Computer Graphics PLOTTER

Masao Kodama
mkodama@mable.ne.jp

Contents

1 Introduction **2**

1.1 Characteristics of Plotter 2

1.2 Software and hardware necessary to use Plotter 3

2 The subroutines of Plotter **3**

2.1 pfnbegin: beginning Plotter 4

2.2 pfnend: ending Plotter 4

2.3 pfsorigin: setting the origin 4

2.4 pfscolor: setting the color 5

2.5 plstype: setting the line type 5

2.6 plswidth: setting the line width 6

2.7 plsframe: setting the clipping frame 6

2.8 plsclip: setting the clipping indicator 6

2.9 pldline: drawing a polygonal line 6

2.10 pldsgmnt: drawing a segment 7

2.11 pldarrow: drawing an arrowhead 7

2.12 pldcrcl: drawing a circle 7

2.13 pldcrclarc: drawing a circular arc 7

2.14 pldarcarrow: drawing an arrowhead on a circular arc 7

2.15 pldrctgl: drawing a rectangle 8

2.16 pldcross: drawing a cross 8

2.17 pldhatching: drawing hatching 8

2.18 prprctgl: painting a rectangular region 8

2.19 prpcrcl: painting a circular region 8

2.20 prpplygn: painting a polygonal region 9

2.21 pcsheight: setting the height of character patterns 9

2.22 pcspstn: setting the internal position of character patterns 9

2.23 pcsangle: setting the slant angle of character patterns 9

2.24 pcwrite: writing a character pattern 10

3 Sample programs for application of Plotter 13

3.1 File sample1.f90: application of every subroutine 13

3.2 File sample2.f90: a figure of scattering of a plane wave 14

3.3 File sample3.f90: a figure of a parallel plate condenser 14

3.4 File sample4.f90: a graph whose y-axis has an anti-logarithmic scale 14

3.5 File sample5.f90: a graph whose y-axis has a logarithmic scale 15

3.6 File sample6.f90: a figure of 2-dimensional temperature distribution 16

1 Introduction

1.1 Characteristics of Plotter

Computer graphics Plotter is an interface between Fortran 90 and the PostScript language. Plotter provides 24 Fortran subroutines. With a Fortran program including these subroutines, we make a figure. The kernel program of Plotter interprets the Fortran program and expresses the figure with the PostScript language.

We actually know many application programs for the computer graphics, that is, we know GKS, PLOT, gnuplot and so forth. Computer graphics Plotter has the following characteristics.

(1) The computer graphics Plotter is suitable for drawing 2-dimensional chromatic stationary figures that are used for illustrations of papers and books for technology.

(2) Since the kernel program of Plotter is written in Fortran 90, Plotter can work under any OS for example, Windows, UNIX or LINUX, and Plotter can work under any Fortran 90 compiler.

(3) Since the pictures outputted from Plotter are expressed by page description language PostScript, the pictures can be inserted to documents made by Tex.

(4) Since Plotter works under Fortran 90 programs, in graphs or diagrams Plotter can show results computed by a Fortran program instantly if some statements necessary to Plotter are added to this Fortran program.

(5) The kernel program of Plotter is open and can be modified freely by users.

(6) When drawing a curve for making a graph, we can easily erase the part of the curve that protrudes from the frame set for the graph.

(7) When we write characters, we can control the font of each character, and can freely control the interval between any adjacent two characters. We can easily write superscripts and subscripts on each character.

(8) GKS is an ISO standard and is a computer graphics for many purposes. Programs written in GKS are too complicated. If only the illustrations of papers or books are necessary, Plotter can achieve this purpose with very simple programs. The functions of Plotter are reduced in comparison of GKS, but Plotter has the functions sufficient to draw the illustrations.

(9) In Plotter, the unit of lengths is a millimeter. Every angle in Plotter is measured by radians, and its ground line is the horizontal axis.

(10) Plotter can also provide Kanji that are Japanese characters. The codes for Kanji are sift JIS codes, which are usually used in Windows of the Japanese edition. Hence if we use Kanji in LINUX and UNIX, we must use a converter of codes, which editors usually provide in order to obtain the sift JIS codes.

1.2 Software and hardware necessary to use Plotter

The following software and hardware are necessary to use Plotter.

(1) A Fortran 90 compiler.

(2) Ghostscript. Using software Ghostscript, we can preview the pictures outputted from Plotter on a display. We can download this software from the site:

http://pages.cs.wisc.edu/~ghost/

(3) A PostScript printer when OS is UNIX or LINUX. A usual printer is also available when OS is Windows.

2 The subroutines of Plotter

In this section, the subroutines provided by Plotter are introduced. These subroutines work in Fortran programs. When an error in Plotter occurs, a message will be outputted to the standard output device. If the error is serious, the execution will stop.

2.1 pfnbegin

SUBROUTINE pfnbegin(char)

CHARACTER(LEN=∗), INTENT(IN):: char

Subroutine pfnbegin begins Plotter and opens a file that will store the picture output. Character argument char must store the name of this file. The extension of the file name is .ps. Hence, if char='abc' for example, then the full name of the output file is abc.ps.

2.2 pfnend

SUBROUTINE pfnend

Subroutine pfnend ends Plotter and closes the PostScript file that has been opened by subroutine pfnbegin. In Plotter, more than one PostScript file cannot be open at the same time, but can be open if these files are not open at the same time. For example, the following program is permitted.

```
PROGRAM figures
.....
CALL pfnbegin('fig1')
.....
CALL pfnend                    ! Line 1
.....
CALL pfnbegin('fig2')          ! Line 2
.....
CALL pfnend
.....
END PROGRAM figures
```

In program figures, file fig2.ps can be opened at Line 2 because file fig1.ps was closed at Line 1.

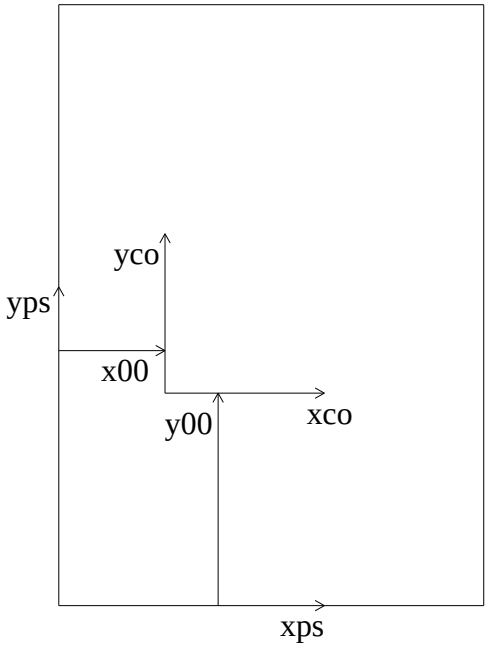
2.3 pfsorigin

SUBROUTINE pfsorigin(x00, y00)

REAL, INTENT(IN):: x00, y00

See Fig. 1. Suppose that the rectangle in Fig. 1 is the boundary of printing in PostScript. Orthogonal coordinates xps and yps are used in PostScript, and their axes are put on sides of the rectangle. The origin of orthogonal coordinates (xco, yco) exists at point xps=x00 and

Figure 1: Suppose that the rectangle in the figure indicates the boundary of printing in PostScript. Orthogonal coordinates xps and yps are used in PostScript, and orthogonal coordinates xco and yco are used in Plotter.



yps=y00. We have the expressions xps=xco+x00 and yps=yco+y00 for arbitrary xco and yco. Every position appearing in Plotter except (x00, y00) in this subroutine is expressed by the orthogonal coordinates (xco, yco). Both the default values of real variables x00 and y00 are set at 0. Subroutine pfsorigin can re-set the values of x00 and y00.

2.4 pfscolor

SUBROUTINE pfscolor(redc, greenc, bluec)

REAL, INTENT(IN):: redc, greenc, bluec

Argument redc indicates the strength of red, and argument greenc indicates the strength of green, and argument bluec indicates the strength of blue. The color of a combination of redc, greenc and bluec is determined by the additive combination of colors. It must be satisfied that $0 \leq$redc≤ 100, $0 \leq$greenc≤ 100 and $0 \leq$bluec≤ 100. The color for redc=greenc=bluec=0 is black, and the color for redc=greenc=bluec=100 is white. Subroutine pfscolor determines the colors of the lines, the regions and the characters outputted from Plotter. All the default values of redc, greenc and bluec are 0. This subroutine re-sets the values of redc, greenc and bluec.

2.5 plstype

SUBROUTINE plstype(line_type)

INTEGER, INTENT(IN):: line_type

Integer line_type sets the line type as follows.

line_type=1: solid line line_type=2: broken line

line_type=3: dotted line line_type=4: chain line

The default value of line_type is set at 1. Subroutine plstype can re-set the value of integer variable line_type.

2.6 plswidth

SUBROUTINE plswidth(width)

REAL, INTENT(IN):: width

Real argument width indicates the line width. The default value of variable width is set at 0.35. Subroutine plswidth can re-set the value of variable width.

2.7 plsframe

SUBROUTINE plsframe(xcomax, ycomax)

REAL, INTENT(IN):: xcomax, ycomax

The clipping frame is a rectangle. The vertexes of the rectangle are at (0., 0.), (xcomax, 0.), (xcomax, ycomax) and (0., ycomax). It must be satisfied that xcomax>0 and ycomax>0. The clipping frame will be used in subroutine plsclip. Both the default values of xcomax and ycomax are set at 80. This subroutine re-sets xcomax and ycomax.

2.8 plsclip

SUBROUTINE plsclip(iclip)

INTEGER, INTENT(IN):: iclip

Integer iclip is the clipping indicator. When iclip=0, there is no clipping. When iclip=1, the lines are clipped outside the clipping frame, which is given by subroutine plsframe. The clipping is effective only for the lines drawn by subroutines pldline, pldsgmnt and pldrctgl. The default value of iclip is set at 0. Subroutine plsclip can re-set the value of iclip.

2.9 pldline

SUBROUTINE pldline(nn, dxx, dyy)

INTEGER, INTENT(IN):: nn

REAL, INTENT(IN):: dxx(∗), dyy(∗)

This subroutine draws a polygonal line. Argument nn is 2 plus the number of broken points of the polygonal line. It must be satisfied that nn≥2. The polygonal line connects the

points {dxx(1),dyy(1)}, {dxx(2),dyy(2)}, $\cdots$ and {dxx(nn),dyy(nn)} by segments. It must be satisfied that ABS{dxx(i)}<30000 and ABS{dyy(i)}< 30000 for i=1, 2, $\cdots$, nn.

2.10 pldsgmnt

SUBROUTINE pldsgmnt(x1, y1, x2, y2)

REAL, INTENT(IN):: x1, y1, x2, y2

This subroutine draws a segment. Both the ends of the segment are put at the positions (x1, y1) and (x2, y2).

2.11 pldarrow

SUBROUTINE pldarrow(x1, y1, x2, y2)

REAL, INTENT(IN):: x1, y1, x2, y2

This subroutine draws an arrowhead. The point of the arrowhead is at (x2, y2). The arrowhead points from (x1, y1) to (x2, y2). It is necessary that ABS(x1−x2)+ABS(y1−y2)>0.

2.12 pldcrcl

SUBROUTINE pldcrcl(x1, y1, rr)

REAL, INTENT(IN):: x1, y1, rr

Subroutine pldcrcl draws a circle. The center of the circle is at (x1, y1). Argument rr is the radius of the circle.

2.13 pldcrclarc

SUBROUTINE pldcrclarc(x1, y1, rr, phi1, phi2)

REAL, INTENT(IN):: x1, y1, rr, phi1, phi2

This subroutine draws a circular arc anticlockwise from the starting angle phi1 to the arriving angle phi2. Point (x1, y1) indicates the center of the circular arc. Real argument rr means the radius of the circular arc. It must be satisfied that phi1<phi2<phi1+2∗pi, ABS(phi1)<4∗pi and ABS(phi2)<4∗pi. Here variable pi is the circle ratio.

2.14 pldarcarrow

SUBROUTINE pldarcarrow(x1, y1, rr, phi1, phi2)

REAL, INTENT(IN):: x1, y1, rr, phi1, phi2

This subroutine draws an arrowhead on a circular arc. The center of the circular arc is at (x1, y1). Argument rr is the radius of the circular arc. Argument phi2 indicates the angle of

the position of the point of the arrowhead. If phi1≤phi2, the arrowhead points anticlockwise. If not, it points clockwise. It must be satisfied that ABS(phi1)<4∗pi and ABS(phi2)<4∗pi.

2.15 pldrctgl

SUBROUTINE pldrctgl(x1, y1, x2, y2)

REAL, INTENT(IN):: x1, y1, x2, y2

This subroutine draws a rectangle. The vertexes of the rectangle are at (x1, y1), (x2, y1), (x2, y2) and (x1, y2).

2.16 pldcross

SUBROUTINE pldcross(x1, y1, rr)

REAL, INTENT(IN):: x1, y1, rr

This subroutine draws a cross ×. The center of the cross is at (x1, y1). The length of the cross lines is 2∗rr.

2.17 pldhatching

SUBROUTINE pldhatching(nn, dxx, dyy, sp, theta)

INTEGER, INTENT(IN):: nn

REAL, INTENT(IN):: dxx(∗), dyy(∗), sp, theta

This subroutine draws hatching in a polygon. Argument nn is the number of vertexes of the polygon. It is necessary that nn≥3. Argument sp is the interval of the parallel lines. Argument theta is the slant angle of the parallel lines. Arguments dxx and dyy are the dimensions storing the xco and yco coordinates of the vertexes of the polygon, that is, the vertexes of the polygon are at {dxx(1),dyy(1)}, {dxx(2),dyy(2)}, ⋯ and {dxx(nn),dyy(nn)}.

2.18 prprctgl

SUBROUTINE prprctgl(x1, y1, x2, y2)

REAL, INTENT(IN):: x1, y1, x2, y2

This subroutine paints a rectangular region with the color given by subroutine pfscolor. The vertexes of the rectangle are at (x1, y1), (x2, y1), (x2, y2) and (x1, y2).

2.19 prpcrcl

SUBROUTINE prpcrcl(x1, y1, rr)

REAL, INTENT(IN):: x1, y1, rr

This subroutine paints a circular region with the color given by subroutine pfscolor. The position (x1, y1) is the center of the circle. Argument rr is the radius of the circle.

2.20 prpplygn

SUBROUTINE prpplygn(nn, dxx, dyy)

INTEGER, INTENT(IN):: nn

REAL, INTENT(IN):: dxx(∗), dyy(∗)

This subroutine paints a polygonal region with the color given by subroutine pfscolor. Argument nn is the number of vertexes of the polygon. It is necessary that nn≥3. Arguments dxx and dyy are the dimensions storing the xco and yco coordinates of the vertexes of the polygon, that is, the vertexes of the polygon are at {dxx(1),dyy(1)}, {dxx(2),dyy(2)}, ⋯ and {dxx(nn),dyy(nn)}.

2.21 pcsheight

SUBROUTINE pcsheight(height)

REAL, INTENT(IN):: height

The height of character patterns is determined by variable height which is illustrated in Fig. 2. The default value of variable height is set at 4. This subroutine re-sets variable height.

2.22 pcspstn

SUBROUTINE pcspstn(nx, ny)

INTEGER, INTENT(IN):: nx, ny

Integers nx and ny are illustrated in Fig. 2 and determine the internal position of a character pattern, which will be referred to again in section 2.24. Integer nx determines the abscissa of the internal position of the character pattern, and takes 0, 1 or 2. Integer ny determines the ordinate of the internal position of the character pattern, and takes −1, 0, 1, 2 or 3. The internal position is used in subroutines pcsangle and pcwrite, which will be introduced below. Both the default values of nx and ny are set at 0. Subroutine pcspstn can re-set the values of nx and ny.

2.23 pcsangle

SUBROUTINE pcsangle(theta)

REAL, INTENT(IN):: theta

Figure 2: Integers nx and ny determine the internal position of a character pattern.

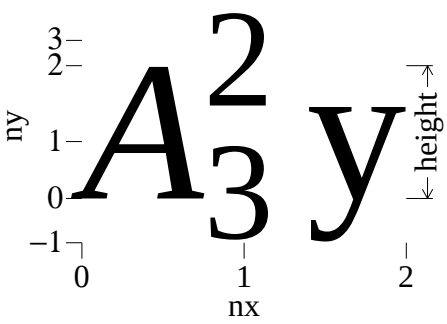
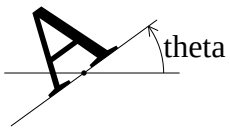


Figure 3: Slant angle theta of a character pattern. The figure shows the case that the internal position is nx=1 and ny=0, and the black dot indicates the center of the rotation, which is the internal position.



Real argument theta indicates the slant angle of a character pattern as shown in Fig. 3. The center of rotation of the character pattern is its internal position. The default value of variable theta is set at 0. This subroutine re-sets the value of variable theta. It must be satisfied that ABS(theta)<2∗pi.

2.24 pcwrite

SUBROUTINE pcwrite(x1, y1, char)

REAL, INTENT(IN):: x1, y1

CHARACTER(LEN=∗), INTENT(IN):: char

This subroutine writes a character pattern that is obtained from argument char. Arguments x1 and y1 determine the position of the character pattern outputted from subroutine pcwrite, so that the internal position of the character pattern is put at (x1, y1).

Argument char is expressed by the following statement

char=cstrg(1)//cstrg(2)//$\cdots$//cstrg(kk) ! Line 3

where cstrg(i) (i=1, 2, $\cdots$, kk) are character strings and are sorted into control strings and text strings. Integer kk is the total number of control strings and text strings in Line 3. The control string and the text string are explained below.

(1) Control strings

Let us define a function cst(i) first. Function cst(i) converts integer i to a character data. For example, cst(−3)='−3' and cst(6)='6'. The function cst(i) does not have any blank in its character data.

The control strings mean character strings for control. If cstrg(i) is a control string,

this cstrg(i) begins at character '\' and ends at character '}', and it does not have any '\' or any '}' except '\' and '}' that exist at both ends of this cstrg(i). Plotter provides the following three kinds of control strings.

(a) Font control strings

The font control strings mean character strings for font control. A font control string generally has a form '\f'//cst(i)//'}', where i=1, 2, $\cdots$, 12. The font control string '\f'//cst(i)//'}' means writing the text strings with Font i until the next control string '\f'//cst(j)//'}', where i $\neq$ j. The text strings will be explained later. Table 1 shows the correspondence between Fonts i (i=1, 2, $\cdots$, 12) and font names. Fonts 1, 2, $\cdots$, 10 correspond to English letters. Fonts 11 and 12 correspond to Kanji that are Japanese letters. The default font is set at Font 1.

Table 1: Fonts 1, 2, $\cdots$, 12 and the font names.

Fonts i	Font names	Fonts i	Font names
Font 1	Times-Roman	Font 7	Helvetica-Oblique
Font 2	Times-Bold	Font 8	Helvetica-BoldOblique
Font 3	Times-Italic	Font 9	Symbol
Font 4	Times-BoldItalic	Font 10	Symbol-Italic
Font 5	Helvetica	Font 11	Ryumin-Light-H
Font 6	Helvetica-Bold	Font 12	GothicBBB-Medium-H

(b) Index control strings

The index control strings mean character strings for control of superscripts or subscripts. Plotter provides the following three kinds of index control strings.

i. '\u}' : At the beginning of superscripts, an index control string '\u}' must be put.

ii. '\l}' : At the beginning of subscripts, an index control string '\l}' must be put.

iii. '\e}' : At the end of superscripts or subscripts, an index control string '\e}' must be put.

(c) Spacing control strings

The spacing control strings mean character strings for spacing control. The general form of the spacing control string is '\s'//cst(i)//'}'. This spacing control

string puts spacing of width (i/20.)∗height, where height is given by subroutine pcsheight, and i is an arbitrary integer and is also permitted to be negative.

(2) Text strings

The text strings mean character strings for texts. Here the texts mean characters that we want to write to the output file. If cstrg(i) is not any control string, this cstrg(i) must be a text string.

Code input is also possible for the characters whose keyboard input is impossible. The code of a character is composed of '\' and an octal number of three places. We can know the octal code numbers of all the characters from [2]. It is impossible to write '\' with keyboard input. If it is necessary to write '\', the code input for '\' is available.

Using examples, let us explain the control strings and the text strings. If argument char is given, then cstrg(1), $\cdots$, cstrg(kk) and kk are determined uniquely. If char=`'\f1}ab'`, then kk=2 and Line 3 becomes char=cstrg(1)//cstrg(2), where cstrg(1)=`'\f1}'` and cstrg(2)= 'ab'. In this case, cstrg(1) is a font control string, and cstrg(2) is a text string.

Next, let us consider the following statement:

```
CALL pcwrite(10., 0., '\f1}cd\f3}ef')          ! Line 4
```

In this example, the font control strings are `'\f1}'` and `'\f3}'`, and the text strings are 'cd' and 'ef'. According to the rules of Plotter, the font control string `'\f1}'` is effective until the next font control string `'\f3}'`. Hence 'cd' is written with Font 1. The text string 'ef' is written with Font 3 because there is no font control string after `'\f3}'`. The character pattern outputted from Line 4 is given by

cdef

In the following statement,

```
CALL pcwrite(10., 0., &
        '\f3}b\f1}\u}2\l}3\e}\s8}\f3}b\f1}\u}2\e}\l}3\e}')      ! Line 5
```

the outputted character pattern is given by

b_3^2 $b^2{}_3$

We know from this character pattern that b_3^2 and $b^2{}_3$ are different in the positions of the subscripts, and that the spacing between b_3^2 and $b^2{}_3$ is effective owing to the spacing control string `'\s8}'` in Line 5.

The character pattern outputted from the following statement is illustrated in Fig. 2.

```
CALL pcwrite(10., 0., '\f3}A\f1}\u}2\l}3\e}\s5}y' )
```

See Fig. 2. The point (nx=0, ny=0) corresponds to the starting point of the character A. The starting point is defined in the PostScript language. The line nx=2 indicates the right end of the character y. Hence, the position of line nx=2 is influenced by the superscript, the subscript and the spacing appearing in Fig. 2. However, the position of ny=2 is not influenced by a superscript or a subscript. The line of ny=2 is determined by only variable height given by subroutine pcsheight.

3 Sample programs for application of Plotter

Files sample1.f90–sample6.f90 are examples of application of Plotter. Let us state a method of implementation of sample1.f90 as an example. First, compile sample1.f90 and plotter.f90 with a Fortran 90 compiler, and link the two object programs, and execute the executable program. Then we get the output file sample1.ps. File sample1.ps is observable with software Ghostscript, and we can print out the picture of sample1.ps with a printer stated in section 1.2.

3.1 File sample1.f90

Implementation of file sample1.f90 storing program sample1 outputs file sample1.ps, which stores picture data Sample 1, where examples of application of all the subroutines except subroutines pldline, plsframe and plsclip are shown. Subroutines pldline, plsframe and plsclip are used in program sample4 stated in section 3.4.

Picture data Sample 1 illustrates examples of usage of subroutines of Plotter in Items 1, 2, $\cdots$, 11. Item 1 shows examples of usage of subroutines pldhatching, prprctgl, prpcrcl, prpplygn and pfscolor. Item 2 shows an example of using subroutine pldcross. Item 3 shows an example of using subroutines pldsgmnt and pldarrow. Item 4 illustrates examples of usage of subroutines pldcrcl, prpcrcl, pldcrclarc and pldarcarrow. Item 5 shows an example of using subroutine pldrctgl. Examples of using subroutines plstype, plswidth and pfscolor are shown in Item 6. Item 7 illustrates examples of usage of subroutines prprctgl, prpcrcl, prpplygn and pfscolor. The figures outputted for Item 7 are chromatic. Item 8 illustrates samples of characters by code inputs. Item 9 shows an example of using subroutines pcspstn, pcsheight, pcsangle and pfscolor. The fonts for Fonts i (i=1, $\cdots$, 10) are shown in Item 10. If the OS of your computer is Windows of the Japanese edition, remove mark ! in Line 129 of file sample1.f90. Then you can get output of Kanji characters in Item 10. Item 11 shows examples of application of subroutine pcwrite. Subroutines pfnbegin, pfnend and pfsorigin

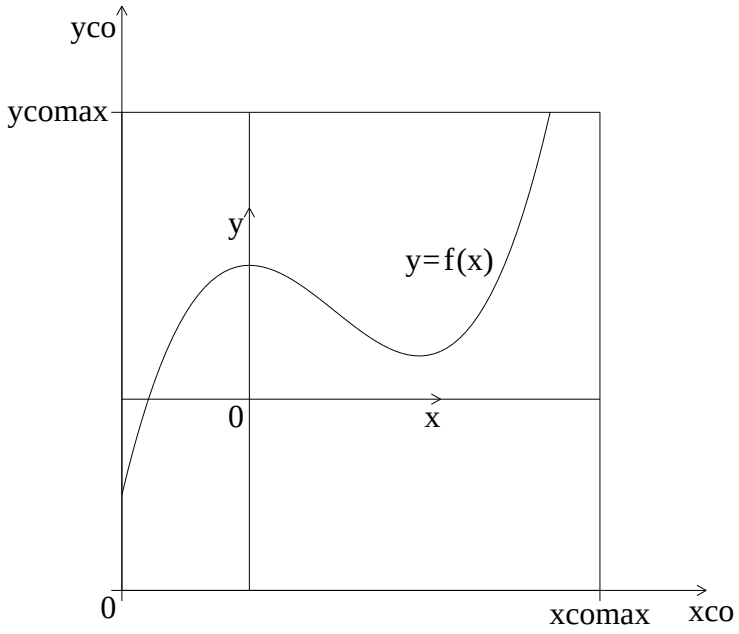


Figure 4: An orthogonal coordinates (x, y), and the orthogonal coordinates (xco, yco).

are also applied to program sample1.

3.2 File sample2.f90

Implementation of file sample2.f90 storing program sample2 outputs file sample2.ps, which stores picture data Sample 2, where a figure of a plane wave incident to a conductive wedge is drawn. We used subroutine prpcrcl for depicting the z axis. The subscript of character φ can be easily written.

3.3 File sample3.f90

Implementation of file sample3.f90 storing program sample3 outputs file sample3.ps, which stores picture data Sample 3, where a parallel plate condenser is drawn. Here subroutines pldhatching and prprctgl are applied.

3.4 File sample4.f90

First, let us explain a method of drawing a graph of y=f(x) in range xmin≤x≤xmax and range ymin≤y≤ymax. As shown in Fig. 4, the graph is drawn in a rectangular region 0≤xco≤xcomax, 0≤yco≤ycomax, where coordinates xco and yco are defined in section 2.3. Here we suppose that xmin, xmax, ymin, ymax, xcomax and ycomax are known. A function funxco determines xco when x is given, and a function funyco determines yco when y is given. Next we must determine the functions funxco and funyco. We suppose that

xco=funxco(x)=xfact∗x+xt0

yco=funyco(y)=yfact∗y+yt0

where the unknown constants xfact, xt0, yfact and yt0 are determined by solving the following equations.

0=funxco(xmin) xcomax=funxco(xmax)

0=funyco(ymin) ycomax=funyco(ymax)

Then functions funxco and funyco can be calculated. If function y=f(x) is known, we can draw the curve shown in Fig. 4.

Implementation of file sample4.f90 storing program sample4 outputs file sample4.ps, which stores picture data Sample 4. In the graph of Sample 4, xmin=−5, xmax=5, ymin=−1.2, ymax=1.5, xcomax=100 and ycomax=90. When constants xfact, xt0, yfact and yt0 are determined by the method stated above, the functions funxco and funyco are calculated by internal functions funxco and funyco of program sample4. The solid curve expresses function y=0.05∗x∗∗3+0.2∗x∗∗2+0.5, and the broken line expresses function y=sin(x). The curve of function y=0.05∗x∗∗3+0.2∗x∗∗2+0.5 is clipped in the region y≥1.5 owing to subroutines plsframe and plsclip.

3.5 File sample5.f90

Implementation of file sample5.f90 storing program sample5 outputs file sample5.ps, which stores picture data Sample 5, where a graph of y=cosh(x) is drawn. The y-axis of the graph has a logarithmic scale. Function funxco gives the relation between xco and x, and function funyco gives the relation between yco and y. These functions are given by

xco=funxco(x)=xfact∗x+xt0

yco=funyco(y)=yfact∗LOG10(y)+yt0

The unknown constants xfact, xt0, yfact and yt0 are determined below. The graph of y=cosh(x) in ranges xmin≤x≤xmax and ymin≤y≤ymax is depicted in the rectangular region 0≤xco≤xcomax and 0≤yco≤ycomax. Accordingly, the following four equations are satisfied.

0=funxco(xmin) xcomax=funxco(xmax)

0=funyco(ymin) ycomax=funyco(ymax)

Here we suppose that xmin=−1, xmax=10, ymin=0.7, ymax=3E4, xcomax=100 and ycomax=90. Hence, we can determine xfact, xt0, yfact and yt0 by the above four equations. Then the functions funxco and funyco are calculated by internal functions funxco and funyco of program sample5. Then we can draw the curve of y=cosh(x), and the curve is illustrated in picture data Sample 5.

3.6 File sample6.f90

Implementation of file sample6.f90 storing program sample6 outputs file sample6.ps, which stores picture data Sample 6, which illustrates a cross section of 2-dimensional temperature distribution between a high temperature cylinder and a low temperature plane. In program sample6, subroutines pfscolor, prpcrcl, prprctgl and so forth are applied. In this figure, the red color indicates high temperature, and the black color indicates low temperature.

References

[1] M. Metcalf and J. Reid, "Fortran 90/95 Explained *Second Edition*", Oxford University Press, 1999.

[2] Adobe Systems Inc., "PostScript Language Reference", Addison-Wesley, 1999.

[3] P.R. Bono and I. Herman, "GKS Theory and Practice", Springer-Verlag, 1987.

Version: 2.0

Date: February 20, 2009