*y0*dt;
52:           plot(x1,y1,2);
53:           x0=x1;y0=y1;
54:           }
55:        }
56:       }
57:       stroke();
58:       return;
59: }
```

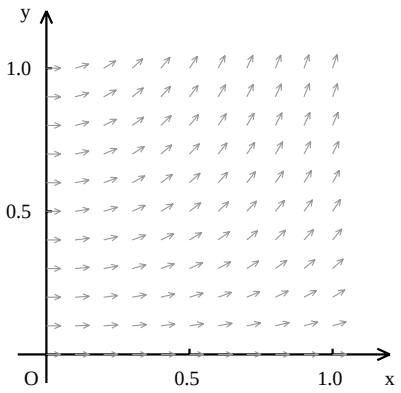

図 34: 方向場.

まず、メインプログラムについてみていく。12 – 15 行目でメモリ線を書いている。16、17 行目では軸を描き、18 – 24 行目でメモリを入れている。25 行目で方向場を描くサブルーチンを呼び出している。サブルーチン plotd1 においては、36 行目で上の式の各々の右辺を代入している。

サンプル 5-2(samp5s.cpp)

```
1: #include <stdio.h>
2: #include <stdlib.h>
3: #include <string.h>
4: #include <math.h>
5: #include "ps.h"
6: void plotd1(void);
7: void main()
```

```
 8: {
 9:      init();
10:      viewport(0.2, 0.2, 0.8, 0.8);
11:      xyworld(-0.2, -0.2, 1.2, 1.2);
12:      linewidth(2.0);
13:      line1(0.5, 0.0, 0.5, 0.02);
14:      line1(1.0, 0.0, 1.0, 0.02);
15:      line1(0.0, 0.5, 0.02, 0.5);
16:      line1(0.0, 1.0, 0.02, 1.0);
17:      arrow1(-0.1, 0.0, 1.2, 0.0, 0.02);
18:      arrow1(0.0, -0.1, 0.0, 1.2, 0.02);
19:      textx(1.2,-0.02,1,"x");
20:      textx(0.5,-0.02,3,"0.5");
21:      textx(1.0,-0.02,3,"1.0");
22:      texty(-0.02,1.2,1,"y");
23:      texty(0.0,0.5,3,"0.5");
24:      texty(0.0,1.0,3,"1.0");
25:      textx(-0.06,-0.02,1,"0");
26:      plotd1();
27:      fin();
28: }
29: void  plotd1(void)
30: {
31:       int n,m,ix,iy,it;double dt,dx,dy,ds,x0,y0,x1,y1;
32:       linewidth(1.0); setgray(0.6); linety(1);
33:       m=10;
34:       for (ix=0;ix<=m;ix++){
35:        for (iy=0; iy<=m;iy++){
36:          x0=0.0+1.0*double(ix)/double(m); y0=0.0+1.0*float(iy)/float(m);
37:          dt=0.05; dx=1.0; dy=3.0*x0*y0; ds=sqrt(dx*dx+dy*dy);
38:          if(ds!=0.0) {
39:            x1=x0+dx/ds*0.05; y1=y0+dy/ds*0.05; arrow1(x0,y0,x1,y1,0.01);
40:            }
41:         }
42:       }
43:        stroke(); newpath(); setgray(0.4); linewidth(1.2); linety(1);
44:      n=40; m=1;
45:      for(ix=0;ix<=0;ix++){
46:       for(iy=0;iy<=0;iy++){
47:         x0=0.0; y0=0.1;
48:         dt=0.0128;
49:         plot(x0,y0,3);
50:          for(it=1;it<=2*n;it++){
51:           x1= x0+1.0*dt;y1= y0+3.0*x0*y0*dt;
52:           plot(x1,y1,2);
53:           x0=x1;y0=y1;
54:           }
```

```
55:         }
56:        }
57:        stroke();
58:        return;
59: }
```

回路図

このプログラムを用いれば簡単に回路図を描くことができる。その一例を下に示す。

サンプル 6(samp6.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:       init();
 9:      viewport(0.2, 0.2, 0.8, 0.8);
10:       xyworld(0.0, 0.0, 1.0, 1.0);
11:       linewidth(2.0);
12:       line(0.0, 0.0, 0.1, 0.0, 0.0, 2.0, 1);
13:       coil(0.1, 0.0, 0.3, 0.0, 0.03, 6, 2.0);
14:       line(0.3, 0.0, 0.4, 0.0, 0.0, 2.0, 1);
15:       line(0.4, 0.0, 0.4, 0.3, 0.0, 2.0, 1);
16:       line(0.4, 0.3, 0.3, 0.3, 0.0, 2.0, 1);
17:       resist(0.3, 0.3, 0.1, 0.3, 2.0, 2.0);
18:       line(0.1, 0.3, 0.0, 0.3, 0.0, 2.0, 1);
19:       line(0.0, 0.3, 0.0, 0.2, 0.0, 2.0, 1);
20:       battery(0.0, 0.2, 0.0, 0.1, 0.1, 2.0);
21:       line(0.0, 0.1, 0.0, 0.0, 0.0, 2.0, 1);
22:       fin();
23: }
```

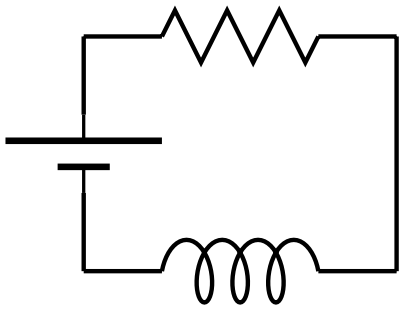

図 35: 回路図.

このプログラム例の出力結果を図 35 に示す。このプログラムは、13 行目でコイル、17 行目で抵抗、そして 20 行目で電池を描いている。サブルーチン line でそれぞれをつないで回路にしている。

サンプル 6-2(samp6s.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void main()
 7: {
 8:      init();
 9:      viewport(0.2, 0.2, 0.8, 0.8);
10:      xyworld(0.0, 0.0, 1.0, 1.0);
11:      linewidth(2.0);
12:      line1(0.0, 0.0, 0.1, 0.0);
13:      coil(0.1, 0.0, 0.3, 0.0, 0.03, 6, 2.0);
14:      linewidth(2.0);
15:      line1(0.3, 0.0, 0.4, 0.0);
16:      line1(0.4, 0.0, 0.4, 0.3);
17:      line1(0.4, 0.3, 0.3, 0.3);
18:      resist(0.3, 0.3, 0.1, 0.3, 2.0, 2.0);
19:      linewidth(2.0);
20:      line1(0.1, 0.3, 0.0, 0.3);
21:      line1(0.0, 0.3, 0.0, 0.2);
22:      battery(0.0, 0.2, 0.0, 0.1, 0.1, 2.0);
23:      linewidth(2.0);
24:      line1(0.0, 0.1, 0.0, 0.0);
25:      fin();
26: }
```

正弦関数

次のプログラムは $y = \sin x$ の関数を描いたものである。

サンプル 7(samp7.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: void frame(void);
 7: void drawline(void);
 8: void main()
 9: {
10:      init();
11:      viewport(0.2, 0.2, 0.8, 0.8);
12:      xyworld(-1.2, -1.2, 1.2, 1.2);
13:      frame();
14:      drawline();
```

```
15:      fin();
16: }
17: void frame()
18: {
19:      int n,i;double x,y,dx,dy;
20:      arrow( -1.1, 0.0, 1.1, 0.0, 0.02, 0.0, 1.0);
21:      arrow( 0.0, -1.1, 0.0, 1.1, 0.02, 0.0, 1.0);
22:      text(1.0,-0.1,1,"x");
23:      text(-0.1,1.0,1,"y");
24:      text(-0.1,-0.1,1,"O");
25:      n=10;
26:        for (i=1;i<=n;i++){
27:         x=0.1*i; if(i/5*5==i) dy=0.02; else  dy=0.01;
28:         line( -x, -dy, -x, dy, 0.0, 1.0, 1);
29:         line( x, -dy, x, dy, 0.0, 1.0, 1);
30:         }
31:      for(i=1;i<=n;i++){
32:         y=0.1*i; if(i/5*5==i) dx=0.02; else  dx=0.01;
33:         line( -dx, -y, dx, -y, 0.0, 1.0, 1);
34:         line( -dx, y, dx, y, 0.0, 1.0, 1);
35:         } stroke();
36:      return;
37: }
38: void drawline(void)
39: {
40:      int i,nx=100;double x,y,t;
41:        x=-1.0;  y=0.0;
42:       linewidth(1.5);
43:       plot( x, y, 3);
44:         for (i=-nx+1;i<=nx;i++){
45:           t = double(i)/double(nx);x = t; y = sin(t*3.14159);
46:         plot( x, y, 2);
47:          } stroke();
48:     return;
49: }
```

まず、サブルーチン frameで軸を描き、メモリを入れて、そしてメモリ線を引いている。次に、サブルーチン drawlineで与えられた関数を描いている。45 行目に関数を与える。出力結果を図 36 に示す。

サンプル 7-2(samp7s.cpp)

```
1: #include <stdio.h>
2: #include <stdlib.h>
3: #include <string.h>
4: #include <math.h>
5: #include "ps.h"
6: void frame(void);
7: void drawline(void);
8: void main()
```

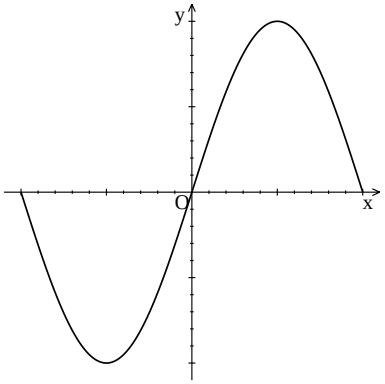

図 36: 正弦関数.

```
 9: {
10:      init();
11:      viewport(0.2, 0.2, 0.8, 0.8);
12:      xyworld(-1.2, -1.2, 1.2, 1.2);
13:      frame();
14:      drawline();
15:      fin();
16: }
17: void frame()
18: {
19:      int n,i;double x,y,dx,dy;
20:      linewidth(1.0);
21:      arrow1( -1.1, 0.0, 1.1, 0.0, 0.02);
22:      arrow1( 0.0, -1.1, 0.0, 1.1, 0.02);
23:      text(1.0,-0.1,1,"x");
24:      text(-0.1,1.0,1,"y");
25:      text(-0.1,-0.1,1,"O");
26:      n=10;
27:        for (i=1;i<=n;i++){
28:         x=0.1*i; if(i/5*5==i) dy=0.02; else  dy=0.01;
29:         line1( -x, -dy, -x, dy);
30:         line1(  x, -dy,  x, dy);
31:         }
32:      for(i=1;i<=n;i++){
33:         y=0.1*i; if(i/5*5==i) dx=0.02; else  dx=0.01;
34:         line1( -dx, -y, dx, -y);
35:         line1( -dx,  y, dx,  y);
36:         } stroke();
37:      return;
38: }
39: void drawline(void)
40: {
41:      int i,nx=100;double x,y,t;
42:       x=-1.0;  y=0.0;
```

```
43:       linewidth(1.5);
44:       plot( x, y, 3);
45:         for (i=-nx+1;i<=nx;i++){
46:           t = double(i)/double(nx);x = t; y = sin(t*3.14159);
47:         plot( x, y, 2);
48:          } stroke();
49:     return;
50: }
```

関数の作図 1

次のプログラムは $y = \exp -x \sin 4\pi x$ の関数を描いたものである。
紙の都合上、右端までに書ききれなかった行は、'[継続行]' として折り返してすぐ下の行に書いた。

サンプル 8(samp8.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: double f(double);
 7: void plotd(void);
 8: void frame(void);
 9: void main()
10: {
11:      init();
12:      viewport(0.2,0.2,0.8,0.6); xyworld(0.0,-1.0,1.0,1.0);
13:       frame();
14:       plotd();
15:      fin();
16: }
17: double f(double x)
18: {       double pi, fx;
19:          pi=atan(1.0)*4.0;
20:          fx=exp(-x)*sin(4.0*pi*x);
21:       return(fx);
22: }
23: void plotd()
24: {
25:       double x1[21],y1[21];int n,i;
26:       n=20;
27:       for(i=0;i<=n;i++){
28:         x1[i]=double(i)/double(n);y1[i]=f(x1[i]); }
29:       linewidth(1.5);linety(1);
30:       setgray(0.0);
31:       plot(x1[0],y1[0],3);
32:       for (i=1;i<=n-2;i++){
33:       spline1(x1[i-1],y1[i-1],x1[i],y1[i],x1[i+1],y1[i+1]
              [継続行]                ,x1[i+2],y1[i+2],-1); }
```

```
34:       spline1(x1[n-3],y1[n-3],x1[n-2],y1[n-2]
              [継続行]                ,x1[n-1],y1[n-1],x1[n],y1[n],0);
35:       spline1(x1[n-3],y1[n-3],x1[n-2],y1[n-2]
              [継続行]                ,x1[n-1],y1[n-1],x1[n],y1[n],1);
36:       stroke();
37: }
38: void frame()
39: {
40:       int ix,nx,iy,ny; double x,dx,y,dy;
41:       linewidth(1.0);
42:       rect(0.0,-1.0,1.0,1.0,0.0,1.0,1);
43:       nx=2;
44:       for(ix=1;ix<=nx*10-1;ix++){
45:        x=double(ix)/double(nx*10);dy=1.0;
46:        if(ix/10*10==ix) setgray(0.5); else setgray(0.8);
47:        plot(x,-dy,3);plot(x,dy,2);stroke(); }
48:       ny=2;
49:       for(iy=1;iy<=ny*10-1;iy++){
50:        y=double(iy)/double(ny*10)*2.0-1.0;dx=1.0;
51:        if(iy/10*10==iy) setgray(0.5); else setgray(0.8);
52:        plot(0.0,y,3);plot(dx,y,2);stroke(); }
53:       setgray(0.0);
54:       textx(0.75,0.0,1,"x");
55:       textx(0.05,0.02,1,"0");textx(0.55,0.02,3,"0.5")
              [継続行]       ;textx(0.95,0.02,1,"1");
56:       texty(0.0,0.5,1,"y");
57:          texty(0.0,-1.0,2,"-1");texty(0.0,0.0,1,"0");texty(0.0,1.0,1,"1");
58: }
```

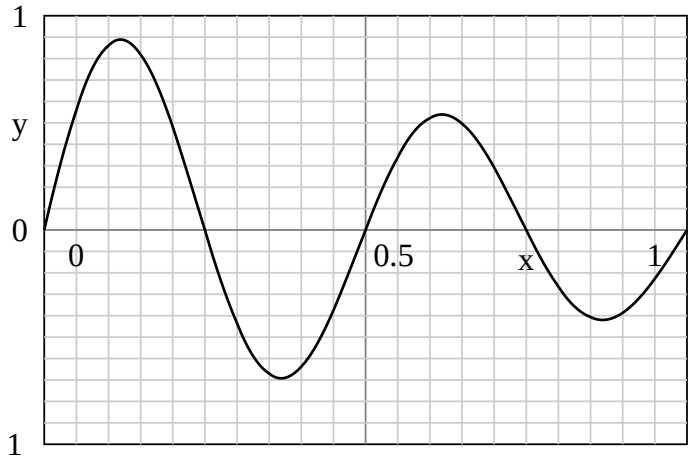

図 37: グラフ 1.

このプログラムでは、サブルーチン frame で軸を描き、メモリを入れて、そしてメモリ線を引いている。46 行目でメモリ線の色の設定を行っている。17 行目から 22 行目で描きたい関数を与える。出力結果を図 37 に示す。

関数の作図 2

次のプログラムは関数 $y = x^3/(5.0 + x^2) + 10.0$ を両対数で出力する。

サンプル 9(samp9.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: double f(double);
 7: void plotd(void);
 8: void frame(void);
 9: void main()
10: {
11:      init();
12:      viewport(0.2,0.2,0.8,0.8); xyworld(0.0,0.0,4.0,4.0);
13:       frame();
14:       plotd();
15:      fin();
16: }
17: double f(double x)
18: {     double pi, fx;
19:         fx=x*x*x/(5.0+x*x)+10.0;
20:       return(fx);
21: }
22: void plotd()
23: {
24:       double x1[21],y1[21];int n,i;
25:       n=20;
26:       for(i=0;i<=n;i++){
27:         x1[i]=1.0*pow(10.0,double(i)/double(n)*4.0);y1[i]=f(x1[i]); }
28:       for(i=0;i<=n;i++){
29:         x1[i]=log10(x1[i]);y1[i]=log10(y1[i]);}
30:       linewidth(1.5);linety(1);
31:       setgray(0.0);
32:       plot(x1[0],y1[0],3);
33:       for (i=1;i<=n-2;i++){
34:        spline1(x1[i-1],y1[i-1],x1[i],y1[i]
              [継続行]       ,x1[i+1],y1[i+1],x1[i+2],y1[i+2],-1); }
35:       spline1(x1[n-3],y1[n-3],x1[n-2],y1[n-2]
              [継続行]       ,x1[n-1],y1[n-1],x1[n],y1[n],0);
36:       spline1(x1[n-3],y1[n-3],x1[n-2],y1[n-2]
              [継続行]       ,x1[n-1],y1[n-1],x1[n],y1[n],1);
37:       stroke();
38: }
39: void frame()
40: {
41:       int ix,nx,iy,ny,ixx,iyy; double x,dx,y,dy;
```

```
42:       linewidth(1.0);
43:       rect(0.0,0.0,4.0,4.0,0.0,1.0,1);
44:       nx=4;
45:       for(ix=0;ix<=nx-1;ix++){
46:         setgray(0.3);
47:         x=double(ix);dy=4.0;
48:         if(ix!=0) {plot(x,0.0,3);plot(x,dy,2);stroke(); }
49:         for(ixx=2;ixx<=9;ixx++) {
50:              setgray(0.8);x=double(ix)+log10(double(ixx));
51:              plot(x,0.0,3);plot(x,dy,2);stroke();  } }
52:       ny=4;
53:       for(iy=0;iy<=ny-1;iy++){
54:         setgray(0.5);
55:         y=double(iy);dx=4.0;
56:         if(iy!=0) {plot(0.0,y,3);plot(dx,y,2);stroke(); }
57:         for(iyy=2;iyy<=9;iyy++) {
58:              setgray(0.8);y=double(iy)+log10(double(iyy));
59:              plot(0.0,y,3);plot(dx,y,2);stroke();  } }
60:       setgray(0.0);
61:       textx(3.5,0.0,1,"x");
62:          textx(0.0,0.02,1,"1");textx(1.0,0.02,2,"10")
           [継続行]       ;textx(2.0,0.02,3,"100");
63:          textx(3.0,0.02,4,"1000");textx(4.0,0.02,5,"10000");
64:       texty(0.0,3.5,1,"y");
65:          texty(0.0,0.0,1,"1");texty(0.0,1.0,2,"10")
           [継続行]       ;texty(0.0,2.0,3,"100");
66:          texty(0.0,3.0,4,"1000");texty(0.0,4.0,5,"10000");
67:  }
```

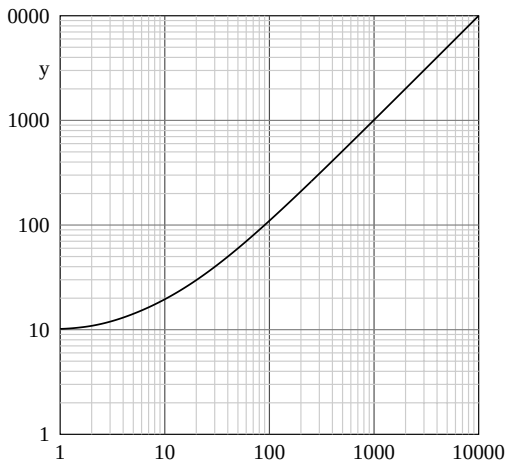

図 38: 両対数グラフ.

このプログラムでは、サブルーチン frame で軸を描き、メモリを入れて、そしてメモリ線を引いている。50 行目と 54 行目でメモリ線の色の設定を行っている。17 行目から 21 行目で描きたい関数を与える。出力結果を図 38 に示す。

関数の作図 3

次のプログラムは関数 $y = \sin(2.0\pi x) \times \sin(\pi y)$ を等高線で出力する。

サンプル 10(samp10.cpp)

```
 1: #include <stdio.h>
 2: #include <stdlib.h>
 3: #include <string.h>
 4: #include <math.h>
 5: #include "ps.h"
 6: double f(double,double);
 7: void draw(double, double, int);
 8: void frame(void);
 9: void main()
10: {
11:      init();
12:      viewport(0.2,0.2,0.8,0.5); xyworld(0.0,0.0,2.0,1.0);
13:      draw(1.0,-1.0,20);
14:      frame();
15:      fin();
16: }
17: void frame()
18: {
19:       int ix,nx,iy,ny,ixx,iyy; double x,dx,y,dy;
20:       linewidth(1.0);
21:       rect(0.0,0.0,2.0,1.0,0.0,1.0,1);
22:       nx=4;
23:       for(ix=0;ix<=2*nx-1;ix++){
24:         setgray(0.0);
25:         x=double(ix)/double(nx)*2.0;dy=0.05;
26:         plot(x,0.0,3);plot(x,dy,2);stroke(); }
27:       ny=4;
28:       for(iy=0;iy<=ny-1;iy++){
29:         setgray(0.0);
30:         y=double(iy);dx=0.05;
31:         plot(0.0,y,3);plot(dx,y,2);stroke(); }
32:       textx(1.5,0.0,1,"x");
33:         textx(0.0,0.02,1,"0");textx(1.0,0.02,1,"1");textx(2.0,0.02,1,"2");
34:       texty(0.0,0.75,1,"y");
35:          texty(0.0,0.0,1,"0");texty(0.0,0.5,3,"0.5");texty(0.0,1.0,1,"1");
36:   }
37: double f(double x,double y)
38: {     double pi, fx;
39:         pi=3.14159;
40:         fx=sin(2.0*pi*x)*sin(pi*y);
41:       return(fx);
42: }
43: void draw(double fmax,double fmin, int nline)
44: {
45:       double x,y,dx,dy,color,r,g,b;int nx,ny,ix,iy;
```

```
46:       nx=200;ny=100;dx=2.0/double(nx);dy=1.0/double(ny);
47:       for(iy=0;iy<=ny;iy++){ y=double(iy)/double(ny);
48:         for(ix=0;ix<=nx;ix++){ x=double(ix)/double(nx);
49:          color=(f(x,y)-fmin)/(fmax-fmin);
50:          plot((ix-1)*dx,(iy-1)*dy,3);plot(ix*dx,(iy-1)*dy,2);
51:          plot(ix*dx,iy*dy,2);plot((ix-1)*dx,iy*dy,2);closepath();
52:          setrgb(color*0.5,0.1,1.0-color*0.5);
53:          fill();stroke(); }
54:          }
55: }
```

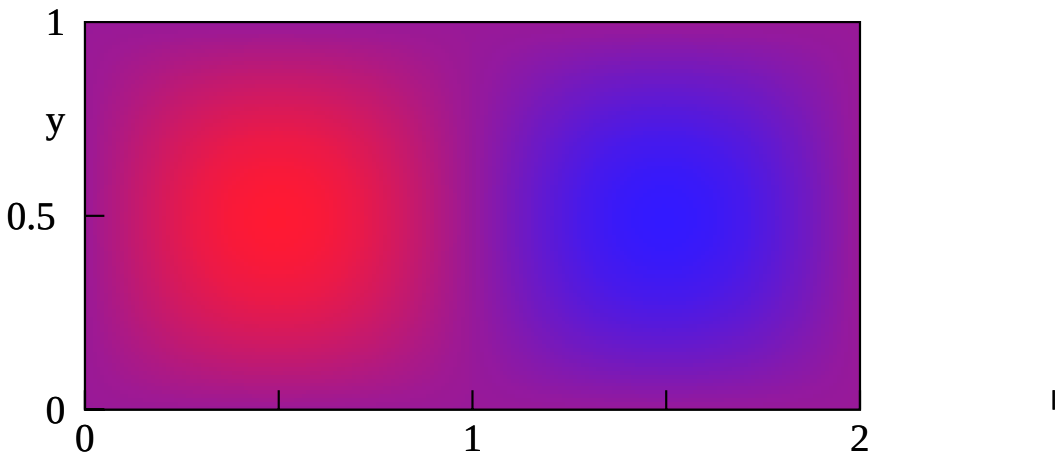

図 39: 等高線.

このプログラムでは、サブルーチン frame で軸を描き、メモリを入れて、そしてメモリ線を引いている。38 行目から 42 行目で描きたい関数を与える。出力結果を図 39 に示す。

8　作図プログラム

データには、実験結果や数値計算によって得られるデータなどさまざまなデータがあるだろう。そのようなデータを一定の形式に従ってデータファイルを作るだけで簡単に、2 次元平面グラフを描くプログラムを紹介する。

データファイルの作り方

この作図プログラムを使うためには、まずデータファイルを用意しなければならない。データファイルの例を下に示す。このデータファイルから作図プログラム 1(fig1) を用いて作成されたグラフが図 40 である。データファイルの作り方は、実に簡単でグラフを描く平面を (x, y) とするとデータは (1.0,1.2)、(2.0,3.2)、(3.25,5.6)、... と縦に並べていく。x のデータと y のデータは、1 つ以上隙間を空けておく。そして、データファイル名を、作図プログラムの中で指定したデータファイル名と対応させればよい。作図プログラム 1(fig1) を用いる場合は、pg.data とすればよい。

データファイル (pg.data)

```
1.0  1.2
2.0  3.2
3.25 5.6
5.1  4.5
6.8  2.1
```

プログラムのコンパイル・リンク操作

作図プログラムをコンパイルし、psbasic.for(psbasic.cpp) とのリンク操作を行う。ここでは、コンピュータの D ドライブにディレクトリ \FIGURE があり、このディレクトリに作図プログラム fig.for(fig.cpp) とポストスクリプト基本命令 psbasic.for(psbasic.cpp) とがあるとする。

(Fortran) の場合
D : \FIGURE ＞ f90 △fig.for △ psbasic.for
(C) の場合
D : \FIGURE ＞ cl △fig.cpp △ psbasic.cpp

データファイルに問題がなければ、実行ファイル fig.exeが作成される。ここで、psbasic.for(psbasic.cpp) は、すでにシステムにインストールされているとする。もし、このディレクトリに psbasic.for(psbasic.cpp) がないときは、ホームページ

http://saffman.doshisha.ac.jp/

よりダウンロードする。

プログラムの実行

fig.exe を実行することにより、計算が実行できる。グラフィック出力があるときには、temp1.ps という名の PostScript ファイルが作成される。

D : \FIGURE ＞fig.exe

プログラムを実行すれば、temp1.ps というファイル名のポストスクリプトファイルが出来上がる。

8.1　作図プログラムの紹介

作図プログラムそれぞれについての細かい説明や特徴を説明する。どの作図プログラムもデータの作り方やコンパイル方法は、上で示した通りである。

作図プログラム 1(fig1)

最も簡単にグラフを描画する作図プログラムである（付録 C に示した）。この作図プログラム 1(fig1) は、データの中から (x, y) のそれぞれの最小値と最大値を見つけ出し、その領域内で 1 つのデータのグラフを描画する。そのため、いつも枠いっぱいに広がった 1 つのグラフができあがる。論文作成などの清書の図としては不向きであると思うが、実験

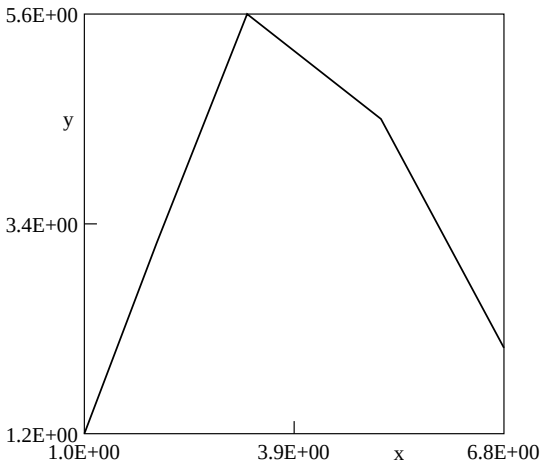

図 40: 出力図.

においてその結果の推移を一時的に見るときや未知のデータのグラフを描くときには、便利である。前ページのデータファイル pg.data を用いて作成したグラフの出力が図 40 である。

9 TEX にポストスクリプトファイルを貼りこむには

本書では、文書整形は TEX で、作図はポストスクリプトを用いている。この章では、本書のようにポストスクリプトファイルを TEX に貼りこむ方法について説明する。

まずは、貼りこむための準備をする。次のようにスタイルファイルをそれぞれのフォルダに置く。 `Indent.sty` と `Psfrag.sty` を `\TeX\texmf\tex\platex\base` の下に置き、 `Psfrag.pro` を `\TeX\texmf\dvips` の下に置く。次に張りこみたいポストスクリプトファイルを EPS(Encapsulated PostScript) ファイルに変えなければならない。その方法は、ポストスクリプトファイルを開けて File の PStoEPS を実行する。したがって、実際に貼りこむファイルはポストスクリプトファイルではなくてこの EPS ファイルである。

ここから実際に貼りこむ方法について述べていく。まず、TEX を使用して文書整形する際には、例えば次のような入力ファイルを作ることになる（TEX についての詳しい説明はここでは省く。他の書物を参考されたい）。

```
\documentclass[a4paper,10pt]{jarticle}

（プリアンブル）

\begin{document}

（本文）

\end{document}
```

（プリアンブル）は、標準的な文書スタイルの一部を変更するためのパラメータの設定を行うところである。ここに、以下の 3 行を追加する。

```
\usepackage[dvips]{graphicx}
\usepackage{psfrag}
\usepackage{indent}
```

これにより準備しておいたスタイルファイルを用いてポストスクリプトファイルを貼りこむことを宣言している。次に、ポストスクリプトファイルを挿入したい文章中に

```
1: \begin{figure}
2: \begin{center}
3: \psfrag{A}[][][1.5]{$X$}
4: \psfrag{B}[][][1.5]{$Y$}
5: \includegraphics*[図の大きさの指定]{ファイル名.eps}\\
6: \caption{図の見出し}
7: \end{center}
8: \end{figure}
```

を挿入すればポストスクリプトファイルが貼りこめる。ポストスクリプトファイルを貼りこむには、figure 環境（1 行目、8 行目）のもと、5 行目の `\includegraphics` を用いる。このファイルはこの章の最初でも述べたが、EPS ファイルである。5 行目の（図の大きさの指定）では、 `height=5cm` 、 `width=5cm` 、 `scale=0.7` などと記述する。 `height` では高さを、 `width` では幅を、そして `scale` では図の縮尺を決定する。3 と 4 行目の `\psfrag` では、ポストスクリプトで書いた A と B を TEX に貼りこむ際に、それぞれ 1.5 倍して X と Y に変換するということを行っている。

こうして出来上がった TEX ファイルをコンパイルすれば TEX ファイルと同じ名前の dvi ファイルが出来上がる。最後にこの dvi ファイルを wdvips などによってポストスクリプトファイルに変換してやれば完成である。

A Fortran ライブラリー (psbasic.for)

```
C     plotting routines for general purpose using postscript
      subroutine init
      common /paper/xsize,ysize
      common /viewp/xv1,yv1,xv2,yv2
      common /world/xw1,yw1,xw2,yw2
      open(1,file='temp1.ps')
C     aspect ratio of the paper is 1.4143
       write(1,*) '%! Created by Jiro Mizushima'
       write(1,*) '/Times-Roman findfont 18 scalefont setfont'
       write(1,*) '/cm {28.35 mul } def '
       xsize= 21.0
       ysize= 29.7
       ysize= 21.0
       call viewport(0.2,0.2,0.8,0.8)
       call xyworld(0.0,0.0,1.0,1.0)
       call linety(1)
       call linewidth(1.0)
      return
      end
      subroutine viewport(xv1d,yv1d,xv2d,yv2d)
      common /viewp/xv1,yv1,xv2,yv2
       xv1=xv1d
       yv1=yv1d
       xv2=xv2d
       yv2=yv2d
      return
      end
      subroutine xyworld(xw1d,yw1d,xw2d,yw2d)
      common /world/xw1,yw1,xw2,yw2
       xw1=xw1d
       yw1=yw1d
       xw2=xw2d
       yw2=yw2d
      return
      end
      subroutine fin
       call stroke
       write(1,*) ' showpage '
       close(1)
      return
      end
      subroutine linety(ichar)
      if (ichar.eq.1) then
       write(1,*) '[] 0 setdash'
       else if (ichar.eq.2) then
       write(1,*) '[2 2] 0 setdash'
```

```
 else if (ichar.eq.3) then
 write(1,*) '[8 2] 0 setdash'
 else if (ichar.eq.4) then
 write(1,*) '[8 1 1 1] 0 setdash'
end if
return
end
subroutine linewidth(w)
 write(1,*) w,' setlinewidth'
return
end
subroutine setgray(g)
 write(1,*) g,' setgray'
return
end
subroutine setrgb(r,g,b)
write(1,*) r,g,b,' setrgbcolor'
return
end
subroutine newpath
 write(1,*) ' newpath'
return
end
subroutine closepath
 write(1,*) ' closepath'
return
end
subroutine stroke
 write(1,*) ' stroke'
return
end
subroutine rotate(itheta)
 write(1,*) itheta,' rotate'
return
end
subroutine scale(tx,ty)
 write(1,*) tx, ty,' scale'
return
end
subroutine translate(x,y)
common /paper/xsize,ysize
common /viewp/xv1,yv1,xv2,yv2
common /world/xw1,yw1,xw2,yw2
 x1=((x-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 y1=((y-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
write(1,*) x1,' cm ',y1,' cm  translate '
return
```

```
end
subroutine clipon(x1,y1,x2,y2)
 call newpath
 call plot(x1, y1, 3)
 call plot(x2, y1, 2)
 call plot(x2, y2, 2)
 call plot(x1, y2, 2)
 write(1,*) 'closepath clip'
return
end
subroutine eoclipon(x1,y1,x2,y2)
 call newpath
 call plot(x1, y1, 3)
 call plot(x2, y1, 2)
 call plot(x2, y2, 2)
 call plot(x1, y2, 2)
write(1,*) 'closepath eoclip'
return
end
subroutine clipoff
 write(1,*) 'initclip'
return
end
subroutine plot(x,y,ipen)
common /paper/xsize,ysize
common /viewp/xv1,yv1,xv2,yv2
common /world/xw1,yw1,xw2,yw2
 x1=((x-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 y1=((y-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 if(ipen.eq.3) write(1,*) x1,' cm ',y1,' cm  moveto'
 if(ipen.eq.2) write(1,*) x1,' cm ',y1,' cm  lineto'
return
end
subroutine line(x1,y1,x2,y2,g,w,it)
 call linety(it)
 call linewidth(w)
 call setgray(g)
 call plot(x1, y1, 3)
 call plot(x2, y2, 2)
 call stroke
 call linety(1)
 call linewidth(1.0)
 call setgray(0.0)
return
end
subroutine line1(x1,y1,x2,y2)
 call plot(x1, y1, 3)
```

```
 call plot(x2, y2, 2)
 call stroke
return
end
subroutine rect(x1,y1,x2,y2,g,w,it)
 call linety(it)
 call linewidth(w)
 call setgray(g)
 call newpath
 call plot(x1, y1, 3)
 call plot(x2, y1, 2)
 call plot(x2, y2, 2)
 call plot(x1, y2, 2)
 call closepath
 if(g.ne.0.0) write(1,*) ' fill'
 call stroke
 call linety(1)
 call linewidth(1.0)
 call setgray(0.0)
return
end
subroutine rect1(x1,y1,x2,y2)
 call newpath
 call plot(x1, y1, 3)
 call plot(x2, y1, 2)
 call plot(x2, y2, 2)
 call plot(x1, y2, 2)
 call closepath
return
end
subroutine rectfill1(x1,y1,x2,y2)
 call newpath
 call plot(x1, y1, 3)
 call plot(x2, y1, 2)
 call plot(x2, y2, 2)
 call plot(x1, y2, 2)
 call closepath
 write(1,*) ' fill'
 call stroke
return
end
subroutine circ(x1,y1,r1,g,w,it)
common /paper/xsize,ysize
common /viewp/xv1,yv1,xv2,yv2
common /world/xw1,yw1,xw2,yw2
 xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
```

```
 rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
 call linety(it)
 call linewidth(w)
 call setgray(g)
 call newpath
 write(1,*) xx1, ' cm ',yy1, ' cm ',rr1,' cm ', '0 360 arc '
 if(g.ne.0.0) write(1,*) ' fill'
 call stroke
 call linety(1)
 call linewidth(1.0)
 call setgray(0.0)
return
end
subroutine circ1(x1,y1,r1)
common /paper/xsize,ysize
common /viewp/xv1,yv1,xv2,yv2
common /world/xw1,yw1,xw2,yw2
 xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
 call newpath
 write(1,*) xx1, ' cm ',yy1, ' cm ',rr1,' cm ', '0 360 arc '
 call stroke
return
end
subroutine circfill1(x1,y1,r1)
common /paper/xsize,ysize
common /viewp/xv1,yv1,xv2,yv2
common /world/xw1,yw1,xw2,yw2
 xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
 call newpath
 write(1,*) xx1, ' cm ',yy1, ' cm ',rr1,' cm ', '0 360 arc '
 write(1,*) 'fill'
 call stroke
return
end
subroutine ellipse(x1,y1,rx,ry,g,w,it)
 call linety(it)
 call linewidth(w)
 call setgray(g)
 call newpath
 x=x1+rx
 y=y1
 dt=3.14156*2.0/float(20)
 call plot(x,y,3)
```

```
     do 10 i=1,20
      x=x1+rx*cos(dt*i)
      y=y1+ry*sin(dt*i)
      call plot(x,y,2)
10   continue
     call stroke
     call linety(1)
     call linewidth(1.0)
     call setgray(0.0)
    return
    end
    subroutine ellipse1(x1,y1,rx,ry)
     call newpath
     x=x1+rx
     y=y1
     dt=3.14156*2.0/float(20)
     call plot(x,y,3)
     do 10 i=1,20
      x=x1+rx*cos(dt*i)
      y=y1+ry*sin(dt*i)
      call plot(x,y,2)
10   continue
     call stroke
    return
    end
    subroutine ellipsefill1(x1,y1,rx,ry)
     call newpath
     x=x1+rx
     y=y1
     dt=3.14156*2.0/float(20)
     call plot(x,y,3)
     do 10 i=1,20
      x=x1+rx*cos(dt*i)
      y=y1+ry*sin(dt*i)
      call plot(x,y,2)
10   continue
     write(1,*) 'fill'
     call stroke
     return
    end
    subroutine arc(x1,y1,r1,t1,t2,g,w,it)
    common /paper/xsize,ysize
    common /viewp/xv1,yv1,xv2,yv2
    common /world/xw1,yw1,xw2,yw2
     xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
     yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
     rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
```

```
 call linety(it)
 call linewidth(w)
 call setgray(g)
 call newpath
 write(1,*) xx1, ' cm ',yy1, ' cm ',rr1,' cm ',t1,t2,' arc '
 call stroke
 call linety(1)
 call linewidth(1.0)
 call setgray(0.0)
return
end
subroutine arc1(x1,y1,r1,t1,t2)
common /paper/xsize,ysize
common /viewp/xv1,yv1,xv2,yv2
common /world/xw1,yw1,xw2,yw2
 xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
 call newpath
 write(1,*) xx1, ' cm ',yy1, ' cm ',rr1,' cm ',t1,t2,' arc '
 call stroke
return
end
subroutine curv(x1,y1,x2,y2,x3,y3,x4,y4,g,w,it)
common /paper/xsize,ysize
common /viewp/xv1,yv1,xv2,yv2
common /world/xw1,yw1,xw2,yw2
 xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 xx2=((x2-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy2=((y2-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 xx3=((x3-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy3=((y3-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 xx4=((x4-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
 yy4=((y4-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
 call linety(it)
 call linewidth(w)
 call setgray(g)
 call newpath
 write(1,*) xx1,' cm ',yy1,' cm moveto'
 write(1,*) xx2,' cm ',yy2,' cm ',xx3,' cm ',yy3,' cm ',
&    xx4,' cm',yy4,' cm curveto'
 call stroke
 call linety(1)
 call linewidth(1.0)
 call setgray(0.0)
return
```

```
 end
 subroutine curv1(x1,y1,x2,y2,x3,y3,x4,y4)
 common /paper/xsize,ysize
 common /viewp/xv1,yv1,xv2,yv2
 common /world/xw1,yw1,xw2,yw2
  xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
  yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
  xx2=((x2-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
  yy2=((y2-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
  xx3=((x3-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
  yy3=((y3-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
  xx4=((x4-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
  yy4=((y4-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
  call newpath
  write(1,*) xx1,' cm ',yy1,' cm moveto'
  write(1,*) xx2,' cm ',yy2,' cm ',xx3,' cm ',yy3,' cm ',
&    xx4,' cm',yy4,' cm curveto'
  call stroke
 return
 end
 subroutine triangl(x1,y1,r1,a1,g,w,it)
 common /paper/xsize,ysize
 common /viewp/xv1,yv1,xv2,yv2
 common /world/xw1,yw1,xw2,yw2
  xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
  yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
  rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
  aa1=a1*3.14159/180.0
  aa2=aa1+3.14159/3.0
  call linety(it)
  call linewidth(w)
  call setgray(g)
  call newpath
  write(1,*) xx1,' cm ',yy1,' cm  moveto '
  write(1,*) xx1+rr1*cos(aa1),' cm',yy1+rr1*sin(aa1),' cm lineto '
  write(1,*) xx1+rr1*cos(aa2),' cm',yy1+rr1*sin(aa2),' cm lineto '
  call closepath
  if(g.ne.0.0) write(1,*) ' fill'
  call stroke
  call linety(1)
  call linewidth(1.0)
  call setgray(0.0)
 return
 end
 subroutine triangl1(x1,y1,r1,a1)
 common /paper/xsize,ysize
 common /viewp/xv1,yv1,xv2,yv2
```

```
 common /world/xw1,yw1,xw2,yw2
  xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
  yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
  rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
  aa1=a1*3.14159/180.0
  aa2=aa1+3.14159/3.0
  call newpath
  write(1,*) xx1,' cm ',yy1,' cm  moveto '
  write(1,*) xx1+rr1*cos(aa1),' cm',yy1+rr1*sin(aa1),' cm lineto '
  write(1,*) xx1+rr1*cos(aa2),' cm',yy1+rr1*sin(aa2),' cm lineto '
  call closepath
  call stroke
 return
 end
 subroutine trianglfill1(x1,y1,r1,a1)
 common /paper/xsize,ysize
 common /viewp/xv1,yv1,xv2,yv2
 common /world/xw1,yw1,xw2,yw2
  xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize
  yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize
  rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize
  aa1=a1*3.14159/180.0
  aa2=aa1+3.14159/3.0
  call newpath
  write(1,*) xx1,' cm ',yy1,' cm  moveto '
  write(1,*) xx1+rr1*cos(aa1),' cm',yy1+rr1*sin(aa1),' cm lineto '
  write(1,*) xx1+rr1*cos(aa2),' cm',yy1+rr1*sin(aa2),' cm lineto '
  call closepath
  write(1,*) ' fill'
  call stroke
 return
 end
 subroutine spline(x1,y1,x2,y2,x3,y3,x4,y4,ipart,g,w,it)
  ns=10
  call linety(it)
  call linewidth(w)
  call setgray(g)
  write(1,*) 'newpath'
  u=float(ipart)
  x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
&  -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4
  y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2
&  -(u+1.)*u*(u-2.)/2.*y3+(u+1.)*u*(u-1.)/6.*y4
  call plot(x,y,3)
  do 10 i=1,ns
   u=float(ipart)+float(i)/float(ns)
   x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
```

```
  &   -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4
     y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2
  &   -(u+1.)*u*(u-2.)/2.*y3+(u+1.)*u*(u-1.)/6.*y4
     call plot(x,y,2)
10   continue
     call stroke
     call linety(1)
     call linewidth(1.0)
     call setgray(0.0)
    return
    end
    subroutine spline1(x1,y1,x2,y2,x3,y3,x4,y4,ipart)
     ns=10
     write(1,*) 'newpath'
     u=float(ipart)
     x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
  &  -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4
     y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2
  &  -(u+1.)*u*(u-2.)/2.*y3+(u+1.)*u*(u-1.)/6.*y4
     call plot(x,y,3)
     do 10 i=1,ns
      u=float(ipart)+float(i)/float(ns)
      x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
  &   -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4
      y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2
  &   -(u+1.)*u*(u-2.)/2.*y3+(u+1.)*u*(u-1.)/6.*y4
     call plot(x,y,2)
10   continue
     call stroke
    return
    end
    subroutine parabola(x1,y1,x2,y2,x3,y3,g,w,it)
     u=0.0
     ns=20
     call linety(it)
     call linewidth(w)
     call setgray(g)
     call newpath
     x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3
     y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3
     call plot(x,y,3)
     do 10 i=1,2*ns
      u=float(i)/float(ns)
      x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3
      y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3
      call plot(x,y,2)
10   continue
```

```
     call stroke
     call linety(1)
     call linewidth(1.0)
     call setgray(0.0)
    return
    end
    subroutine parabola1(x1,y1,x2,y2,x3,y3)
     u=0.0
     ns=20
     call newpath
     x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3
     y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3
     call plot(x,y,3)
     do 10 i=1,2*ns
      u=float(i)/float(ns)
      x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3
      y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3
      call plot(x,y,2)
10   continue
     call stroke
    return
    end
    subroutine arrow(x1,y1,x2,y2,d,g,w)
     s=sqrt((x2-x1)**2+(y2-y1)**2)
     alpha=2.2
     x3=x2-(x2-x1)/s*d*alpha-(y2-y1)/s*d
     y3=y2-(y2-y1)/s*d*alpha+(x2-x1)/s*d
     x4=x2-(x2-x1)/s*d*alpha+(y2-y1)/s*d
     y4=y2-(y2-y1)/s*d*alpha-(x2-x1)/s*d
     call linewidth(w)
     call setgray(g)
     call plot(x1, y1, 3)
     call plot(x2, y2, 2)
     call plot(x2, y2, 3)
     call plot(x3, y3, 2)
     call plot(x2, y2, 3)
     call plot(x4, y4, 2)
     call stroke
     call linety(1)
     call linewidth(1.0)
     call setgray(0.0)
    return
    end
    subroutine arrow1(x1,y1,x2,y2,d)
     s=sqrt((x2-x1)**2+(y2-y1)**2)
     alpha=2.2
     x3=x2-(x2-x1)/s*d*alpha-(y2-y1)/s*d
```

```
      y3=y2-(y2-y1)/s*d*alpha+(x2-x1)/s*d
      x4=x2-(x2-x1)/s*d*alpha+(y2-y1)/s*d
      y4=y2-(y2-y1)/s*d*alpha-(x2-x1)/s*d
      call plot(x1, y1, 3)
      call plot(x2, y2, 2)
      call plot(x2, y2, 3)
      call plot(x3, y3, 2)
      call plot(x2, y2, 3)
      call plot(x4, y4, 2)
      call stroke
    return
    end
    subroutine resist(x1,y1,x2,y2,d,w)
      s=sqrt((x2-x1)**2+(y2-y1)**2)
      call linewidth(w)
      dx=(x2-x1)/12.0
      dy=(y2-y1)/12.0
      ex=-d*dy
      ey=d*dx
      call plot(x1, y1, 3)
      xx=x1+dx+ex
      yy=y1+dy+ey
      call plot(xx, yy, 2)
      do 10 i=1, 5
       xx=x1+(i*2+1)*dx+(-1)**i*ex
       yy=y1+(i*2+1)*dy+(-1)**i*ey
       call plot(xx, yy, 2)
10    continue
      call plot(x2, y2, 2)
      call stroke
      call linety(1)
    return
    end
    subroutine battery(x1,y1,x2,y2,d,w)
      s=sqrt((x2-x1)**2+(y2-y1)**2)
      call linewidth(w)
      dx=(x2-x1)/3.0
      dy=(y2-y1)/3.0
      ex=-dy
      ey=dx
      call plot(x1, y1, 3)
      xx=x1+dx
      yy=y1+dy
      call plot(xx, yy, 2)
      call stroke
      call linewidth(2.0*w)
      xx=x1+dx+3.0*ex
```

```
     yy=y1+dy+3.0*ey
     call plot(xx, yy, 3)
     xx=x1+dx-3.0*ex
     yy=y1+dy-3.0*ey
     call plot(xx, yy, 2)
     call stroke
     xx=x1+2.0*dx+ex
     yy=y1+2.0*dy+ey
     call plot(xx, yy, 3)
     xx=x1+2.0*dx-ex
     yy=y1+2.0*dy-ey
     call plot(xx, yy, 2)
     call stroke
     call linewidth(w)
     call plot(x2, y2, 3)
     xx=x2-dx
     yy=y2-dy
     call plot(xx, yy, 2)
     call stroke
     call linety(1)
    return
    end
    subroutine coil(x1,y1,x2,y2,d,n,w)
     s=sqrt((x2-x1)**2+(y2-y1)**2)
     call linewidth(w)
     ex=(x2-x1)/s
     ey=(y2-y1)/s
     a=2.0
     ds=(s-2.0*d)/(float(n)+0.5)
     call plot(x1, y1, 3)
     do 10 i=1, 40*n+20
      t=2.0*3.14159*float(i)/40.0
      xx=x1+((d-d*cos(t))+t/2.0/3.14159*ds)*ex-a*d*sin(t)*ey
      yy=y1+((d-d*cos(t))+t/2.0/3.14159*ds)*ey+a*d*sin(t)*ex
      call plot(xx, yy, 2)
10   continue
     call stroke
     call linety(1)
    return
    end
    subroutine text(x,y,n,string)
     character*80 string
     call plot(x, y, 3)
     write(1,*) '(',string(1:n),') show'
     call stroke
    return
    end
```

```
 subroutine textx(x,y,n,string)
 character*12 string
  call plot(x, y, 3)
  write(1,*) -n/2.0*9.0,-3.0/2.0*15.0, ' rmoveto '
  write(1,*) '(',string(1:n),') show'
 return
 end
 subroutine texty(x,y,n,string)
 character*12 string
  call plot(x, y, 3)
  write(1,*) -(n+1)*9.0,-1.0/2.0*12.0, ' rmoveto '
  write(1,*) '(',string(1:n),') show'
 return
 end
 subroutine setchar(ichar,ip)
 if (ichar.eq.1) then
  write(1,*) '/Times-Roman findfont',ip,' scalefont setfont'
  else if (ichar.eq.2) then
  write(1,*) '/Times-Bold findfont',ip,' scalefont setfont'
  else if (ichar.eq.3) then
  write(1,*) '/Times-Italic findfont',ip,' scalefont setfont'
  else if (ichar.eq.4) then
  write(1,*) '/Times-BoldItalic findfont',ip,' scalefont setfont'
  else if (ichar.eq.5) then
  write(1,*) '/Helvetica findfont',ip,' scalefont setfont'
  else if (ichar.eq.6) then
  write(1,*) '/Helvetica-Bold findfont',ip,' scalefont setfont'
  else if (ichar.eq.7) then
  write(1,*) '/Helvetica-Oblique findfont',ip,' scalefont setfont'
  else if (ichar.eq.8) then
  write(1,*) '/Helvetica-BoldOblique findfont',ip,
&  ' scalefont setfont'
  else if (ichar.eq.9) then
  write(1,*) '/Courier findfont',ip,' scalefont setfont'
  else if (ichar.eq.10) then
  write(1,*) '/Courier-Bold findfont',ip,' scalefont setfont'
  else if (ichar.eq.11) then
  write(1,*) '/Courier-Oblique findfont',ip,' scalefont setfont'
  else if (ichar.eq.12) then
  write(1,*) '/Courier-BoldOblique findfont',ip,
&  ' scalefont setfont'
  else if (ichar.eq.13) then
  write(1,*) '/Symbol findfont',ip,' scalefont setfont'
  end if
 return
 end
```

B　C ライブラリー (psbasic.cpp)

紙の都合上、右端までに書ききれなかった行は、'[継続行]' として折り返してすぐ下の行に書いた。

```
#include "ps.h"

//papersize
double xsize, ysize;

//viewpoint
double xv1, yv1, xv2, yv2;

//world coordinate
double xw1, yw1, xw2, yw2;

FILE* stream;

// plotting routines for general purpose using postscript

void init(void)
{ stream = fopen("temp1.ps", "w");

  // aspect ratio of the paper is 1.4143
    fprintf(stream, "%%! Created by Jiro Mizushima \n");
    fprintf(stream, "/Times-Roman findfont 18 scalefont setfont \n");
    fprintf(stream, "/cm {28.35 mul} def \n");

  xsize = 21.0;/* ysize = 29.7;*/ ysize = 21.0;
  viewport(0.2, 0.2, 0.8, 0.8);
  xyworld(0.0, 0.0, 1.0, 1.0);
  linety(1); linewidth(1);
  return;
}

void viewport(double xv1d, double yv1d, double xv2d, double yv2d)
{
  xv1 = xv1d; yv1 = yv1d; xv2 = xv2d; yv2 = yv2d;
  return;
}

void xyworld(double xw1d, double yw1d, double xw2d, double yw2d)
{
  xw1 = xw1d; yw1 = yw1d; xw2 = xw2d; yw2 = yw2d;
  return;
}

void fin(void)
{
```

```
  stroke(); fprintf(stream, "showpage\n"); fclose(stream);
  return;
}

void linety(int ichar)
{
  switch (ichar) {
    case 1: fprintf(stream, "[] 0 setdash \n"); break;
    case 2: fprintf(stream, "[2 2] 0 setdash \n");break;
    case 3: fprintf(stream, "[8 2] 0 setdash \n");break;
    case 4: fprintf(stream, "[8 1 1 1] 0 setdash \n");break;
  }
  return;
}

void linewidth(double w)
{
  fprintf(stream, "%lf setlinewidth\n", w); return;
}

void setgray(double g)
{
  fprintf(stream, "%lf setgray\n", g); return;
}

void setrgb(double r,double g,double b)
{
  fprintf(stream, "%lf %lf %lf setrgbcolor\n",r,g,b); return;
}

void newpath(void)
{
  fprintf(stream, "newpath\n"); return;
}

void closepath(void)
{
  fprintf(stream, "closepath\n"); return;
}

void fill(void)
{
  fprintf(stream, "fill\n"); return;
}

void stroke(void)
{
```

```
  fprintf(stream, "stroke\n");
}

void rotate(int itheta)
{
  fprintf(stream, "%d rotate\n",itheta);
}

void scale(double tx, double ty)
{
  fprintf(stream, "%lf %lf scale\n",tx, ty);
}

void transrate(double x, double y)
{ double x1,y1;
  x1 = ((x-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  y1 = ((y-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  fprintf(stream, "%lf cm %lf cm translate\n", x1, y1);
}

void clipon(double x1, double y1, double x2, double y2)
{
  newpath();
  plot(x1, y1, 3);plot(x2, y1, 2);plot(x2, y2,2); plot(x1, y2, 2);
  closepath();
  fprintf(stream, "clip\n");
}

void eoclipon(double x1, double y1, double x2, double y2)
{
  newpath();
  plot(x1, y1, 3);plot(x2, y1, 2);plot(x2, y2,2); plot(x1, y2, 2);
  closepath();
  fprintf(stream, "eoclip\n");
}

void clipoff(void)
{
  fprintf(stream, "clip\n");
}

void plot(double x, double y, int ipen)
{
  double x1, y1;

  x1 = ((x-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  y1 = ((y-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
```

```
  switch (ipen) {
    case 3: fprintf(stream, "%lf cm %lf cm moveto\n", x1,y1);
      break;
    case 2: fprintf(stream, "%lf cm %lf cm lineto\n", x1,y1);
      break;
     }
  return;
}

void line(double x, double y, double xx, double yy, double g, double w, int it)
{
  double x1, y1, xx1, yy1;

  x1 = ((x-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  y1 = ((y-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  xx1 = ((xx-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  yy1 = ((yy-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  linewidth(w);linety(it);setgray(g);
  fprintf(stream, "%lf cm %lf cm moveto\n", x1, y1);
  fprintf(stream, "%lf cm %lf cm lineto\n", xx1, yy1);
  stroke();
  linety(1); linewidth(1);setgray(0.0);
  return;
}

void line1(double x, double y, double xx, double yy)
{
  double x1, y1, xx1, yy1;

  x1 = ((x-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  y1 = ((y-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  xx1 = ((xx-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  yy1 = ((yy-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  fprintf(stream, "%lf cm %lf cm moveto\n", x1, y1);
  fprintf(stream, "%lf cm %lf cm lineto\n", xx1, yy1);
  stroke();
  return;
}


void rect(double x1, double y1, double x2, double y2, double g, double w, int it)
{
  linewidth(w);linety(it);setgray(g);
  newpath();plot(x1,y1,3);plot(x2,y1,2);plot(x2,y2,2);plot(x1,y2,2);
  closepath();
  if(g!=0.0) fprintf(stream, " fill\n");
  stroke();
```

```
  linety(1); linewidth(1);setgray(0.0);
  return;
}

void rect1(double x1, double y1, double x2, double y2)
{
  newpath();plot(x1,y1,3);plot(x2,y1,2);plot(x2,y2,2);plot(x1,y2,2);
  closepath();
  stroke();
  return;
}

void rectfill1(double x1, double y1, double x2, double y2)
{
  newpath();plot(x1,y1,3);plot(x2,y1,2);plot(x2,y2,2);plot(x1,y2,2);
  closepath();
  fprintf(stream, " fill\n");
  stroke();
  return;
}

void circ(double x1, double y1, double r1, double g, double w, int it)
{
  double xx1,yy1,rr1;
  xx1 = ((x1-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  yy1 = ((y1-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  rr1 = (r1-xw1)/(xw2-xw1)*(xv2-xv1)*xsize;
  linewidth(w);linety(it);setgray(g);
  newpath();
  fprintf(stream, "%lf cm %lf cm %lf cm 0 360 arc\n", xx1, yy1, rr1);
  if(g!=0.0) fprintf(stream, " fill\n");
  stroke();
  linety(1); linewidth(1);setgray(0.0);
  return;
}

void circ1(double x1, double y1, double r1)
{
  double xx1,yy1,rr1;
  xx1 = ((x1-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  yy1 = ((y1-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  rr1 = (r1-xw1)/(xw2-xw1)*(xv2-xv1)*xsize;
  newpath();
  fprintf(stream, "%lf cm %lf cm %lf cm 0 360 arc\n", xx1, yy1, rr1);
  stroke();
  return;
}
```

```
void circfill1(double x1, double y1, double r1)
{
  double xx1,yy1,rr1;
  xx1 = ((x1-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  yy1 = ((y1-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  rr1 = (r1-xw1)/(xw2-xw1)*(xv2-xv1)*xsize;
  newpath();
  fprintf(stream, "%lf cm %lf cm %lf cm 0 360 arc\n", xx1, yy1, rr1);
  fprintf(stream, " fill\n");
  stroke();
  return;
}

void ellipse(double x1, double y1, double rx, double ry, double g, double w, int it)
{
      double x, y, dt; int i;

      linety(it); linewidth(w); setgray(g);
      newpath();
      x=x1+rx; y=y1; dt=3.14156*2.0/double(20);
      plot(x,y,3);
      for (i=1;i<=20;i++) {
             x=x1+rx*cos(dt*i);y=y1+ry*sin(dt*i); plot(x,y,2);
             }
      if(g!=0.0) {fprintf(stream, "fill\n");}
      stroke();
      linety(1); linewidth(1.0); setgray(0.0); return;
}

void ellipse1(double x1, double y1, double rx, double ry)
{
      double x, y, dt; int i;

      newpath();
      x=x1+rx; y=y1; dt=3.14156*2.0/double(20);
      plot(x,y,3);
      for (i=1;i<=20;i++) {
             x=x1+rx*cos(dt*i);y=y1+ry*sin(dt*i); plot(x,y,2);
             }
      stroke();
}

void ellipsefill1(double x1, double y1, double rx, double ry)
{
      double x, y, dt; int i;
```

```
      newpath();
      x=x1+rx; y=y1; dt=3.14156*2.0/double(20);
      plot(x,y,3);
      for (i=1;i<=20;i++) {
             x=x1+rx*cos(dt*i);y=y1+ry*sin(dt*i); plot(x,y,2);
             }
      fprintf(stream, "fill\n");
      stroke();
}

void arc(double x1, double y1, double r1, double t1
              [継続行]    , double t2, double g, double w, int it)
{
   double xx1,yy1,rr1;

   xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
   yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
   rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize;
   linety(it); linewidth(w); setgray(g);
   newpath();
   fprintf(stream,"%lf cm %lf cm %lf cm %lf %lf arc \n",xx1,yy1,rr1,t1,t2);
   stroke(); linety(1); linewidth(1.0); setgray(0.0);
   return;
}

void arc1(double x1, double y1, double r1, double t1, double t2)
{
   double xx1,yy1,rr1;

   xx1=((x1-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
   yy1=((y1-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
   rr1=(r1/(xw2-xw1)*(xv2-xv1))*xsize;
   newpath();
   fprintf(stream,"%lf cm %lf cm %lf cm %lf %lf arc \n",xx1,yy1,rr1,t1,t2);
   return;
}

void curv(double x1, double y1, double x2, double y2, double x3, double y3
              [継続行]    , double x4, double y4, double g, double w, int it)
{
      double xx2,yy2,xx3,yy3,xx4,yy4;

     xx2=((x2-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
     yy2=((y2-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
     xx3=((x3-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
     yy3=((y3-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
     xx4=((x4-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
```

```
     yy4=((y4-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
     linety(it); linewidth(w); setgray(g);
     newpath(); plot(x1,y1,3);
     fprintf(stream, "%lf cm %lf cm %lf cm %lf cm %lf cm %lf cm
              [継続行]    curveto\n", xx2 ,yy2,xx3,yy3,xx4,yy4);
      stroke();
      linety(1);linewidth(1.0); setgray(0.0);
      return;
}

void curv1(double x1, double y1, double x2, double y2
              [継続行]    , double x3, double y3, double x4, double y4)
{
      double xx2,yy2,xx3,yy3,xx4,yy4;

     xx2=((x2-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
     yy2=((y2-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
     xx3=((x3-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
     yy3=((y3-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
     xx4=((x4-xw1)/(xw2-xw1)*(xv2-xv1)+xv1)*xsize;
     yy4=((y4-yw1)/(yw2-yw1)*(yv2-yv1)+yv1)*ysize;
     newpath(); plot(x1,y1,3);
     fprintf(stream, "%lf cm %lf cm %lf cm %lf cm %lf cm %lf cm
              [継続行]    curveto\n", xx2 ,yy2,xx3,yy3,xx4,yy4);
      stroke();
      return;
}

void triangl(double x1, double y1, double r1, double a1
              [継続行]    , double g, double w, int it)
{
    double aa1,aa2;

     aa1=a1*3.14159/180.0;
     aa2=aa1+3.14159/3.0;
     linety(it); linewidth(w); setgray(g);
     newpath(); plot(x1,y1,3); plot(x1+r1*cos(aa1), y1+r1*sin(aa1), 2);
     plot(x1+r1*cos(aa2), y1+r1*sin(aa2), 2);
     closepath();
       if(g!=0.0) {fprintf(stream, "fill\n");}
     stroke();
     linety(1); linewidth(1.0); setgray(0.0);
     return;
}

void triangl1(double x1, double y1, double r1, double a1)
{
```

```
    double aa1,aa2;

     aa1=a1*3.14159/180.0;
     aa2=aa1+3.14159/3.0;
     newpath(); plot(x1,y1,3); plot(x1+r1*cos(aa1), y1+r1*sin(aa1), 2);
     plot(x1+r1*cos(aa2), y1+r1*sin(aa2), 2);
     closepath();
     stroke();
     return;
}

void trianglfill1(double x1, double y1, double r1, double a1)
{
    double aa1,aa2;

     aa1=a1*3.14159/180.0;
     aa2=aa1+3.14159/3.0;
     newpath(); plot(x1,y1,3); plot(x1+r1*cos(aa1), y1+r1*sin(aa1), 2);
     plot(x1+r1*cos(aa2), y1+r1*sin(aa2), 2);
     closepath();
     fprintf(stream, "fill\n");
     stroke();
     return;
}



void spline(double x1, double y1, double x2, double y2, double x3, double y3,
              [継続行]    double x4, double y4, int ipart, double g, double w)
{ int ns=10,i; double u,x,y;

  linewidth(w); setgray(g);
  newpath();
  u=double(ipart);
  x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
              [継続行]    -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4;
  y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2-(u+1.)*u*(u-2.)/2.*y3
              [継続行]    +(u+1.)*u*(u-1.)/6.*y4;
  plot(x,y,3);
   for (i=1;i<=ns;i++){
    u=double(ipart)+double(i)/double(ns);
    x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
              [継続行]    -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4;
    y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2
              [継続行]    -(u+1.)*u*(u-2.)/2.*y3+(u+1.)*u*(u-1.)/6.*y4;
    plot(x,y,2);
    }
   stroke();
```

```
   linewidth(1.0); setgray(0.0);
   return;
}

void spline1(double x1, double y1, double x2, double y2, double x3, double y3,
              [継続行]    double x4, double y4, int ipart)
{ int ns=10,i; double u,x,y;

  newpath();
  u=double(ipart);
  x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
              [継続行]    -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4;
  y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2
              [継続行]    -(u+1.)*u*(u-2.)/2.*y3+(u+1.)*u*(u-1.)/6.*y4;
  plot(x,y,3);
   for (i=1;i<=ns;i++){
    u=double(ipart)+double(i)/double(ns);
    x=-u*(u-1.)*(u-2.)/6.*x1+(u+1.)*(u-1.)*(u-2.)/2.*x2
              [継続行]    -(u+1.)*u*(u-2.)/2.*x3+(u+1.)*u*(u-1.)/6.*x4;
    y=-u*(u-1.)*(u-2.)/6.*y1+(u+1.)*(u-1.)*(u-2.)/2.*y2
              [継続行]    -(u+1.)*u*(u-2.)/2.*y3+(u+1.)*u*(u-1.)/6.*y4;
    plot(x,y,2);
    }
   stroke();
   return;
}

void parabola(double x1, double y1, double x2, double y2, double x3, double y3,
              [継続行]     double g, double w, int it)
{
      double u,x,y; int i,ns=20; u=0.0;
      linewidth(w); setgray(g); linety(it);
      newpath();
      x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3;
      y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3;
      plot(x,y,3);
      for (i=1; i<=2*ns; i++)
        { u=double(i)/double(ns);
          x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3;
          y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3;
          plot(x,y,2);
         } stroke();
      linewidth(1.0); setgray(0.0); linety(1);
      return;
}

void parabola1(double x1, double y1, double x2, double y2, double x3, double y3)
```

```
{
      double u,x,y; int i,ns=20; u=0.0;
      newpath();
      x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3;
      y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3;
      plot(x,y,3);
      for (i=1; i<=2*ns; i++)
        { u=double(i)/double(ns);
          x=(u-1.)*(u-2.)/2.*x1-u*(u-2.)*x2+u*(u-1.)/2.*x3;
          y=(u-1.)*(u-2.)/2.*y1-u*(u-2.)*y2+u*(u-1.)/2.*y3;
          plot(x,y,2);
         } stroke();
      return;
}

void arrow(double x1, double y1, double x2, double y2,
              [継続行]    double d, double g, double w)
{
   double alpha=2.2,s,x3,y3,x4,y4;
   s=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
   x3=x2-(x2-x1)/s*d*alpha-(y2-y1)/s*d;
   y3=y2-(y2-y1)/s*d*alpha+(x2-x1)/s*d;
   x4=x2-(x2-x1)/s*d*alpha+(y2-y1)/s*d;
   y4=y2-(y2-y1)/s*d*alpha-(x2-x1)/s*d;
   linewidth(w); setgray(g);
   plot(x1, y1, 3); plot(x2, y2, 2);
   plot(x2, y2, 3); plot(x3, y3, 2); plot(x2, y2, 3); plot(x4, y4, 2);
   stroke(); linety(1); linewidth(1.0); setgray(0.0);
   return;
}

void arrow1(double x1, double y1, double x2, double y2, double d)
{
   double alpha=2.2,s,x3,y3,x4,y4;
   s=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
   x3=x2-(x2-x1)/s*d*alpha-(y2-y1)/s*d;
   y3=y2-(y2-y1)/s*d*alpha+(x2-x1)/s*d;
   x4=x2-(x2-x1)/s*d*alpha+(y2-y1)/s*d;
   y4=y2-(y2-y1)/s*d*alpha-(x2-x1)/s*d;
   plot(x1, y1, 3); plot(x2, y2, 2);
   plot(x2, y2, 3); plot(x3, y3, 2); plot(x2, y2, 3); plot(x4, y4, 2);
   stroke();
   return;
}


void resist(double x1, double y1, double x2, double y2, double d, double w)
```

```
{
   double s, dx, dy, ex, ey, xx, yy;int i;
      s=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
      linewidth(w);
      dx=(x2-x1)/12.0; dy=(y2-y1)/12.0;
      ex=-d*dy; ey=d*dx;
       plot(x1, y1, 3); xx=x1+dx+ex; yy=y1+dy+ey; plot(xx, yy, 2);
      for(i=1;i<=5; i++){
        xx=x1+(i*2+1)*dx+pow(-1,i)*ex; yy=y1+(i*2+1)*dy+pow(-1,i)*ey;
        plot(xx, yy, 2);
        }
      plot(x2, y2, 2);
      stroke(); linety(1); linewidth(1.0);
     return;
}

void battery(double x1, double y1, double x2, double y2, double d, double w)
{
    double s,dx,dy,ex,ey,xx,yy;

    s=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
    linewidth(w);
     dx=(x2-x1)/3.0; dy=(y2-y1)/3.0;
     ex=-dy; ey=dx;
     plot(x1, y1, 3); xx=x1+dx; yy=y1+dy; plot(xx, yy, 2); stroke();
    linewidth(2.0*w);
     xx=x1+dx+3.0*ex; yy=y1+dy+3.0*ey;
      plot(xx, yy, 3); xx=x1+dx-3.0*ex; yy=y1+dy-3.0*ey; plot(xx, yy, 2); stroke();
     xx=x1+2.0*dx+ex; yy=y1+2.0*dy+ey;
      plot(xx, yy, 3); xx=x1+2.0*dx-ex; yy=y1+2.0*dy-ey; plot(xx, yy, 2); stroke();
    linewidth(w);
      plot(x2, y2, 3); xx=x2-dx; yy=y2-dy; plot(xx, yy, 2); stroke();
    linewidth(1.0); linety(1);
    return;
}

void coil(double x1, double y1, double x2, double y2, double d, int n, double w)
{
   double s,ex,ey,ds,a,xx,yy,t;int i;

   s=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
   linewidth(w);
   ex=(x2-x1)/s; ey=(y2-y1)/s;
   a=2.0; ds=(s-2.0*d)/(double(n)+0.5);
     plot(x1, y1, 3);
     for(i=1;i<=40*n+20;i++) {
      t=2.0*3.14159*double(i)/40.0;
```

```
      xx=x1+((d-d*cos(t))+t/2.0/3.14159*ds)*ex-a*d*sin(t)*ey;
      yy=y1+((d-d*cos(t))+t/2.0/3.14159*ds)*ey+a*d*sin(t)*ex;
      plot(xx, yy, 2);
      }  stroke();
    linewidth(1.0);linety(1);
    return;
}

void text(double x, double y, int n, char* str1)
{
  double x1, y1;

  x1 = ((x-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  y1 = ((y-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
    fprintf(stream, "%lf cm %lf cm moveto\n", x1, y1);
    fprintf(stream, "(%s) show\n", str1);
  return;
}

void textx(double x, double y, int n, char* str1)
{
      plot(x, y, 3);
      fprintf(stream, "%lf %lf rmoveto\n", -n/2.0*9.0,-3.0/2.0*15.0);
      fprintf(stream, "(%s) show\n", str1);
      return;
}

void  texty(double x, double y, int n, char* str1)
{
      plot(x, y, 3);
       fprintf(stream, "%lf %lf rmoveto\n", -(n+1)*9.0,-1.0/2.0*12.0);
      fprintf(stream, "(%s) show\n", str1);
      return;
}

void setchar(int ichar, int ip)
{
 switch (ichar) {
  case 1:fprintf(stream, "/Times-Roman findfont %d scalefont setfont \n",ip);
      break;
  case 2:fprintf(stream, "/Times-Bold findfont %d scalefont setfont \n",ip);
      break;
  case 3:fprintf(stream, "/Times-Italic findfont %d scalefont setfont \n",ip);
      break;
  case 4:fprintf(stream, "/Times-BoldItalic findfont %d scalefont setfont \n",ip);
      break;
  case 5:fprintf(stream, "/Helvetica findfont %d scalefont setfont \n",ip);
```

```
      break;
  case 6:fprintf(stream, "/Helvetica-Bold findfont %d scalefont setfont \n",ip);
      break;
  case 7:fprintf(stream, "/Helvetica-Oblique findfont %d scalefont setfont \n",ip);
      break;
  case 8:fprintf(stream, "/Helvetica-BoldOblique findfont %d scalefont setfont \n",ip);
      break;
  case 9:fprintf(stream, "/Courier findfont %d scalefont setfont \n",ip);
      break;
  case 10:fprintf(stream, "/Courier-Bold findfont %d scalefont setfont \n",ip);
      break;
  case 11:fprintf(stream, "/Courier-Oblique findfont %d scalefont setfont \n",ip);
      break;
  case 12:fprintf(stream, "/Courier-BoldOblique findfont %d scalefont setfont \n",ip);
      break;
  case 13:fprintf(stream, "/Symbol findfont %d scalefont setfont \n",ip);
      break;
   }
  return;
}

void square(double x, double y)
{
  double x1, y1, delta;

  x1 = ((x-xw1)/(xw2-xw1)*(xv2-xv1) + xv1)*xsize;
  y1 = ((y-yw1)/(yw2-yw1)*(yv2-yv1) + yv1)*ysize;
  delta = 0.1;
  fprintf(stream, "%lf cm %lf cm moveto\n", x1-delta,y1-delta);
  fprintf(stream, "%lf cm %lf cm lineto\n", x1+delta,y1-delta);
  fprintf(stream, "%lf cm %lf cm moveto\n", x1+delta,y1+delta);
  fprintf(stream, "%lf cm %lf cm lineto\n", x1-delta,y1+delta);
  closepath();stroke();
  return;
}
```

C　作図プログラム 1

C.1　Fortran(fig1.for)

```
C     Plotting program for general purpose using postscript
      call init
      call readdata
      call viewport(0.2,0.2,0.8,0.8)
      call xyworld(0.0,0.0,1.0,1.0)
      call frame
      call plotd
      call fin
      stop
      end
      subroutine readdata
      common /maxmin/xmax,xmin,ymax,ymin,n
      common /dataxy/x1(100),y1(100)
      open (2,file='pg.data',status='OLD')
      do 10 i=1,100
        read(2,*,end=99) x1(i),y1(i)
   10 continue
   99 continue
      n=i-1
      xmax=x1(1)
      xmin=x1(1)
      ymax=y1(1)
      ymin=y1(1)
      do 20 i=2,n
       if(xmax.lt.x1(i)) xmax=x1(i)
       if(xmin.gt.x1(i)) xmin=x1(i)
       if(ymax.lt.y1(i)) ymax=y1(i)
       if(ymin.gt.y1(i)) ymin=y1(i)
   20 continue
      return
      end
      subroutine plotd
      common /maxmin/xmax,xmin,ymax,ymin,n
      common /dataxy/x1(100),y1(100)
      call linewidth(1.5)
      call linety(1)
      do 20 i=1,n
       x1(i)=(x1(i)-xmin)/(xmax-xmin)
       y1(i)=(y1(i)-ymin)/(ymax-ymin)
   20 continue
      call plot(x1(1),y1(1),3)
      do 30 i=2,n
        call plot(x1(i),y1(i),2)
   30 continue
```

```
    call stroke()
    call plot(0.9,-0.06,3)
    write(1,100) xmax
    call plot(0.4,-0.06,3)
    write(1,100) (xmax+xmin)/2.0
    call plot(-0.1,-0.06,3)
    write(1,100) xmin
    call plot(-0.2,0.98,3)
    write(1,100) ymax
    call plot(-0.2,0.48,3)
    write(1,100) (ymax+ymin)/2.0
    call plot(-0.2,-0.02,3)
    write(1,100) ymin
100 format(1x,'(',1pe8.1,') show')
    continue
    return
    end
    subroutine frame
    common /maxmin/xmax,xmin,ymax,ymin,n
    common /dataxy/x1(100),y1(100)
    call linewidth(1.0)
    call rect(0.0,0.0,1.0,1.0,0.0,1.0,1)
    call textx(0.75,0.0,1,'x')
    nx=2
    do 10 ix=1,nx-1
     x=float(ix)/float(nx)
     dy=0.03
     call plot(x,0.0,3)
     call plot(x,dy,2)
     call stroke()
 10 continue
    call texty(0.0,0.75,1,'y')
    ny=2
    do 20 iy=1,ny-1
     y=float(iy)/float(ny)
     dx=0.03
     call plot(0.0,y,3)
     call plot(dx,y,2)
     call stroke()
 20 continue
    return
    end
```

C.2 C(fig1.cpp)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "ps.h"

int readdata(double*,double*,double*,double*);
void frame(int,double*,double*,double*,double*);
void plotd(int,double*,double*,double*,double*);

void main()
{
   int n;
   double xymax[2],xymin[2],x1[100],y1[100];
   init();
   n=readdata(x1,y1,xymax,xymin);
   viewport(0.2,0.2,0.8,0.8); xyworld(0.0,0.0,1.0,1.0);
   frame(n,x1,y1,xymax,xymin);
   plotd(n,x1,y1,xymax,xymin);
   fin();
}

int readdata(double x1[], double y1[], double xymax[], double xymin[])
{
      FILE* stream;int i;char sx[10],sy[10];
      stream=fopen("pg.data","r");
      int n=0;
      while(!feof(stream)) {
        fscanf(stream,"%s%s",sx,sy);
        x1[n]=(double)atof(sx);y1[n]=(double)atof(sy);n++;
        }
      n--;

      xymax[0]=x1[0];xymax[1]=y1[0];
      xymin[0]=x1[0];xymin[1]=y1[0];
      for(i=1;i<n;i++) {
         if(xymax[0]<x1[i]) xymax[0]=x1[i];
         if(xymin[0]>x1[i]) xymin[0]=x1[i];
         if(xymax[1]<y1[i]) xymax[1]=y1[i];
         if(xymin[1]>y1[i]) xymin[1]=y1[i];
         }
      fclose(stream);
      return (n);
}
void plotd(int n,double x1[], double y1[], double xymax[], double xymin[])
{
```

```
      int i;char s[10];
      linewidth(1.5);linety(1);
      for(i=0;i<n;i++) {
        x1[i]=(x1[i]-xymin[0])/(xymax[0]-xymin[0]);
        y1[i]=(y1[i]-xymin[1])/(xymax[1]-xymin[1]);
        }
      plot(x1[0],y1[0],3);
       for(i=1;i<n;i++){ plot(x1[i],y1[i],2); } stroke();

      _gcvt(xymax[0],5,s);text(0.95,-0.06,5,s);
      _gcvt((xymax[0]+xymin[0])/2.0,5,s);text(0.45,-0.06,5,s);
      _gcvt(xymin[0],5,s);text(-0.05,-0.06,5,s);
      _gcvt(xymax[1],5,s);text(-0.1,0.98,5,s);
      _gcvt((xymax[1]+xymin[1])/2.0,5,s);text(-0.1,0.48,5,s);
      _gcvt(xymin[1],5,s);text(-0.1,-0.02,5,s);
}
void frame(int n, double x1[], double y1[], double xymax[], double xymin[])
{

      int nx=2,ny=2,ix,iy;double x,dx=0.03,y,dy=0.03;
      linewidth(1.0);rect(0.0,0.0,1.0,1.0,0.0,1.0,1);
      textx(0.75,0.0,1,"x");
      for(ix=1;ix<nx;ix++) {
       x=((double)ix)/((double)nx);
       plot(x,0.0,3); plot(x,dy,2);stroke();
       }
      texty(0.0,0.75,1,"y");
      for(iy=1;iy<ny;iy++) {
       y=((double)iy)/((double)ny);
       plot(0.0,y,3);plot(dx,y,2);stroke();
       }
}
```

C.3 ヘッダファイル (ps.h)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define PI 3.141592653585793

void init(void);
void viewport(double, double, double, double);
void xyworld(double, double, double, double);
void fin(void);
void linety(int);
void linewidth(double);
void setgray(double);
void setrgb(double, double, double);
void newpath(void);
void fill(void);
void closepath(void);
void stroke(void);
void rotate(int);
void scale(double, double);
void transrate(double, double);
void clipon(double, double, double, double);
void eoclipon(double, double, double, double);
void clipoff(void);
void plot(double, double, int);
void line(double, double, double, double, double, double, int);
void line1(double, double, double, double);
void rect(double, double, double, double, double, double, int);
void rect1(double, double, double, double);
void rectfill1(double, double, double, double);
void circ(double, double, double, double, double, int);
void circ1(double, double, double);
void circfill1(double, double, double);
void ellipse(double, double, double, double, double, double, int);
void ellipse1(double, double, double, double);
void ellipsefill1(double, double, double, double);
void arc(double, double, double, double, double, double, double, int);
void arc1(double, double, double, double, double);
void curv(double, double, double, double, double, double, double, double, double, double, int);
void curv1(double, double, double, double, double, double, double, double);
void triangl(double, double, double, double, double, double, int);
void triangl1(double, double, double, double);
void trianglfill1(double, double, double, double);
void spline(double, double, double, double, double, double, double, double, int, double, double);
void spline1(double, double, double, double, double, double, double, double, int);
void parabola(double, double, double, double, double, double, double, double, int);
```

```
void parabola1(double, double, double, double, double, double);
void arrow(double, double, double, double, double, double, double);
void arrow1(double, double, double, double, double);
void resist(double, double, double, double, double, double);
void battery(double, double, double, double, double, double);
void coil(double, double, double, double, double, int, double);
void text(double, double, int, char* );
void textx(double, double, int, char* );
void texty(double, double, int, char* );

void setchar(int, int);
void square(double, double);
```

参考文献

[1] Adobe Systems: “Post Script チュートリアル & クックブック” アスキー出版局, 1989 年.

[2] 安居院猛、永江孝規: “Post Script グラフィックス” 新紀元社, 1993 年.

[3] 江口庄英: “Ghostscript – Another Manual” ソフトバンク, 1997 年.

[4] 乙部厳己、江口庄英: “pLaTex 2e for WINDOWS Another Manual‘’ ソフトバンク, 1996 年.

PostScript	– Fortran, C からの利用法 –
発行年	2001 年 9 月 8 日
編集	同志社大学工学部水島研究室
発行責任者	水島二郎
発行所	茫洋出版