D I S L I N 9.1

A Data Plotting

Library

by

Helmut Michels

Contents

Preface to Version 9.1

This manual describes the data plotting library DISLIN written in the programming languages Fortran and C. The name DISLIN is an abbreviation for Device-Independent Software LINdau since applications were designed to run on different computer systems without any changes. The library contains subroutines and functions for displaying data graphically as curves, bar graphs, pie charts, 3-D colour plots, surfaces, contours and maps.

DISLIN is intended to be a powerful and easy to use software package for programmers and scientists that does not require knowledge of hardware features of output devices. The routines in the graphics library are the result of my own work on many projects with different computers and many plotting packages. There are only a few graphics routines with a short parameter list needed to display the desired graphical output. A large variety of parameter setting routines can then be called to create individually customized graphics.

Since the first version of DISLIN was released in Dec. 1986, many changes and corrections have been made and new features and standards have been added to the software. Some of the new features are elementary image routines, a graphical user interface, filled contour lines, flat and smooth shaded surfaces and a C interface for reading binary data from Fortran programs. DISLIN supports now several hardware platforms, operating systems and compilers. A real Fortran 90 library is available for most Fortran 90 compilers.

Although nearly all the routines and utilities of the software package are written by myself, DISLIN would not have been possible without the help of many people. I would like to thank several people at the Max-Planck-Institut in Lindau. First, Dr. W. Degenhardt, Dr. H. J. Mueller and Dr. I. Pardowitz who gave their friendly assistance. To all the users of DISLIN, I am grateful for your helpful suggestions and comments. I would especially like to thank the members of the computer center, Friederich Both, Terry Ho, Godehard Monecke and Michael Bruns for their co-operation. Finally, I am grateful to Linda See and Erika Eschebach who corrected the English and German manuals with great carefulness. To all of them, my sincere thanks.

H. Michels   Lindau, 15.11.2006

Chapter 1

Introduction

DISLIN is a library of subroutines and functions that display data graphically. The routines can be used with any display device capable of drawing straight lines with the exception of routines that generate 3-D colour graphics which require special devices. Fortran 77, Fortran 90 and C versions of the library are available.

DISLIN can display graphic information directly on graphic terminals or store them in metafiles. The supported display types are VGA, X Windows, Windows API and Tektronix. The supported file formats are GKSLIN, CGM, HPGL, PostScript, PDF, WMF, PNG, SVG, PPM, BMP, GIF and TIFF. DISLIN metafiles can be printed on various devices using the DISLIN driver program DISDRV.

Chapter 2 describes the file and page formats and the overall structure of DISLIN programs.

Chapter 3 describes routines for the initialization, termination and plotting of text, numbers and symbols.

Chapter 4 presents the format of two-dimensional axis systems. Axes can be linearly or logarithmically scaled and labeled with linear, logarithmic, date, time, map and user-defined formats.

Chapter 5 describes the routines for plotting curves. Several curves can appear in one axis system and can be differentiated by colour, line style and pattern.

Chapter 6 summarizes parameter setting routines that overwrite default plotting parameters such as fonts, character size and angle, colours, line styles and patterns.

Chapter 7 presents routines to request the values of plot parameters.

Chapter 8 describes the routines for plotting lines, circles, ellipses, vectors and shaded regions.

Chapter 9 describes the utilities available to transform coordinates, sort data and calculate the lengths of numbers and character strings. Elementary image routines and some special routines that are only useful for terminal output are also described in this chapter.

Chapter 10 introduces business graphic routines to create bar graphs and pie charts.

Chapter 11 presents 3-D colour graphics where points can be plotted with coloured or shaded rectangles.

Chapter 12 describes routines for 3-D coordinate systems. Axis systems, curves and surfaces can be drawn from various angular perspectives. All 2-D plotting routines can be used in a 3-D axis system.

Chapter 13 presents 14 different methods to project geographical coordinates onto a plane surface. Several base maps are stored in the library for map plotting.

Chapter 14 describes routines for contouring three-dimensional functions of the form Z = F(X,Y). Contours can be filled with solid lines.

Chapter 15 offers routines for creating graphical user interfaces in Fortran and C programs.

Chapter 16 presents some quickplots that are collections of DISLIN routines for displaying data with one statement.

Chapter 2

Basic Concepts and Conventions

2.1 Page Format

In DISLIN, the graphics are limited to a rectangular area called the page. All lines outside of or crossing page borders will be suppressed.

The size of the page is determined by the routines SETPAG and PAGE. SETPAG corresponds to a predefined page while PAGE defines a global page setting. In default mode, there are 100 points per centimeter and the point (0, 0) is located in the upper left corner (Figure 2.1):

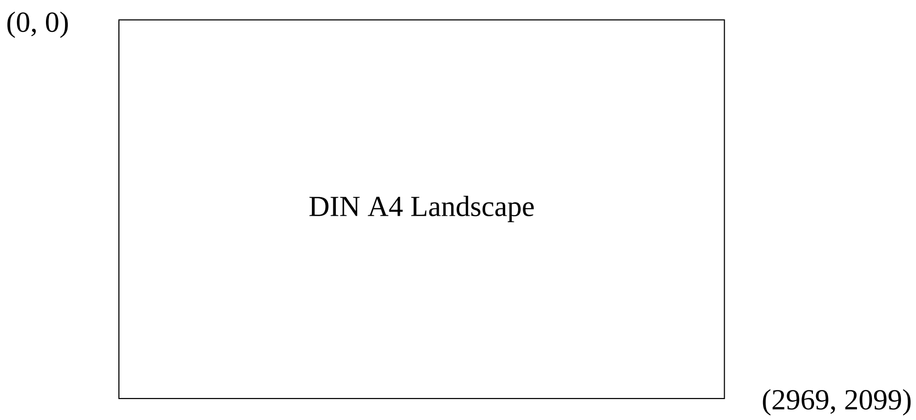


Figure 2.1: Default Page (DA4L)

2.2 File Format

DISLIN can create several types of plotfiles. Device-independent plotfiles or metafiles can be coded in ASCII or binary format. Device-dependent plotfiles are available for several printers and plotters.

The file formats are:

a) a CGM metafile according to the ANSI standard
Plot vectors are coded in binary format as non negative integers with 200 points per cm. Because of binary coding, CGM metafiles are smaller than other plotfiles.

b) a GKSLIN metafile
Plot vectors are stored as floating-point numbers between 0 and 1 in ASCII format. These files are easily transferable from one computer to another.

c) a PostScript file
PostScript is an international standard language that has been developed for laserprinters in the last few years. Some of the PostScript features such as hardware fonts and shading can be used within DISLIN.

d) an EPS file
the Encapsulated PostScript file format is simular to the PostScript format. It is useful for importing PostScript files into other appications.

e) a PDF file
The Portable Document Format is the de facto standard for the electronic exchange of documents. Compressed and non compressed PDF files can be created by DISLIN. PostScript fonts can be used for PDF files in the same way as for PostScript files.

f) a HPGL file
Plot vectors and colours are coded in a language recognized by Hewlett-Packard plotters.

g) a WMF file
The Windows metafile format is also supported by DISLIN. Plot vectors are converted to 1/1440 inch. WMF files can contain hardware fonts defined with the DISLIN routine WINFNT.

h) a SVG file
Scalable Vector Graphics (SVG) is a language for describing graphics in XML. SVG files can be displayed directly by some browsers if a corresponding plug-in is installed. The most of the standard PostScript fonts are supported by the DISLIN SVG files.

i) a GIF file
The Graphics Interchange Format (c) is the Copyright property of Compuserve Incorporated.

j) a TIFF file
The raster format TIFF can be used for storing graphical output. DISLIN can create 8 bit palette and truecolour TIFF files.

k) a PNG file
The Portable Network Graphics format is a compressed and therefore very small raster format for storing graphical output. PNG files can be displayed directly by several Internet browsers. The compression of PNG files is done in DISLIN with the zlib compression routines written by Jean-loup Gailly and Mark Adler. DISLIN supports 8 bit palette and truecolour PNG files.

l) a PPM file
The portable pixmap format is a well-known colour image file format in the UNIX world. There are many tools for converting PPM files into other image formats. The pixel values are stored in DISLIN PPM files in plain bytes as RGB values.

m) a BMP file
The Windows Bitmap format can be used for storing graphical output. DISLIN can create uncompressed 8 and 24 bit BMP files.

n) an IMAGE file
This easy raster format is used by DISLIN to store images. The files contain an ASCII header of 80 bytes and the following image data.

o) a Tektronix, X Window and VGA emulation
Data can be displayed on graphic terminals such as X Window, VGA and Tektronix 4010/4014.

File formats can be set with the routine METAFL. The filename consists of the keyword 'DISLIN' and an extension that depends on the file format. An alternate filename can be chosen by calling the routine SETFIL. Both subroutines must be called before the initialization routine DISINI.

2.3 Level Structure of DISLIN

Most routines in DISLIN can be called anywhere during program execution. Certain routines, however, use parameters from other routines and must be called in a fixed order. DISLIN uses a level structure to control the order in which routines are called. The levels are:

0	before initialization or after termination
1	after initialization or a call to ENDGRF
2	after a call to GRAF or POLAR
3	after a call to GRAF3 or GRAF3D.

Generally, programs should have the following structure:

(1)	setting of page format, file format and filename
(2)	initialization
(3)	setting of plot parameters
(4)	plotting of the axis system
(5)	plotting the title
(6)	plotting data points
(7)	termination.

2.4 Conventions

The following conventions appear throughout this manual for the description of routine calls:

- INTEGER variables begin with the character N or I
- CHARACTER variables begin with the character C
- other variables are REAL
- arrays end with the keyword 'RAY'.

Additional notes:

- CHARACTER keywords may be specified in upper or lower case and may be shortened to four characters.
- DISLIN stores parameters in common blocks whose names begin with the character 'C'. Common block names in user programs should not begin with the character 'C' to avoid possible name equalities.
- The Fortran logical units 15, 16 and 17 are reserved by DISLIN for plot and parameter files.
- Two types of coordinates are continually referred to throughout the manual: plot coordinates which correspond to the page and have by default 100 points per cm, and user coordinates which correspond to the scaling of the axis system.

2.5 Error Messages

When a DISLIN subroutine or function is called with an illegal parameter or not according to the level structure, DISLIN writes a warning to the screen. The call of the routine will be ignored and program execution resumed. Points lying outside of the axis system will also be listed on the screen. Error messages can be suppressed or written to a file with the routines ERRMOD and ERRDEV.

2.6 Programming in C

There are different DISLIN libraries for the programming languages Fortran 77, Fortran 90 and C. The DISLIN C library is written in the programming language C and useful for C programmers.

Though it is possible to call C routines in Fortran programs and Fortran subroutines in C programs, it is easier to use the corresponding library. Especially, the passing of strings can be complicate in mixed language programming.

The number and meaning of parameters passed to DISLIN routines are identical for all libraries. The Fortran versions use INTEGER, REAL and CHARACTER variables while the C library uses int, float and char variables. A detailed description of the syntax of C routines is given by the utility program DISHLP or can be found in the header file 'dislin.h' which must be included in all C programs.
Here is a short example for a DISLIN C program:

```
#include <stdio.h>
#include "dislin.h"
main()
{
    disini ();
    messag ("This is a test", 100, 100);
    disfin ();
}
```

An example for a DISLIN C++ programm is:

```
#include <iosteam>
namespace dislin {
#include "dislin.h"
}
main()
{
    dislin::disini ();
    dislin::messag ("This is a test", 100, 100);
    dislin::disfin ();
}
```

2.7 Programming in Fortran 90

Several DISLIN distributions contain native libraries for the programming language Fortran 90 where the source code of DISLIN is written in Fortran 90. Since the passing of parameters to subroutines and functions can be different in Fortran 90 and Fortran 77, you should not link Fortran 77 programs with Fortran 90 libraries and vice versa.

Additional notes:

- All program units in Fortran 90 programs that contain calls to DISLIN routines should include the statement 'USE DISLIN'. The module 'DISLIN' contains interfaces for all DISLIN routines and enables the compiler to check the number and type of parameters passed to DISLIN routines.
- Since version 9.1 of DISLIN, the array declarations in the DISLIN module file are changed from assumed-shape arrays to explicit-shape arrays for native Fortran 90 libraries. All DISLIN Fortran 90 libaries (native of wrapper) use now the same interfaces. A missing 'USE DISLIN' statement for a native Fortran 90 library of DISLIN should no longer cause a general protection fault.

For example:

```
PROGRAM TEST
    USE DISLIN
    CALL DISINI ()
    CALL MESSAG ('This is a test', 100, 100)
    CALL DISFIN ()
END PROGRAM TEST
```

2.8 Linking Programs

The linking of programs with the graphics library depends upon the operating system of the computer. Therefore, DISLIN offers a system-independent link procedure that can be used on all computers in the same way.

Command: DLINK [option] main

option — is an optional parameter containing a minus sign and a character. The following options can be used on all computers:

- -c — for compiling programs before linking.
- -cpp — for compiling a C++ program before linking.
- -r — for running programs after linking.
- -a — for compiling, linking and running programs.
- -r8 — for using the double precision libraries of DISLIN.

main — is the name of the main program.

Additional notes:

- If DLINK is called without parameters, the description of the program will be printed on the screen. There may be other local features available depending upon the operating system used.
- Linking of C programs should be done with the procedure CLINK.
- Linking of Fortran 90 programs should be done with the procedure F90LINK.
- The most DISLIN distributions contain libraries for single precision (32 bit) and double precision (64 bit) floatingpoint parameters. The double precision libraries can be identified by the term '_d' in the library filename.

2.9 Utility Programs

The following programs are useful for working with DISLIN. They send plotfiles to devices and and print the description of routines on the screen.

D I S H L P

DISHLP prints the description of a DISLIN routine on the screen.

Command: DISHLP routine [options]

routine — is the name of a DISLIN routine or a question mark. For a question mark, all routine names will be listed. An empty input terminates the program.

options — is an optional field of keywords (see DISHLP).

D I S M A N

DISMAN prints an ASCII version of the DISLIN manual on the screen.

Command:	DISMAN [options]
options	is an optional field of keywords (see DISMAN).

D I S D R V

DISDRV sends a plotfile to a device. CGM and GKSLIN files can be used for all devices while device-dependent plotfiles can only be sent to corresponding devices.

Command:	DISDRV filename[.MET] [device] [options]
filename	is the name of a plotfile.
device	is the name of a device where CONS refers to the graphics screen and XWIN to a smaller graphics window.
options	is an optional field of keywords (see DISDRV).

D I S I M G

DISIMG displays an image file on the screen, or converts it to PostScript and TIFF.

Command:	DISIMG filename[.IMG] [device] [options]
filename	is the name of the image file. The file must be created with the routine RIMAGE.
device	is the device name.
options	is an optional field of keywords (see DISIMG).

D I S M O V

DISMOV displays a sequence of image files.

Command:	DISMOV filename[.MOV] [device] [options]
filename	is the name of a data file where the filenames of the images are stored (1 line for each filename). The images must be created with the routine RIMAGE.
device	is the device name.
options	is an optional field of keywords (see DISMOV).

D I S T I F

DISTIF displays a TIFF file created by DISLIN on the screen, or converts it to PostScript and an image format.

Command:	DISTIF filename[.TIF] [device] [options]
filename	is the name of the TIFF file. The file must be created with DISLIN.
device	is the device name.
options	is an optional field of keywords (see DISTIF).

D I S G I F

DISGIF displays a GIF file, or converts it to another format.

Command:	DISGIF filename[.GIF] [device] [options]
filename	is the name of the GIF file.
device	is the device name.
options	is an optional field of keywords (see DISGIF).

D I S A P S

DISAPS converts an ASCII file to a PostScript file. Several page layouts can be defined.

Command:	DISAPS filename [output] [options]
filename	is the name of the ASCII file.
output	is the name of the output file. By default, the name of the input file and the extension ps will be used.
options	is an optional field of keywords (see DISAPS).
Additional note:	If a utility program is called without parameters, a description of possible parameters will be printed on the screen. DISDRV, for example, lists the local output devices available.

D I S G C L

DISGCL is an interpreter for DISLIN. All DISLIN statements can be written to a script file and then be executed with DISGCL, or can be entered in an interactive mode. High-level language elements such variables, operators, expressions, array operations, loops and user-defined functions van be used within DISGCL.

Command:	DISGCL [filename[.gcl]] [args] [options]
filename	is the name of a DISGCL script file. The extension '.gcl' is optional.
args	are optional arguments that can be passed to DISGCL scripts (see DISGCL).
options	is an optional field of keywords separated by blanks (see DISGCL).

2.10 WWW Homepage

DISLIN is available from the Web sites

http://www.dislin.de http://www.mps.mpg.de/dislin

2.11 Reporting Bugs

DISLIN is well tested by many users and should be very bug free. However, no software is perfect and every change can cause new bugs. If you have any problems with DISLIN, contact the author:

Helmut Michels
Max-Planck-Institut fuer Sonnensystemforschung
D-37191 Katlenburg-Lindau, Max-Planck-Str. 2, Germany
E-Mail: michels@mps.mpg.de
Tel.: +49 5556 979 334
Fax: +49 5556 979 240

2.12 License Information

DISLIN is free for non-commercial use. Licenses for commercial use are available from the site http://www.dislin.de. Commercial use means selling of programs linked with DISLIN or using DISLIN in an environment related to business.

This manual of the data plotting software DISLIN can be copied and distributed freely.

Chapter 3

Introductory Routines

3.1 Initialization and Termination

DISINI initializes DISLIN by setting default parameters and creating a plotfile. The level is set to 1. DISINI must be called before any other DISLIN routine except for those noted throughout the manual.

The call is:	CALL DISINI	level 0
or:	void disini ();	

DISFIN terminates DISLIN and prints a message on the screen. The level is set back to 0.

The call is:	CALL DISFIN	level 1, 2, 3
or:	void disfin ();	

Additional note: The printing of the protocol in DISFIN can be suppressed with the routine ERRMOD.

3.2 Plotting of Text and Numbers

M E S S A G

MESSAG plots text.

The call is:	CALL MESSAG (CSTR, NX, NY)	level 1, 2, 3
or:	void messag (char *cstr, int nx, int ny);	

CSTR	is a character string ($\leq$ 256 characters).
NX, NY	are the plot coordinates of the upper left corner.

N U M B E R

NUMBER plots a floating-point number or integer.

The call is:	CALL NUMBER (X, NDIG, NX, NY)	level 1, 2, 3
or:	void number (float x, int ndig, int nx, int ny);	

X	is a floating-point number.
NDIG	is the number of digits plotted after the decimal point. If NDIG = -1, X will be plotted as an integer. The last digit of X will be rounded up.

NX, NY	are the coordinates of the upper left corner.

RLMESS and RLNUMB are corresponding routines for user coordinates. They can be used for plotting text and numbers in an axis system after a call to GRAF.

The calls are:	CALL RLMESS (CSTR, XP, YP)	level 2, 3
	CALL RLNUMB (X, NDIG, XP, YP)	level 2, 3
or:	void rlmess (char *cstr, float xp, float yp);	
	void rlnumb (float x, int ndig, float xp, float yp);	

Additional notes:

- To continue character strings and numbers on the same line, the coordinates (999, 999) should be sent to MESSAG and NUMBER. The text or numbers will be plotted after the last plotted text character or number.
- The angle and height of the characters can be changed with the routines ANGLE and HEIGHT.
- The format of numbers can be modified with the routines NUMFMT and NUMODE.
- Text and numbers can be plotted in a box if the routine FRMESS is used.
- The starting point of text and numbers can be interpreted as upper left, upper center and upper right point if the routine TXTJUS is used.

3.3 Plotting Symbols

S Y M B O L

The routine SYMBOL plots symbols.

The call is:	CALL SYMBOL (NSYM, NX, NY)	level 1, 2, 3
or:	void symbol (int nsym, int nx, int ny);	

NSYM	is a symbol number between 0 and 23. Available symbols are given in the Appendix B.
NX, NY	is the centre of the symbol in plot coordinates.

Additional notes:

- The size of symbols can be set with HSYMBL.
- SYMROT (ANGLE) defines a rotation angle for symbols (in degrees). The symbol is rotated in a counter-clockwise direction.
- An user-defined symbol can be specified with the routine MYSYMB.

R L S Y M B

RLSYMB plots a symbol where the centre is specified by user coordinates.

The call is:	CALL RLSYMB (NSYM, XP, YP)	level 2, 3
or:	void rlsymb (int nsym, float xp, float yp);	

3.4 Plotting a Page Border, Background and Header

P A G E R A

PAGERA plots a border around the page.

The call is:	CALL PAGERA	level 1, 2, 3
or:	void pagera ();	

P A G F L L

The routine PAGFLL fills the page with a colour.

The call is:	CALL PAGFLL (NCLR)	level 1, 2, 3
or:	void pagfll (int nclr);	

NCLR is a colour value.

P A G H D R

PAGHDR plots a page header at a corner of the page. The header line contains date, time and user-defined information.

The call is:	CALL PAGHDR (CSTR1, CSTR2, IOPT, IDIR)	level 1, 2, 3
or:	void paghdr (char *cstr1, char *cstr2, int iopt, int idir);	

CSTR1		is a character string preceding the header line.
CSTR2		is a character string following the header line.
IOPT		is the page corner where the header is plotted:
	= 1	is the lower left corner.
	= 2	is the lower right corner.
	= 3	is the upper right corner.
	= 4	is the upper left corner.
IDIR		is the direction of the header line:
	= 0	is horizontal.
	= 1	is vertical.

Additional note: The character size of the header line is 0.6 * NH where NH is the parameter used in HEIGHT.

3.5 Sending a Metafile to a Device

A metafile can be converted with a driver program and sent from the operating system to several devices. From within a user program, the SYMFIL routine is used for this purpose.

S Y M F I L

SYMFIL sends a metafile to a device. It must be called after DISFIN.

The call is:	CALL SYMFIL (CDEV, CSTAT)	level 0
or:	void symfil (char *cdev, char *cstat);	

CDEV	is the name of the device. ’CONS’ refers to the graphics screen, ’XWIN’ to a X Window terminal, ’PSCi’ to a PostScript printer, ’KYOi’ to a Kyocera laserprinter with Prescribe and ’HPLi’ to a HP-plotter. The keyword ’NONE’ can be used to delete a metafile with no device plotting.
CSTAT	is a status parameter and can have the values ’DELETE’ and ’KEEP’.
Additional note:	SYMFIL calls the DISLIN driver utility DISDRV. The parameter ’REVERS’ can be passed to DISDRV from SYMFIL if the routine SCRMOD is called before with the parameter ’REVERS’.

3.6 Including Meta- and Bitmap files into a Graphics

GKSLIN and CGM metafiles created by DISLIN and general BMP files can be included into a graphics with the routine INCFIL.

I N C F I L

The routine INCFIL includes a GKSLIN or CGM metafile created by DISLIN, or general BMP and GIF files into a graphics.

The call is:	CALL INCFIL (CFIL)	level 1, 2, 3
or:	void incfil (char *cfil);	

CFIL is a character string that contains the filename.

Additional notes:

- For including BMP or GIF files, the output format must be a raster, PostScript or PDF format.
- The routine FILBOX (NX, NY, NW, NH) defines a rectangular area on the page where the file will be included. (NX, NY) are the plot coordinates of the upper left corner, (NW, NH) are the width and length of the box in plot coordinates. By default, the entire page will be used. If the file is a bitmap and the output format a raster format, the file will be included at the point (NX, NY) while NW and NH will be ignored. If the output format is PostScript or PDF, the BMP/GIF file will be scaled into the box defined by the parameters NX, NY, NW and NH. Therefore, NW and NH should have the same ratio as the width and height of the BMP/GIF file.
- INCFIL draws by default a frame around the included file that can be modified with the routine FRAME.
- With the statement CALL FILCLR (’NONE’), colour values in GKSLIN and CGM metafiles will be ignored and the current colour is used. The default is FILCLR (’ALL’).

Chapter 4

Plotting Axis Systems and Titles

4.1 Plotting Axis Systems

An axis system defines an area on the page for plotting data. Various axis systems can be plotted to accommodate different applications. For two-dimensional graphics, a maximum of two parallel X- and Y-axes can be drawn. The axis system is scaled to fit the range of data points and can be labeled with values, names and ticks. Two-dimensional axis systems are plotted with a call to the routines GRAF or POLAR.

G R A F

GRAF plots a two-dimensional axis system.

The call is:	CALL GRAF (XA, XE, XOR, XSTEP, YA, YE, YOR, YSTEP)	level 1
or:	void graf (float xa, float xe, float xor, float xstep, float ya, float ye, float yor, float ystep);	

XA, XE	are the lower and upper limits of the X-axis.
XOR, XSTEP	are the first X-axis label and the step between labels.
YA, YE	are the lower and upper limits of the Y-axis.
YOR, YSTEP	are the first Y-axis label and the step between labels.

Additional notes:

- GRAF must be called in level 1 and automatically sets the level to 2. When plotting more than 1 axis system on a page, ENDGRF must be called in between each new set of axes in order to set the level back to 1.
- The position of the lower left corner and the size of an axis system can be changed with the routines AXSPOS and AXSLEN.
- The axis scaling is linear by default and can be changed with AXSSCL. For logarithmic scaling, the corresponding parameters in GRAF must be exponents of base 10.
- One of several label types can be chosen with the routine LABELS or user-defined with MYLAB. Single labels can be suppressed by calling AXENDS.
- The routine NAME defines axis titles.
- The number of ticks between axis labels can be changed with the routine TICKS.
- SETGRF can be used to remove a piece of or complete axis from an axis system.

- If the numerical value of the lower limit of an axis is larger than the upper limit and the label step is negative, axis scaling will be in descending order.
- The routine FRAME defines the thickness of a frame plotted around an axis system. A frame can also be plotted outside of GRAF with the statement CALL BOX2D.
- A crossed axis system can be defined with CALL AXSTYP ('CROSS').

The following routine POLAR can be used to plot a polar axis system and set up a scale for polar axes.

P O L A R

The routine POLAR plots a two-dimensional polar axis system.

The call is:	CALL POLAR (XE, XOR, XSTEP, YOR, YSTEP)	level 1
or:	void polar (float xe, float xor, float xstep, float yor, float ystep);	
XE	is upper limit of the X-axis (radius coordinate).	
XOR, XSTEP	are the first X-axis label and the step between labels.	
YOR, YSTEP	are the first Y-axis label and the step between labels specified in degrees. The Y-axis is scaled from 0 to 360 degrees.	
Additional note:	The direction and position of the angle labels can be modified with the routine POLMOD.	

4.2 Termination of Axis Systems

E N D G R F

The routine ENDGRF terminates an axis system and sets the level back to 1.

The call is:	CALL ENDGRF	level 2, 3
or:	void endgrf ();	

4.3 Plotting Titles

T I T L E

This routine plots a title over an axis system. The title may contain up to four lines of text designated with TITLIN.

The call is:	CALL TITLE	level 2, 3
or:	void title ();	
Additional note:	All lines are centred by default but can be left- or right-justified using TITJUS.	

4.4 Plotting Grid Lines

G R I D

The routine GRID overlays a grid on an axis system.

The call is:	CALL GRID (IXGRID, IYGRID)	level 2, 3
or:	void grid (int ixgrid, int iygrid);	

IXGRID, IYGRID	are the numbers of grid lines between labels.
Additional note:	GRID uses automatically GRDPOL for a polar axis system.

G R D P O L

The routine GRDPOL plots a polar grid.

The call is:	CALL GRDPOL (IXGRID, IYGRID)	level 2, 3
or:	void grdpol (int ixgrid, int iygrid);	

IXGRID	is the numbers of circles between labels.
IYGRID	is the numbers of sector lines between 360 degrees.

Example:

The statements

```
CALL  AXSLEN     (1400,1400)
CALL  GRAF       (-3., 3., -3., 1., -3., 3., -3., 1.)
CALL  GRDPOL     (3, 16)
```

produce the following figure:

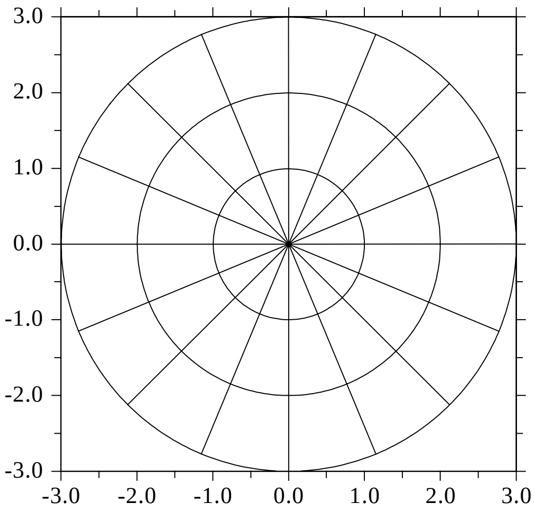


Figure 4.1: GRDPOL

A X G I T

The routine AXGIT plots vertical and horizontal lines through X = 0 and Y = 0.

The call is:	CALL AXGIT	level 2, 3

or:	void axgit ();	

Additional note: The statement CALL XAXGIT plots only the line Y = 0 while CALL YAXGIT plots only X = 0.

C R O S S

The routine CROSS plots vertical and horizontal lines with additional ticks through X = 0 and Y = 0.

The call is:	CALL CROSS	level 2, 3
or:	void cross ();	

Additional note: The statement CALL XCROSS plots only the line Y = 0 while CALL YCROSS plots only X = 0.

4.5 Plotting Additional Labels

A D D L A B

Additional single labels can be plotted on an axis system with the routine ADDLAB.

The call is:	CALL ADDLAB (CSTR, V, ITIC, CAX)	level 2, 3
or:	void addlab (char *cstr, float v, int itic, char *cax);	

CSTR	is a character string containing a label.
V	is an user coordinate that defines the axis position of the label.
ITIC	is an integer option that defines if a tick mark is plotted. ITIC = 0 means that no tick is plotted, ITIC = 1 defines a minor tick and ITICK = 2 defines a major tick.
CAX	is a character string that defines the axis. CAX can have the values ’X’, ’Y’, ’Z’, ’XTOP’ and ’YRIGHT’.

4.6 Secondary Axes

The following routines plot single X- and Y-axes; they are called secondary axes because they do not define or change any of the axis scaling parameters. Secondary axes can be used to add additional labels to the axis systems.

The plotting routines for secondary axes are:

XAXIS	plots a linear X-axis.	level 1, 2, 3
YAXIS	plots a linear Y-axis.	level 1, 2, 3
XAXLG	plots a logarithmic X-axis.	level 1, 2, 3
YAXLG	plots a logarithmic Y-axis.	level 1, 2, 3

The call is:	CALL XAXIS (A, B, OR, STEP, NL, CSTR, IT, NX, NY)
or:	void xaxis (float a, float b, float or, float step, int nl, char *cstr, int it, int nx, int ny);

A, B	are the lower and upper limits of the axis.
OR, STEP	are the first label and the step between labels.
NL	is the length of the axis in plot coordinates.
CSTR	is a character string containing the axis name.
IT	indicates how ticks, labels and the axis name are plotted. If IT = 0, they are plotted in a clockwise direction. If IT = 1, they are plotted in an counter-clockwise direction.
NX, NY	are the plot coordinates of the axis start point. The X-axis will be plotted from left to right and the Y-axis from bottom to top.

Analog: YAXIS, XAXLG, YAXLG

Additional notes:

- Secondary axes can be called from level 1, 2 or 3. Note again that secondary axes do not change the scaling of an axis system defined by GRAF. Similarly, curves cannot be plotted with only secondary axes, they require a call to GRAF.
- As in GRAF, the parameters of logarithmic axes must be exponents of base 10.
- User-defined labels may also be plotted on secondary axes with MYLAB and the argument 'USER' in the routine LABELS. The number of ticks can be changed by calling TICKS.

Chapter 5

Plotting Curves

This chapter describes how to plot curves with lines and symbols. Several curves can be plotted in one axis system and can be differentiated by colour, line style and pattern. Curve attributes can be plotted in a legend.

5.1 Plotting Curves

C U R V E

CURVE connects data points with lines or plots them with symbols.

The call is:	CALL CURVE (XRAY, YRAY, N)	level 2, 3
or:	void curve (float *xray, float *yray, int n);	

XRAY, YRAY	are arrays that contain X- and Y-coordinates. For a polar scaling, XRAY must hold the radial values and YRAY the angular values expressed in radians.
N	is the number of data points.

Additional notes:

- CURVE must be called after GRAF or POLAR from level 2 or 3.
- By default, data points that lie outside of an axis system are listed on the screen. The listing can be suppressed with the routine NOCHEK.
- For a logarithmic scaling of an axis, CURVE suppresses the plotting of curves and prints a warning if some corresponding data coordinates have non positive values. After the statement CALL NEGLOG (EPS), where EPS is a small positiv floating-point number, CURVE will use the value EPS for non positive values.
- CURVE suppresses lines outside the borders of an axis system. Suppressing can be disabled with NOCLIP or the margins of suppression can be changed with GRACE.
- INCMRK determines if CURVE plots lines or symbols.
- When plotting several curves, attributes such as colour and line style can be changed automatically by DISLIN or directly by the user. The routine CHNCRV defines which attributes are changed automatically. The routines COLOR or SETCLR are used to define colours, SOLID, DOT, DASH, CHNDOT, CHNDSH, DOTL, DASHM and DASHL to define line styles and MARKER to define symbols plotted with the routine CURVE.
- Different data interpolation methods can be chosen with POLCRV.

5.2 Plotting Legends

To differentiate multiple curves in an axis system, legends with text can be plotted. DISLIN can store up to 30 curve attributes such as symbols, thicknesses, line styles and colours and these can be incorporated in a legend.

Legends are created with the following steps:

(1) define a character variable used to store the lines of text in the legend

(2) initialize the legend

(3) define the lines of text

(4) plot the legend.

The corresponding routines are:

L E G I N I

LEGINI initializes a legend.

The call is:	CALL LEGINI (CBUF, NLIN, NMAXLN)	level 1, 2, 3
or:	void legini (char *cbuf, int nlin, int nmaxln);	

CBUF	is a character variable used to store the lines of text in the legend. The variable must be defined by the user to have at least NLIN * NMAXLN characters.
NLIN	is the number of text lines in the legend.
NMAXLN	is the number of characters in the longest line of text.

L E G L I N

LEGLIN stores lines of text for the legend.

The call is:	CALL LEGLIN (CBUF, CSTR, ILIN)	level 1, 2, 3
or:	void leglin (char *cbuf, char *cstr, int ilin);	

CBUF	see LEGINI.
CSTR	is a character string that contains a line of text for the legend.
ILIN	is the number of the legend line between 1 and NLIN.

L E G E N D

LEGEND plots legends.

The call is:	CALL LEGEND (CBUF, NCOR)	level 2, 3
or:	void legend (char *cbuf, int ncor);	

CBUF		see LEGINI.
NCOR		indicates the position of the legend:
	= 1	is the lower left corner of the page.
	= 2	is the lower right corner of the page.
	= 3	is the upper right corner of the page.
	= 4	is the upper left corner of the page.
	= 5	is the lower left corner of the axis system.
	= 6	is the lower right corner of the axis system.
	= 7	is the upper right corner of the axis system.
	= 8	is the upper left corner of the axis system.

Additional notes: The following routines change the position and appearance of a legend. They must be called after LEGINI except for the routines FRAME and LINESP.

- LEGTIT (CTIT) sets the title of the legend.
  Default: CTIT = 'Legende'.
- LEGPOS (NX, NY) defines a global position for the legend where NX and NY are the plot coordinates of the upper left corner. After a call to LEGPOS, the second parameter in LEGEND will be ignored.
- NLX = NXLEGN (CBUF) and NYL = NYLEGN (CBUF) return the length and the height of a legend in plot coordinates.
- FRAME (NFRA) defines the thickness of a frame plotted around a legend.
- LINESP (XF) changes the spacing of lines in a legend.
- LEGCLR retains the same colour for curves and lines of text in the legend.

- The statement CALL MIXLEG enables multiple text lines in legends. By default, the character '/' is used as a newline character but can be changed with the routine SETMIX.

L E G P A T

The routine LEGPAT stores curve attributes plotted in legends. Normally, this is done automatically by routines such as CURVE and BARS.

The call is: CALL LEGPAT (ITYP, ITHK, ISYM, ICLR, IPAT, ILIN) level 1, 2, 3
or: void legpat (int ityp, int ithk, int isym, int iclr, long ipat, int ilin);

ITYP is the line style between -1 and 7 (see LINTYP). IF ITYP = -1, no line will be plotted in the legend line.

ITHK defines the thickness of lines (> 0).

ISYM is the symbol number between -1 and 21. If ISYM = -1, no symbol will be plotted in the legend line.

ICLR is the colour value. If ICLR = -1, the current colour will be used.

IPAT is the shading pattern (see SHDPAT). If IPAT = -1, no pattern will be plotted in the legend line.

ILIN is the legend line between 1 and NLIN.

Additional notes:
- The routine LEGPAT is useful to create legends without calls to CURVE.
- LEGPAT must be called after LEGINI.

L E G O P T

The routine LEGOPT modifies the appearance of legends.

The call is: CALL LEGOPT (XF1, XF2, XF3) level 1, 2, 3
or: void legopt (float xf1, float xf2, float xf3);

XF1 is a multiplier for the length of the pattern field. The length is XF1 * NH, where NH is the current character height. If XF1 = 0., the pattern field will be suppressed.

XF2 — is a multiplier for the distance between legend frames and text. The distance is XF2 * NH * XSPC, where XSPC is the spacing between legend lines (see LINESP).

XF3 — is a multiplier for the spacing between multiple text lines. The space is XF3 * NH * XSPC.

Default: (4.0, 0.5, 1.0).

5.3 Plotting Shaded Areas between Curves

S H D C R V

SHDCRV plots a shaded area between two curves.

The call is: CALL SHDCRV (X1RAY, Y1RAY, N1, X2RAY, Y2RAY, N2) level 2, 3

or: void shdcrv (float *x1ray, float *y1ray, int n1, float *x2ray, float *y2ray, int n2);

X1RAY, Y1RAY — are arrays with the X- and Y-coordinates of the first curve. Values are not changed by SHDCRV.

N1 — is the number of points in the first curve.

X2RAY, Y2RAY — are arrays with the X- and Y-coordinates of the second curve. Values are not changed by SHDCRV.

N2 — is the number of points in the second curve.

Additional notes:

- The maximum number of data points cannot be greater than 25000 in Fortran 77 programs. There is no restriction for Fortran 90 and C.
- Different shading patterns can be selected with SHDPAT. The pattern number will automatically be incremented by 1 after a call to SHDCRV.
- Legends may be plotted for shaded curves.
- The routine NOARLN will suppress border lines around shaded areas.

5.4 Plotting Error Bars

E R R B A R

The routine ERRBAR plots error bars.

The call is: CALL ERRBAR (XRAY, YRAY, E1RAY, E2RAY, N) level 2, 3

or: void errbar (float *xray, float *yray, float *e1ray, float *e2ray, int n);

XRAY, YRAY — are arrays that contain the X- and Y-coordinates.

E1RAY, E2RAY — are arrays that contain the errors. Lines will be drawn from YRAY - E1RAY to YRAY + E2RAY.

N — is the number of data points.

Additional notes:

- Horizontal bars will be drawn after CALL BARTYP ('HORI').
- A symbol can be selected with MARKER and the symbol size with HSYMBL.

5.5 Plotting Vector Fields

F I E L D

The routine FIELD plots a vector field.

The call is:	CALL FIELD (X1RAY, Y1RAY, X2RAY, Y2RAY, N, IVEC)	level 2, 3
or:	void field (float *x1ray, float *y1ray, float *x2ray, float *y2ray, int n, int ivec);	

X1RAY, Y1RAY	are arrays that contain the X- and Y-coordinates of the start points.
X2RAY, Y2RAY	are arrays that contain the X- and Y-coordinates of the end points.
N	is the number of vectors.
IVEC	is a four digit number that specifies the vector (see VECTOR).

Chapter 6

Parameter Setting Routines

All parameters in DISLIN have default values set by the initialization routine DISINI. This chapter summarizes subroutines that allow the user to alter default values. The following routines can be called from level 1, 2 or 3 except for those noted throughout the chapter. Subroutines that can only be called from level 0 must appear before DISINI. In general, parameter setting routines should be called between DISINI and the plotting routines they affect.

6.1 Basic Routines

6.1.1 Resetting Parameters

R E S E T

RESET sets parameters back to their default values.

The call is:	CALL RESET (CNAME)	level 1, 2, 3
or:	void reset (char *cname);	

CNAME is a character string containing the name of the routine whose parameters will be set back to default values. If CNAME = 'ALL', all parameters in DISLIN will be reset.

6.1.2 Changing the Plot Units

U N I T S

The routine UNITS defines the plot units.

The call is:	CALL UNITS (COPT)	level 0
or:	void units (char *copt);	

COPT is a character string that can have the values 'CM', 'INCH', 'POINTS' and 'TWIPS'. 'CM' means 100 points per centimeter, 'INCH' means 100 points per inch, 'POINTS' means 720 points per inch and 'TWIPS' means 1440 points per inch.

Default: COPT = 'CM'.

6.1.3 Modifying the Origin

P A G O R G

The routine PAGORG sets the origin of the page. By default, the page origin is located in the upper left corner of the page.

The call is:	CALL PAGORG (COPT)	level 1, 2, 3
or:	void pagorg (char *copt);	

COPT — is a character string that can have the values 'TOP' and 'BOTTOM'. The keyword 'TOP' sets the page origin to the upper left corner, 'BOTTOM' to the lower left corner.
Default: COPT = 'TOP'.

O R I G I N

In DISLIN, all lines are plotted relative to a point on the page which is by default identical with the page origin. Modifying this point by ORIGIN produces a shifting of plot vectors on the page.

The call is:	CALL ORIGIN (NX0, NY0)	level 1
or:	void origin (int nx0, int ny0);	

NX0, NY0 — are the coordinates of the origin. Default: (0, 0).

6.1.4 File Format Control

M E T A F L

METAFL defines the metafile format.

The call is:	CALL METAFL (CFMT)	level 0
or:	void metafl (char *cfmt);	

CFMT	is a character string that defines the file format.
= 'GKSL'	defines a GKSLIN metafile.
= 'CGM'	defines a CGM metafile.
= 'PS'	defines a coloured PostScript file.
= 'EPS'	defines an Encapsulated PostScript file. The format is nearly the same as for 'PS'.
= 'PDF'	defines a PDF file.
= 'HPGL'	defines a HPGL file.
= 'SVG'	defines a Scalable Vector Graphics file.
= 'JAVA'	defines a Java applet file.
= 'WMF'	defines a Windows metafile.
= 'GIF'	defines a GIF file.
= 'TIFF'	defines a TIFF file.
= 'PNG'	defines a PNG file.
= 'PPM'	defines a portable pixmap format.
= 'IMAG'	defines a DISLIN image format.

= 'BMP' defines a Windows Bitmap format.

= 'VIRT' defines a virtual file. The metafile is hold in a raster format in computer memory.

= 'CONS' defines a graphics output on the screen. If the screen is a windows display, a graphical window is used that has nearly the size of the screen.

= 'XWIN' defines a window for graphical output. By default, the size of the window is nearly 2/3 of the size of the screen.

Default: CFMT = 'GKSL'.

Notes:

- The default size of TIFF, GIF, PNG, PPM, BMP, IMAGE, SVG and virtual files is set to 853 x 603 points but can be modified with the routine WINSIZ. The size of graphical windows can also be changed with WINSIZ.
- The default background colour for graphical windows and image formats such as TIFF, GIF and PNG is black but can be changed to white with the routine SCRMOD.
- The format of VIRT, TIFF, PNG, BMP and IMAGE is by default a 8 bit palette format, but can be changed to a truecolour format with the parameter 'RGB' in the routine IMGFMT. GIF files created by DISLIN have always a 8 bit palette format.

SETFIL

By default, the plotfile name consists of the keyword 'dislin' and an extension that depends on the file format. An alternate filename can be set with SETFIL.

The call is:	CALL SETFIL (CFIL)	level 0
or:	void setfil (char *cfil);	

CFIL is a character string that contains the filename.

FILMOD

The routine FILMOD determines if a new plotfile name is created for existing files.

The call is:	CALL FILMOD (CMOD)	level 0, 1, 2, 3
or:	void filmod (char *cmod);	

CMOD is a character string containing the mode.

= 'COUNT' means that a new file version will be created.

= 'DELETE' means that the existing file will be overwritten.

= 'BREAK' means that the program will be terminated by DISINI.

Default: CMOD = 'COUNT'.

SCRMOD

Normally, the background of screens and image formats such as TIFF, GIF, BMP and PNG is set to 'BLACK'. With the routine SCRMOD, the back and foreground colours can be swapped.

The call is:	CALL SCRMOD (CMOD)	level 0
or:	void scrmod (char *cmod);	

CMOD = 'AUTO' uses a 'BLACK' background colour for screen output and image files.

CMOD = 'REVERS'	means that the background colour is set to 'WHITE' and the foreground colour to 'BLACK'.
CMOD = 'NOREV'	means that the background colour is set to 'BLACK' and the foreground colour to 'WHITE'.

Default: CMOD = 'AUTO'.

C G M B G D

The routine CGMBGD sets the background colour for CGM files.

The call is:	CALL CGMBGD (XR, XG, XB)	level 0, 1, 2, 3
or:	void cgmbgd (float xr, floar xg, float xb);	

XR, XG, XB	are the RGB coordinates of the background colour in the range 0 to 1.

Default: (1., 1., 1.).

C G M P I C

The routine CGMPIC modifies the picture ID in CGM files. The picture ID may be referenced by some browsers.

The call is:	CALL CGMPIC (CSTR)	level 0, 1, 2, 3
or:	void cgmpic (char *cstr);	

CSTR	is a character string containing the picture ID ($\leq$ 256 characters). By default, the ID 'Picture n' is used where n is the picture number beginning with 1.

T I F M O D

The routine TIFMOD modifies the physical resolution of TIFF files.

The call is:	CALL TIFMOD (N, CVAL, COPT)	level 0
or:	void tifmod (int n, char *cval, char *copt);	

N	is an integer value containing the number of pixels per resolution unit.
CVAL	is a character string containing the resolution unit. CVAL can have the values 'INCH' and 'CM'.
COPT	is a character string that can have the value 'RESOLUTION'.

Default: (100, 'INCH', 'RESOLUTION').

W M F M O D

The routine WMFMOD modifies the appearance of WMF files.

The call is:	CALL WMFMOD (CMOD, CKEY)	level 0
or:	void wmfmod (char *cmod, char *ckey);	

CMOD	is a character string containing the values 'STANDARD' or 'PLACEABLE'. If CMOD = 'PLACEABLE', an additional leading header of 22 byte is added to the WMF file. The format is also known as Aldus Placeable Metafile.
CKEY	is a character string that can have the value 'FORMAT'.

Default: CMOD = 'STANDARD'.

P D F M O D

The routine PDFMOD selects between compressed and non compressed PDF files, and can enable PDF buffer output instead of file output.

The call is:	CALL PDFMOD (CMOD, CKEY)	level 0
or:	void pdfmod (char *cmod, char *ckey);	

CMOD — is a character string that can have the values 'ON' and 'OFF'.

CKEY — is a character string that can have the values 'COMPRESSION' and 'BUFFER'. For CKEY = 'BUFFER' and CMOD = 'ON', the PDF file is hold in memory and can be copied to an user buffer with the routine PDFBUF after DISFIN.

Default: ('ON', 'COMPRESSION'),
Default: ('OFF', 'BUFFER').

P D F M R K

The routine PDFMRK writes bookmarks to PDF files. This makes it possible to navigate through PDF files that contain multiple pages.

The call is:	CALL PDFMRK (CSTR, COPT)	level 1 ,2 ,3
or:	void pdfmrk (char *cstr, char *copt);	

CSTR — is a character string that contains the text of the bookmark.

COPT — is a character string that can have the values 'CHAPTER', 'SECTION', 'SUBSECTION', 'PARAGRAPH' and 'SUBPARAGRAPH'. This option defines the level of a bookmark in the hierarchy of bookmarks. A bookmark with the option 'SECTION' can only be defined if a bookmark with the option 'CHAPTER' is defined before, and so on.

G I F M O D

The routine GIFMOD enables transparency for GIF files.

The call is:	CALL GIFMOD (CMOD, CKEY)	level 0
or:	void gifmod (char *cmod, char *ckey);	

CMOD — is a character string that can have the values 'ON' and 'OFF'.

CKEY — is a character string that can have the value 'TRANSPARENCY'.

Default: ('OFF', 'TRANSPARENCY').

P N G M O D

The routine PNGMOD enables transparency for PNG files.

The call is:	CALL PNGMOD (CMOD, CKEY)	level 0
or:	void pngmod (char *cmod, char *ckey);	

CMOD — is a character string that can have the values 'ON' and 'OFF'.

CKEY — is a character string that can have the value 'TRANSPARENCY'.

Default: ('OFF', 'TRANSPARENCY').

Additional note:	For indexed PNG files, the colour table entry 0 is used for transparency. For RGB files, the colour White is used for transparency.

I M G F M T

The routine IMGFMT defines palette or truecolour mode for DISLIN image formats such as TIFF, PNG, BMP and IMAGE.

The call is:	CALL IMGFMT (CMOD)	level 0
or:	void imgfmt (char *cmod);	

CMOD	is a character string that can have the values 'INDEX' and 'RGB'. For TIFF files, the additional keyword 'BILEVEL' is allowed for creating bilevel TIFF files. Default: CMOD = 'INDEX'.

6.1.5 Page Control

P A G E

PAGE determines the size of the page.

The call is:	CALL PAGE (NXP, NYP)	level 0
or:	void page (int nxp, int nyp);	

NXP, NYP	are the length and height of the page in plot coordinates. The lower right corner of the page is the point (NXP-1, NYP-1). Default: (2970, 2100).

S E T P A G

SETPAG selects a predefined page format.

The call is:	CALL SETPAG (CPAGE)	level 0
or:	void setpag (char *cpage);	

CPAGE is a character string that defines the page format.

= 'DA4L'	DIN A4,	landscape,	2970 * 2100 points.
= 'DA4P'	DIN A4,	portrait,	2100 * 2970 points.
= 'DA3L'	DIN A3,	landscape,	4200 * 2970 points.
= 'DA3P'	DIN A3,	portrait,	2970 * 4200 points.
= 'DA2L'	DIN A2,	landscape,	5940 * 4200 points.
= 'DA2P'	DIN A2,	portrait,	4200 * 5940 points.
= 'DA1L'	DIN A1,	landscape,	8410 * 5940 points.
= 'DA1P'	DIN A1,	portrait,	5940 * 8410 points.
= 'DA0L'	DIN A0,	landscape,	11890 * 8410 points.
= 'DA0P'	DIN A0,	portrait,	8410 * 11890 points.
= 'USAL'	US paper size A,	landscape,	2790 * 2160 points.
= 'USAP'	US paper size A,	portrait,	2160 * 2790 points.
= 'USBL'	US paper size B,	landscape,	4320 * 2790 points.
= 'USBP'	US paper size B,	portrait,	2790 * 4320 points.
= 'USCL'	US paper size C,	landscape,	5590 * 4320 points.
= 'USCP'	US paper size C,	portrait,	4320 * 5590 points.
= 'USDL'	US paper size D,	landscape,	8640 * 5590 points.
= 'USDP'	US paper size D,	portrait,	5590 * 8640 points.

= 'USEL'	US paper size E,	landscape,	11180 * 8640 points.
= 'USEP'	US paper size E,	portrait,	8640 * 11180 points.
= 'PS4L'	PostScript A4,	landscape,	2800 * 1950 points.
= 'PS4P'	PostScript A4,	portrait,	1950 * 2800 points.
= 'HP4L'	HP-plotter A4,	landscape,	2718 * 1900 points.
= 'HP4P'	HP-plotter A4,	portrait,	1900 * 2718 points.
= 'HP3L'	HP-plotter A3,	landscape,	3992 * 2718 points.
= 'HP3P'	HP-plotter A3,	portrait,	2718 * 3992 points.
= 'HP2L'	HP-plotter A2,	landscape,	5340 * 3360 points.
= 'HP2P'	HP-plotter A2,	portrait,	3360 * 5340 points.
= 'HP1L'	HP-plotter A1,	landscape,	7570 * 5340 points.
= 'HP1P'	HP-plotter A1,	portrait,	5340 * 7570 points.

Default: CPAGE = 'DA4L'.

S C L F A C

SCLFAC sets the scaling factor for an entire plot.

The call is: CALL SCLFAC (XFAC) level 0

or: void sclfac (float xfac);

XFAC is the scaling factor by which the entire plot is scaled up or down.

Default: XFAC = 1.

S C L M O D

The method by which graphics are scaled to the hardware pages of devices such as a graphics terminal can be selected with the routine SCLMOD.

The call is: CALL SCLMOD (CMOD) level 0

or: void sclmod (char *cmod);

CMOD = 'DOWN' means that graphics will be scaled down if the hardware page of a device is smaller than the plotting page.

= 'FULL' means that the graphics will be scaled up or down depending upon the size of the hardware page.

Default: CMOD = 'DOWN'.

Additional notes:
- The size of a graphics screen will be interpreted as DIN A4 landscape. This means that by default graphics which are smaller than DIN A4 will not fill the entire screen.
- SCLFAC and SCLMOD can affect each other.

P A G M O D

GKSLIN and CGM files can be rotated by 90 degrees to use the full hardware page of a device. In general, this is done automatically by the driver program.

The call is: CALL PAGMOD (CMOD) level 0

or: void pagmod (char *cmod);

CMOD = 'LAND' means that the metafile is not rotated.

= 'PORT' means that the metafile is rotated by 90 degrees.

= 'NONE' can be used to disable automatic plotfile rotation in the driver program (i.e. for PostScript files).

Default: CMOD = 'LAND'.

Figure 6.1 shows the effect of PAGMOD:

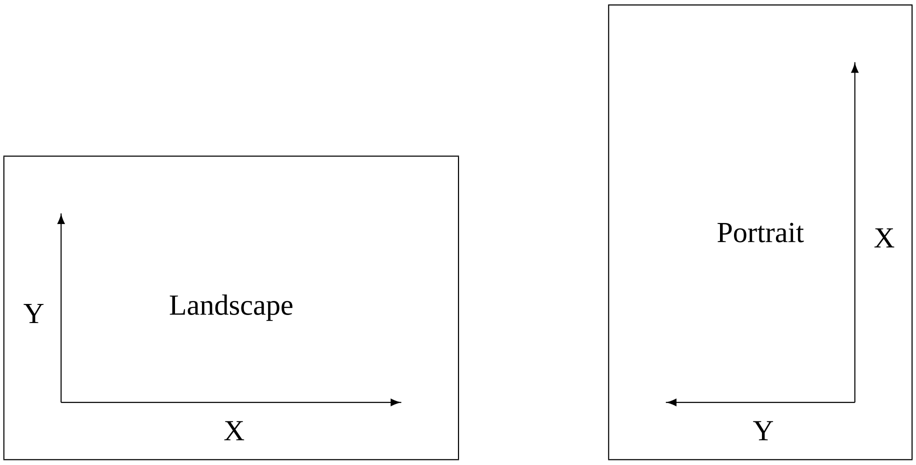


Figure 6.1: PAGMOD

NEWPAG

NEWPAG creates a new page.

The call is:	CALL NEWPAG	level 1
or:	void newpag ();	

Additional notes:
- PostScript, PDF and CGM files can store multiple pages. For other output formats, NEWPAG is not useful.
- On X Window terminals, NEWPAG is waiting for a mouse button 2 event before displaying the next page. This mode can be changed with the routine WINMOD. On other terminals, NEWPAG has the same effect as ERASE.

HWPAGE

The routine HWPAGE defines the size of the PostScript hardware page.

The call is:	CALL HWPAGE (NW, NH)	level 0
or:	void hwpage (int nw, int nh);	

NW, NH — are the width and height of the PostScript hardware page in plot coordinates. Default: (1950, 2800).

HWORIG

The routine HWORIG defines the hardware origin of the PostScript hardware page.

The call is:	CALL HWORIG (NX, NY)	level 0
or:	void hworig (int nx, int ny);	

NX, NY — are the plot coordinates of the hardware origin. Default: (75, 100).

6.1.6 Error Handling

E R R M O D

The printing of warnings and the output of the protocol in DISFIN can be disabled with the routine ERRMOD.

The call is:	CALL ERRMOD (CKEY, CMOD)	level 1, 2, 3
or:	void errmod (char *ckey, char *cmod);	

CKEY — is a character string that can have the values 'WARNINGS', 'CHECK', 'PROTOCOL' and 'ALL'. 'WARNINGS' means the error messages about bad parameters passed to DISLIN routines, 'CHECK' the out of range check of coordinates passed to plotting routines such as CURVE and 'PROTOCOL' the output of the protocol in DISFIN.

CMOD — is a character string that can have the values 'ON' and 'OFF'.
Default: ('ALL', 'ON')

E R R D E V

The routine ERRDEV defines the output device for DISLIN warnings. By default, warnings are written to the screen.

The call is:	CALL ERRDEV (COPT)	level 0
or:	void errdev (char *copt);	

COPT — is a character string that can have the values 'CONS' and 'FILE'.
Default: COPT = 'CONS'.

E R R F I L

By default, the name of the error file is 'dislin.err'. An alternate filename can be set with ERRFIL.

The call is:	CALL ERRFIL (CFIL)	level 0
or:	void errfil (char *cfil);	

CFIL — is a character string that contains the filename.

U N I T

UNIT defines the logical unit used for printing error messages and listing data points that lie outside of the axis scaling.

The call is:	CALL UNIT (NU)	level 1, 2, 3
or:	void unit (FILE *nu);	

NU — is the logical unit. If NU = 0, all messages will be suppressed.
Default: NU = 6

Additional note: UNIT is an old DISLIN routine for suppressing error messages. It should be replaced by the newer routines ERRMOD, ERRDEV and ERRFIL.

W I N A P P

The routine WINAPP defines if a DISLIN program should look like a Windows console, or more like a Windows program. If Windows mode is selected, all warnings are written to an error file and the protocol in disfin is displayed in a widget.

The call is: CALL WINAPP (COPT) level 0

or: void winapp (char *copt);

COPT is a character string that can have the values 'CONSOLE' and 'WINDOWS'.
Default: COPT = 'CONSOLE'.

6.1.7 Viewport Control

W I N D O W

This routine defines, for X Window terminals, a region on the screen where the graphics will be displayed. By default, the window size is set to 2/3 of the screen size and located in the lower right corner of the screen.

The call is: CALL WINDOW (NX, NY, NW, NH) level 0, 1, 2, 3

or: void window (int nx, int ny, int nw, int nh);

NX, NY are the screen coordinates of the upper left corner.

NW, NH are the width and height of the window in screen coordinates.

Additional note: In general, the screen size is 1280 * 1024 pixels.

W I N S I Z

This routine defines the size of windows and the resolution of DISLIN image formats such as TIFF, PNG, BMP, PPM and IMAGE. By default, the window size is set to 2/3 of the screen size, and the resolution of image formats is 853 x 603 pixels.

The call is: CALL WINSIZ (NW, NH) level 0, 1, 2, 3

or: void winsiz (int nw, int nh);

NW, NH are the width and height of the window in pixels.

C L R M O D

The routine CLRMOD defines the colour mode used for output on window terminals.

The call is: CALL CLRMOD (CMOD) level 0

or: void clrmod (char *cmod);

CMOD is a character string defining the mode.

= 'NONE' means that a colour table with 256 colours will be reduced to 129 colours to conserve current screen and window colours. The colour values will be reduced by the formula (0 ⇔ 0, i = (iclr + 1) / 2, iclr = 1, ... 255).

= 'FULL' means that all 256 colours will be displayed.

= 'CONT' means that a colour table with less than 129 entries will be used.
Default: CMOD = 'NONE'.

X 1 1 M O D

The routine X11MOD enables or disables backing store for graphic windows.

The call is: CALL X11MOD (CMOD) level 0

or:	void x11mod (char *cmod);

CMOD	is a character string containing the mode.
= ’NOSTORE’	means that graphical output is sent directly to the graphics window.
= ’STORE’	means that graphical output is sent to a pixmap that will be copied to the graphics window.
= ’AUTO’	means that ’NOSTORE’ will be used on X11 and ’STORE’ on Windows terminals.
= ’PIXMAP’	means that only a pixmap is used. The graphics window will be invisible.

Default: CMOD = ’AUTO’.

W I N M O D

The routine WINMOD affects the handling of windows in the termination routine DISFIN.

The call is:	CALL WINMOD (CMOD)	level 1, 2, 3
or:	void winmod (char *cmod);	

CMOD	is a character string containing the mode.
= ’FULL’	means that DISFIN is waiting for a mouse button 2 event. After program continuation, all windows are deleted.
= ’NOHOLD’	means that DISFIN is not waiting for a mouse button 2 event. After a call to DISFIN, all windows are deleted.
= ’NOERASE’	means that the program is still blocked in DISFIN but windows will not be deleted after program continuation.
= ’NONE’	means that the program is not blocked in DISFIN and windows are not deleted.
= ’DELAY’	means that the program is blocked for a short time in DISFIN before it is continued. The delay time can be defined with the routine WINOPT.

Default: CMOD = ’FULL’.

W I N O P T

The routine WINOPT sets the delay time for the keyword ’DELAY’ in WINMOD.

The call is:	CALL WINOPT (IOPT, CKEY)	level 1, 2, 3
or:	void winopt (int iopt, char *ckey);	

IOPT	is the delay time in seconds.
CKEY	is a character string that can have the value ’DELAY’.

Default: (10, ’DELAY’).

W I N K E Y

The routine WINKEY enables a an additional key that can be used for program continuation is DISFIN. Normally, the mouse button 2 can be used for closing the graphics window.

The call is:	CALL WINKEY (CKEY)	level 1, 2, 3
or:	void winkey (char *ckey);	

CKEY is a character string that can have the values 'NONE', 'RETURN' and 'ESCAPE'.

Default: CKEY = 'NONE'.

S E T X I D

The routine SETXID defines an external graphics window for X11 and Windows displays. All graphical output is sent to the external window. For X11 displays, an external pixmap can also be defined.

The call is: CALL SETXID (ID, CTYPE) level 0, 1, 2, 3

or: void setxid (int id, char *ctype);

ID is the window or pixmap ID.

CTYPE is a character string that can have the values 'NONE', 'WINDOW', 'PIXMAP' and 'WIDGET'. For the keyword 'WIDGET', the ID of a DISLIN draw widget can be used.

Default: (0, 'NONE').

Additional notes:

- If an external pixmap is used, backing store must also be enabled with the routine X11MOD.
- An external window is not erased by DISINI. This can be done with the routine ERASE.
- External windows are not blocked in DISFIN (see WINMOD).
- External windows can also be used for multiple DISLIN windows that are defined with the routine OPNWIN.

6.2 Axis Systems

This section describes subroutines that allow the user to modify axis systems. The position of an axis system, the size, the scaling, ticks, labels and axis titles can be altered in any way. Some of the routines defining axis attributes can also be used with secondary axes. Routines that set axis attributes can be used for one or for any combination of axes. The axes are identified by a character string that can contain the characters ’X’, ’Y’ and ’Z’ in any combination.

6.2.1 Modifying the Type

A X S T Y P

The routine AXSTYP defines the type of an axis system. Axis systems can be plotted as rectangles or in a crossed form. For crossed axis systems, the scaling must be linear and the axis limits must contain the origin.

The call is:	CALL AXSTYP (COPT)	level 1
or:	void axstyp (char *copt);	

COPT	is a character string defining the type.
= ’RECT’	defines a rectangular axis system.
= ’CROSS’	defines a crossed axis system.

Default: COPT = ’RECT’.

The following figure shows a rectangular and a crossed axis system:

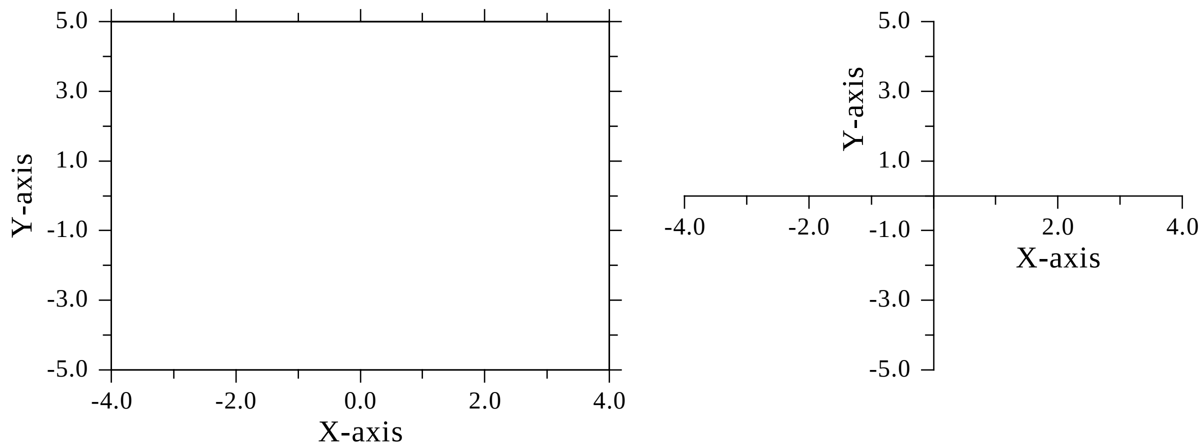


Figure 6.2: Rectangular and Crossed Axis Systems

6.2.2 Modifying the Position and Size

A X S P O S

AXSPOS determines the position of an axis system.

The call is:	CALL AXSPOS (NXA, NYA)	level 1
or:	void axspos (int nxa, int nya);	

NXA, NYA	are plot coordinates that define the lower left corner of an axis system. By default, axis systems are centred in the X-direction while NYA is set to the value (page height - 300).

A X S O R G

AXSORG is an alternate routine for defining the position of a crossed axis system.

The call is:	CALL AXSORG (NX, NY)	level 1
or:	void axsorg (int nx, int ny);	

NX, NY — are plot coordinates that define the position of the origin of a crossed axis system.

A X S L E N

AXSLEN defines the size of an axis system.

The call is:	CALL AXSLEN (NXL, NYL)	level 1
or:	void axslen (int nxl, int nyl);	

NXL, NYL — are the length and height of an axis system in plot coordinates. The default values are set to 2/3 of the page length and height.

C E N T E R

A call to the routine CENTER will centre the axis system on the page. All elements of an axis system, including titles, axis labels and names, will be taken into consideration. The centralisation is done by GRAF through changing the position of the origin. Therefore, all plotting routines called after GRAF will work with the new origin.

The call is:	CALL CENTER	level 1, 2, 3
or:	void center ();	

Additional notes:

- If there are several axis systems on the page, the origin will be changed only by the first call to GRAF.
- The character height of titles should be defined with HTITLE if it is different from the current character height in GRAF.

6.2.3 Axis Scaling

A X S S C L

This routine sets the axis scaling to logarithmic or linear.

The call is:	CALL AXSSCL (CSCL, CAX)	level 1, 2, 3
or:	void axsscl (char *cscl, char *cax);	

CSCL = ’LIN’ — denotes linear scaling.
= ’LOG’ — denotes logarithmic scaling.

CAX — is a character string that defines the axes.

Default: (’LIN’, ’XYZ’).

Additional notes: - For logarithmic scaling, the corresponding parameters in GRAF must be exponents of base 10.

- The routine AXSSCL replaces the DISLIN routine SCALE because SCALE is also a Fortran 90 intrinsic function.

S E T S C L

The parameters in GRAF will be calculated automatically by DISLIN if the routine SETSCL is used. In this case, GRAF must have dummy parameters in which DISLIN returns the calculated values.

The call is:	CALL SETSCL (XRAY, N, CAX)	level 1
or:	void setscl (float *xray, int n, char *cax);	

XRAY	is a vector that contains user coordinates. SETSCL calculates the minimum and maximum values of the data and stores them in a common block.
N	is the number of points in XRAY.
CAX	is a character string that defines the axes. CAX can have the additional values 'XRESET', 'YRESET', 'ZRESET' and 'RESET' for disabling automatic scaling. The parameter 'RESET' resets automatic scaling for all axes.

Additional notes:

- SETSCL can be used with linear and logarithmic scaling and with all label types.
- The calculation of scaling and label values is done by GRAF. The minimum and maximum of the data are always used for the lower and upper limits of an axis while even values are calculated for the labels.
- The number of digits after the decimal point will be set automatically.
- If the scaling of an axis is logarithmic, labels will be plotted with the format 'LOG'.

6.2.4 Modifying Ticks

T I C K S

This routine is used to define the number of ticks between axis labels.

The call is:	CALL TICKS (NTIC, CAX)	level 1, 2, 3
or:	void ticks (int ntic, char *cax);	

NTIC	is the number of ticks (≥ 0).
CAX	is a character string that defines the axes.

Default: (2, 'XYZ').

T I C P O S

This routine defines the position of ticks.

The call is:	CALL TICPOS (CPOS, CAX)	level 1, 2, 3
or:	void ticpos (char *cpos, char *cax);	

CPOS	is a character string defining the position.
= 'LABELS'	means that ticks will be plotted on the same side as labels.
= 'REVERS'	means that ticks will be plotted inside of an axis system.
= 'CENTER'	means that ticks will be centred on the axis line.
CAX	is a character string that defines the axes.

Default: ('LABELS', 'XYZ').

T I C L E N

TICLEN sets the lengths of major and minor ticks.

The call is:	CALL TICLEN (NMAJ, NMIN)	level 1, 2, 3
or:	void ticlen (int nmaj, int nmin);	

NMAJ	is the length of major ticks in plot coordinates (> 0).
NMIN	is the length of minor ticks in plot coordinates (> 0).

Default: (24, 16).

T I C M O D

The routine TICMOD modifies the plotting of minor tick marks on calendar axes. By default, a major tick is plotted at each date label and no minor ticks are plotted.

The call is:	CALL TICMOD (COPT, CAX)	level 1, 2, 3
or:	void ticmod (char *copt, char *cax);	

COPT	is a character string defining the tick marks.
= 'NONE'	means that no minor ticks will be plotted.
= 'DAYS'	means that ticks will be plotted for every day.
= 'MONTH'	means that ticks will be plotted for every month.
= 'DMONTH'	means that ticks will be plotted for every second month.
= 'QUARTER'	means that ticks will be plotted on the first of January, April, July and October.
= 'HALF'	means that ticks will be plotted on the first of January and July.
= 'YEAR'	means that ticks will be plotted for every year.
CAX	is a character string that defines the axes.

Default: ('NONE', 'XYZ').

L O G T I C

The appearance of minor ticks on logarithmic axes differs slightly from linear axes. By default, logarithmic minor ticks are generated automatically if the label step is 1 or -1 and if the number of ticks in TICKS is greater than 1. If the step has another value, minor ticks are plotted as specified in TICKS. This algorithm can be modified with LOGTIC.

The call is:	CALL LOGTIC (CMOD)	level 1, 2, 3
or:	void logtic (char *cmod);	

CMOD	is a character string defining the appearance of logarithmic ticks.
= 'AUTO'	defines default ticks.
= 'FULL'	means that logarithmic minor ticks will be generated for every cycle even if the label step is not 1 but some other integer.

Default: CMOD = 'AUTO'.

6.2.5 Modifying Labels

L A B E L S

LABELS determines which label types will be plotted on an axis.

The call is:	CALL LABELS (CLAB, CAX)	level 1, 2, 3
or:	void labels (char *clab, char *cax);	

CLAB	is a character string that defines the labels.
= 'NONE'	will suppress all axis labels.
= 'FLOAT'	will plot labels in floating-point format.
= 'EXP'	will plot floating-point labels in exponential format where fractions range between 1 and 10.
= 'FEXP'	will plot labels in the format fEn where f ranges between 1 and 10.
= 'LOG'	will plot logarithmic labels with base 10 and the corresponding exponents.
= 'CLOG'	is similar to 'LOG' except that the entire label is centred below the tick mark; with 'LOG', only the base '10' is centred.
= 'ELOG'	will plot only the logarithmic values of labels.
= 'TIME'	will plot time labels in the format 'hhmm'.
= 'HOURS'	will plot time labels in the format 'hh'.
= 'SECONDS'	will plot time labels in the format 'hhmmss'.
= 'DATE'	defines date labels.
= 'MAP'	defines geographical labels which are plotted as non negative floating-point numbers with the following characters 'W', 'E', 'N' and 'S'.
= 'LMAP'	is simular to 'MAP' except that lowercase characters are used.
= 'DMAP'	selects labels that are plotted as floating-point numbers with degree symbols.
= 'MYLAB'	selects labels that are defined with the routine MYLAB.
CAX	is a character string that defines the axes.

Default: ('FLOAT', 'XYZ').

Additional notes:

- The values 'LOG', 'CLOG' and 'ELOG' can be only used with logarithmic scaling. If these label types are used with linear scaling, DISLIN will change them to 'FLOAT'.
- For the values 'TIME', 'HOURS' and 'SECONDS', the corresponding parameters in GRAF must be in seconds since midnight.
- For the value 'DATE', the corresponding parameters in GRAF must be in days since a base date. The base date can be defined with the routine BASDAT while the number of days since the base date can be calculated with the routine INCDAT. Date labels can be modified with the routine LABMOD.

M Y L A B

MYLAB defines user labels.

The call is:	CALL MYLAB (CSTR, ITICK, CAX)	level 1, 2, 3
or:	void mylab (char *cstr, int itick, char *cax);	

CSTR	is a character string containing a label ($\leq$ 32 characters).
ITICK	is the tick number where the label will be plotted ($\leq$ 20). Tick numbering starts with 1.
CAX	is a character string that defines the axes.

L A B T Y P

LABTYP defines horizontal or vertical labels.

The call is:	CALL LABTYP (CTYPE, CAX)	level 1, 2, 3
or:	void labtyp (char *ctype, char *cax);	

CTYPE	is a character string defining the direction.
= 'HORI'	defines horizontal labels.
= 'VERT'	defines vertical labels.
CAX	is a character string that defines the axes.

Default: ('HORI', 'XYZ').

L A B P O S

LABPOS defines the position of labels.

The call is:	CALL LABPOS (CPOS, CAX)	level 1, 2, 3
or:	void labpos (char *cpos, char *cax);	

CPOS	is a character string defining the position.
= 'TICKS'	means that labels will be plotted at major ticks.
= 'CENTER'	means that labels will be centred between major ticks.
= 'SHIFT'	means that the starting and end labels will be shifted.
CAX	is a character string that defines the axes.

Default: ('TICKS', 'XYZ').

L A B J U S

LABJUS defines the alignment of axis labels.

The call is:	CALL LABJUS (CJUS, CAX)	level 1, 2, 3
or:	void labjus (char *cjus, char *cax);	

CJUS	is a character string defining the alignment of labels.
= 'AUTO'	means that labels are automatically justified.
= 'LEFT'	means that labels are left-justified.
= 'RIGHT'	means that labels are right-justified.
= 'OUTW'	means that labels are left-justified on the left and lower axes of an axis system. On the right and upper axes, labels are right-justified.
= 'INWA'	means that labels are right-justified on the left and lower axes of an axis system. On the right and upper axes, labels are left-justified.
CAX	is a character string that defines the axes.

Default: ('AUTO', 'XYZ').

L A B D I G

This routine sets the number of digits after the decimal point displayed in labels.

The call is:	CALL LABDIG (NDIG, CAX)	level 1, 2, 3
or:	void labdig (int ndig, char *cax);	

NDIG	= -1	defines integer labels.
	= 0	defines integer labels followed by a decimal point.
	= n	defines the number of digits after the decimal point. The last digit will be rounded up.
CAX		is a character string that defines the axes.

Default: (1, 'XYZ').

Additional note: The routine LABDIG replaces the DISLIN routine DIGITS because DIGITS is also a Fortran 90 intrinsic function.

I N T A X

With the routine INTAX, all axes will be labeled with integers.

The call is:	CALL INTAX	level 1, 2, 3
or:	void intax ();	

L A B D I S

This routine sets the distance between labels and ticks.

The call is:	CALL LABDIS (NDIS, CAX)	level 1, 2, 3
or:	void labdis (int ndis, char *cax);	

NDIS	is the distance in plot coordinates.
CAX	is a character string that defines the axes.

Default: (24, 'XYZ').

L A B M O D

The routine LABMOD modifies the appearance of date labels enabled with the keyword 'DATE' in the routine LABELS. Normally, date labels will be plotted in the form dd-mmm-yyyy.

The call is:	CALL LABMOD (CKEY, CVAL, CAX)	level 1, 2, 3
or:	void labmod (char *ckey, char *cval, char *cax);	

CKEY	is a character string containing one of the following keywords:
= 'YEAR'	means that the century field will be modified in date labels. For CKEY = 'YEAR', CVAL can have the values 'NONE', 'SHORT' and 'FULL'. 'NONE' suppresses the year field while 'SHORT' suppresses the century in the year field. The default value is 'FULL'.
= 'DAYS'	means that the day field will be modified. CVAL can have the values 'NONE', 'SHORT', 'LONG', 'NAME' and 'FULL'. For CVAL = 'NONE', the day field will be suppressed, for CVAL = 'SHORT', the day will be plotted as a number without a leading zero. CVAL = 'LONG' means that the day will be plotted as a number with two digits, CVAL = 'NAME' means that abbreviations of the weekday names will be plotted and CVAL = 'FULL' means that the full weekday names will be displayed. The default value is CVAL = 'LONG'.

= 'MONTH'	means that the month field will be modified. CVAL can have the values 'NONE', 'SHORT', 'LONG', 'NAME', 'TINY' and 'FULL'. For CVAL = 'NONE', the month field will be suppressed, for CVAL = 'SHORT', the month will be plotted as a number without a leading zero. CVAL = 'LONG' means that the month will be plotted as a number with two digits, CVAL = 'NAME' means that abbreviations of the month names will be plotted, CVAL = 'TINY' means that only the first character of month names will be plotted and CVAL = 'FULL' means that the full month names will be displayed. The default value is CVAL = 'NAME'.
= 'LANG'	defines the language used for weekdays and month names in date labels. CVAL can have the values 'ENGLISH' and 'GERMAN'. The default value for CVAL is 'ENGLISH'.
= 'FORM'	defines the order of the date fields. CVAL can have the values 'DMY', 'DYM', 'YDM', 'YMD', 'DYM' and 'MDY'. The default is CVAL = 'DMY'.
= 'SEPA'	defines a separator character used in date labels. CVAL is a character string containing the separator character. The default is CVAL = '-'.
= 'CASE'	defines if weekdays and month names are plotted in uppercase characters or in lowercase characters with a leading uppercase character. CVAL can have the values 'UPPER' and 'NONE'. The default value is 'NONE'.
= 'STEP'	defines a step between labels. CVAL can have the values 'DAYS', 'MONTH', 'DMONTH', 'QUARTER', 'HALF' and 'YEAR'. For CVAL = 'DAYS', the label step specified in the routine GRAF will be used. The default value is CVAL = 'DAYS'.
CAX	is a character string that defines the axes.

P O L M O D

The routine POLMOD modifies the appearance of angle labels plotted with the routine POLAR.

The call is:	CALL POLMOD (CPOS, CDIR)	level 1, 2, 3
or:	void polmod (char *cpos, char *cdir);	

CPOS	is a character string that defines the position of the first label. CPOS can have the values 'RIGHT', 'TOP', 'LEFT' and 'BOTTOM'.
CDIR	defines the direction of the labels. CDIR can have the values 'CLOCKWISE' and 'ANTICLOCK'.

Default: ('RIGHT', 'ANTICLOCK').

T I M O P T

With TIMOPT time labels can be plotted in the format 'hh:mm'. The default is 'hhmm'.

The call is:	CALL TIMOPT	level 1, 2, 3
or:	void timopt ();	

R G T L A B

The routine RGTLAB right-justifies user labels. By default, user labels are left-justified.

The call is:	CALL RGTLAB	level 1, 2, 3
or:	void rgtlab ();	

6.2.6 Modifying Axis Titles

N A M E

NAME defines axis titles.

The call is:	CALL NAME (CSTR, CAX)	level 1, 2, 3
or:	void name (char *cstr, char *cax);	

CSTR	is a character string containing the axis title (≤ 132 characters).
CAX	is a character string that defines the axes.

Default: (' ', 'XYZ').

H N A M E

HNAME defines the character height for axis names.

The call is:	CALL HNAME (NHNAME)	level 1, 2, 3
or:	void hname (int nhname);	

NHNAME	is the character height in plot coordinates.

Default: NHNAME = 36

N A M D I S

NAMDIS sets the distance between axis names and labels.

The call is:	CALL NAMDIS (NDIS, CAX)	level 1, 2, 3
or:	void namdis (int ndis, char *cax);	

NDIS	is the distance in plot coordinates.
CAX	is a character string that defines the axes.

Default: (30, 'XYZ').

N A M J U S

The routine NAMJUS defines the alignment of axis titles.

The call is:	CALL NAMJUS (CJUS, CAX)	level 1, 2, 3
or:	void namjus (char *cjus, char *cax);	

CJUS	is a character string that can have the values 'CENT', 'LEFT' and 'RIGHT'.
CAX	is a character string that defines the axes.

Default: ('CENT', 'XYZ').

R V Y N A M

The routine RVYNAM is used to plot names on right Y-axes and colour bars at an angle of 90 degrees. By default, they are plotted at an angle of 270 degrees.

The call is:	CALL RVYNAM	level 1, 2, 3
or:	void rvynam ();	

6.2.7 Suppressing Axis Parts

N O L I N E

After a call to NOLINE the plotting of axis lines will be suppressed.

The call is:	CALL NOLINE (CAX)	level 1, 2, 3
or:	void noline (char *cax);	

CAX	is a character string that defines the axes.

A X E N D S

With a call to AXENDS certain labels can be suppressed.

The call is:	CALL AXENDS (COPT, CAX)	level 1, 2, 3
or:	void axends (char *copt, char *cax);	

COPT	is a character string that defines which labels will be suppressed.
= ’NONE’	means that all labels will be displayed.
= ’FIRST’	means that only the starting label will be plotted.
= ’NOFIRST’	means that the starting label will not be plotted.
= ’LAST’	means that only the ending label will be plotted.
= ’NOLAST’	means that the ending label will not be plotted.
= ’ENDS’	means that only the start and end labels will be plotted.
= ’NOENDS’	means that start and end labels will be suppressed.
CAX	is a character string that defines the axes.

Default: (’NONE’, ’XYZ’).

N O G R A F

The routine NOGRAF suppresses the plotting of an axis system.

The call is:	CALL NOGRAF	level 1
or:	void nograf ();	

A X 2 G R F

The routine AX2GRF suppresses the plotting of the upper X- and left Y-axis.

The call is:	CALL AX2GRF	level 1, 2, 3
or:	void ax2grf ();	

S E T G R F

SETGRF removes a part of an axis or a complete axis from an axis system.

The call is:	CALL SETGRF (C1, C2, C3, C4)	level 1, 2, 3
or:	void setgrf (char *c1, char *c2, char *c3, char *c4);	

Ci — are character strings corresponding to the four axes of an axis system. C1 corresponds to the lower X-axis, C2 to the left Y-axis, C3 to the upper X-axis and C4 to the right Y-axis. The parameters can have the values 'NONE', 'LINE', 'TICKS', 'LABELS' and 'NAME'. With 'NONE', complete axes will be suppressed, with 'LINE', only axis lines will be plotted, with 'TICKS', axis lines and ticks will be plotted, with 'LABELS' axis lines, ticks and labels will be plotted and with 'NAME', all axis elements will be displayed.

Default: ('NAME', 'NAME', 'TICKS', 'TICKS').

Additional notes:

- By default, GRAF plots a frame of thickness 1 around axis systems. Therefore, in addition to the parameter 'NONE', FRAME should be called with the parameter 0 for suppressing complete axes.
- SETGRF does not reset the effect of NOGRAF and NOLINE. This must be done using RESET.

6.2.8 Modifying Clipping

C L P W I N

The routine CLPWIN defines a rectangular clipping area on the page.

The call is:	CALL CLPWIN (NX, NY, NW, NH)	level 1, 2, 3
or:	void clpwin (int nx, int ny, int nw, int nh);	

NX, NY — are the plot coordinates of the upper left corner.

NW, NH — are the width and height of the rectangle in plot coordinates.

C L P B O R

The routine CLPBOR sets the clipping area to the entire page or to the axis system.

The call is:	CALL CLPBOR (COPT)	level 1, 2, 3
or:	void clpbor (char *copt);	

COPT — is a character string that can have the values 'PAGE' and 'AXIS'.

Default: COPT = 'PAGE'.

N O C L I P

The suppressing of lines outside of the borders of an axis system can be disabled with NOCLIP.

The call is:	CALL NOCLIP	level 1, 2, 3
or:	void noclip ();	

G R A C E

GRACE defines a margin around axis systems where lines will be clipped.

The call is:	CALL GRACE (NGRA)	level 1, 2, 3
or:	void grace (int ngra);	

NGRA — is the width of the margin in plot coordinates. If NGRA is negative, lines will be clipped inside the axis system.

Default: NGRA = -1

6.2.9 Framing Axis Systems

F R A M E

FRAME defines the thickness of frames plotted by routines such as GRAF and LEGEND.

The call is:	CALL FRAME (NFRM)	level 1, 2, 3
or:	void frame (int nfrm);	

NFRM — is the thickness of the frame in plot coordinates. If NFRM is negative, the frame will be thickened from the inside. If positive, the frame will be thickened towards the outside.

Default: NFRM = 1

F R M C L R

The colour of frames can be defined with the routine FRMCLR.

The call is:	CALL FRMCLR (NCLR)	level 1, 2, 3
or:	void frmclr (int nclr);	

NCLR — is a colour value. If NCLR = -1, the current colour is used.

Default: NCLR = -1

6.2.10 Setting Colours

A X S B G D

The routine AXSBGD defines a background colour for axis systems.

The call is:	CALL AXSBGD (NCLR)	level 1, 2, 3
or:	void axsbgd (int nclr);	

NCLR — is a colour value. If NCLR = -1, the background of an axis system is not filled in GRAF.

Default: NCLR = -1

A X C L R S

AXCLRS selects colours for single parts of axes.

The call is:	CALL AXCLRS (NCLR, COPT, CAX)	level 1, 2, 3
or:	void axclrs (int nclr, char *copt, char *cax);	

NCLR — is a colour value. If NCLR = -1, the actual colour is used.

COPT — is a character string that can have the values ’LINE’, ’TICKS’, ’LABELS’, ’NAME’ and ’ALL’.

CAX — is a character string that defines the axes.

Default: (-1, ’ALL’, ’XYZ’).

Additional note: By default, a frame of thickness 1 is plotted around axis systems. This may overplot the colour of axis lines (see FRAME, FRMCLR).

6.2.11 Axis System Titles

T I T L I N

This subroutine defines up to four lines of text used for axis system titles. The text can be plotted with TITLE after a call to GRAF.

The call is:	CALL TITLIN (CSTR, N)	level 1, 2, 3
or:	void titlin (char *cstr, int n);	

CSTR — is a character string ($\leq$ 132 characters).

N — is an integer that contains a value between 1 and 4 or -1 and -4. If N is negative, the line will be underscored.

Default: All lines are filled with blanks.

T I T J U S

The routine TITJUS defines the alignment of title lines.

The call is:	CALL TITJUS (CJUS)	level 1, 2, 3
or:	void titjus (char *cjus);	

CJUS — is a character string that can have the values 'CENT', 'LEFT' and 'RIGHT'.

Default: CJUS = 'CENT'.

L F T T I T

Title lines are centred above axis systems by default but can be left-justified with a call to LFTTIT. This routine has the same meaning as TITJUS ('LEFT').

The call is:	CALL LFTTIT	level 1, 2, 3
or:	void lfttit ();	

T I T P O S

The routine TITPOS defines the position of title lines which can be plotted above or below axis systems.

The call is:	CALL TITPOS (CPOS)	level 1, 2, 3
or:	void titpos (char *cpos);	

CPOS — is a character string that can have the values 'ABOVE' and 'BELOW'.

Default: CPOS = 'ABOVE'.

L I N E S P

LINESP defines the spacing between title and legend lines.

The call is:	CALL LINESP (XFAC)	level 1, 2, 3
or:	void linesp (float xfac);	

XFAC — The space between lines is set to XFAC * character height.

Default: XFAC = 1.5

H T I T L E

HTITLE defines the character height for titles. The character height defined by HEIGHT will be used if HTITLE is not called.

The call is:	CALL HTITLE (NHCHAR)	level 1, 2, 3
or:	void htitle (int nhchar);	

NHCHAR is the character height in plot coordinates.

V K Y T I T

The space between titles and axis systems can be enlarged or reduced with VKYTIT. By default, the space is 2 * character height.

The call is:	CALL VKYTIT (NV)	level 1, 2, 3
or:	void vkytit (int nv);	

NV is an integer that determines the spacing between axis systems and titles. If NV is negative, the space will be reduced by NV plot coordinates. If NV is positive, the space will be enlarged by NV plot coordinates.

Default: NV = 0

6.3 Colours

This paragraph describes routines that modify colours. A colour value in DISLIN may be an entry of the current colour table, or an explicit RGB value. When specifying an explicit RGB value, the colour value must have the following hexadecimal form: 01bbggrr. The low-order byte contains the intensity of red, the second byte the intensity of green and the third byte the intensity of blue. The high-order byte must have the value 1. The function INTRGB creates an explicit RGB value from RGB coordinates. If the output device can only display 256 colours and an explicit RGB value is given, the nearest entry in the current colour table that matches the RGB coordinates will be used. Some routines define colours also by name such as COLOR, or by RGB coordinates such as SETRGB.

6.3.1 Changing the Foreground Colour

C O L O R

COLOR defines the colours used for plotting text and lines.

The call is:	CALL COLOR (CNAME)	level 1, 2, 3
or:	void color (char *cname);	

CNAME is a character string that can have the values ’BLACK’, ’RED’, ’GREEN’, ’BLUE’, ’CYAN’, ’YELLOW’, ’ORANGE’, ’MAGENTA’, ’WHITE’, ’FORE’ and ’BACK’. The keyword ’FORE’ resets the color to the default value, while the keyword ’BACK’ sets the colour to the background colour.

Additional note: The values ’BLACK’ and ’WHITE’ define not absolute colours. If the output format is in reverse mode, ’BLACK’ is interpreted as ’WHITE’ and ’WHITE’ is interpreted as ’BLACK’. If you want to use true black and true white, you can use the routine SETRGB (0., 0., 0.) and SETRGB (1., .1., 1.).

S E T C L R

The routine SETCLR sets the foreground colour where the colour can be specified as a colour table entry or as an explicit RGB colour.

The call is:	CALL SETCLR (NCOL)	level 1, 2, 3
or:	void setclr (int ncol);	

NCOL is a colour value.

Default: NCOL = 255 (White).

S E T R G B

The routine SETRGB defines the foreground colour specified in RGB coordinates.

The call is:	CALL SETRGB (XR, XG, XB)	level 1, 2, 3
or:	void setrgb (float xr, float xg, float xb);	

XR, XG, XB are the RGB coordinates of a colour in the range 0 to 1. If the output device cannot display true colours, SETRGB sets the nearest entry in the colour table that matches the RGB coordinates.

6.3.2 Modifying Colour Tables

S E T V L T

SETVLT selects a colour table.

The call is:	CALL SETVLT (CVLT)	level 1, 2, 3
or:	void setvlt (char *cvlt);	

CVLT	is a character string that defines the colour table.
= 'SMALL'	defines a small colour table with the 8 colours: 1 = BLACK, 2 = RED, 3 = GREEN, 4 = BLUE, 5 = YELLOW, 6 = ORANGE, 7 = CYAN and 8 = MAGENTA.
= 'VGA'	defines the 16 standard colours of a VGA graphics card.
= 'RAIN'	defines 256 colours arranged in a rainbow where 0 means black and 255 means white.
= 'SPEC'	defines 256 colours arranged in a rainbow where 0 means black and 255 means white. This colour table uses more violet colours than 'RAIN'.
= 'GREY'	defines 256 grey scale colours where 0 means black and 255 is white.
= 'RRAIN'	is the reverse colour table of 'RAIN'.
= 'RSPEC'	is the reverse colour table of 'SPEC'.
= 'RGREY'	is the reverse colour table of 'GREY'.
= 'TEMP'	defines a temperature colour table. The default colour table is 'RAIN'.

M Y V L T

The routine MYVLT changes the current colour table.

The call is:	CALL MYVLT (XR, XG, XB, N)	level 1, 2, 3
or:	void myvlt (float *xr, float *xg, float *xb, int n);	

XR, XG, XB	are arrays containing RGB coordinates in the range 0 to 1.
N	is the number of colour entries.

S E T I N D

The routine SETIND allows the user to change the current colour table.

The call is:	CALL SETIND (INDEX, XR, XG, XB)	level 1, 2, 3
or:	void setind (int index, float xr, float xg, float xb);	

INDEX	is an index between 0 and 255.
XR, XG, XB	are the RGB coordinates of a colour in the range 0 to 1.

V L T F I L

The routine VLTFIL saves the current colour table to a file, or loads a colour table from a file.

The call is:	CALL VLTFIL (CFIL, COPT)	level 1, 2, 3
or:	void vltfil (char *cfil, char *copt);	

CFIL	is a character string containing a filename. Colour entries are stored in the file as RGB coordinates in the range 0 to 1.
COPT	is a character string that can have the values 'SAVE' and 'LOAD'.

6.3.3 Utitily Routines for Colours

I N T R G B

The function INTRGB creates an explicit colour value from RGB coordinates.

The call is:	NCLR = INTRGB (XR, XG, XB)	level 1, 2, 3
or:	int intrgb (float xr, float xg, float xb);	

XR, XG, XB	are the RGB coordinates of a colour in the range 0 to 1.
NCLR	is the returned colour value.

I N D R G B

The function INDRGB returns the nearest entry in the current colour table that matches given RGB coordinates.

The call is:	N = INDRGB (XR, XG, XB)	level 1, 2, 3
or:	int indrgb (float xr, float xg, float xb);	

XR, XG, XB	are the RGB coordinates of a colour in the range 0 to 1.
N	is the returned colour index.

Sometimes, it is easier to specify colours as HSV coordinates where H is the hue, S the saturation and V the value of a colour. The following routines convert coordinates from the HSV to the RGB model and vice versa.

H S V R G B

The routine HSVRGB converts HSV coordinates to RGB coordinates.

The call is:	CALL HSVRGB (XH, XS, XV, XR, XG, XB)	level 1, 2, 3
or:	void hsvrgb (float xh, float xs, float xv, float *xr, float *xg, float *xb);	

XH, XS, XV	are the hue, saturation and value of a colour. XH must be in the range 0 to 360 degrees while XS and XV can have values between 0 and 1. In the HSV model, colours lie in a spectral order on a six-sided pyramid where red corresponds to the angle 0, green to 120 and blue to 240 degrees.
XR, XG, XB	are the RGB coordinates in the range 0 to 1 calculated by HSVRGB.

R G B H S V

The routine RGBHSV converts RGB coordinates to HSV coordinates.

The call is:	CALL RGBHSV (XR, XG, XB, XH, XS, XV)	level 1, 2, 3
or:	void rgbhsv (float xr, float xg, float xb, float *xh, float *xs, float *xv);	

6.4 Text and Numbers

H E I G H T

HEIGHT defines the character height.

The call is:	CALL HEIGHT (NHCHAR)	level 1, 2, 3
or:	void height (int nhchar);	

NHCHAR	is the character height in plot coordinates.

Default: NHCHAR = 36

A N G L E

This routine modifies the direction of text plotted with the routines MESSAG, NUMBER, RLMESS and RLNUMB.

The call is:	CALL ANGLE (NDEG)	level 1, 2, 3
or:	void angle (int ndeg);	

NDEG	is an angle measured in degrees and a counter-clockwise direction.

Default: NDEG = 0

T X T J U S

The routine TXTJUS defines the alignment of text plotted with the routines MESSAG and NUMBER.

The call is:	CALL TXTJUS (CJUS)	level 1, 2, 3
or:	void txtjus (char *cjus);	

CJUS is a character string that can have the values 'LEFT', 'RIGHT' and 'CENT'. The starting point of text and numbers will be interpreted as upper left, upper right and upper centre point.

Default: CJUS = 'LEFT'.

F R M E S S

FRMESS defines the thickness of frames around text plotted by MESSAG.

The call is:	CALL FRMESS (NFRM)	level 1, 2, 3
or:	void frmess (int nfrm);	

NFRM is the thickness of frames in plot coordinates. If NFRM is negative, frames will be thickened from the inside. If positive, frames will be thickened towards the outside.

Default: NFRM = 0

N U M F M T

NUMFMT modifies the format of numbers plotted by NUMBER and RLNUMB.

The call is:	CALL NUMFMT (COPT)	level 1, 2, 3
or:	void numfmt (char *copt);	

COPT is a character string defining the format.

= 'FLOAT' will plot numbers in floating-point format.

= 'EXP' will plot numbers in exponential format where fractions range between 1 and 10.

= 'FEXP' will plot numbers in the format fEn where f ranges between 1 and 10.

= 'LOG' will plot numbers logarithmically with base 10 and the corresponding exponents. The exponents must be passed to NUMBER and RLNUMB.

Default: COPT = 'FLOAT'.

Additional note: SETEXP and SETBAS alter the position and size of exponents.

N U M O D E

NUMODE alters the appearance of numbers plotted by NUMBER and RLNUMB.

The call is:	CALL NUMODE (CDEC, CGRP, CPOS, CFIX)	level 1, 2, 3
or:	void numode (char *cdec, char *cgrp, char *cpos, char *cfix);	

CDEC is a character string that defines the decimal notation.

= 'POINT' defines a point.

= 'COMMA' defines a comma.

CGRP is a character string that defines the grouping of 3 digits.

= 'NONE' means no grouping.

= 'SPACE' defines a space as separator.

= 'POINT' defines a point as separator.

= 'COMMA' defines a comma as separator.

CPOS is a character string that defines the sign preceding positive numbers.

= 'NONE'	means no preceding sign.
= 'SPACE'	defines a space as a preceding sign.
= 'PLUS'	defines a plus as a preceding sign.
CFIX	is a character string specifying character spacing.
= 'NOEQUAL'	is used for proportional spacing.
= 'EQUAL'	is used for non-proportional spacing.

Default: ('POINT','NONE','NONE','NOEQUAL').

CHASPC

CHASPC affects intercharacter spacing.

The call is: CALL CHASPC (XSPC) level 1, 2, 3

or: void chaspc (float xspc);

XSPC is a real number that contains a multiplier. If XSPC <0, the intercharacter spacing will be reduced by XSPC * NH plot coordinates where NH is the current character height. If XSPC >0, the spacing will be enlarged by XSPC * NH plot coordinates.

Default: XSPC = 0.

CHAWTH

CHAWTH affects the width of characters.

The call is: CALL CHAWTH (XWTH) level 1, 2, 3

or: void chawth (float xwth);

XWTH is a real number between 0 and 2. If XWTH <1, the character width will be reduced. If XWTH >1, the character width will be enlarged.

Default: XWTH = 1.

CHAANG

CHAANG defines an inclination angle for characters.

The call is: CALL CHAANG (ANGLE) level 1, 2, 3

or: void chaang (float angle);

ANGLE is the inclination angle between characters and the vertical direction in degrees (-60. $\leq$ ANGLE $\leq$60).

Default: ANGLE = 0.

FIXSPC

All fonts in DISLIN except for the default font are proportional. After a call to FIXSPC the characters of a proportional font will also be plotted with a constant character width.

The call is: CALL FIXSPC (XFAC) level 1, 2, 3

or: void fixspc (float xfac);

XFAC is a real number containing a scaling factor. Characters will be centred in a box of width XFAC * XMAX where XMAX is the largest character width of the current font.

6.5 Fonts

The following routines define character sets of varying style and plot velocity. All fonts except for the default font DISALF are proportional. Each font provides 6 alphabets.

The calls are:	CALL DISALF	- default font, single stroke, low resolution
	CALL SIMPLX	- single stroke font
	CALL COMPLX	- complex font
	CALL DUPLX	- double stroke font
	CALL TRIPLX	- triple stroke font
	CALL GOTHIC	- gothic font
	CALL SERIF	- complex shaded font
	CALL HELVE	- shaded font
	CALL HELVES	- shaded font with small characters

Additional note: If one of the shaded fonts SERIF, HELVE or HELVES is used, only the outlines of characters are plotted to minimize plotting time. With the statement CALL SHDCHA characters will be shaded.

P S F O N T

PSFONT defines a PostScript font.

The call is: CALL PSFONT (CFONT) level 1, 2, 3

or: void psfont (char *cfont);

CFONT is a character string containing the font. Standard font names in PostScript are:

Times-Roman	Courier
Times-Bold	Courier-Bold
Times-Italic	Courier-Oblique
Times-BoldItalic	Courier-BoldOblique
Helvetica	AvantGarde-Book
Helvetica-Bold	AvantGarde-Demi
Helvetica-Oblique	AvantGarde-BookOblique
Helvetica-BoldOblique	AvantGarde-DemiOblique
Helvetica-Narrow	Bookman-Light
Helvetica-Narrow-Bold	Bookman-LightItalic
Helvetica-Narrow-Oblique	Bookman-Demi
Helvetica-Narrow-BoldOblique	Bookman-DemiItalic
NewCenturySchlbk-Roman	Palatino-Roman
NewCenturySchlbk-Italic	Palatino-Italic
NewCenturySchlbk-Bold	Palatino-Bold
NewCenturySchlbk-BoldItalic	Palatino-BoldItalic
ZapfChancery-MediumItalic	Symbol
ZapfDingbats	

Additional notes: - The file format must be set to 'PS', 'EPS', 'PDF' or 'SVG' with the routine METAFL. For SVG files, the Times, Helvetica and Courier fonts can be used.

- Font names cannot be shortened. Some printers provide additional non-standard fonts. These fonts should be specified exactly in upper and lower characters as they are described in the printer manuals. PostScript suppresses any graphics if there is a syntax error in the font name. Standard font names are not case-sensitive.
- A call to a DISLIN font resets PostScript fonts.

W I N F N T

WINFNT defines a TrueType font for WMF files and screen output on Windows displays.

The call is:	CALL WINFNT (CFONT)	level 1, 2, 3
or:	void winfnt (char *cfont);	

CFONT is a character string containing the font. The following fonts can normally be used on the Windows 9x/NT/2000 operating system:

Courier New	Times New Roman Italic
Courier New Bold	Times New Roman Bold Italic
Courier New Italic	Arial
Courier New Bold Italic	Arial Bold
Times New Roman	Arial Italic
Times New Roman Bold	Arial Bold Italic

X 1 1 F N T

X11FNT defines an X11 font for screen output on X11 displays.

The call is:	CALL X11FNT (CFONT, COPT)	level 1, 2, 3
or:	void x11fnt (char *cfont, char *copt);	

CFONT is a character string containing the first part of an X11 font.

COPT is a character string containing the last part of an X11 font. IF COPT = 'STANDARD', the value '-*-*-*-*-iso8859-1' is used for the last part of an X11 font.

Additional note: - CFONT must begin and end with the separator '-' and must contain the first five fields of an X11 font. DISLIN adds then the point size and a transformation matrix to the font. IF COPT has not the value 'STANDARD', it must begin with the character '-' and contain the last 6 fields of an X11 font.

Here are some examples for the contents of CFONT:

-Adobe-Times-Medium-R-Normal-
-Adobe-Times-Bold-R-Normal-
-Adobe-Times-Bold-I-Normal-
-Adobe-Helvetica-Bold-R-Normal-
-Adobe-Courier-Medium-R-Normal-

B M P F N T

DISLIN contains some bitmap fonts that can be set with the routine BMPFNT. Bitmap fonts are allowed for screen output and for a bitmap file format. They can be used to increase the quality of directly created raster formats such as PNG and TIFF.

The call is:	CALL BMPFNT (CFONT)	level 1, 2, 3

or: void bmpfnt (char *cfont);

CFONT is a character string that can have the values 'COMPLEX', 'SIMPLEX' and 'HELVE'.

H W F O N T

The routine HWFONT sets a standard hardware font if hardware fonts are supported by the current file format. For example, if the file format is PostScript, the font 'Times-Roman' is used, if the file format is 'CONS' or 'XWIN', 'Times New Roman' is used for Windows and '-*-Times-Bold-R-Normal-' is used for X11. If no hardware fonts are supported, COMPLX is used.

The call is: CALL HWFONT level 1, 2, 3

or: void hwfont ();

C H A C O D

The routine CHACOD defines the coding of characters.

The call is: CALL CHACOD (COPT) level 1, 2, 3

or: void chacod (char *copt);

COPT is a character string that can have the values 'STANDARD', 'ISO1', 'ISO2' and 'ISO3'. If COPT = 'ISO1', characters in strings will be interpreted as ISO-Latin-1 coded, and if COPT = 'ISO2' or 'ISO3', characters will be interpreted as ISO-Latin-2 or ISO-Latin-3 coded. The DISLIN vector font 'COMPLX' and fonts defined by BMPFNT contain ISO-Latin-2 and ISO-Latin-3 characters.

Default: 'STANDARD'.

B A S A L F

BASALF defines the base alphabet.

The call is: CALL BASALF (CALPH) level 1, 2, 3

or: void basalf (char *calph);

CALPH is a character string that can have the values 'STANDARD', 'ITALIC', 'GREEK', 'SCRIPT', 'RUSSIAN' and 'MATHEMATIC'. These alphabets can be used with all fonts.

Default: 'STANDARD'.

S M X A L F

SMXALF defines shift characters to shift between the base and an alternate alphabet.

The call is: CALL SMXALF (CALPH, C1, C2, N) level 1, 2, 3

or: void smxalf (char *calph, char *c1, char *c2, int n);

CALPH is a character string containing an alphabet. In addition to the names in BASALF, CALPH can have the value 'INSTRUCTION'.

C1 is a character that shifts to the alternate alphabet.

C2 is a character that shifts back to the base alphabet. C1 and C2 may be identical. After the last plotted character of a character string, DISLIN automatically shifts back to the base alphabet.

N — is an integer between 1 and 6. Up to 6 alternate alphabets can be defined.

P S M O D E

The routine PSMODE enables Greek and Italic PostScript characters in Postscript fonts. By default, DISLIN vector characters are used for Greek and Italic characters in PostScript fonts

The call is: CALL PSMODE (COPT) level 1, 2, 3

or: void psmode (char *copt);

COPT — is a character string that can have the values 'NONE', 'GREEK', 'ITALIC' and 'BOTH'.

Default: 'NONE'.

E U S H F T

European characters can be plotted by using their character codes in text strings where different character codings are available (see CHACOD), or by defining a shift character that converts the following character into a European character. The routine EUSHFT defines shift characters for European characters.

The call is: CALL EUSHFT (COPT, CSHIFT) level 1, 2, 3

or: void eushft (char *copt, char *cshift);

COPT — is a character string that can have the values 'GERMAN', 'FRENCH', 'SPANISH', 'DANISH', 'ACUTE', 'GRAVE', 'CIRCUM' and 'TURKISH'.

CSHIFT — is a shift character. The character placed directly after CSHIFT will be plotted as the corresponding European character. Figure 6.3 shows a table of the possible European characters.

Additional notes:

- Shift characters can be defined multiple where the characters must be different.
- The Turkish characters are only supported by COMPLX and by the bitmap fonts defined with BITMAP. The other European characters are also supported by PostScript.
- If the shift characters should be plotted in a text string, they must be doubled.

The following table shows all possible European characters. The characters on the left side of a column are shifted to the characters on the right side of that column:

GERMAN		DANISH		SPANISH		FRENCH		ACUTE		GRAVE		CIRCUM	
A	Ä	A	Å	N	Ñ	C	Ç	A	Á	A	À	A	Â
O	Ö	O	Ø	n	ñ	c	ç	E	É	E	È	E	Ê
U	Ü	E	Æ	!	¡	E	Ë	I	Í	I	Ì	I	Î
a	ä	a	å	?	¿	I	Ï	O	Ó	O	Ò	O	Ô
o	ö	o	ø			e	ë	U	Ú	U	Ù	U	Û
u	ü	e	æ			i	ï	a	á	a	à	a	â
s	ß							e	é	e	è	e	ê
								i	í	i	ì	i	î
								o	ó	o	ò	o	ô
								u	ú	u	ù	u	û

Figure 6.3: EUSHFT Character Set

Example:

```
PROGRAM EUSHFT
CALL METAFL ('CONS')
CALL DISINI
CALL PAGERA
CALL HWFONT

CALL EUSHFT ('GERMAN', '!')
CALL MESSAG ('!A, !O, !U, !a, !o, !u, !s', 100, 100)
CALL DISFIN
END
```

The next figures show several software and PostScript fonts that can be used in DISLIN. The full set of special European characters (ASCII code $>$ 126) is available in the software font COMPLX and in PostScript, X11 and TrueType fonts. The coding of the characters in figure 6.10 is the default character coding in DISLIN. An ISO-Latin-1 coding of characters can be defined with the DISLIN routine CHACOD.

DISALF

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	*B*	B	100	d	*d*	δ
33	!	*!*	!	67	C	*C*	Γ	101	e	*e*	ε
34	"	*"*	"	68	D	*D*	Δ	102	f	*f*	φ
35	#	*#*	#	69	E	*E*	E	103	g	*g*	χ
36	$	*$*	$	70	F	*F*	Φ	104	h	*h*	η
37	%	*%*	%	71	G	*G*	X	105	i	*i*	ι
38	&	*&*	&	72	H	*H*	H	106	j	*j*	?
39	'	*'*	'	73	I	*I*	I	107	k	*k*	κ
40	(	*(*	(	74	J	*J*	?	108	l	*l*	λ
41	)	*)*	)	75	K	*K*	K	109	m	*m*	μ
42	*	***	*	76	L	*L*	Λ	110	n	*n*	ν
43	+	*+*	+	77	M	*M*	M	111	o	*o*	ο
44	,	*,*	,	78	N	*N*	N	112	p	*p*	π
45	-	*-*	-	79	O	*O*	O	113	q	*q*	ϑ
46	.	*.*	.	80	P	*P*	Π	114	r	*r*	ρ
47	/	*/*	/	81	Q	*Q*	Θ	115	s	*s*	σ
48	0	*0*	0	82	R	*R*	P	116	t	*t*	τ
49	1	*1*	1	83	S	*S*	Σ	117	u	*u*	ψ
50	2	*2*	2	84	T	*T*	T	118	v	*v*	?
51	3	*3*	3	85	U	*U*	Ψ	119	w	*w*	ω
52	4	*4*	4	86	V	*V*	?	120	x	*x*	ξ
53	5	*5*	5	87	W	*W*	Ω	121	y	*y*	υ
54	6	*6*	6	88	X	*X*	Ξ	122	z	*z*	ζ
55	7	*7*	7	89	Y	*Y*	Y	123	{	*{*	{
56	8	*8*	8	90	Z	*Z*	Z	124	|	*|*	|
57	9	*9*	9	91	[	*[*	[	125	}	*}*	}
58	:	*:*	:	92	\	**	\	126	~	*~*	~
59	;	*;*	;	93	]	*]*	]	127	Ä	*Ä*	?
60	<	*<*	<	94	^	*^*	^	128	Ö	*Ö*	?
61	=	*=*	=	95	_	*_*	_	129	Ü	*Ü*	?
62	>	*>*	>	96	`	*`*	`	130	ä	*ä*	?
63	?	*?*	?	97	a	*a*	α	131	ö	*ö*	?
64	@	*@*	@	98	b	*b*	β	132	ü	*ü*	?
65	A	*A*	A	99	c	*c*	γ	133	ß	*ß*	?

Figure 6.4: DISALF Character Set

SIMPLX

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	*B*	Β	100	d	*d*	δ
33	!	*!*	!	67	C	*C*	Γ	101	e	*e*	ε
34	"	"	"	68	D	*D*	Δ	102	f	*f*	φ
35	#	#	#	69	E	*E*	Ε	103	g	*g*	χ
36	$	$	$	70	F	*F*	Φ	104	h	*h*	η
37	%	%	%	71	G	*G*	Χ	105	i	*i*	ι
38	&	&	&	72	H	*H*	Η	106	j	*j*	?
39	'	'	'	73	I	*I*	Ι	107	k	*k*	κ
40	(	(	(	74	J	*J*	?	108	l	*l*	λ
41	)	)	)	75	K	*K*	Κ	109	m	*m*	μ
42	*	*	*	76	L	*L*	Λ	110	n	*n*	ν
43	+	+	+	77	M	*M*	Μ	111	o	*o*	ο
44	,	,	,	78	N	*N*	Ν	112	p	*p*	π
45	—	—	—	79	O	*O*	Ο	113	q	*q*	θ
46	.	.	.	80	P	*P*	Π	114	r	*r*	ρ
47	/	/	/	81	Q	*Q*	Θ	115	s	*s*	σ
48	0	*0*	0	82	R	*R*	Ρ	116	t	*t*	τ
49	1	*1*	1	83	S	*S*	Σ	117	u	*u*	ψ
50	2	*2*	2	84	T	*T*	Τ	118	v	*v*	?
51	3	*3*	3	85	U	*U*	Ψ	119	w	*w*	ω
52	4	*4*	4	86	V	*V*	?	120	x	*x*	ξ
53	5	*5*	5	87	W	*W*	Ω	121	y	*y*	υ
54	6	*6*	6	88	X	*X*	Ξ	122	z	*z*	ζ
55	7	*7*	7	89	Y	*Y*	Υ	123	{	{	{
56	8	*8*	8	90	Z	*Z*	Ζ	124	|	|	|
57	9	*9*	9	91	[	[	[	125	}	}	}
58	:	:	:	92	\	\	\	126	~	~	~
59	;	;	;	93	]	]	]	127	Ä	*Ä*	?
60	<	<	<	94	^	^	^	128	Ö	*Ö*	?
61	=	=	=	95	_	_	_	129	Ü	*Ü*	?
62	>	>	>	96	`	`	`	130	ä	*ä*	?
63	?	?	?	97	a	*a*	α	131	ö	*ö*	?
64	@	@	@	98	b	*b*	β	132	ü	*ü*	?
65	A	*A*	Α	99	c	*c*	γ	133	ß	*ß*	?

Figure 6.5: SIMPLX Character Set

COMPLX

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	*B*	Β	100	d	*d*	δ
33	!	*!*	!	67	C	*C*	Γ	101	e	*e*	ε
34	''	*''*	''	68	D	*D*	Δ	102	f	*f*	φ
35	#	*#*	#	69	E	*E*	Ε	103	g	*g*	χ
36	$	*$*	$	70	F	*F*	Φ	104	h	*h*	η
37	%	*%*	%	71	G	*G*	Χ	105	i	*i*	ι
38	&	*&*	&	72	H	*H*	Η	106	j	*j*	?
39	'	*'*	'	73	I	*I*	Ι	107	k	*k*	κ
40	(	*(*	(	74	J	*J*	?	108	l	*l*	λ
41	)	*)*	)	75	K	*K*	Κ	109	m	*m*	μ
42	*	*	*	76	L	*L*	Λ	110	n	*n*	ν
43	+	*+*	+	77	M	*M*	Μ	111	o	*o*	ο
44	,	*,*	,	78	N	*N*	Ν	112	p	*p*	π
45	—	—	—	79	O	*O*	Ο	113	q	*q*	θ
46	.	.	.	80	P	*P*	Π	114	r	*r*	ρ
47	/	*/*	/	81	Q	*Q*	Θ	115	s	*s*	σ
48	0	*0*	0	82	R	*R*	Ρ	116	t	*t*	τ
49	1	*1*	1	83	S	*S*	Σ	117	u	*u*	ψ
50	2	*2*	2	84	T	*T*	Τ	118	v	*v*	?
51	3	*3*	3	85	U	*U*	Ψ	119	w	*w*	ω
52	4	*4*	4	86	V	*V*	?	120	x	*x*	ξ
53	5	*5*	5	87	W	*W*	Ω	121	y	*y*	υ
54	6	*6*	6	88	X	*X*	Ξ	122	z	*z*	ζ
55	7	*7*	7	89	Y	*Y*	Υ	123	{	*{*	{
56	8	*8*	8	90	Z	*Z*	Ζ	124	|	*/*	|
57	9	*9*	9	91	[	*[*	[	125	}	*}*	}
58	:	*:*	:	92	\	**	\	126	~	*~*	~
59	;	*;*	;	93	]	*]*	]	127	Ä	*Ä*	?
60	<	*<*	<	94	^	*^*	^	128	Ö	*Ö*	?
61	=	*=*	=	95	_	*_*	_	129	Ü	*Ü*	?
62	>	*>*	>	96	`	*`*	`	130	ä	*ä*	?
63	?	*?*	?	97	a	*a*	α	131	ö	*ö*	?
64	@	*@*	@	98	b	*b*	β	132	ü	*ü*	?
65	A	*A*	A	99	c	*c*	γ	133	ß	*ß*	?

Figure 6.6: COMPLX Character Set

COMPLX

ASCII	SCRI.	RUSS.	MATH.	ASCII	SCRI.	RUSS.	MATH.	ASCII	SCRI.	RUSS.	MATH.
32				66	B	Б	≡	100	d	д	↓
33	!	!	!	67	C	Э	⟩	101	e	й	⇒
34	''	''	''	68	D	Д	×	102	f	ф	⇐
35	#	#	#	69	E	Й	÷	103	g	г	⇔
36	$	Ъ	$	70	F	Ф	±	104	h	ж	⊕
37	%	Ы	%	71	G	Г	∓	105	i	и	⊖
38	&	Ь	&	72	H	Ж	≤	106	j	ч	⊙
39	'	'	'	73	I	И	≠	107	k	к	∀
40	(	(	(	74	J	Ч	≥	108	l	л	∞
41	)	)	)	75	K	К	⊥	109	m	м	∂
42	*	*	*	76	L	Л	∩	110	n	н	∇
43	+	+	+	77	M	М	∪	111	o	о	¬
44	,	,	,	78	N	Н	⊂	112	p	п	∨
45	—	—	—	79	O	О	⊃	113	q	ш	∧
46	.	.	.	80	P	П	∧	114	r	р	∑
47	/	/	/	81	Q	Ш	∨	115	s	с	∏
48	0	0	0	82	R	Р	⊆	116	t	т	⋃
49	1	1	1	83	S	С	⊇	117	u	ю	⋂
50	2	2	2	84	T	Т	∥	118	v	в	∫
51	3	3	3	85	U	Ю	∃	119	w	щ	∮
52	4	4	4	86	V	В	∄	120	x	х	√
53	5	5	5	87	W	Щ	∈	121	y	у	∅
54	6	6	6	88	X	Х	∉	122	z	з	≅
55	7	7	7	89	Y	У	∋	123	{	е	{
56	8	8	8	90	Z	З	∌	124	/	ё	$\vert$
57	9	9	9	91	[	Е	[	125	}	ц	}
58	:	:	:	92	\	Ё	\	126	~	я	~
59	;	;	;	93	]	Ц	]	127	Ä	?	?
60	<	ъ	<	94	^	Я	^	128	Ö	?	?
61	=	ы	=	95	_	_	_	129	Ü	?	?
62	>	ь	>	96	`	`	`	130	ä	?	?
63	?	?	?	97	a	а	→	131	ö	?	?
64	@	@	@	98	b	б	↑	132	ü	?	?
65	A	А	⟨	99	c	э	←	133	β	?	?

Figure 6.7: COMPLX Character Set

GOTHIC

ASCII	STAN.	ITAL.	SCRI.	ASCII	STAN.	ITAL.	SCRI.	ASCII	STAN.	ITAL.	SCRI.
32				66	B	B	B	100	d	d	d
33	!	!	!	67	C	C	C	101	e	e	e
34	''	''	''	68	D	D	D	102	f	f	f
35	#	#	#	69	E	E	E	103	g	g	g
36	$	$	$	70	F	F	F	104	h	h	h
37	%	%	%	71	G	G	G	105	i	i	i
38	&	&	&	72	H	H	H	106	j	j	j
39	'	'	'	73	I	I	I	107	k	k	k
40	(	(	(	74	J	J	J	108	l	l	l
41	)	)	)	75	K	K	K	109	m	m	m
42	*	*	*	76	L	L	L	110	n	n	n
43	+	+	+	77	M	M	M	111	o	o	o
44	,	,	,	78	N	N	N	112	p	p	p
45	—	—	—	79	O	O	O	113	q	q	q
46	.	.	.	80	P	P	P	114	r	r	r
47	/	/	/	81	Q	Q	Q	115	s	s	ſ
48	0	0	0	82	R	R	R	116	t	t	t
49	1	1	1	83	S	S	S	117	u	u	u
50	2	2	2	84	T	T	T	118	v	v	v
51	3	3	3	85	U	U	U	119	w	w	w
52	4	4	4	86	V	V	V	120	x	x	x
53	5	5	5	87	W	W	W	121	y	y	y
54	6	6	6	88	X	X	X	122	z	z	z
55	7	7	7	89	Y	Y	Y	123	{	{	{
56	8	8	8	90	Z	Z	Z	124	|	|	|
57	9	9	9	91	[	[	[	125	}	}	}
58	:	:	:	92	\	\	\	126	~	~	~
59	;	;	;	93	]	]	]	127	?	?	Ä
60	<	<	<	94	^	^	^	128	?	?	Ö
61	=	=	=	95	_	_	_	129	?	?	Ü
62	>	>	>	96	`	`	`	130	?	?	ä
63	?	?	?	97	a	a	a	131	?	?	ö
64	@	@	@	98	b	b	b	132	?	?	ü
65	A	A	A	99	c	c	c	133	?	?	ß

Figure 6.8: GOTHIC Character Set

HELVE

ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK	ASCII	STAN.	ITAL.	GREEK
32				66	B	B	Β	100	d	d	δ
33	!	!	!	67	C	C	Γ	101	e	e	ε
34	"	"	"	68	D	D	Δ	102	f	f	φ
35	#	#	#	69	E	E	Ε	103	g	g	χ
36	$	$	$	70	F	F	Φ	104	h	h	η
37	%	%	%	71	G	G	Χ	105	i	i	ι
38	&	&	&	72	H	H	Η	106	j	j	?
39	'	'	'	73	I	I	Ι	107	k	k	κ
40	(	(	(	74	J	J	?	108	l	l	λ
41	)	)	)	75	K	K	Κ	109	m	m	μ
42	*	*	*	76	L	L	Λ	110	n	n	ν
43	+	+	+	77	M	M	Μ	111	o	o	ο
44	,	,	,	78	N	N	Ν	112	p	p	π
45	-	-	-	79	O	O	Ο	113	q	q	θ
46	.	.	.	80	P	P	Π	114	r	r	ρ
47	/	/	/	81	Q	Q	Θ	115	s	s	σ
48	0	0	0	82	R	R	Ρ	116	t	t	τ
49	1	1	1	83	S	S	Σ	117	u	u	ψ
50	2	2	2	84	T	T	Τ	118	v	v	?
51	3	3	3	85	U	U	Ψ	119	w	w	ω
52	4	4	4	86	V	V	?	120	x	x	ξ
53	5	5	5	87	W	W	Ω	121	y	y	υ
54	6	6	6	88	X	X	Ξ	122	z	z	ζ
55	7	7	7	89	Y	Y	Υ	123	{	{	{
56	8	8	8	90	Z	Z	Ζ	124	|	|	|
57	9	9	9	91	[	[	[	125	}	}	}
58	:	:	:	92	\	\	\	126	~	~	~
59	;	;	;	93	]	]	]	127	Ä	Ä	?
60	<	<	<	94	^	^	^	128	Ö	Ö	?
61	=	=	=	95	_	_	_	129	Ü	Ü	?
62	>	>	>	96	‘	‘	‘	130	ä	ä	?
63	?	?	?	97	a	a	α	131	ö	ö	?
64	@	@	@	98	b	b	β	132	ü	ü	?
65	A	A	Α	99	c	c	γ	133	ß	ß	?

Figure 6.9: HELVE Character Set

Times-Roman

ASCII	CHAR	ASCII	CHAR	ASCII	CHAR	ASCII	CHAR	ASCII	CHAR
32		63	?	94	^	125	}	156	ó
33	!	64	@	95	_	126	~	157	ú
34	"	65	A	96	‘	127	Ä	158	À
35	#	66	B	97	a	128	Ö	159	È
36	$	67	C	98	b	129	Ü	160	Ì
37	%	68	D	99	c	130	ä	161	Ò
38	&	69	E	100	d	131	ö	162	Ù
39	’	70	F	101	e	132	ü	163	à
40	(	71	G	102	f	133	ß	164	è
41	)	72	H	103	g	134	Å	165	ì
42	*	73	I	104	h	135	Ø	166	ò
43	+	74	J	105	i	136	Æ	167	ù
44	,	75	K	106	j	137	å	168	Â
45	-	76	L	107	k	138	ø	169	Ê
46	.	77	M	108	l	139	æ	170	Î
47	/	78	N	109	m	140	Ñ	171	Ô
48	0	79	O	110	n	141	ñ	172	Û
49	1	80	P	111	o	142	Ç	173	â
50	2	81	Q	112	p	143	ç	174	ê
51	3	82	R	113	q	144	Ë	175	î
52	4	83	S	114	r	145	Ï	176	ô
53	5	84	T	115	s	146	ë	177	û
54	6	85	U	116	t	147	ï	178	Ã
55	7	86	V	117	u	148	Á	179	ã
56	8	87	W	118	v	149	É	180	Õ
57	9	88	X	119	w	150	Í	181	õ
58	:	89	Y	120	x	151	Ó	182	Ý
59	;	90	Z	121	y	152	Ú	183	ý
60	<	91	[	122	z	153	á	184	ÿ
61	=	92	\	123	{	154	é	185	¡
62	>	93	]	124	|	155	í	186	¿

Figure 6.10: Times-Roman Character Set

PostScript Fonts

This is Times-Roman
This is Times-Bold
This is Times-Italic
This is Times-BoldItalic
This is Helvetica
This is Helvetica-Bold
This is Helvetica-Oblique
This is Helvetica-BoldOblique
This is Helvetica-Narrow
This is Helvetica-Narrow-Bold
This is Helvetica-Narrow-Oblique
This is Helvetica-Narrow-BoldOblique
This is NewCenturySchlbk-Roman
This is NewCenturySchlbk-Italic
This is NewCenturySchlbk-Bold
This is NewCenturySchlbk-BoldItalic
This is ZapfChancery-MediumItalic

This is Courier
This is Courier-Bold
This is Courier-Oblique
This is Courier-BoldOblique
This is AvantGarde-Book
This is AvantGarde-Demi
This is AvantGarde-BookOblique
This is AvantGarde-DemiOblique
This is Bookman-Light
This is Bookman-LightItalic
This is Bookman-Demi
This is Bookman-DemiItalic
This is Palatino-Roman
This is Palatino-Italic
This is Palatino-Bold
This is Palatino-BoldItalic
Τηισ ισ Σψμβολ

Figure 6.11: PostScript Fonts

6.6 Indices and Exponents

Indices and exponents can be plotted by using control characters in characters strings, or by using the TeX syntax described in paragraph 6.7. There are 3 predefined control characters in DISLIN which can be altered with the routines NEWMIX and SETMIX. The predefined character

[is used for exponents. The character height is reduced by the scaling factor FEXP and the pen is moved up FBAS * NH plot coordinates where NH is the current character height.

] is used for indices. The pen is moved down FBAS * NH plot coordinates and the character height is reduced by the scaling factor FEXP.

$ is used to move the pen back to the base-line. This will automatically be done at the end of a character string.

FBAS and FEXP have the default values 0.6 and 0.8, respectively, these values can be changed with the routines SETBAS and SETEXP.

M I X A L F

This routine instructs DISLIN to search for control characters in character strings.

The call is:	CALL MIXALF	level 1, 2, 3
or:	void mixalf ();	

S E T B A S

SETBAS defines the position of indices and exponents. This routine also affects logarithmic axis labels.

The call is:	CALL SETBAS (FBAS)	level 1, 2, 3
or:	void setbas (float fbas);	

FBAS is a real number used as a scaling factor. The pen will be moved up or down by FBAS * NH plot coordinates to plot exponents or indices. NH is the current character height.

Default: FBAS = 0.6.

S E T E X P

SETEXP sets the character height of indices and exponents.

The call is:	CALL SETEXP (FEXP)	level 1, 2, 3
or:	void setexp (float fexp);	

FEXP is a real number used as a scaling factor. The character height of indices and exponents is set to FEXP * NH where NH is the current character height.

Default: FEXP = 0.8

N E W M I X

NEWMIX defines an alternate set of control characters for plotting indices and exponents. The default characters '[', ']' and '$' are replaced by '^', '_' and '%'.

The call is:	CALL NEWMIX	level 1, 2, 3

or:	void newmix ();

S E T M I X

SETMIX defines global control characters for plotting indices and exponents.

The call is:	CALL SETMIX (C, CMIX)	level 1, 2, 3
or:	void setmix (char *c, char *cmix);	

C	is a new control character.
CMIX	is a character string that defines the function of the control character. CMIX can have the values ’EXP’, ’IND’, ’RES’, ’LEG’ and ’TEX’ for exponents, indices, resetting the base-line, for multiple text lines in legends and for TeX instructions, respectively.
Additional note:	The routines NEWMIX and SETMIX only modify the control characters. A call to MIXALF is always necessary to plot indices and exponents.

6.7 Instruction Alphabet

The instruction alphabet contains commands that control pen movements and character sizes during the plotting of character strings. It is provided for the representation of complicated formulas. An alternate method for plotting of complicated formulas is described in paragraph 6.7, “TeX Instructions for Mathematical Formulas”.
The instruction alphabet can be used in the same way as other alphabets in DISLIN. Shift characters must be defined with the routine SMXALF to switch between the base and the instruction alphabet.
The commands of the instruction alphabet consist of a single character and an optional parameter. If the parameter is omitted, DISLIN will use default values. A parameter can be a real number, an integer or the character ’X’ which resets the parameter back to the entry value at the beginning of the character string.
Commands of the instruction alphabet can only change plot parameters temporarily within a character string. At the end of a character string, all parameters are reset to their entry values.

The following table summarizes all instruction commands. The character r means a real parameter and i an integer. The base-line of character strings is placed directly below them. Commands can be given in uppercase or lowercase letters. Real parameters can be specified without decimal points while integer parameters cannot have decimal points. Several commands can follow one another. Blanks between commands will be ignored.

Instruction-Alphabet

Command	Parameter	Default	Description
A	real	1.	moves the pen horizontally by r * NH plot coordinates where NH is the current character height. If r < 0, the pen will be moved backwards.
C	integer	1	moves the pen horizontally by i character spaces. If i < 0, the pen will be moved backwards.
D	real	1.	moves the pen down from the base-line by r * NH plot coordinates. If r > 0, NH is the entry character height. If r < 0, NH is the current character height.
E			moves the pen up by 0.75 * character height and reduces the character height by the scaling factor 0.6 (for exponents).
F	integer	1	moves the pen horizontally by i spaces. If i is negative, the pen is moved backwards.
G	integer	1	moves the pen horizontally to the tab position with the index i, where $1 \leq i \leq 20$.
H	real	0.6	sets the character height to r * NH. If r > 0, NH is the entry character height. If r < 0, NH is the current character height.
I			moves the pen down by 0.35 * character height and multiplies the character height by 0.6 (for indices).
J	integer	1	underscores twice from the tab position i to the current pen position.
K	real	0.8	is used to plot characters with constant widths. Characters will be centred in a box with the width r * W where W is the largest character length in the current font. The global routine is FIXSPC.
L	integer	1	underscores from the tab position i to the current pen position.
M	integer	1	defines the base alphabet. (1 = STAND., 2 = GREEK, 3 = MATH., 4 = ITAL., 5 = SCRIPT, 6 = RUSSIAN).

Command	Parameter	Default	Description
N	integer	1	sets a colour i, where $0 \leq i \leq 255$). The global routine is SETCLR.
O	real	0.	moves the base-line vertically by r * character height. If $r < 0$ the base-line is moved down.
P	integer	1	defines a horizontal tab position with the index i at the current pen position, where $1 \leq i \leq 20$. All tab positions are initialized to the beginning of the string.
R			resets the character height and the base-line to their entry values.
S	integer	0	plots a symbol with the number i, where $0 \leq i \leq 21$.
T	integer	0	moves the pen horizontally from the beginning of the string by i plot coordinates.
U	real	1.	moves the pen up from the base-line by r * NH plot coordinates. If $r > 0$, NH is the entry character height. If $r < 0$, NH is the current character height.
V	integer	1	plots a horizontal line from the tab position i to the current pen position. The line is moved up from the base-line by 0.5 * character height plot coordinates.
W	real	1.	affects the width of characters. The global routine is CHAWTH.
Y	real	0.	affects the character spacing. The global routine is CHASPC.
Z	real	0.	defines an inclination angle for characters, where $-60 \leq r \leq 60$. The global routine is CHAANG.

For the following examples, the characters '{' and '}' are defined with

CALL SMXALF ('INST', '{', '}', 1)

to switch between the instruction and the base alphabet.

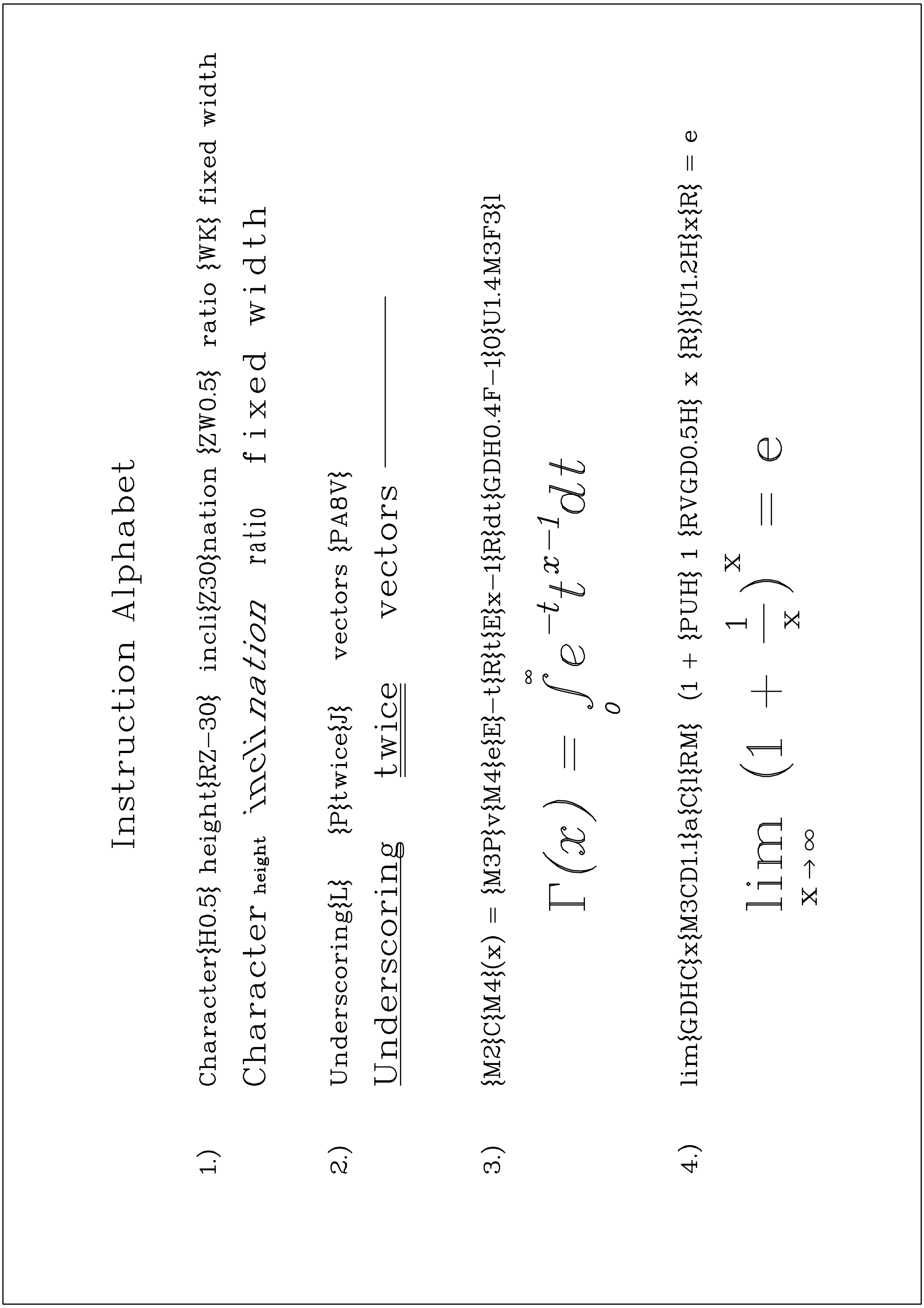


Figure 6.12: Instruction Alphabet

6.8 TeX Instructions for Mathematical Formulas

6.8.1 Introduction

This paragraph presents an alternate method to the DISLIN instruction alphabet for plotting mathematical formulas. The text formatting language TeX has a very easy method for describing mathematical formulas. Since this method is well-known by many scientists, an emulation mode for TeX instructions is added to DISLIN with version 7.4.

TeX instructions can be enabled in DISLIN with the statement CALL TEXMOD ('ON'). If TeX mode is enabled, mixed alphabets defined with SMXALF and the control characters for indices and exponents described in paragraph 6.5 will be ignored.

Mathematical formulas in TeX mode are produced in DISLIN by some special descriptive text. This means that DISLIN must be informed that the following text is to be interpreted as a mathematical formula. The character $ in a text switches from text to math mode, and from math to text mode. Therefore, mathematical formulas must be enclosed in a pair of dollar signs.

Numbers that appear within formulas are called constants, whereas simple variables are represented by single letters. The universal practice in mathematical typesetting is to put constants in Roman typeface and variables in italics. DISLIN uses this rule by default in math mode. The rule can be modified with the routine TEXOPT. Blanks are totally ignored in math mode and spaces are included automatically by DISLIN between constants, variables and operators.

The characters $, {, } and \ have a special meaning in TeX mode and therefore cannot act as printable characters. To include them in normal text, the commands \$, \{, \} and \\ must be used. Additional, the characters _ and ^ have a special meaning in math mode and can be handled in the same way.

Note: Some Fortran compilers treat the character '\' as a special control character, so that an additional flag has to be used for compiling (i.e. -fno-backslash for g77), or the TeX control character '\' can be replaced by another character with the routine SETMIX.

6.8.2 Enabling TeX Mode and TeX Options

T E X M O D

The routine TEXMOD can be used to enable TeX mode in DISLIN. In TeX mode, all character strings passed to DISLIN routines can contain TeX instructions for plotting mathematical formulas.

The call is:	CALL TEXMOD (CMODE)	level 1, 2, 3
or:	void texmod (char *cmode);	

CMODE is a character string that can have the values 'ON' and 'OFF'. CMODE = 'ON' enables TeX mode and CMODE = 'OFF' disables TeX mode.
Default: CMODE = 'OFF'.

T E X O P T

The routine TEXOPT sets some TeX options.

The call is:	CALL TEXOPT (COPT, CTYPE)	level 1, 2, 3
or:	void texopt (char *copt, char *ctype);	

COPT is a character string that can have the values 'ON' and 'OFF'.

CTYPE	is a character string that can contain the keywords 'LIMITS' and 'ITALIC'. 'LIMITS' means that the limits for sums and integrals will be placed above and below the sum and integral signs instead of following them. 'ITALIC' means that for math mode variables will be put in italics.

Default: ('ON', 'LIMITS'),
('ON', 'ITALIC').

T E X V A L

The routine TEXVAL sets a factor for the size of indices and exponents.

The call is:	CALL TEXVAL (X, COPT)	level 1, 2, 3
or:	void texval (float x, char *copt);	

X	is a floatingpoint variable containing the factor.
COPT	is a character string that can contain the keyword 'EXP'.

Default: (1.0, 'EXP').

6.8.3 Exponents and Indices

Exponents and indices are characters that are either raised or lowered relative to the base line of the text. The character ˆ sets the next character as an exponent, while the character _ sets it as an index:

x^2 x^2 a_n a_n x_i^n x_i^n

When exponents and indices occur together, their order is unimportant. If the exponent or index contains more than one character, the group of characters must be inclosed in braces { }:

x^{2n} x^{2n} x_{2y} x_{2y} $A_{i,j,k}^{-n+2}$ A_{i,j,k}^{-n+2}

Multiple raisings and lowerings are generated by applying ˆ and _ to the exponents and indices:

x^{y^2} x^{y^2}

Additional note: The commands ˆ and _ are only allowed in math mode.

6.8.4 Fractions

The instruction \frac{numerator}{denominator} can be used in TeX math mode for plotting fractions. The numerator is plotted on top of the denominator with a horizontal fraction line between them.

$$\frac{1}{x+y}$$ \frac{1}{x+y}

$$\frac{a^2-b^2}{a+b} = a - b$$ \frac{a^2 - b^2}{a+b} = a - b

Fractions may be nested to a depth of 8 within one another:

$$\frac{\frac{a}{x-y} + \frac{b}{x+y}}{1 + \frac{a-b}{a+b}}$$ \frac{\frac{a}{x-y} + \frac{b}{x+y}
{1 + \frac{a-b}{a+b}}

6.8.5 Roots

Roots can be plotted with the syntax \sqrt[n]{arg} where the optional part [n] can be omitted.

Examples:

$\sqrt[3]{8} = 2$	\sqrt[3]{8} = 2
$\sqrt{x^2 + y^2 + 2xy} = x + y$	\sqrt{x^2 + y^2 + 2xy} = x + y

Roots may be nested inside one another to a depth of 8:

$\sqrt{-q + \sqrt{q^2 + p^2}}$	\sqrt{-q +\sqrt{q^2 + p^2}}

6.8.6 Sums and Integrals

Summation and integral signs can be plotted with the two instructions \sum and \int. Sums and integrals can posses upper and lower limits that can be plotted with the exponent and index instructions ^ and _. By default, the limits are placed below and above the summation and integral signs. This can be modfified with the routine TEXMOD or with the instruction \nolimits following the summation and integral signs.

Examples:

$2\sum\limits_{i=0}^{n} a_i$	\sum_{i=1}^n a_i
$\int_a^b f_i(x)g_i(x)dx$	\int\nolimits_a^b f_i(x)g_i(x)dx

6.8.7 Greek Letters

The following Greek letters are available in text and in math mode. If they are used in text mode, the first blank character after the letter will be interpreted as a seperator and will be ignored.

α	\alpha	θ	\theta	o	o	χ	\chi
β	\beta	ι	\iota	π	\pi	ψ	\psi
γ	\gamma	κ	\kappa	ρ	\rho	ω	\omega
δ	\delta	λ	\lambda	σ	\sigma		
ϵ	\epsilon	μ	\mu	τ	\tau		
ζ	\zeta	ν	\nu	υ	\upsilon		
η	\eta	ξ	\xi	φ	\phi		

Γ	\Gamma	Λ	\Lambda	Σ	\Sigma	Ψ	\Psi
Δ	\Delta	Ξ	\Xi	Υ	\Upsilon	Ω	\Omega
Θ	\Theta	Π	\Pi	Φ	\Phi		

6.8.8 Mathematical Symbols

The following mathematical symbols are available in text and in math mode.

$\pm$	\pm	$\cdot$	\cdot	$\cup$	\cup	$\odot$	\odot
$\mp$	\mp	$\ast$	\ast	$\vee$	\vee	$\oplus$	\oplus
$\times$	\times	$\star$	\star	$\wedge$	\wedge	$\ominus$	\ominus
$\div$	\div	$\cap$	\cap	$\setminus$	\setminus		

$\leq$	\le \leq	$\geq$	\ge \geq	$\neq$	\neq	$\sim$	\sim
$\subset$	\subset	$\supset$	\supset	$\cong$	\cong	$\mid$	\mid
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\equiv$	\equiv	$\notin$	\notin
$\in$	\in	$\ni$	\ni	$\parallel$	\parallel	$\neq$	\not=
$\leftarrow$	\leftarrow	$\rightarrow$	\rightarrow	$\Leftrightarrow$	\Leftrightarrow	$\downarrow$	\downarrow
$\Leftarrow$	\Leftarrow	$\Rightarrow$	\Rightarrow	$\uparrow$	\uparrow		
$\emptyset$	\emptyset	$\surd$	\surd	$\forall$	\forall	$\backslash$	\backslash
∇	\nabla	∂	\partial	$\exists$	\exists	∞	\infty
$\perp$	\perp						

6.8.9 Alternate Alphabets

The DISLIN alphabets 'STANDARD', 'ITALIC', 'GREEK', 'SCRIPT' and 'RUSSIAN' can be used in TeX mode with the instructions \rm, \it, \gr, \cal and \ru.

6.8.10 Function Names

The standard for mathematical formulas is to set variable names in italics but the names of functions in Roman. The following function names will be recognized by DISLIN and plotted in Roman.

\arccos	\arcsin	\arctan	\arg	\cos	\cosh	\cot
\coth	\csc	\dec	\dim	\exp	\hom	\ln
\log	\sec	\sin	\sinh	\tan	\tanh	

6.8.11 Accents

Accents are available in TeX mode in the same way as in normal DISLIN mode (see EUSHFT).

6.8.12 Lines above and below Formulas

The commands \overline{arg} and \underline{arg} can be used to draw lines over and under a formula. The command \vec{arg} draws a vector over a formula. All commands can be used in TeX text and math mode.

6.8.13 Horizontal Spacing

Small amounts of horizontal spacing can be added in TeX mode with the following commands:

\,	small space	= 3/18 of the current character size
\:	medium space	= 4/18 of the current character size
\;	large space	= 5/18 of the current character size
\!	negative space	= -3/18 of the current character size

Larger amounts of horizontal spacing can be added with the commands:

\quad	extra space	= 1/1 of the current character size
\qquad	extra space	= 2/1 of the current character size

6.8.14 Selecting Character Size in TeX Mode

The commands \tiny, \scriptsize, \footnotesize, \small, \normalsize, \large, \Large, \LARGE, \huge and \Huge can be used in TeX mode for modifying the character size. The command \normalsize is corresponding to the current character size before the call of the text plotting routine. The character size is decreased or increased by a factor of 1.2 for neighbouring character size commands.

6.8.15 Colours in TeX Mode

The commands \black, \red, \green, \blue, \cyan, \yellow, \orange, \magenta, \white, \fore and \back set the corresponding colours in TeX mode.

6.8.16 Example

```
      PROGRAM EX6_2
      CHARACTER CSTR*80

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX
      CALL HEIGHT(40)

      CSTR='TeX Instructions for Mathematical Formulas'
      NL=NLMESS(CSTR)
      CALL MESSAG(CSTR, (2100 - nl)/2, 100)

      CALL TEXMOD('ON')
      CALL MESSAG('$\frac{1}{x+y}$', 150, 400)
      CALL MESSAG('$\frac{a^2 - b^2}{a+b} = a - b$', 1200, 400)

      CALL MESSAG('$r = \sqrt{x^2 + y^2}', 150, 700)
      CALL MESSAG('$\cos \phi = \frac{x}{\sqrt{x^2 + y^2}}$',
     *            1200, 700)

      CALL MESSAG('$\Gamma(x) = \int_0^\infty e^{-t}t^{x-1}dt$',
     *            150, 1000)
      CALL MESSAG('$\lim_{x \to \infty} (1 + \frac{1}{x})^x = e$',
     *            1200, 1000)

      CALL MESSAG('$\mu = \sum_{i=1}^n x_i p_i$', 150, 1300)
      CALL MESSAG('$\mu = \int_{-\infty}^ \infty x f(x) dx$',
     *            1200, 1300)

      CALL MESSAG('$\overline{x} = \frac{1}{n} \sum_{i=1}^n x_i$',
     *            150, 1600)
      CALL MESSAG('$s^2 = \frac{1}{n-1} \sum_{i=1}^n' //
     *            '(x_i - \overline{x})^2$', 1200, 1600)

      CALL MESSAG('$\sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}}$',
     *            150, 1900)
      CALL MESSAG('$\sqrt[3]{-q + \sqrt{q^2 + p^3}}$', 1200, 1900)
```

```
 CALL MESSAG('$\int \frac{dx}{1+x^2} = \arctan x + C$',
*             150, 2200)
 CALL MESSAG('$\int \frac{dx}{\sqrt{1+x^2}} = ' //
*             ' {\rm arsinh} x + C$', 1200, 2200)

 CALL MESSAG('$\overline{P_1P_2} = \sqrt{(x_2-x_1)^2 + '//
*             '(y_2-y_1)^2}$', 150,2500)
 CALL MESSAG('$x = \frac{x_1 + \lambda x_2}{1 + \lambda}$',
*             1200, 2500)
 CALL DISFIN
 END
```

TeX Instructions for Mathematical Formulas

$$\frac{1}{x+y} \qquad \frac{a^2-b^2}{a+b}=a-b$$

$$r=\sqrt{x^2+y^2} \qquad \cos\varphi=\frac{x}{\sqrt{x^2+y^2}}$$

$$\Gamma(x)=\int_0^\infty e^{-t}t^{x-1}dt \qquad \lim_{x\to\infty}(1+\frac{1}{x})^x=e$$

$$\mu=\sum_{i=1}^n x_i p_i \qquad \mu=\int_{-\infty}^\infty xf(x)dx$$

$$\overline{x}=\frac{1}{n}\sum_{i=1}^n x_i \qquad s^2=\frac{1}{n-1}\sum_{i=1}^n(x_i-\overline{x})^2$$

$$\sqrt[n]{\frac{x^n-y^n}{1+u^{2n}}} \qquad \sqrt[3]{-q+\sqrt{q^2+p^3}}$$

$$\int\frac{dx}{1+x^2}=\arctan x+C \qquad \int\frac{dx}{\sqrt{1+x^2}}=\mathrm{arsinh}x+C$$

$$\overline{P_1P_2}=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \qquad x=\frac{x_1+\lambda x_2}{1+\lambda}$$

Figure 6.13: TeX Instructions for Mathematical Formulas

6.9 Curve Attributes

C H N C R V

CHNCRV defines attributes that will be automatically changed by CURVE after a certain number of calls to the routine CURVE.

The call is:	CALL CHNCRV (CATT)	level 1, 2, 3
or:	void chncrv (char *catt);	

CATT	= 'NONE'	means that CURVE changes no attributes.
	= 'COLOR'	means that colours will be changed.
	= 'LINE'	means that line styles will be changed.
	= 'BOTH'	means that colours and line styles will be changed.

Default: CATT = 'NONE'.

Additional note: The sequence of colours is WHITE/BLACK, RED, GREEN, YELLOW, BLUE, ORANGE, CYAN and MAGENTA.
The sequence of line styles is SOLID, DOT, DASH, CHNDSH, CHNDOT, DASHM, DOTL and DASHL.
The symbol number is always changed. It will be incremented by 1 starting with the current symbol defined by MARKER.

The following three routines are useful when automatic attribute setting is selected and the routine CURVE is called several times to plot a single curve.

I N C C R V

INCCRV defines the number of calls after which CURVE will automatically change attributes.

The call is:	CALL INCCRV (NCRV)	level 1, 2, 3
or:	void inccrv (int ncrv);	

NCRV is the number of curves that will be plotted with identical attributes.

Default: NCRV = 1

C H N A T T

CHNATT is an alternative routine to INCCRV. It is useful when the number of curves plotted with identical attributes varies. CHNATT defines new attributes that will be used by CURVE during the next call.

The call is:	CALL CHNATT	level 1, 2, 3
or:	void chnatt ();	

Additional notes:

- CHNATT changes only attributes specified with CHNCRV.
- Attributes cannot be skipped by calling CHNATT several times; the order of the attribute cycles must be changed.

R E S A T T

In general, curve attributes will be repeated after 8 changes. With the routine RESATT, the attributes can be reset earlier.

The call is:	CALL RESATT	level 1, 2, 3
or:	void resatt ();	

I N C M R K

INCMRK selects line or symbol mode for CURVE.

The call is:	CALL INCMRK (NMRK)	level 1, 2, 3
or:	void incmrk (int nmrk);	

NMRK	= - n	means that CURVE plots only symbols. Every n-th point will be marked by a symbol.
	= 0	means that CURVE connects points with lines.
	= n	means that CURVE plots lines and marks every n-th point with a symbol.

Default: NMRK = 0

M A R K E R

The symbols used to plot points can be selected with the routine MARKER. The symbol number will be incremented by 1 after a certain number of calls to CURVE defined by INCCRV.

The call is:	CALL MARKER (NSYM)	level 1, 2, 3
or:	void marker (int nsym);	

NSYM — is the symbol number between 0 and 21. The symbols are shown in appendix B.

Default: NSYM = 0

H S Y M B L

HSYMBL defines the size of symbols.

The call is:	CALL HSYMBL (NHSYM)	level 1, 2, 3
or:	void hsymbl (int nhsym);	

NHSYM — is the size of symbols in plot coordinates.

Default: NHSYM = 35

M Y S Y M B

MYSYMB sets an user-defined symbol.

The call is:	CALL MYSYMB (XRAY, YRAY, N, ISYM, IFLAG)	level 1, 2, 3
or:	void mysymb (float *xray, float *yray, int n, int isym, int iflag);	

XRAY, YRAY	are the X- and Y-coordinates of the symbol in the range -1 and 1.
N	is the number of coordinates in XRAY and YRAY.
ISYM	is a non negative number that will be used as symbol number.
IFLAG	is an Integer that can have the values 0 and 1. If IFLAG = 1, the symbol will be filled.
Additional note:	The number of points in MYSYMB is limited to 100 for Fortran 77. There is no limitation for the C and Fortran 90 versions of DISLIN.

T H K C R V

THKCRV defines the thickness of curves.

The call is:	CALL THKCRV (NTHK)	level 1, 2, 3
or:	void thkcrv (int nthk);	

NTHK — is the thickness of curves in plot coordinates.

Default: NTHK = 1

G A P C R V

GAPCRV defines a data gap used in the routine CURVE. If the distance between two neightbouring X coordinates is greater than the gap value, CURVE will not connect these data points.

The call is:	CALL GAPCRV (XGAP)	level 1, 2, 3
or:	void gapcrv (float xgap);	

XGAP — is the gap value.

P O L C R V

POLCRV defines an interpolation method used by CURVE to connect points.

The call is:	CALL POLCRV (CPOL)	level 1, 2, 3
or:	void polcrv (char *cpol);	

CPOL	is a character string containing the interpolation method.
= ’LINEAR’	defines linear interpolation.
= ’STEP’	defines step interpolation.
= ’STAIRS’	defines step interpolation.
= ’BARS’	defines bar interpolation.
= ’FBARS’	defines filled bar interpolation.
= ’STEM’	defines stem interpolation.
= ’SPLINE’	defines spline interpolation.
= ’PSPLINE’	defines parametric spline interpolation.

Default: CPOL = ’LINEAR’.

Additional notes:

- The width of bars can be set with BARWTH.
- For spline interpolation, the X-coordinates must have different values and be in ascending order. There is no restriction for a parametric spline. The order of spline polynomials and the number of interpolated points can be modified with SPLMOD.
- The interpolation medthods ’LINEAR’, ’BARS’, ’FBARS’ and ’STEM’ can also be used for polar scaling.

S P L M O D

SPLMOD defines the order of polynomials and the number of interpolated points used for the interpolation methods ’SPLINE’ and ’PSPLINE’.

The call is:	CALL SPLMOD (NGRAD, NPTS)	level 1, 2, 3

or:	void splmod (int ngrad, int npts);
NGRAD	is the order of the spline polynomials (2 - 10). It affects the number of points accepted by CURVE which is determined by the formula (2 * NGRAD + 1) * N $\leq$ 1000. For example, with a cubic spline, up to 142 points can be passed to CURVE.
NPTS	is the number of points that will be interpolated in the range XRAY(1) to XRAY(N).

Default: (3, 200).

B A R W T H

BARWTH sets the width of bars plotted by CURVE.

The call is:	CALL BARWTH (XWTH)	level 1, 2, 3
or:	void barwth (float xwth);	

XWTH	defines the bar width. If positive, the absolute value of XWTH * (XRAY(2)-XRAY(1)) is used. If negative, the absolute value of XWTH is used where XWTH is specified in plot coordinates.

Default: XWTH = 0.75

Additional note:	If XWTH is positive and polar scaling is enabled, the absolute value of XWTH * (YRAY(2) - YRAY(1)) defines the width of bars. If XWTH is negative for polar scaling, the absolute value of XWTH is used where XWTH must be specified in degrees.

N O C H E K

The routine NOCHEK can be used to suppress the listing of points that lie outside of the axis scaling.

The call is:	CALL NOCHEK	level 1, 2, 3
or:	void nochek ();	

6.10 Line Attributes

L I N E   S T Y L E S

The routines SOLID, DOT, DASH, CHNDSH, CHNDOT, DASHM, DOTL and DASHL define different line styles. They are called without parameters. The routine LINTYP (NTYP) can also be used to set line styles where NTYP is an integer between 0 and 7 and corresponds to the line styles above. The routine MYLINE sets user-defined line styles.

M Y L I N E

MYLINE defines a global line style.

The call is:	CALL MYLINE (NRAY, N)	level 1, 2, 3
or:	void myline (int *nray, int n);	

NRAY	is an array of positive integers characterizing the line style. Beginning with pen-down, a pen-down and pen-up will be done alternately according to the specified lengths in NRAY. The lengths must be given in plot coordinates.

N	is the number of elements in NRAY.
Examples:	The values of NRAY for the predefined line styles are given below:

SOLID :	NRAY = {1}
DOT :	NRAY = {1, 10}
DASH :	NRAY = {10, 10}
CHNDSH:	NRAY = {30, 15, 10, 15}
CHNDOT:	NRAY = {1, 15, 15, 15}
DASHM :	NRAY = {20, 15}
DOTL :	NRAY = {1, 20}
DASHL :	NRAY = {30, 20}

L I N W I D

The routine LINWID sets the line width.

The call is: CALL LINWID (NWIDTH) level 1, 2, 3

or: void linwid (int nwidth);

NWIDTH is the line width in plot coordinates. Default: NWIDTH = 1

Additional note: To define smaller line widhts than 1 (i.e. for PostScript files), the routine PENWID (XWIDTH) can be used where XWIDTH has the same meaning as NWIDTH.

L N C A P

The routine LNCAP sets the current line cap parameter.

The call is: CALL LNCAP (CAP) level 1, 2, 3

or: void lncap (char *cap);

CAP is a character string defining the line cap.

= ’ROUND’ defines rounded caps.

= ’CUT’ defines square caps.

= ’LONG’ defines square caps where stroke ends will be continued equal to half the line width.

Default: CAP = ’LONG’.

L N J O I N

The routine LNJOIN sets the current line join parameter.

The call is: CALL LNJOIN (CJOIN) level 1, 2, 3

or: void lnjoin (char *cjoin);

CJOIN is a character string containing the the line join.

= ’SHARP’ defines sharp corners between path segments.

= ’TRUNC’ defines truncated corners between path segments.

Default: CJOIN = ’TRUNC’.

L N M L T

The routine LNMLT sets the current miter limit parameter. This routine can be useful if the line join is set to 'SHARP'.

The call is:	CALL LNMLT (XFC)	level 1, 2, 3
or:	void lnmlt (float xfc);	

XFC	is a floatingpoint number where XFC * line width will be used as the miter limit. The miter length is the distance between the inner and outside edge of a path corner. Default: XFC = 2.

6.11 Shading

S H D P A T

SHDPAT selects shading patterns used by routines such as SHDCRV and AREAF.

The call is:	CALL SHDPAT (IPAT)	level 1, 2, 3
or:	void shdpat (long ipat);	

IPAT	is an integer between 0 and 17. The predefined patterns are shown in appendix B.

M Y P A T

MYPAT defines a global shading pattern.

The call is:	CALL MYPAT (IANGLE, ITYPE, IDENS, ICROSS)	level 1, 2, 3
or:	void mypat (int iangle, int itype, int idens, int icross);	

IANGLE		is the angle of shading lines (0 - 179).
ITYPE		defines the type of shading lines:
	= 0	no shading lines.
	= 1	equidistant lines.
	= 2	double shading lines.
	= 3	triple shading lines.
	= 4	thick shading lines.
	= 5	dotted lines.
	= 6	dashed lines.
	= 7	dashed-dotted lines.
IDENS		defines the distance between shading lines (0: small distance, 9: big distance).
ICROSS		indicates whether shading lines are hatched (0: not hatched, 1: hatched).

Examples:	The following calls to MYPAT show the predefined shading patterns used by SHDPAT:
IPAT = 0:	CALL MYPAT (0, 0, 0, 0)
IPAT = 1:	CALL MYPAT (45, 1, 5, 0)
IPAT = 2:	CALL MYPAT (150, 4, 5, 0)
IPAT = 3:	CALL MYPAT (135, 1, 5, 0)

IPAT = 4:	CALL MYPAT (45, 4, 5, 0)
IPAT = 5:	CALL MYPAT (45, 1, 5, 1)
IPAT = 6:	CALL MYPAT (135, 2, 1, 0)
IPAT = 7:	CALL MYPAT (45, 4, 5, 1)
IPAT = 8:	CALL MYPAT (30, 1, 4, 0)
IPAT = 9:	CALL MYPAT (45, 2, 1, 1)
IPAT = 10:	CALL MYPAT (0, 1, 5, 1)
IPAT = 11:	CALL MYPAT (45, 3, 1, 0)
IPAT = 12:	CALL MYPAT (70, 4, 7, 0)
IPAT = 13:	CALL MYPAT (45, 3, 1, 1)
IPAT = 14:	CALL MYPAT (0, 4, 5, 1)
IPAT = 15:	CALL MYPAT (45, 2, 1, 0)
IPAT = 16:	CALL MYPAT (0, 1, 0, 0)
IPAT = 17:	CALL MYPAT (0, 5, 5, 0)

N O A R L N

With the routine NOARLN the outlines of shaded regions can be suppressed.

The call is:	CALL NOARLN	level 1, 2, 3
or:	void noarln ();	

6.12 Attribute Cycles

The attributes line style, colour and shading pattern can be changed automatically by routines such as CURVE, SHDCRV, BARS and PIEGRF according to a predefined cycle.
The cycles are:

Line styles:	SOLID, DOT, DASH, CHNDSH, CHNDOT, DASHM, DOTL and DASHL.
Colours:	WHITE/BLACK, RED, GREEN, YELLOW, BLUE, ORANGE, CYAN and MAGENTA.
Shading:	Pattern numbers from 0 to 17.

The following subroutines allow the redefining of cycles.

L I N C Y C

LINCYC changes the line style cycle.

The call is:	CALL LINCYC (INDEX, ITYP)	level 1, 2, 3
or:	void lincyc (int index, int ityp);	

INDEX	is an index between 1 and 30.
ITYP	is an integer between 0 and 7 containing the line style (0 = SOLID, 1 = DOT, 2 = DASH, 3 = CHNDSH, 4 = CHNDOT, 5 = DASHM, 6 = DOTL, 7 = DASHL).

C L R C Y C

CLRCYC changes the colour cycle.

The call is:	CALL CLRCYC (INDEX, ICLR)	level 1, 2, 3
or:	void clrcyc (int index, int iclr);	

INDEX	is an index between 1 and 30.
ICLR	is a colour value (see SETCLR).

P A T C Y C

PATCYC changes the shading pattern cycle.

The call is:	CALL PATCYC (INDEX, IPAT)	level 1, 2, 3
or:	void patcyc (int index, long ipat);	

INDEX	is an index between 1 and 30.
IPAT	is a pattern number between 0 and 17 or is determined by the formula IANGLE * 1000 + ITYPE * 100 + IDENS * 10 + ICROSS with the parameters described in MYPAT.

6.13 Base Transformations

The following subroutines create a transformation matrix that affects plot vectors contained within page borders. Vectors may be scaled, shifted and rotated and the transformations can be combined in any order.

T R F S H F

TRFSHF affects the shifting of plot vectors.

The call is:	CALL TRFSHF (NXSHFT, NYSHFT)	level 1, 2, 3
or:	void trfshf (int nxshft, int nyshft);	

NXSHFT, NYSHFT	are plot coordinates that define the magnitude of shifting in the X- and Y-direction.

T R F S C L

TRFSCL affects the scaling of plot vectors.

The call is:	CALL TRFSCL (XSCL, YSCL)	level 1, 2, 3
or:	void trfscl (float xscl, float yscl);	

XSCL, YSCL	are scaling factors for the X- and Y-direction.

T R F R O T

TRFROT affects the rotation of plot vectors around a point.

The call is:	CALL TRFROT (XANG, NX, NY)	level 1, 2, 3
or:	void trfrot (float xang, int nx, int ny);	

XANG	is the rotation angle measured in degrees in a counter-clockwise direction.
NX, NY	are the plot coordinates of the rotation point.

T R F R E S

TRFRES resets base transformations.

The call is:	CALL TRFRES	level 1, 2, 3
or:	void trfres ();	

6.14 Shielded Regions

This section describes how to protect regions from being overwritten. Shielded regions can be defined automatically by DISLIN or explicitly by the user. Shielded regions are stored in a buffer which can then be manipulated by the user.

S H I E L D

SHIELD selects shielded regions which are set automatically by DISLIN.

The call is:	CALL SHIELD (CAREA, CMODE)	level 1, 2, 3
or:	void shield (char *carea, char *cmode);	

CAREA	is a character string defining the regions:
= ’MESSAG’	is used for text and numbers plotted by MESSAG and NUMBER.
= ’SYMBOL’	will shield symbols.
= ’BARS’	will shield bars plotted by BARS.
= ’PIE’	will shield pie segments plotted by PIEGRF.
= ’LEGEND’	will protect legends. All legend attributes should be set before calling CURVE because the shielded region of a legend is defined by CURVE. If there is no legend position defined with LEGPOS, CURVE assumes that the legend lies in the upper right corner of the axis system.
CMODE	is a character string defining a status:
= ’ON’	means that the regions defined above will be written to the shielding buffer and are protected.
= ’OFF’	means that regions will not be written to the shielding buffer. Regions that are still stored in the buffer will be shielded.
= ’DELETE’	removes regions from the shielding buffer.
= ’RESET’	is a combination of ’OFF’ and ’DELETE’. Regions are removed from and will not be written to the shielding buffer. To save computing time, this command should always be used when shielding is no longer needed.
= ’NOVIS’	The shielding of regions held in the shielding buffer is disabled. This is not valid for regions newly written to the buffer.
= ’VIS’	Disabled regions will be protected. This is the default value for regions newly written to the buffer.

The following routines set user-defined regions:

The calls are:	CALL SHLREC	(NX, NY, NW, NH)	for rectangles
	CALL SHLRCT	(NX, NY, NW, NH, THETA)	for rotated rectangles
	CALL SHLCIR	(NX, NY, NR)	for circles
	CALL SHLELL	(NX, NY, NA, NB, THETA)	for rotated ellipses
	CALL SHLPIE	(NX, NY, NR, ALPHA, BETA)	for pie segments
	CALL SHLPOL	(NXRAY, NYRAY, N)	for polygons.

NX, NY	are plot coordinates of the upper left corner or the centre point.
NW, NH	are the width and height of rectangles.
NR, NA, NB	are radii in plot coordinates.

THETA	is a rotation angle measured in degrees in a counter-clockwise direction.
ALPHA, BETA	are starting and ending angles for pie segments measured in degrees in a counter-clockwise direction.
NXRAY, NYRAY	are arrays of the dimension N containing the corner points of a polygon.

S H L I N D

The index of shielded regions in the buffer can be requested with SHLIND. It returns the index of the region last written to the buffer.

The call is: CALL SHLIND (ID) level 1, 2, 3

or: int shlind ();

ID is the returned index.

S H L D E L

SHLDEL removes entries from the shielding buffer.

The call is: CALL SHLDEL (ID) level 1, 2, 3

or: void shldel (int id);

ID is the index of a shielded region. If ID is 0, all regions defined by the user will be deleted.

S H L R E S

SHLRES deletes regions last written to the shielding buffer.

The call is: CALL SHLRES (N) level 1, 2, 3

or: void shlres (int n);

N is the number of regions to delete.

S H L V I S

SHLVIS disables or enables shielded regions. Disabled regions are no longer protected but are still held in the shielding buffer.

The call is: CALL SHLVIS (ID, CMODE) level 1, 2, 3

or: void shlvis (int id, char *cmode);

ID	is the index of a shielded region. If ID is 0, all entries are disabled or enabled.
CMODE = ’ON’	enables shielded regions. This is the default value for regions newly written to the buffer.
= ’OFF’	disables shielded regions.

Additional notes:

- A frame is plotted around regions defined by the user. The thickness of frames can be set with FRAME. Regions defined automatically by DISLIN are not enclosed by a frame but frames plotted by MESSAG after using FRMESS and shielded regions defined by MESSAG are identical.
- Shielded regions can overlap each other.

- The statement CALL RESET ('SHIELD') resets shielding. All regions defined by DISLIN and the user are removed from the shielding buffer and no new regions will be written to the buffer.
- The number of shielded regions is limited to the size of the shielding buffer which is set to 1000 words. The number of words used by regions are: SHLREC = 6, SHLRCT = 7, SHLCIR = 5, SHLELL = 7, SHLPIE = 7 and SHLPOL = 2*N+3.
- Shielding of regions is computer intensive. Therefore, shielding should be used very carefully and shielded regions should be deleted from the buffer when no longer needed.
- Base transformations do not affect the position of shielded regions.
- SHLPOL can be used between the routines GRFINI and GRFFIN. The shielded region will be projected into 3-D space. This is not valid for other shielded regions.

Chapter 7

Parameter Requesting Routines

This chapter describes subroutines that return the current values of plot parameters. All routines correspond to parameter setting routines described in the last chapter or handled in chapter 11, ”3-D Colour Graphics”. For a complete description of parameters, the user is referred to these chapters. If a character string is returned, it will appear in uppercase letters and be shortened to four characters.

G E T P A G

This routine returns the page size (see SETPAG, PAGE).

The call is:	CALL GETPAG (NXPAG, NYPAG)	level 1, 2, 3
or:	void getpag (int *nxpag, int *nypag);	

G E T F I L

The routine GETFIL returns the current plotfile name (see SETFIL).

The call is:	CALL GETFIL (CFIL)	level 1, 2, 3
or:	char *getfil ();	
CFIL	is a character variable containing the filename.	

G E T M F L

GETMFL returns the file format (see METAFL).

The call is:	CALL GETMFL (CDEV)	level 1, 2, 3
or:	char *getmfl ();	
CDEV	is a character variable containing the file format.	

G E T O R

GETOR returns the coordinates of the origin (see ORIGIN).

The call is:	CALL GETOR (NX0, NY0)	level 1, 2, 3
or:	void getor (int *nx0, int *ny0);	

G E T P O S

This routine returns the position of the lower left corner of an axis system in plot coordinates (see AXSPOS).

The call is:	CALL GETPOS (NXA, NYA)	level 1, 2, 3

or: void getpos (int *nxa, int *nya);

G E T L E N

GETLEN returns the length of the X-, Y- and Z-axes (see AXSLEN, AX3LEN).

The call is: CALL GETLEN (NXL, NYL, NZL) level 1, 2, 3

or: void getlen (int *nxl, int *nyl, int *nzl);

G E T H G T

GETHGT returns the character height (see HEIGHT).

The call is: CALL GETHGT (NHCHAR) level 1, 2, 3

or: int gethgt ();

G E T H N M

GETHNM returns the character height of axis titles (see HNAME).

The call is: CALL GETHNM (NHNAME) level 1, 2, 3

or: int gethnm ();

G E T A N G

GETANG returns the current character angle used for text and numbers (see ANGLE).

The call is: CALL GETANG (NANG) level 1, 2, 3

or: int getang ();

G E T A L F

GETALF returns the base alphabet (see BASALF).

The call is: CALL GETALF (CALF) level 1, 2, 3

or: char *getalf ();

CALF is a character variable containing the returned base alphabet.

G E T M I X

GETMIX returns control characters used for plotting indices and exponents (see SETMIX, NEWMIX).

The call is: CALL GETMIX (CHAR, CMIX) level 1, 2, 3

or: char *getmix (char *cmix);

CHAR is a character string containing the control character.

CMIX is a character string that defines the function of the control character. CMIX can have the values 'EXP', 'IND', 'RES' and 'LEG' for exponents, indices, resetting the base-line, and for multiple text lines in legends.

G E T S H F

GETSHF returns shift characters used for plotting special European characters (see EUSHFT).

The call is: CALL GETSHF (CNAT, CHAR) level 1, 2, 3

or: char *getshf (char *cnat);

CNAT	is a character string that can have the values 'GERMAN', 'FRENCH', 'SPANISH', 'DANISH', 'ACUTE' , 'GRAVE' and 'CIRCUM'.
CHAR	is a character string containing the returned shift character.

GMXALF

GMXALF returns shift characters used for shifting between the base and an alternate alphabet (see SMXALF).

The call is:	CALL GMXALF (CALPH, C1, C2, N)	level 1, 2, 3
or:	int gmxalf (char *calph, char *c1, char *c2);	

CALPH	is a character string containing an alphabet. In addition to the names in BASALF, CALPH can have the value 'INSTRUCTION'.
C1, C2	are characters strings that contain the returned shift characters.
N	is the returned index of the alphabet between 0 and 6. If N = 0, no shift characters are defined for the alphabet CALPH.

GETDIG

This routine returns the number of decimal places that are displayed in axis labels (see LABDIG).

The call is:	CALL GETDIG (NXDIG, NYDIG, NZDIG)	level 1, 2, 3
or:	void getdig (int *nxdig, int *nydig, int *nzdig);	

GETGRF

The routine GETGRF returns the current scaling of an axis system.

The call is:	CALL GETGRF (XA, XE, XOR, XSTP, CAX)	level 2, 3
or:	void getgrf (float *xa, float *xe, float *xor, float *xstp, char *cax);	

XA, XE	are the lower and upper limits of the axis.
XOR, XSTP	are the first axis label and the step between labels.
CAX	select the axis and can have the values 'X', 'Y' and 'Z'.

GETTIC

GETTIC returns the number of ticks that are plotted between axis labels (see TICKS).

The call is:	CALL GETTIC (NXTIC, NYTIC, NZTIC)	level 1, 2, 3
or:	void gettic (int *nxtic, int *nytic, int *nztic);	

GETTCL

GETTCL returns tick lengths (see TICLEN).

The call is:	CALL GETTCL (NMAJ, NMIN)	level 1, 2, 3
or:	void gettcl (int *nmaj, int *nmin);	

GETSP1

GETSP1 returns the distance between axis ticks and labels (see LABDIS).

The call is:	CALL GETSP1 (NXDIS, NYDIS, NZDIS)	level 1, 2, 3

or: void getsp1 (int *nxdis, int *nydis, int *nzdis);

G E T S P 2

GETSP2 returns the distance between axis labels and names (see NAMDIS).

The call is: CALL GETSP2 (NXDIS, NYDIS, NZDIS) level 1, 2, 3

or: void getsp2 (int *nxdis, int *nydis, int *nzdis);

G E T S C L

This routine returns the type of axis scaling used. For linear scaling, the value 0 is returned and for logarithmic scaling, the value 1 is returned (see AXSSCL).

The call is: CALL GETSCL (NXLOG, NYLOG, NZLOG) level 1, 2, 3

or: void getscl (int *nxlog, int *nylog, int *nzlog);

G E T L A B

GETLAB returns the label types used for axis numbering (see LABELS).

The call is: CALL GETLAB (CXLAB, CYLAB, CZLAB) level 1, 2, 3

or: void getlab (char *cxlab, char *cylab, char *czlab);

G E T C L R

GETCLR returns the current colour as an index from the colour table (see SETCLR).

The call is: CALL GETCLR (NCOL) level 1, 2, 3

or: int getclr ();

G E T U N I

GETUNI returns the logical unit used for error messages.

The call is: CALL GETUNI (NU) level 1, 2, 3

or: FILE *getuni ();

G E T V E R

GETVER returns the version number of the currently used DISLIN library.

The call is: CALL GETVER (XVER) level 1, 2, 3

or: float getver ();

G E T P L V

GETPLV returns the patch level of the currently used DISLIN library.

The call is: CALL GETPLV (IPLV) level 1, 2, 3

or: int getplv ();

G E T L E V

GETLEV returns the level.

The call is: CALL GETLEV (NLEV) level 1, 2, 3

or:	int getlev ();	

G E T S Y M

GETSYM returns the current symbol number and height of symbols.

The call is:	CALL GETSYM (NSYM, NHSYMB)	level 1, 2, 3
or:	void getsym (int *nsym, int *nhsymb);	

G E T T Y P

GETTYP returns the current line style (see LINTYP).

The call is:	CALL GETTYP (NTYP)	level 1, 2, 3
or:	int gettyp ();	

G E T L I N

The routine GETLIN returns the current line width (see LINWID).

The call is:	CALL GETLIN (NWIDTH)	level 1, 2, 3
or:	int getlin ();	

G E T P A T

The routine GETPAT returns the current shading pattern (see SHDPAT).

The call is:	CALL GETPAT (NPAT)	level 1, 2, 3
or:	long getpat ();	

G E T R E S

GETRES returns the width and height of rectangles plotted in 3-D colour graphics (see SETRES, AUTRES).

The call is:	CALL GETRES (NPB, NPH)	level 1, 2, 3
or:	void getres (int *npb, int *nph);	

G E T V L T

GETVLT returns the current colour table (see SETVLT).

The call is:	CALL GETVLT (CVLT)	level 1, 2, 3
or:	char *getvlt ();	

G E T I N D

For a colour index, the routine GETIND returns the corresponding RGB coordinates stored in the current colour table (see SETIND). If an explicit RGB value is specified, GETIND returns the RGB coordinates of the RGB value.

The call is:	CALL GETIND (I, XR, XG, XB)	level 1, 2, 3
or:	void getind (int i, float *xr, float *xg, float *xb);	

G E T R G B

GETRGB returns the RGB coordinates of the current colour.

The call is:	CALL GETRGB (XR, XG, XB)	level 1, 2, 3
or:	void getrgb (float *xr, float *xg, float *xb);	

G E T S C R

GETSCR returns the width and height of the screen in pixels.

The call is:	CALL GETSCR (NWPIX, NHPIX)	level 0, 1, 2, 3
or:	void getscr (int *nwpix, int *nhpix);	

G E T B P P

GETBPP returns the number of bits per pixel used by graphics card.

The call is:	CALL GETBPP (NBPP)	level 0, 1, 2, 3
or:	int getbpp ();	

G E T D S P

The routine GETDSP returns the terminal type.

The call is:	CALL GETDSP (CDSP)	level 0, 1, 2, 3
or:	char *getdsp ();	

CDSP	is a returned character string that can have the values ’XWIN’ for X Window terminals, ’WIND’ for Windows terminals and ’NONE’ for none of them.

G E T R A N

GETRAN returns the colour range of colour bars (see COLRAN).

The call is:	CALL GETRAN (NCA, NCE)	level 1, 2, 3
or:	void getran (int *nca, int *nce);	

G E T W I D

GETWID returns the width of the colour bar plotted in 3-D colour graphics (see BARWTH).

The call is:	CALL GETWID (NZB)	level 1, 2, 3
or:	int getwid ();	

G E T V K

This routine returns the lengths used for shifting titles and colour bars (see VKYTIT, VKXBAR, VKYBAR).

The call is:	CALL GETVK (NYTIT, NXBAR, NYBAR)	level 1, 2, 3
or:	void getvk (int *nytit, int *nxbar, int *nybar);	

G E T W I N

This routine returns the upper left corner and the size of the graphics window (see WINDOW, WINSIZ).

The call is:	CALL GETWIN (NX, NY, NW, NH)	level 1, 2, 3
or:	void getwin (int *nx, int *ny, int *nw, int *nh);	

G E T C L P

The routine GETCLP returns the upper left corner and the size of the current clipping window (see CLPWIN).

The call is:	CALL GETCLP (NX, NY, NW, NH)	level 1, 2, 3
or:	void getclp (int *nx, int *ny, int *nw, int *nh);	

G E T X I D

The routine GETXID returns the ID of the current X graphics window or pixmap.

The call is:	CALL GETXID (ID, CTYPE)	level 1, 2, 3
or:	int getxid (char *ctype);	

ID	is the returned window ID.
CTYPE	is a character string that can have the values ’WINDOW’ and ’PIXMAP’.

Chapter 8

Elementary Plot Routines

This chapter describes elementary subroutines that plot lines, vectors, circles, ellipses, pie segments and polygons. There are versions for plot and user coordinates; the routines for user coordinates begin with the keyword ’RL’. These routines can only be called from level 2 or 3 after an axis system has been defined.

8.1 Lines

XMOVE and XDRAW are simple subroutines for plotting lines. They require absolute page coordinates and are, therefore, not affected by a call to ORIGIN. Different line styles cannot be used. The routine XMOVE moves the pen to a point while XDRAW draws a line to a point.

The calls are:	CALL XMOVE (X, Y)	level 1, 2, 3
	CALL XDRAW (X, Y)	level 1, 2, 3
or:	void xmove(float x, float y);	
	void xdraw (float x, float y);	

X, Y	are absolute page coordinates.

The subroutines STRTPT and CONNPT require plot coordinates as real numbers and allow different line styles to be used.

The calls are:	CALL STRTPT (X, Y)	level 1, 2, 3
	CALL CONNPT (X, Y)	level 1, 2, 3
or:	void strtpt (float x, float y);	
	void connpt (float x, float y);	

X, Y	are real numbers containing the plot coordinates.

The corresponding routines for user coordinates are:

The calls are:	CALL RLSTRT (X, Y)	level 2, 3
	CALL RLCONN (X, Y)	level 2, 3
or:	void rlstrt (float x, float y);	
	void rlconn (float x, float y);	

Additional note:	Lines plotted with RLSTRT and RLCONN will not be cut off at the borders of an axis system. This can be enabled with the routine CLPBOR. Points lying outside of the axis scaling will not be listed by RLSTRT and RLCONN.

L I N E

LINE joins two points with a line. Different line styles can be used.

The call is:	CALL LINE (NX1, NY1, NX2, NY2)	level 1, 2, 3
or:	void line (int nx1, int ny1, int nx2, int ny2);	

NX1, NY1	are the plot coordinates of the first point.
NX2, NY2	are the plot coordinates of the second point.

R L I N E

RLINE is the corresponding routine for user coordinates.

The call is:	CALL RLINE (X1, Y1, X2, Y2)	level 2, 3
or:	void rline (float x1, float y1, float x2, float y2);	

X1, Y1	are the user coordinates of the first point.
X2, Y2	are the user coordinates of the second point.
Additional note:	RLINE draws only that part of the line lying inside the axis system. If NOCHEK is not used, points lying outside the axis scaling will be listed.

8.2 Vectors

V E C T O R

VECTOR plots vectors with none, one or two arrow heads.

The call is:	CALL VECTOR (IX1, IY1, IX2, IY2, IVEC)	level 1, 2, 3
or:	void vector (int ix1, int iy1, int ix2, int iy2, int ivec);	

IX1, IY1	are the plot coordinates of the start point.
IX2, IY2	are the plot coordinates of the end point.
IVEC	is a four digit number ’wxyz’ specifying the arrow heads where the digits have the following meaning: (see appendix B for examples)

w:	determines the ratio of width and length (0 - 9).
x:	determines the size (0 - 9).
y:	determines the form:
= 0	filled
= 1	not filled
= 2	opened
= 3	closed.
z:	determines the position:
= 0	no arrow heads are plotted
= 1	at end points
= 2	at start and end points
= 3	at start and end points and in the same direction.

R L V E C

RLVEC is the corresponding routine for user coordinates.

The call is:	CALL RLVEC (X1, Y1, X2, Y2, IVEC)	level 2, 3
or:	void rlvec (float x1, float y1, float x2, float y2, int ivec);	

8.3 Filled Triangles

T R I F L L

The routine TRIFLL plots solid filled triangles.

The call is:	CALL TRIFLL (XRAY, YRAY)	level 1, 2, 3
or:	void trifll (float *xray, float *yray;	

XRAY, YRAY	are floatingpoint arrays containing the three corners of a triangle.

8.4 Wind Speed Symbols

W I N D B R

The routine WINDBR plots wind speed symbols.

The call is:	CALL WINDBR (X, NXP, NYP, NW, A)	level 1, 2, 3
or:	void windbr (float x, int nxp, int nyp, int nw, float a);	

X	is the wind speed in knots.
NXP, NYP	are the plot coordinates of the lower left corner of the wind speed symbol.
NW	is the length of the symbol in plot coordinates.
A	is the wind direction in degrees.

R L W I N D

RLWIND is the corresponding routine to WINDBR for user coordinates.

The call is:	CALL RLWIND (X, XP, YP, NW, A)	level 2, 3
or:	void rlwind (float x, float yp, float xp, int nw, float a);	

8.5 Geometric Figures

The following subroutines plot geometric figures such as rectangles, circles, ellipses, pie segments and polygons. These routines can be used to plot only the outlines of figures or the figures can be filled in with shaded patterns.

R E C T A N

RECTAN plots rectangles.

The call is:	CALL RECTAN (NX, NY, NW, NH)	level 1, 2, 3
or:	void rectan (int nx, int ny, int nw, int nh);	

NX, NY	are the plot coordinates of the upper left corner.
NW, NH	are the width and height in plot coordinates.

R N D R E C

RECTAN plots an rectangle where the corners will be rounded.

The call is:	CALL RNDREC (NX, NY, NW, NH, IOPT)	level 1, 2, 3
or:	void rndrec (int nx, int ny, int nw, int nh, int iopt);	

NX, NY	are the plot coordinates of the upper left corner.
NW, NH	are the width and height in plot coordinates.
IOPT	defines the rounding of corners ($0 \leq$ IOPT ≤ 9). For IOPT = 0, rounding is disabled.

C I R C L E

CIRCLE plots circles.

The call is:	CALL CIRCLE (NX, NY, NR)	level 1, 2, 3
or:	void circle (int nx, int ny, int nr);	

NX, NY	are the plot coordinates of the centre point.
NR	is the radius in plot coordinates.

E L L I P S

ELLIPS plots ellipses.

The call is:	CALL ELLIPS (NX, NY, NA, NB)	level 1, 2, 3
or:	void ellips (int nx, int ny, int na, int nb);	

NX, NY	are the plot coordinates of the centre point.
NA, NB	are the radii in plot coordinates.

P I E

PIE plots pie segments.

The call is:	CALL PIE (NX, NY, NR, ALPHA, BETA)	level 1, 2, 3
or:	void pie (int nx, int ny, int nr, float alpha, float beta);	

NX, NY	are the plot coordinates of the centre point.
NR	is the radius in plot coordinates.
ALPHA, BETA	are the start and end angles measured in degrees in a counter-clockwise direction.

A R C E L L

ARCELL plots elliptical arcs where the arcs can be rotated.

The call is:	CALL ARCELL (NX, NY, NA, NB, ALPHA, BETA, THETA)	level 1, 2, 3

or: void arcell (int nx, int ny, int na, int nb, float alpha, float beta, float theta);

NX, NY	are the plot coordinates of the centre point.
NA, NB	are the radii in plot coordinates.
ALPHA, BETA	are the start and end angles measured in degrees in a counter-clockwise direction.
THETA	is the rotation angle measured in degrees in a counter-clockwise direction.

A R E A F

AREAF draws polygons.

The call is: CALL AREAF (NXRAY, NYRAY, N) level 1, 2, 3

or: void areaf (int *nxray, int *nyray, int n);

NXRAY, NYRAY	are arrays containing the plot coordinates of the corner points. Start and end points can be different.
N	is the number of points.

The corresponding routines for user coordinates are:

The calls are:

```
CALL  RLREC     (X, Y, WIDTH, HEIGHT)
CALL  RLRND     (X, Y, WIDTH, HEIGHT, IOPT)
CALL  RLCIRC    (XM, YM, R)
CALL  RLELL     (XM, YM, A, B)
CALL  RLPIE     (XM, YM, R, ALPHA, BETA)
CALL  RLARC     (XM, YM, A, B, ALPHA, BETA, THETA)
CALL  RLAREA    (XRAY, YRAY, N)
```

or:

```
void  rlrec     (float x, float y, float width, float height);
void  rlrnd     (float x, float y, float width, float height, int iopt);
void  rlcirc    (float xm, float ym, float r);
void  rlell     (float xm, float ym, float a, float b);
void  rplpie    (float xm, float ym, float r, float alpha, float beta);
void  rlarc     (float xm, float ym, float a, float b, float alpha,
                  float beta, float theta);
void  rlarea    (float *xray, float *yray, int n);
```

Additional notes:

- Shading patterns can be defined with SHDPAT and MYPAT. If the pattern number is zero, the figures will only be outlined. With CALL NOARLN, the outline will be suppressed.
- The number of points in AREAF and RLAREA is limited to 25000 for Fortran 77. There is no limitation for the C and Fortran 90 versions of DISLIN.
- For the calculation of the radius in RLCIRC and RLPIE, the X-axis scaling is used.
- The interpolation of circles and ellipses can be altered with CIRCSP (NSPC) where NSPC is the arc length in plot coordinates. The default value is 10.

Chapter 9

Utility Routines

This chapter describes the utilities available to transform coordinates, sort data and calculate the lengths of numbers and character strings.

9.1 Transforming Coordinates

The following functions convert user coordinates to plot coordinates.

The calls are:	IXP = NXPOSN (X)	level 2, 3
	IYP = NYPOSN (Y)	level 2, 3
or:	int nxposn (float x);	
	int nyposn (float y);	

Plot coordinates can also be returned as real numbers.

The calls are:	XP = XPOSN (X)	level 2, 3
	YP = YPOSN (Y)	level 2, 3
or:	float xposn (float x);	
	float yposn (float y);	

The following two functions convert plot coordinates to user coordinates.

The calls are:	XW = XINVRS (NXP)	level 2, 3
	YW = YINVRS (NYP)	level 2, 3
or:	float xinvrs (int nxp);	
	float yinvrs (int nyp);	

T R F R E L

The routine TRFREL converts arrays of user coordinates to plot coordinates.

The call is:	CALL TRFREL (XRAY, YRAY, N)	level 2, 3
or:	void trfrel (float *xray, float *yray, int n);	

XRAY, YRAY are arrays containing the user coordinates. After the call, they contain the calculated plot coordinates.

N is the number of points.

Additional note: The functions above can be used for linear and logarithmic scaling. For polar scaling, TRFREL and POS2PT can be used for getting plot coordinates.

T R F C O 1

The routine TRFCO1 converts one-dimensional coordinates.

The call is: CALL TRFCO1 (XRAY, N, CFROM, CTO) level 0, 1, 2, 3

or: void trfco1 (float *xray, int n, char *cfrom, char *cto);

XRAY is an array containing angles expressed in radians or degrees. After a call to TRFCO1, XRAY contains the converted coordinates.

N is the number of coordinates.

CFROM, CTO are character strings that can have the values 'DEGREES' and 'RADIANS'.

T R F C O 2

The routine TRFCO2 converts two-dimensional coordinates.

The call is: CALL TRFCO2 (XRAY, YRAY, N, CFROM, CTO) level 0, 1, 2, 3

or: void trfco2 (float *xray, float *yray, int n, char *cfrom, char *cto);

XRAY, YRAY are arrays containing rectangular or polar coordinates. For polar coordinates, XRAY contains the angles measured in degrees and YRAY the radii.

N is the number of coordinates.

CFROM, CTO are character strings that can have the values 'RECT' and 'POLAR'.

T R F C O 3

The routine TRFCO3 converts three-dimensional coordinates.

The call is: CALL TRFCO3 (XRAY, YRAY, ZRAY, N, CFROM, CTO)
level 0, 1, 2, 3

or: void trfco3 (float *xray, float *yray, float *zray, int n, char *cfrom, char *cto);

XRAY, YRAY, ZRAY are arrays containing rectangular, spherical or cylindrical coordinates. Spherical coordinates must be in the form (longitude, latitude, radius) where $0 \leq$ longitude ≤ 360 and $-90 \leq$ latitude ≤ 90.
Cylindrical coordinates must be in the form (angle, radius, z).

N is the number of coordinates.

CFROM, CTO are character strings that can have the values 'RECT','SPHER' and 'CYLI'.

T R F M A T

The routine TRFMAT converts a matrix to another matrix by bilinear interpolation.

The call is: CALL TRFMAT (ZMAT, NX, NY, ZMAT2, NX2, NY2)
level 0, 1, 2, 3

or: void trfmat (float *zmat, int nx, int ny, float *zmat2, int nx2, int ny2);

ZMAT is the input matrix of the dimesion (NX, NY).

NX, NY are the dimensions of the matrix ZMAT.

ZMAT2 is the output matrix of the dimesion (NX2, NY2).

NX2, NY2 are the dimensions of the matrix ZMAT2.

9.2 String Arithmetic

N L M E S S

The function NLMESS returns the length of text in plot coordinates.

The call is:	NL = NLMESS (CSTR)	level 1, 2, 3
or:	int nlmess (char *cstr);	

CSTR	is a character string ($\leq$ 256 characters).
NL	is the length in plot coordinates.

T R M L E N

The function TRMLEN returns the number of characters in a character string.

The call is:	NL = TRMLEN (CSTR)	level 0, 1, 2, 3
or:	int trmlen (char *cstr);	

CSTR	is a character string.
NL	is the number of characters.

U P S T R

UPSTR converts a character string to uppercase letters.

The call is:	CALL UPSTR (CSTR)	level 0, 1, 2, 3
or:	void upstr (char *cstr);	

CSTR	is a character string to be converted.

9.3 Number Arithmetic

N L N U M B

NLNUMB calculates the length of numbers in plot coordinates.

The call is:	NL = NLNUMB (X, NDIG)	level 1, 2, 3
or:	int nlnumb (float x, int ndig);	

X	is a real number.
NDIG	is the number of decimal places ($\geq$ -1).
NL	is the returned length in plot coordinates.

I N T L E N

INTLEN calculates the number of digits in integers.

The call is:	CALL INTLEN (NX, NL)	level 0, 1, 2, 3
or:	int intlen (int nx);	

NX	is an integer.
NL	is the returned number of digits.

F L E N

FLEN calculates the number of digits in real numbers.

The call is:	CALL FLEN (X, NDIG, NL)	level 0, 1, 2, 3
or:	int flen (float x, int ndig);	

X	is a real number.
NDIG	is the number of decimal places ($\geq$ -1).
NL	is the number of digits including the decimal point. For negative numbers, it includes the minus sign.

I N T C H A

INTCHA converts integers to character strings.

The call is:	CALL INTCHA (NX, NL, CSTR)	level 0, 1, 2, 3
or:	int intcha (int nx, char *cstr);	

NX	is the integer to be converted.
NL	is the number of digits in NX returned by INTCHA.
CSTR	is the character string containing the integer.

F C H A

FCHA converts real numbers to character strings.

The call is:	CALL FCHA (X, NDIG, NL, CSTR)	level 0, 1, 2, 3
or:	int fcha (float x, int ndig, char *cstr);	

X	is the real number to be converted.
NDIG	is the number of decimal places to be considered ($\geq$ -1). The last digit will be rounded up.
NL	is the number of digits returned by FCHA.
CSTR	is the character string containing the real number.

S O R T R 1

SORTR1 sorts real numbers.

The call is:	CALL SORTR1 (XRAY, N, COPT)	level 0, 1, 2, 3
or:	void sortr1 (float *xray, int n, char *copt);	

XRAY	is an array containing real numbers.
N	is the dimension of XRAY.
COPT	defines the sorting direction. IF COPT = ’A’, the numbers will be sorted in ascending order; if COPT = ’D’, they will be sorted in descending order.

S O R T R 2

SORTR2 sorts two-dimensional points in the X-direction.

The call is:	CALL SORTR2 (XRAY, YRAY, N, COPT)	level 0, 1, 2, 3
or:	void sortr2 (float *xray, float *yray, int n, char *copt);	

XRAY, YRAY	are arrays containing the coordinates.
N	is the number of points.
COPT	defines the sorting direction. IF COPT = 'A', the points will be sorted in ascending order; if COPT = 'D', they will be sorted in descending order.

Additional note:	The Shell-algorithm is used for sorting.

S P L I N E

SPLINE calculates splined points used in CURVE to plot a spline.

The call is:	CALL SPLINE (XRAY, YRAY, N, XSRAY, YSRAY, NSPL)	level 1, 2, 3
or:	void spline (float *xray, float *yray, float *xsray, float *ysray, int *nspl);	

XRAY, YRAY	are arrays containing points of the curve.
N	is the dimension of XRAY and YRAY.
XSRAY, YSRAY	are the splined points returned by SPLINE.
NSPL	is the number of calculated splined points returned by SPLINE. By default, NSPL has the value 200.

Additional note:	The number of interpolated points and the order of the polynomials can be modified with SPLMOD.

B E Z I E R

The routine BEZIER calculates a Bezier interpolation.

The call is:	CALL BEZIER (XRAY, YRAY, N, XPRAY, YPRAY, NP)	level 0, 1, 2, 3
or:	void bezier (float *xray, float *yray, int n, float *xpray, float *ypray, int np);	

XRAY, YRAY	are arrays containing points of the curve.
N	is the dimension of XRAY and YRAY ($1 < N < 21$).
XPRAY, YPRAY	are the Bezier points returned by BEZIER.
NP	is the number of calculated points defined by the user.

H I S T O G

The routine HISTOG calculates a histogram.

The call is:	CALL HISTOG (XRAY, N, XHRAY, YHRAY, NH)	level 0, 1, 2, 3
or:	void histog (float *xray, int n, float *xhray, float *yhray, int *nh);	

XRAY	is an array containing floatingpoint numbers.
N	is the dimension of XRAY.
XHRAY, YHRAY	are arrays containing the calculated histogram. XHRAY contains distinct values from XRAY sorted in ascending order. YHRAY contains the frequency of points.
NH	is the number of points in XHRAY und YHRAY returned by HISTOG.

T R I A N G

The routine TRIANG calculates the Delaunay triangulation of an arbitrary collection of points in the plane. The Delaunay triangulation can directly be used to display surfaces and contour lines of irregularily distributed data points.

The call is:	CALL TRIANG (XRAY, YRAY, N, I1RAY, I2RAY, I3RAY, NMAX, NTRI) level 0, 1, 2, 3
or:	void triang (float *xray, float *yray, int n, int *i1ray, int *i2ray, int *i3ray, int nmax, int *ntri);

XRAY, YRAY	are arrays containing floatingpoint numbers. The dimension of XRAY and YRAY must be greater or equal N + 3.
N	is the number of points in XRAY and YRAY.
I1RAY, I2RAY, I3RAY	are the returned vertices for each triangle in anticlockwise order.
NMAX	is the dimension of I1RAY, I2RAY and I3RAY. NMAX must be greater of equal 2 * N + 1.
NTRI	is the returned number of triangles.

Additional notes:

- The Watson algorithm is used for calculating the Delaunay triangulation. The algorithm increases with the number of points as approximately $O(N^{1.5})$. Reference: S.W. Sloan and G.T. Houlsby, An Implementation of Watson's algorithm for computing 2-dimensional Delaunay triangulations, Advanced Engineering Software, 1984, Vol. 6, No. 4.
- Surfaces and contours can be directly plotted from the triangulation with the routines CRVTRI, SURTRI and CONTRI.

C I R C 3 P

The routine CIRC3P calculates a circle specified by three points.

The call is:	CALL CIRC3P (X1, Y1, X2, Y2, X3, Y3, XM, YM, R) level 0, 1, 2, 3
or:	void circ3p (float x1, float y1, float x2, float y2, float x3, float y3, float *xm, float *ym, float *r);

X1, Y1	are the X- and Y-coordinates of the first point.
X2, Y2	are the X- and Y-coordinates of the second point.
X3, Y3	are the X- and Y-coordinates of the third point.
XM, YM	are the calculated coordinates of the centre point.
R	is the calculated radius of the circle.

9.4 Date Routines

B A S D A T

The routine BASDAT defines the base date. This routine is necessary for plotting date labels and data containing date coordinates.

The call is:	CALL BASDAT (IDAY, IMONTH, IYEAR) level 0, 1, 2, 3

or:	void basbat (int iday, int imonth, int iyear);
IDAY	is the day number of the date between 1 and 31.
IMONTH	is the month number of the date between 1 and 12.
IYEAR	is the four digit year number of the date.

I N C D A T

The function INCDAT returns the number of days between a specified date and the base date. This calculated days can be passed as parameters to the routine GRAF and as coordinates to data plotting routines such as CURVE.

The call is:	N = INCDAT (IDAY, IMONTH, IYEAR)	level 0, 1, 2, 3
or:	int incdat (int iday, int imonth, int iyear);	
N	is the returned number of calculated days.	
IDAY	is the day number of the date between 1 and 31.	
IMONTH	is the month number of the date between 1 and 12.	
IYEAR	is the four digit year number of the date.	

T R F D A T

The routine TRFDAT calculates for a number of days the corresponding date.

The call is:	CALL TRFDAT (N, IDAY, IMONTH, IYEAR)	level 0, 1, 2, 3
or:	int trfdat (int n, int *iday, int *imonth, int *iyear);	
N	is the number of days.	
IDAY	is the returned day number.	
IMONTH	is the returned month number.	
IYEAR	is the returned four digit year number.	

N W K D A Y

The function NWKDAY returns the weekday for a given date.

The call is:	N = NWKDAY (IDAY, IMONTH, IYEAR)	level 0, 1, 2, 3
or:	int nwkday (int iday, int imonth, int iyear);	
N	is the returned weekday between 1 and 7 (1 = Monday, 2 = Tuesday, ...).	
IDAY	is the day number of the date between 1 and 31.	
IMONTH	is the month number of the date between 1 and 12.	
IYEAR	is the four digit year number of the date.	

9.5 Bit Manipulation

B I T S I 2

The routine BITSI2 allows bit manipulation on 16 bit variables.

The call is:	CALL BITSI2 (NBITS, NINP, IINP, NOUT, IOUT, IOPT)	level 0, 1, 2, 3

or:	short bitsi2 (int nbits, short ninp, int iinp, short nout, int iout);

NBITS	is the number of bits to be shifted.
NINP	is a 16 bit variable from which to extract the bit field.
IINP	is the bit position of the leftmost bit of the bit field. The bits are numbered 0 - 15 where 0 is the most significant bit.
NOUT	is a 16 bit variable into which the bit field is placed.
IOUT	is the bit position where to put the bit field.
IOPT	controls whether the bits outside of the field are set to zero or not. If IOPT equal 0, the bits are set to zero. If IOPT not equal 0, the bits are left as they are. For this case, NOUT is also used as input parameter. In the C function, IOPT is missing in the parameter list and internally used with the value 1.

B I T S I 4

The routine BITSI4 allows bit manipulation on 32 bit variables.

The call is:	CALL BITSI4 (NBITS, NINP, IINP, NOUT, IOUT, IOPT)	level 0, 1, 2, 3
or:	int bitsi4 (int nbits, int ninp, int iinp, int nout, int iout);	

NBITS	is the number of bits to be shifted.
NINP	is a 32 bit variable from which to extract the bit field.
IINP	is the bit position of the leftmost bit of the bit field. The bits are numbered 0 - 31 where 0 is the most significant bit.
NOUT	is a 32 bit variable into which the bit field is placed.
IOUT	is the bit position where to put the bit field.
IOPT	controls whether the bits outside of the field are set to zero or not. If IOPT equal 0, the bits are set to zero. If IOPT not equal 0, the bits are left as they are. For this case, NOUT is also used as input parameter. In the C function, IOPT is missing in the parameter list and internally used with the value 1.

9.6 Byte Swapping

S W A P I 2

The routine SWAPI2 swaps the bytes of 16 bit integer variables.

The call is:	CALL SWAPI2 (IRAY, N)	level 0, 1, 2, 3
or:	void swapi2 (short *iray, int n);	

IRAY	is an array containing the 16 bit variables.
N	is the number of variables.

S W A P I 4

The routine SWAPI4 swaps the bytes of 32 bit integer variables.

The call is:	CALL SWAPI4 (IRAY, N)	level 0, 1, 2, 3
or:	void swapi4 (int *iray, int n);	

IRAY	is an array containing the 32 bit variables.
N	is the number of variables.

9.7 Binary I/O

Binary I/O from Fortran can cause some problems: unformatted IO in Fortran is system-dependent and direct access I/O needs a fixed record length. Therefore, DISLIN offers some C routines callable from Fortran.

O P E N F L

The routine OPENFL opens a file for binary I/O.

The call is:	CALL OPENFL (CFILE, NLU, IRW, ISTAT)	level 0, 1, 2, 3
or:	int openfl (char *cfile, int nlu, int irw);	

CFILE	is a character string containing the file name.
NLU	is the logical unit for the I/O ($0 \leq$ NLU ≤ 99). The units 15 and 16 are reserved for DISLIN.
IRW	defines the file access mode (0: READ, 1: WRITE, 2: APPEND).
ISTAT	is the returned status (0: no errors).

C L O S F L

The routine CLOSFL closes a file.

The call is:	CALL CLOSFL (NLU)	level 0, 1, 2, 3
or:	int closfl (int nlu);	

NLU	is the logical unit.

R E A D F L

The routine READFL reads a given number of bytes.

The call is:	CALL READFL (NLU, IBUF, NBYT, ISTAT)	level 0, 1, 2, 3
or:	int readfl (int nlu, unsigned char *ibuf, int nbyt);	

NLU	is the logical unit.
IBUF	is an array where to read the bytes.
NBYT	is the number of bytes.
ISTAT	is the number of bytes read (0 means end of file).

W R I T F L

The routine WRITFL writes a number of bytes.

The call is:	CALL WRITFL (NLU, IBUF, NBYT, ISTAT)	level 0, 1, 2, 3
or:	int writfl (int nlu, unsigned char *ibuf, int nbyt);	

NLU	is the logical unit.
IBUF	is an array containing the bytes.
NBYT	is the number of bytes.
ISTAT	is the number of bytes written (0 means an error).

S K I P F L

The routine SKIPFL skips a number of bytes from the current position.

The call is:	CALL SKIPFL (NLU, NBYT, ISTAT)	level 0, 1, 2, 3
or:	int skipfl (int nlu, int nbyt);	

NLU	is the logical unit.
NBYT	is the number of bytes.
ISTAT	is the returned status (0: OK).

T E L L F L

The routine TELLFL returns the current position in bytes.

The call is:	CALL TELLFL (NLU, NBYT)	level 0, 1, 2, 3
or:	int tellfl (int nlu);	

NLU	is the logical unit.
NBYT	is the returned position in bytes where byte numbering begins with zero. NBYT = -1, if an error occurs.

P O S I F L

The routine POSIFL skips to a certain position relative to the start.

The call is:	CALL POSIFL (NLU, NBYT, ISTAT)	level 0, 1, 2, 3
or:	int posifl (int nlu, int nbyt);	

NLU	is the logical unit.
NBYT	defines the position. Byte numbering begins with zero.
ISTAT	is the returned status (0: OK).

9.8 Window Terminals

9.8.1 Clearing the Screen

E R A S E

The routine ERASE clears the screen, a graphics window or the page of a raster format such as TIFF, PNG, PPM and BMP. In general, this is done by DISINI at the beginning of a plot.

The call is:	CALL ERASE	level 1, 2, 3
or:	void erase ();	

9.8.2 Clearing the Output Buffer

S E N D B F

Normally, the graphical output to the screen is buffered. To send the buffer to the screen, the routine SENDBF can be used.

The call is:	CALL SENDBF	level 1, 2, 3
or:	void sendbf ();	

9.8.3 Multiple Windows

The following routines allow programs to create up to 8 windows for graphics output on X11 and Windows terminals. Note, that multiple windows can be used with graphic windows but are not compatible with other file formats in DISLIN.

O P N W I N

The routine OPNWIN creates a new window for graphics output on the screen.

The call is:	CALL OPNWIN (ID)	level 1, 2, 3
or:	void opnwin (int id);	

ID is the window number between 1 and 8.

Additional notes:

- The file format must be set to X Window emulation in the routine METAFL (i.e. with the keyword 'XWIN').
- The size and position of windows can be changed with the routines WINDOW and WINSIZ. Note that some X11 Window Managers ignore the user-defined position of windows.
- Windows can be closed and selected with the routines CLSWIN and SELWIN.
- A created window with OPNWIN is selected automatically for graphics output.
- External windows can also be used with OPNWIN if the routine SETXID is called before.
- The routine WINMOD affects the handling of windows in the termination routine DISFIN.

C L S W I N

The routine CLSWIN closes a window created with OPNWIN.

The call is:	CALL CLSWIN (ID)	level 1, 2, 3
or:	void clswin (int id);	

ID is the window number between 1 and 8.

S E L W I N

The routine SELWIN selects a window on the screen where the following graphics output will be sent to.

The call is:	CALL SELWIN (ID)	level 1, 2, 3
or:	void selwin (int id);	

ID is the window number between 1 and 8.

W I N I D

The routine WINID returns the ID of the currently selected window.

The call is:	CALL WINID (ID)	level 1, 2, 3
or:	int winid ();	

ID is the returned window number.

W I N T I T

The routine WINTIT changes the window title of the currently selected window.

The call is: CALL WINTIT (CSTR) level 1, 2, 3

or: void wintit (char *cstr);

CSTR is a character string containing the new window title.

9.8.4 Cursor Routines

The following routines allow an user to collect some X- and Y-coordinates in a graphics window with the mouse. The coordinates can be returned in pixels and in DISLIN plot coordinates. All routines are also available in DISLIN draw widgets.

C S R P O S

The routine CSRPOS sets the position of the mouse pointer and returns the position if a character key or a mouse button is pressed. This routine can be used for cursor navigation.

The call is: CALL CSRPOS (NX, NY, IKEY) level 1, 2, 3

or: int csrpos (int *nx, int *ny);

NX, NY are integer coordinates. On entry, the mouse pointer is set to the position (NX, NY). If a character key is pressed, the position of the mouse is returned in NX and NY.

IKEY is the returned ASCII code for the pressed key. The cursor keys can also be used where the following values are returned: 1 for cursor left, 2 for cursor up, 3 for cursor right, 4 for cursor down. The value 5 is returned if the left mouse button is clicked, and the value 6 for the right mouse button. The value -1 is returned if an error occured.

Additional note: The behavior of CSRPOS can be modified with the routine CSRMOD.

C S R P T 1

The routine CSRPT1 returns the position of the mouse pointer if the mouse button 1 is pressed. The mouse pointer is changed to a cross hair pointer in the graphics window if CSRPT1 is active.

The call is: CALL CSRPT1 (NX, NY) level 1, 2, 3

or: void csrpt1 (int *nx, int *ny);

NX, NY are the returned coordinates of the pressed mouse pointer.

C S R P T S

The routine CSRPTS returns an array of mouse positions. The routine is waiting for mouse button 1 clicks and terminates if mouse button 2 is pressed. The mouse pointer is changed to a cross hair pointer in the graphics window.

The call is: CALL CSRPTS (NXRAY, NYRAY, NMAX, N, IRET) level 1, 2, 3

or: void csrpts (int *nxray, int *nyray, int nmax, int *n, int *iret);

NXRAY, NYRAY	are the returned coordinates of the collected mouse positions.
NMAX	is the dimension of NXRAY and NYRAY and defines the maximal number of points that will be stored in NXRAY and NYRAY.
N	is the number of points that are returned in NXRAY and NYRAY.
IRET	is a returned status. IRET not equal 0 means that not all mouse movements could be stored in NXRAY and NYRAY.

C S R M O V

The routine CSRMOV returns an array of mouse movements. The routine collects the mouse movements of mouse button 1 and terminates if mouse button 1 is released. The mouse pointer is changed to a cross hair pointer in the graphics window.

The call is:	CALL CSRMOV (NXRAY, NYRAY, NMAX, N, IRET)	level 1, 2, 3
or:	void csrmov (int *nxray, int *nyray, int nmax, int *n, int *iret);	

NXRAY, NYRAY	are the returned coordinates of the collected mouse movements.
NMAX	is the dimension of NXRAY and NYRAY and defines the maximal number of points that will be stored in NXRAY and NYRAY.
N	is the number of points that are returned in NXRAY and NYRAY.
IRET	is a returned status. IRET not equal 0 means that not all mouse positions could be stored in NXRAY and NYRAY.

C S R M O D

The routine CSRMOD modifies the behavior of CSRPOS.

The call is:	CALL CSRMOD (CMOD, CKEY)	level 1, 2, 3
or:	void csrmod (char *cmod, char *ckey);	

CMOD	is a character string that can have the values ’STANDARD’, ’SET’, ’GET’ and ’READ’. With the keywords ’SET’ and ’GET’ the cursor position can be defined or requested without waiting for an user event. The value ’READ’ means that the cursor position is not set at the entry of CSRPOS. The value ’STANDARD’ means the default behavior of CSRPOS.
CKEY	is a character string that can have the value ’POS’. Default: (’STANDARD’, ’POS’).

C S R U N I

The routine CSRUNI defines if pixels or plot coordinates are returned by the cursor routines.

The call is:	CALL CSRUNI (COPT)	level 1, 2, 3
or:	void csruni (char *copt);	

COPT	is a character string that can have the values ’PIXEL’ and ’PLOT’. Default: COPT = ’PLOT’.
Additional note:	Plot coordinates can be converted to user coordinates with the routines XINVRS and YINVRS.

C S R T Y P

The routine CSRTYP defines the cursor used by the cursor routine.

The call is:	CALL CSRTYP (COPT)	level 1, 2, 3
or:	void csrtyp (char *copt);	

COPT is a character string that can have the values 'NONE', 'CROSS', 'ARROW' and 'VARROW'. 'NONE' means that the current cursor is not changed.
Default: COPT = 'CROSS'.

S E T C S R

The routine SETCSR defines the cursor that is used by the DISLIN graphics window.

The call is:	CALL SETCSR (COPT)	level 1, 2, 3
or:	void setcsr (char *copt);	

COPT is a character string that can have the values 'ARROW', 'CROSS' and 'VARROW'. Default: COPT = 'ARROW'.

9.9 Elementary Image Routines

The following routines allow transfering of image data between windows, files and arrays. The output format must be an image format such as CONS, TIFF, PNG, BMP and PPM, but the writing of image data to PostScript and PDF files is also supported. If the output format is PostScript or PDF, the size of images and the position of an image on the output page can be defined with the routines IMGSIZ and IMGBOX.

I M G I N I

The routine IMGINI initializes transfering of image data with the routines RPIXEL, RPIXLS, RPXROW, WPIXEL, WPIXLS and WPXROW. If the output format is PostScript or PDF, IMGINI creates a virtual image where image data can be written to.

The call is:	CALL IMGINI	level 1, 2, 3
or:	void imgini ();	

I M G F I N

The routine IMGFIN terminates transfering of image data with the routines RPIXEL, RPIXLS, RPXROW, WPIXEL, WPIXLS and WPXROW. If the output format is PostScript or PDF, the virtual image created in IMGINI is copied to the PostScript or PDF file.

The call is:	CALL IMGFIN	level 1, 2, 3
or:	void imgfin ();	

R P I X E L

The routine RPIXEL reads one pixel from memory.

The call is:	CALL RPIXEL (IX, IY, ICLR)	level 1, 2, 3
or:	void rpixel (int ix, int iy, int *iclr);	

IX, IY is the position of the pixel in screen coordinates.

ICLR	is the returned colour value of the pixel. If the parameter 'RGB' is used in the routine IMGMOD before, RPIXEL returns an explicit RGB value, otherwise an entry of the colour table.

W P I X E L

The routine WPIXEL writes one pixel into memory.

The call is:	CALL WPIXEL (IX, IY, ICLR)	level 1, 2, 3
or:	void wpixel (int ix, int iy, int iclr);	

IX, IY	is the position of the pixel in screen coordinates.
ICLR	is the new colour value of the pixel.

R P I X L S

The routine RPIXLS copies colour values from a rectangle in memory to an array.

The call is:	CALL RPIXLS (IRAY, IX, IY, NW, NH)	level 1, 2, 3
or:	void rpixls (unsigned char *iray, int ix, int iy, int nw, int nh);	

IRAY	is a byte array containing the returned colour values.
IX, IY	contain the starting point in screen coordinates.
NW, NH	are the width and height of the rectangle in screen coordinates.

W P I X L S

The routine WPIXLS copies colour values from an array to a rectangle in memory.

The call is:	CALL WPIXLS (IRAY, IX, IY, NW, NH)	level 1, 2, 3
or:	void wpixls (unsigned char *iray, int ix, int iy, int nw, int nh);	

IRAY	is a byte array containing the colour values.
IX, IY	contain the starting point in screen coordinates.
NW, NH	are the width and height of the rectangle in screen coordinates.

R P X R O W

The routine RPXROW copies one line of colour values from memory to an array.

The call is:	CALL RPXROW (IRAY, IX, IY, N)	level 1, 2, 3
or:	void rpxrow (unsigned char *iray, int ix, int iy, int n);	

IRAY	is a byte array containing the returned colour values.
IX, IY	contain the starting point in screen coordinates.
N	is the number of pixels.

W P X R O W

The routine WPXROW copies colour values from an array to a line in memory.

The call is:	CALL WPXROW (IRAY, IX, IY, N)	level 1, 2, 3

or: void wpxrow (unsigned char *iray, int ix, int iy, int n);

IRAY is a byte array containing the colour values.

IX, IY contain the starting point in screen coordinates.

N is the number of pixels.

Additional note: IMGINI and IMGFIN must be used with the routines RPIXEL, WPIXEL, RPIXLS, WPIXLS, RPXROW and WPXROW.

IMGMOD

The routine IMGMOD defines palette or truecolour mode for the routines RPIXLS, WPIXLS, RPXROW and WPXROW. For palette mode, the byte arrays in the routines above must contain colour indices between 0 and 255. For truecolour mode, the byte arrays must contain RGB values (8 bit for each value).

The call is: CALL IMGMOD (CMOD) level 1, 2, 3

or: void imgmod (char *cmod);

CMOD is a character string that can contain the values 'INDEX' and 'RGB'.
Default: CMOD = 'INDEX'.

IMGSIZ

If the output format is PostScript or PDF, the size of images can be defined with the routine IMGSIZ. The routine must be called before IMGINI.

The call is: CALL IMGSIZ (NW, NH) level 1, 2, 3

or: void imgsiz (int nw, int nh);

NW, NH are the image width and height in pixels.
Default: (853, 603).

IMGBOX

If the output format is PostScript or PDF, a rectangle on the output page can be specified where the image is copied to. The routine IMGBOX must be called before IMGINI.

The call is: CALL IMGBOX (NX, NY, NW, NH) level 1, 2, 3

or: void imgbox (int nx, int ny, int nw, int nh);

NX, NY is the upper left corner of the rectangle on the page in plot coordinates.

NW, NH are the width and height of the rectangle in plot coordinates. NW and NH should have the same ratio as the image that is copied to the rectangle. The default rectangle is the full page.

RIMAGE

The routine RIMAGE copies an image from memory to a file.

The call is: CALL RIMAGE (CFIL) level 1, 2, 3

or: void rimage (char *cfil);

CFIL is the name of the output file. A new file version will be created for existing files (see FILMOD).

Additional notes:	-	Images are stored with an ASCII header of 80 bytes length followed by the binary image data. The format of the image data depends on the video mode and is therefore system-dependent.
	-	A single image file can be displayed with the routine WIMAGE or with the utility program DISIMG. A sequence of image files can be displayed with the utility program DISMOV.

W I M A G E

The routine WIMAGE copies an image from a file to memory.

The call is: CALL WIMAGE (CFIL) level 1, 2, 3

or: void wimage (char *cfil);

CFIL is the name of the input file.

R T I F F

The routine RTIFF copies an image from memory to a file. The image is stored in the device-independent TIFF format.

The call is: CALL RTIFF (CFIL) level 1, 2, 3

or: void rtiff (char *cfil);

CFIL is the name of the output file. A new file version will be created for existing files (see FILMOD).

Additional notes:	-	This image format can be used to export images created with DISLIN into other software packages or to transfer them to other computer systems.
	-	A TIFF file created by DISLIN can be displayed with the routine WTIFF or with the utility program DISTIF.

W T I F F

The routine WTIFF copies a TIFF file created by DISLIN from a file to memory.

The call is: CALL WTIFF (CFIL) level 1, 2, 3

or: void wtiff (char *cfil);

CFIL is the name of the input file.

Note: The position of the TIFF file and a clipping window can be defined with the routines TIFORG and TIFWIN.

T I F O R G

The routine TIFORG defines the upper left corner of the screen where the TIFF file is copied to.

The call is: CALL TIFORG (NX, NY) level 1, 2, 3

or: void tiforg (int nx, int ny);

NX, NY is the upper left corner in screen coordinates.

T I F W I N

The routine TIFWIN defines a clipping window of the TIFF file that can be copied with the routine WTIFF to the screen.

The call is:	CALL TIFWIN (NX, NY, NW, NH)	level 1, 2, 3
or:	void tifwin (int nx, int ny, int nw, int nh);	

NX, NY	is the upper left corner of the clipping window in pixels.
NW, NH	are the width and height of the clipping window in pixels.

R G I F

The routine RGIF copies an image from memory to a GIF file.

The call is:	CALL RGIF (CFIL)	level 1, 2, 3
or:	void rgif (char *cfil);	

CFIL	is the name of the output file. A new file version will be created for existing files (see FILMOD).

R P N G

The routine RPNG copies an image from memory to a PNG file.

The call is:	CALL RPNG (CFIL)	level 1, 2, 3
or:	void rpng (char *cfil);	

CFIL	is the name of the output file. A new file version will be created for existing files (see FILMOD).

R B F P N G

The routine RBFPNG copies an image from memory as a PNG file to a buffer.

The call is:	CALL RBFPNG (CBUF, NMAX, N)	level 1, 2, 3
or:	int rbfpng (char *cbuf, int nmax);	

CBUF	is a character buffer where the image is copied to in PNG format.
NMAX	defines how many bytes can be copied to CBUF. If NMAX = 0, the size of the PNG file is returned in N without copying the PNG file to CBUF.
N	is the returned length of the buffer. $N \leq 0$, if an error occurs.

R P P M

The routine RPPM copies an image from memory to a PPM file.

The call is:	CALL RPPM (CFIL)	level 1, 2, 3
or:	void rppm (char *cfil);	

CFIL	is the name of the output file. A new file version will be created for existing files (see FILMOD).

R B M P

The routine RBMP copies an image from memory to a BMP file.

The call is:	CALL RBMP (CFIL)	level 1, 2, 3
or:	void rbmp (char *cfil);	

CFIL	is the name of the output file. A new file version will be created for existing files (see FILMOD).

I M G C L P

The routine IMGCLP defines a clipping region for the routines RTIFF, RGIF, RPNG, RPPM and RBMP for copying the graphics window to an output file.

The call is:	CALL IMGCLP (NX, NY, NW, NH)	level 1, 2, 3
or:	void imgclp (int nx, int ny, int nw, int nh);	

NX, NY	is the upper left corner of the rectangle in pixels.
NW, NH	are the width and height of the rectangle in pixels.

P D F B U F

The routine PDFBUF copies a PDF file from memory to an user buffer. The routine must be called after DISFIN and PDF buffer output must be enabled with the statment CALL PDFMOD ('ON', 'BUFFER') before DISINI.

The call is:	CALL PDFBUF (CBUF, NMAX, N)	level 0
or:	int pdfbuf (char *cbuf, int nmax);	

CBUF	is a character buffer where the PDF format is copied to.
NMAX	defines how many bytes can be copied to CBUF. If NMAX = 0, the size of the PDF file is returned in N without copying the PDF file to CBUF.
N	is the returned length of the buffer. N $\leq$ 0, if an error occurs.
Additional note:	The PDF file is deleted in memory after it is copied to the buffer.

9.10 Using Threads

T H R I N I

The routine THRINI initializes threads. THRINI must be called before any other DISLIN routine.

The call is:	CALL THRINI (N)
or:	void thrini (int n);

N	is the number of threads that are used by the program.

T H R F I N

The routine THRFIN terminates threads. THRFIN should be called after any other DISLIN routine.

The call is:	CALL THRFIN
or:	void thrfin ();

Additional note:	The thread routines above are only available for the DISLIN C libraries.

9.11 Plotting the MPS Logo

Since the Max Planck Institute for Aeronomie was renamed to Max Planck Institute for Solar System Research in July 2004, DISLIN contains a routine for plotting the new MPS logo.

M P S L O G O

The routine MPSLOGO plots the new MPS logo.

The call is:	CALL MPSLOGO (NX, NY, NSIZE, COPT)
or:	void mpslogo (int nx, int ny, int nsize, char *copt);
NX, NY	defines the position of the MPSLOGO (upper left corner, plot coordinates).
NSIZE	defines the size of the MPSLOGO. NSIZE cam have the pixel values 100, 125, 150, 175, 200 and 300.
COPT	is a character option that can have the values 'NOTEXT' and 'TEXT'.
Additional note:	The MPS logo is included as a bitmap file into a DISLIN graphics where the corresponding bitmap files are not included in a DISLIN distribution. They must be copied separately to the subdirectory mps in the DISLIN directory.

Chapter 10

Business Graphics

This chapter presents business graphic routines to create bar graphs and pie charts.

10.1 Bar Graphs

B A R S

BARS plots bar graphs.

The call is:	CALL BARS (XRAY, Y1RAY, Y2RAY, N)	level 2, 3
or:	void bars (float *xray, float *y1ray, float *y2ray, int n);	

XRAY	is an array of user coordinates defining the position of the bars on the X-axis.
Y1RAY	is an array of user coordinates containing the start points of the bars on the Y-axis.
Y2RAY	is an array of user coordinates containing the end points of the bars on the Y-axis.
N	is the number of bars.

Additional notes:
- Shading patterns of bars can be selected with SHDPAT or MYPAT. Shading numbers will be incremented by 1 after every call to BARS.
- Legends can be plotted for bar graphs.

The following routines modify the appearance of bar graphs.

B A R T Y P

The routine BARTYP defines vertical or horizontal bars.

The call is:	CALL BARTYP (CTYP)	level 1, 2, 3
or:	void bartyp (char *ctyp);	

CTYP	is a character string defining the bar type.
= 'VERT'	means that vertical bars will be plotted.
= 'HORI'	means that horizontal bars will be plotted. If this parameter is used, XRAY defines the position of the bars on the Y-axis while Y1RAY and Y2RAY define the position of the bars on the X-axis.
= '3DVERT'	defines vertical 3-D bars.

= '3DHORI' defines horizontal 3-D bars.

Default: CTYP = 'VERT'.

C H N B A R

CHNBAR modifies colours and shading patterns for single bars.

The call is:	CALL CHNBAR (CATT)	level 1, 2, 3
or:	void chnbar (char *catt);	

CATT is a character string defining bar attributes.

= 'NONE' means that all bars will be plotted with the current colour and shading pattern.

= 'COLOR' means that the colour is changed for each bar.

= 'PATTERN' means that the shading pattern is changed for each bar.

= 'BOTH' means that the colour and shading pattern is changed for each bar.

Default: CATT = 'NONE'.

Additional notes:

- The sequence of colours is: WHITE/BLACK, RED, GREEN, YELLOW, BLUE, ORANGE, CYAN, MAGENTA.
  The sequence of shading patterns is 0 - 17.
  Colour and pattern cycles can be changed with CLRCYC and PATCYC.
- If the routine BARCLR is used, the changing of colours will be ignored.

B A R W T H

BARWTH defines the width of the bars.

The call is:	CALL BARWTH (XWTH)	level 1, 2, 3
or:	void barwth (float xwth);	

XWTH is a real number defining the width. If XWTH is positive, the bar width is the absolute value of XWTH * (XRAY(1)-XRAY(2)). If XWTH is negative, the absolute value of XWTH is used where XWTH must be specified in plot coordinates.

Default: XWTH = 0.75

B A R M O D

BARMOD modifies the width of bars.

The call is:	CALL BARMOD (CMOD, COPT)	level 1, 2, 3
or:	void barmod (char *cmod, char *copt);	

CMOD is a character string that can have the values 'FIXED' and 'VARIABLE'. If CMOD = 'VARIABLE', the width of bars plotted by the routine BARS will be variable. In that case, XWTH should have a positive value in BARWTH since the width of bars is calculated in a simular way as described in BARWTH.

COPT is a character string that must contain the value 'WIDTH'. Default: ('FIXED', 'WIDTH').

B A R P O S

The position of the bars is determined by the parameters XRAY, Y1RAY and Y2RAY. The routine BARPOS can be used to select predefined positions. The parameters XRAY, Y1RAY and Y2RAY will contain the calculated positions.

The call is:	CALL BARPOS (COPT)	level 1, 2, 3
or:	void barpos (char *copt);	

COPT	is a character string that defines the position of the bars.
= 'NONE'	means that the positions are defined only by the parameters in BARS.
= 'TICKS'	means that the bars will be centred at major ticks. XRAY must be a dummy vector.
= 'AXIS'	means that vertical bars start at the X-axis and horizontal bars at the Y-axis. Y1RAY must be a dummy vector.
= 'BOTH'	activates the options 'TICKS' and 'AXIS'. XRAY and Y1RAY must be dummy arrays. Default: COPT = 'NONE'.

Bars can be plotted on top of one another if the routine BARS is called several times. To plot bars side by side in groups, the routine BARGRP can be used.

B A R G R P

The routine BARGRP puts bars with the same axis position into groups. The number of group elements should be the same as the number of calls to the routine BARS.

The call is:	CALL BARGRP (NGRP, GAP)	level 1, 2, 3
or:	void bargrp (int ngrp, float gap);	

NGRP	is the number of bars defining one group.
GAP	defines the spacing between group bars. If GAP is positive, the value GAP * W is used where W is the width of a single bar. If GAP is negative, the positive value of GAP is used where GAP must be specified in plot coordinates.

B A R C L R

The routine BARCLR defines the colours of bars. Different colours can be defined for the sides of 3-D bars.

The call is:	CALL BARCLR (IC1, IC2, IC3)	level 1, 2, 3
or:	void barclr (int ic1, int ic2, int ic3);	

IC1, IC2, IC3	are colour values for the front, side and top planes of 3-D bars. The value -1 means that the corresponding plane is plotted with the current colour. Default: (-1, -1, -1).

B A R B O R

The routine BARBOR defines the colour of borders plotted around the bars. By default, a border in the current colour is plotted around 2-D bars, and borders in the foreground colour are plotted around 3-D bars.

The call is:	CALL BARBOR (IC)	level 1, 2, 3
or:	void barbor (int ic);	

IC — is a colour value. If IC = -1, the bar borders will be plotted with the current colour.

Default: IC = -1

B A R O P T

The routine BAROPT modifies the appearance of 3-D bars.

The call is:	CALL BAROPT (XF, ANG)	level 1, 2, 3
or:	void baropt (float xf, float ang);	

XF — is a floatingpoint number that defines the depth of bars. IF XF = -1., the bar width is used for the bar depth. IF XF $>$ 0., XF is interpreted as the bar depth specified in plot coordinates.

ANG — defines an angle measured in degrees between the front and side planes of 3-D bars.

Default: (-1., 45.).

L A B E L S

The routine LABELS defines labels for bar graphs.

The call is:	CALL LABELS (CLAB, 'BARS')	level 1, 2, 3
or:	void labels (char *clab, "BARS");	

CLAB	is a character defining the labels.
= 'NONE'	means that no labels will be plotted.
= 'SECOND'	means that Y2RAY is used for labels.
= 'FIRST'	means that Y1RAY is used for labels.
= 'DELTA'	means that the difference vector (Y2RAY - Y1RAY) is used for labels.

Default: CLAB = 'NONE'.

L A B P O S

The routine LABPOS defines the position of the labels.

The call is:	CALL LABPOS (CPOS, 'BARS')	level 1, 2, 3
or:	void labpos (char *cpos, "BARS");	

CPOS	is a character string that defines the position of the labels.
= 'INSIDE'	means inside at the end of a bar.
= 'OUTSIDE'	means outside at the end of a bar.
= 'LEFT'	defines the upper left side.
= 'RIGHT'	defines the upper right side.
= 'CENTER'	selects the centre of a bar.

= 'AUTO'	means 'INSIDE' if labels are smaller than the bar width, otherwise 'OUTSIDE'.	

Default: CPOS = 'AUTO'.

L A B D I G

The routine LABDIG defines the number of decimal places in the labels.

The call is:	CALL LABDIG (NDIG, 'BARS')	level 1, 2, 3
or:	void labdig (int ndig, "BARS");	

NDIG	is the number of decimal places ($\geq$ -1).

Default: NDIG = 1

L A B C L R

The routine LABCLR defines the colour of labels.

The call is:	CALL LABCLR (NCLR, 'BARS')	level 1, 2, 3
or:	void labclr (int nclr, "BARS");	

NCLR	is a colour value. If NCLR = -1, the bar labels will be plotted with the current colour.

Default: NCLR = -1

10.2 Pie Charts

P I E G R F

PIEGRF plots pie charts.

The call is:	CALL PIEGRF (CBUF, NLIN, XRAY, NSEG)	level 1
or:	void piegrf (char *cbuf, int nlin, float *xray, int nseg);	

CBUF	is a character string containing text lines for segment labels. More than one line can be defined for labels. CBUF must be created with LEGLIN after calling LEGINI. If NLIN is 0 in the parameter list, CBUF can be a dummy variable.
NLIN	is the number of text lines used for one segment label.
XRAY	is an array of user coordinates.
NSEG	is the dimension of XRAY.

Additional notes:

- The centre and the size of pies is defined by a region that can be changed with the routines AXSPOS and AXSLEN.
- PIEGRF sets the level to 2. Titles and legends can be plotted after PIEGRF is called.
- Segment labels can contain several lines of text and the data values specified in PIEGRF. Data values can also be converted to percent values.
- Segment labels are contained within a box where the thickness of the border can be changed with FRAME.

The following routines modify the appearance of pie charts.

P I E T Y P

The routine PIETYP defines 2-D or 3-D pie charts.

The call is:	CALL PIETYP (CTYP)	level 1, 2, 3
or:	void pietyp (char *ctyp);	

CTYP	is a character string defining the pie type.
= ’2D’	defines a 2-D pie chart.
= ’3D’	defines a 3-D pie chart.

Default: CTYP = ’2D’.

C H N P I E

CHNPIE defines colours and shading patterns for pie graphs.

The call is:	CALL CHNPIE (CATT)	level 1, 2, 3
or:	void chnpie (char *catt);	

CATT	is a character string defining segment attributes.
= ’NONE’	means that all pie segments will be plotted with the current colour and shading pattern.
= ’COLOR’	means that every segment will have a different colour.

= 'PATTERN'	means that every segment will have a different shading pattern.
= 'BOTH'	means that every segment will have both a different colour and shading pattern. Default: CATT = 'PATTERN'.

Additional note: The sequence of colours is: WHITE/BLACK, RED, GREEN, YELLOW, BLUE, ORANGE, CYAN, MAGENTA.
The sequence of shading patterns is 0 - 17.
Colour and pattern cycles can be changed with CLRCYC and PATCYC.

L A B E L S

LABELS selects data or percent values used for segment labels.

The call is:	CALL LABELS (CLAB, 'PIE')	level 1, 2, 3
or:	void labels (char *clab, "PIE");	

CLAB	is a character string that defines the values used for segment labels.
= 'NONE'	means that data values will not be displayed.
= 'PERCENT'	means that values will be plotted as percentages.
= 'DATA'	means that the data values specified in PIEGRF will be plotted.
= 'BOTH'	means both 'PERCENT' and 'DATA'. Default: CDOC = 'PERCENT'.

L A B P O S

LABPOS determines the position of segment labels.

The call is:	CALL LABPOS (CPOS, 'PIE')	level 1, 2, 3
or:	void labpos (char *cpos, "PIE");	

CPOS	is a character string defining the position of labels.
= 'INTERNAL'	means that labels will be plotted inside pie segments. If labels are too big, they will be plotted outside.
= 'EXTERNAL'	means that segment labels will be plotted outside pie segments.
= 'ALIGNED'	means that segment labels will be plotted outside pie segments and aligned. Default: CPOS = 'INTERNAL'.

L A B T Y P

LABTYP defines the position of text lines in segment labels.

The call is:	CALL LABTYP (CTYP, 'PIE')	level 1, 2, 3
or:	void labtyp (char *ctyp, "PIE");	

CTYP	is a character string that defines how text lines are justified.
= 'CENTER'	centres text lines.
= 'LEFT'	left-justifies text lines.
= 'RIGHT'	right-justifies text lines.
= 'OUTWARDS'	left-justifies text lines on the left side of pies and right-justifies text lines on the right side of pies.

= 'INWARDS'	right-justifies text lines on the left side of pies and left-justifies text lines on the right side of pies.

Default: CTYP = 'CENTER'.

L A B D I G

The routine LABDIG defines the number of decimal places used in segment labels.

The call is:	CALL LABDIG (NDIG, CDIG)	level 1, 2, 3
or:	void labdig (int ndig, char *cdig);	

NDIG	is the number of decimal places ($\geq$ -1).
CDIG	is a character string selecting the data values.
= 'PIE'	defines the number of decimal places used for percent and data values.
= 'PERCENT'	defines the number of decimal places used for percent values.
= 'DATA'	defines the number of decimal places used for data values.

Default: (1, 'PIE').

L A B C L R

The routine LABCLR defines the colour of labels.

The call is:	CALL LABCLR (NCLR, 'PIE')	level 1, 2, 3
or:	void labclr (int nclr, "PIE");	

NCLR	is a colour value. If NCLR = -1, the pie labels will be plotted with the current colour.

Default: NCLR = -1

P I E C L R

The routine PIECLR defines colours for single pies. Different colours can be defined for the top and front sides of 3-D pies. PIECLR has no effect if the routine CHNPIE is called with the parameters 'COLOR' or 'BOTH'.

The call is:	CALL PIECLR (NC1RAY, NC2RAY, N)	level 1, 2, 3
or:	void pieclr (int *nc1ray, int *nc2ray, int n);	

NC1RAY, NC2RAY	are integer arrays containing colour values for the top and front sides of pies. The value -1 means that the current colour is used.
N	is the dimension of NC1RAY and NC2RAY.

P I E B O R

The routine PIEBOR defines the colour of borders plotted around the pies. By default, a border in the current colour is plotted around 2-D pies, and borders in the foreground colour are plotted around 3-D pies.

The call is:	CALL PIEBOR (IC)	level 1, 2, 3
or:	void piebor (int ic);	

IC	is a colour value. If IC = -1, the pie borders will be plotted with the current colour.

Default: IC = -1

P I E O P T

The routine PIEOPT modifies the appearance of 3-D pies.

The call is:	CALL PIEOPT (XF, ANG)	level 1, 2, 3
or:	void pieopt (float xf, float ang);	

XF	is a scaling number that defines the thickness of pies. The thickness is set to XF * radius.
ANG	defines an view angle measured in degrees.

Default: (0.2, 45.).

P I E L A B

The routine PIELAB defines character strings that can be plotted on the left or right side of data values within segment labels.

The call is:	CALL PIELAB (CLAB, CPOS)	level 1, 2, 3
or:	void pielab (char *clab, char *cpos);	

CLAB	is a character string displayed in segment labels.
CPOS	is a character string that defines the position of CLAB.
= ’LEFT’	means that CLAB will be plotted on the left side of data values.
= ’RIGHT’	means that CLAB will be plotted on the right side of data values.
Additional note:	If percent and data values are plotted in segment labels, PIELAB is only used for data values.

P I E E X P

Pie segments will be offset by 8% of the radius if PIEEXP is called.

The call is:	CALL PIEEXP	level 1, 2, 3
or:	void pieexp ();	

Additional note:	Single segments will be offset if the corresponding values in PIEGRF are negative.

P I E V E C

PIEVEC modifies the arrows plotted between segments and labels that lie outside of segments.

The call is:	CALL PIEVEC (IVEC, COPT)	level 1, 2, 3
or:	void pievec (int ivec, char *copt);	

IVEC	defines the arrow head (see VECTOR).
COPT	is a character string that defines the vector plotted between segments and labels.
= ’NONE’	suppresses vectors.
= ’STRAIGHT’	means that straight vectors will be plotted.
= ’BROKEN’	means that broken vectors will be plotted.

Default: (2301, ’BROKEN’).

U S R P I E

USRPIE is a user-defined subroutine that can modify pie charts such as suppressing certain labels. USRPIE is called by PIEGRF for each segment.

The call is:	CALL USRPIE (ISEG, XDAT, XPER, NRAD, NOFF, ANGLE, NVY, IDRW, IANN)	level 1, 2, 3
or:	void usrpie(int iseg, float xdat, float xper, int *nrad, int *noff, float *angle, int *nvy, int *idrw, int *iann);	

ISEG	is the segment index (starting with 1).
XDAT	is the data value of the segment as specified in PIEGRF.
XPER	is the percent value of XDAT.
NRAD	is the segment radius in plot coordinates.
NOFF	is the segment offset in plot coordinates (default: 0).
ANGLE	is the offset angle measured in degrees in a counter-clockwise direction. The default value is the angle which bisects the segment.
NVY	shifts the segment label in the Y-direction by NVY plot coordinates.
IDRW	defines the plotting of segments. If IDRW = 0, plotting will be suppressed (default: 1).
IANN	defines the plotting of labels. If IANN = 0, labels will be suppressed (default: 1).

Additional note:	The first 3 parameters of USRPIE are only given for information and cannot be changed by the user.

10.3 Examples

```
      PROGRAM EX10_1
      DIMENSION X(9),Y(9),Y1(9),Y2(9),Y3(9)
      CHARACTER*60 CTIT,CBUF*24

      DATA  X/1.,2.,3.,4.,5.,6.,7.,8.,9./ Y/9*0./
     *     Y1/1.,1.5,2.5,1.3,2.0,1.2,0.7,1.4,1.1/
     *     Y2/2.,2.7,3.5,2.1,3.2,1.9,2.0,2.3,1.8/
     *     Y3/4.,3.5,4.5,3.7,4.,2.9,3.0,3.2,2.6/

      NYA=2700
      CTIT='Bar Graphs (BARS)'

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX
      CALL TICKS(1,'X')
      CALL INTAX
      CALL AXSLEN(1600,700)
      CALL TITLIN(CTIT,3)
```

```
CALL LEGINI(CBUF,3,8)
CALL LEGLIN(CBUF,'FIRST',1)
CALL LEGLIN(CBUF,'SECOND',2)
CALL LEGLIN(CBUF,'THIRD',3)
CALL LEGTIT(' ')

CALL SHDPAT(5)
DO I=1,3
  IF(I.GT.1) CALL LABELS('NONE','X')
  CALL AXSPOS(300,NYA-(I-1)*800)

  CALL GRAF(0.,10.,0.,1.,0.,5.,0.,1.)

  IF(I.EQ.1) THEN
    CALL BARGRP(3,0.15)
    CALL BARS(X,Y,Y1,9)
    CALL BARS(X,Y,Y2,9)
    CALL BARS(X,Y,Y3,9)
    CALL RESET('BARGRP')
  ELSE IF(I.EQ.2) THEN
    CALL HEIGHT(30)
    CALL LABELS('DELTA','BARS')
    CALL LABPOS('CENTER','BARS')
    CALL BARS(X,Y,Y1,9)
    CALL BARS(X,Y1,Y2,9)
    CALL BARS(X,Y2,Y3,9)
    CALL HEIGHT(36)
  ELSE IF(I.EQ.3) THEN
    CALL LABELS('SECOND','BARS')
    CALL LABPOS('OUTSIDE','BARS')
    CALL BARS(X,Y,Y1,9)
  END IF

  IF(I.NE.3) CALL LEGEND(CBUF,7)

  IF(I.EQ.3) THEN
    CALL HEIGHT(50)
    CALL TITLE
  END IF

  CALL ENDGRF
END DO

CALL DISFIN
END
```

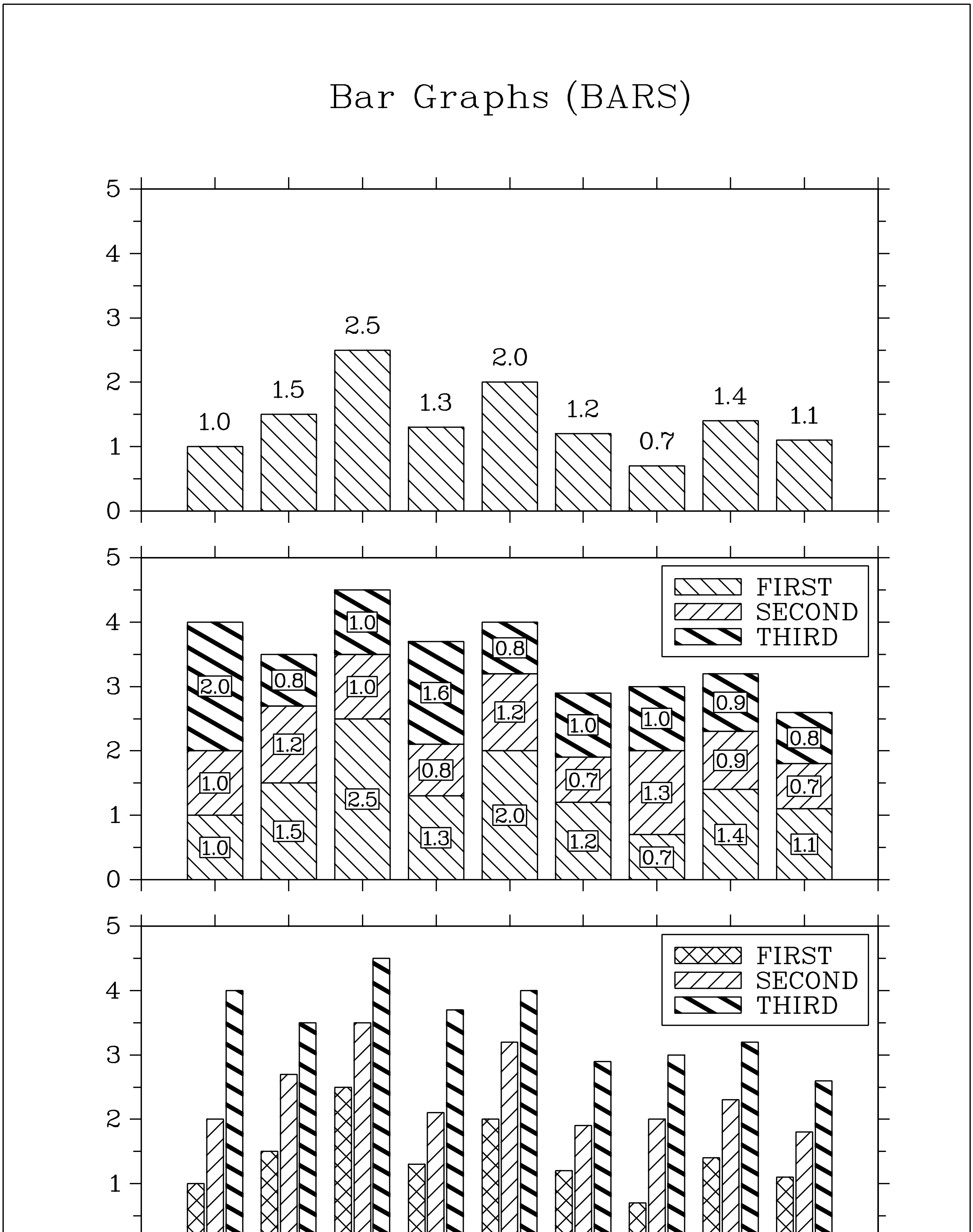


Figure 10.1: Bar Graphs

```
      PROGRAM EX10_2
      DIMENSION XRAY(5)
      CHARACTER*60 CTIT,CBUF*40
      DATA XRAY/1.,2.5,2.,2.7,1.8/

      CTIT='Pie Charts (PIEGRF)'
      NYA=2800

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX
      CALL AXSLEN(1600,1000)
      CALL TITLIN(CTIT,2)

      CALL LEGINI(CBUF,5,8)
      CALL LEGLIN(CBUF,'FIRST',1)
      CALL LEGLIN(CBUF,'SECOND',2)
      CALL LEGLIN(CBUF,'THIRD',3)
      CALL LEGLIN(CBUF,'FOURTH',4)
      CALL LEGLIN(CBUF,'FIFTH',5)

C     Selecting shading patterns
      CALL PATCYC(1,7)
      CALL PATCYC(2,4)
      CALL PATCYC(3,13)
      CALL PATCYC(4,3)
      CALL PATCYC(5,5)

      DO I=1,2
        CALL AXSPOS(250,NYA-(I-1)*1200)
        IF(I.EQ.2) THEN
          CALL LABELS('DATA','PIE')
          CALL LABPOS('EXTERNAL','PIE')
        END IF

        CALL PIEGRF(CBUF,1,XRAY,5)

        IF(I.EQ.2) THEN
          CALL HEIGHT(50)
          CALL TITLE
        END IF
        CALL ENDGRF
      END DO
      CALL DISFIN
      END
```

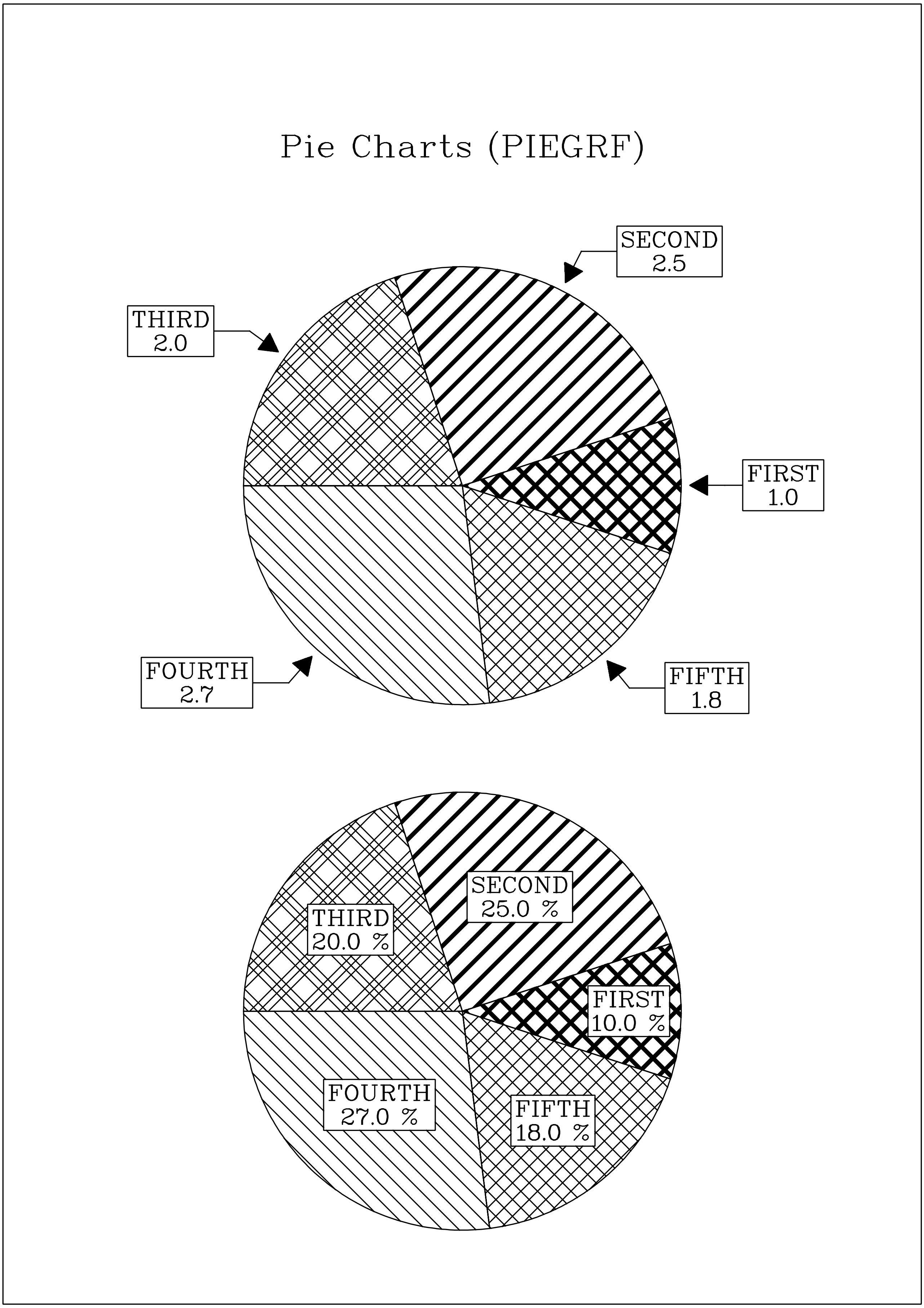


Figure 10.2: Pie Charts

Chapter 11

3-D Colour Graphics

11.1 Introduction

This chapter presents subroutines that plot coloured surfaces in three dimensions. Coloured surfaces are easy to interpret and show the full range of data points. A data point is plotted as a coloured rectangle where the X- and Y-coordinates determine the position of the rectangle and the Z-coordinate defines the colour. Colours are calculated from a scaled colour bar which is, by default, arranged as a rainbow.

11.2 Plotting Coloured Axis Systems

G R A F 3

The routine GRAF3 plots a 3-D axis system where the Z-axis is plotted as a colour bar.

The call is:	CALL GRAF3 (XA, XE, XOR, XSTEP, YA, YE, YOR, YSTEP, ZA, ZE, ZOR, ZSTEP)	level 1
or:	void graf3 (float xa, float xe, float xor, float xstep, float ya, float ye, float yor, float ystep, float za, float ze, float zor, float zstep);	

XA, XE	are the lower and upper limits of the X-axis.
XOR, XSTEP	are the first X-axis label and the step between labels.
YA, YE	are the lower and upper limits of the Y-axis.
YOR, YSTEP	are the first Y-axis label and the step between labels.
ZA, ZE	are the lower and upper limits of the Z-axis.
ZOR, ZSTEP	are the first Z-axis label and the step between labels.
Additional note:	GRAF3 must be called from level 1 and sets the level to 3. For additional notes, the user is referred to the routine GRAF in chapter 4.

11.3 Secondary Colour Bars

GRAF3 plots a vertical colour bar on the right side of a 3-D axis system which can be shifted with the routines VKXBAR and VKYBAR or suppressed with the routine NOBAR. To plot horizontal colour bars at global positions, the routines ZAXIS and ZAXLG can be used. ZAXIS plots a linearly and ZAXLG a logarithmically scaled colour bar.

The call is: CALL ZAXIS (A, B, OR, STEP, NL, CSTR, IT, NDIR, NX, NY) level 1, 2, 3

or: void zaxis (float a, float b, float or, float step, int nl, char *cstr, int nx, int ny);

A, B	are the lower and upper limits of the colour bar.
OR, STEP	are the first label and the step between labels.
NL	is the length of the colour bar in plot coordinates.
CSTR	is a character string containing the axis name.
IT	indicates how ticks, labels and the axis name are plotted. If IT = 0, they are plotted in a clockwise direction. If IT = 1, they are plotted in a counter-clockwise direction.
NDIR	defines the direction of the colour bar. If NDIR = 0, a vertical colour bar will be plotted; if NDIR = 1, a horizontal colour bar will be plotted.
NX, NY	are the plot coordinates of the lower left corner.

Analog: ZAXLG plots a logarithmically scaled colour bar.

Additional note: The user is referred to the notes on secondary axes in chapter 4.

11.4 Plotting Data Points

The routines CURVE3, CURVX3, CURVY3, CRVMAT and CRVTRI plot three-dimensional data points. CURVE3 plots random points from X-, Y- and Z-arrays, CURVY3 plots points as columns, CURVX3 plots data points as rows, CRVMAT plots a coloured surface according to a matrix while CRVTRI plots the surface of the Delaunay triangulation of the points.

The calls are:

CALL CURVE3 (XRAY, YRAY, ZRAY, N)	level 3
CALL CURVX3 (XRAY, Y, ZRAY, N)	level 3
CALL CURVY3 (X, YRAY, ZRAY, N)	level 3
CALL CRVMAT (ZMAT, IXDIM, IYDIM, IXPTS, IYPTS)	level 3
CALL CRVTRI (XRAY, YRAY, ZRAY, N, I1RAY, I2RAY,I3RAY, NTRI)	level 3

or:

```
void curve3 (float *xray, float *yray, float *zray, int n);
void curvx3 (float *xray, float y, float *zray, int n);
void curvy3 (float x, float *yray, float *zray, int n);
void crvmat (float *zmat, int ixdim, int iydim, int ixpts, int iypts);
void crvtri (float *xray, float *yray, float *zray, int n,
             int *i1ray, int *i2ray, int *i3ray, int ntri);
```

XRAY	is an array containing the X-coordinates of data points.
YRAY	is an array containing the Y-coordinates of data points.
ZRAY	is an array containing the Z-coordinates of data points.
N	is the number of data points.
X	is the X-position of a column of data points.
Y	is the Y-position of a row of data points.

ZMAT is a matrix of the dimension (IXDIM, IYDIM) containing Z-coordinates. The coordinates correspond to a linear grid that overlays the axis system. If XA, XE, YA and YE are the axis limits in GRAF3 or values defined with the routine SURSZE, the relationship between the grid points and the matrix elements can be described by the formula:

ZMAT(I,J) = F(X,Y) where

X = XA + (I - 1) * (XE - XA) / (IXDIM - 1) I = 1,..,IXDIM and
Y = YA + (J - 1) * (YE - YA) / (IYDIM - 1) J = 1,..,IYDIM.

IXDIM, IYDIM define the dimension of ZMAT (≥ 2).

IXPTS, IYPTS are the number of interpolation steps between grid lines (≥ 1). CRVMAT can interpolate points linearly.

I1RAY, I2RAY, I3RAY is the Delaunay triangulation of the points (XRAY, YRAY) calculated by the routine TRIANG.

NTRI is the number of triangles in I1RAY, I2RAY and I3RAY.

Additional notes:

- CURVE3, CURVY3 and CRVMAT must be called after GRAF3 from level 3.
- The size of coloured rectangles can be defined with the routine SETRES or calculated automatically by DISLIN using the routine AUTRES.
- Z-coordinates that lie outside of the axis scaling will be plotted with the colour 0 if they are smaller than the lower limit, or with the colour 255 if they are greater than the upper limit. To reduce computing time and the size of plotfiles, the plotting of points with the colour 0 can be suppressed with the routine NOBGD.
- The routines CONMAT and SURMAT are analogs to CRVMAT and plot contours and surfaces of space.
- If SHDMOD ('SMOOTH', 'SURFACE') is called before CRVTRI, the triangles will be plotted with interpolated colours. For that case, a raster format is needed as output format.

11.5 Parameter Setting Routines

S E T R E S

SETRES defines the size of rectangles plotted by CURVE3, CURVY3 and CRVMAT.

The call is: CALL SETRES (NPB, NPH) level 1, 2, 3
or: void setres (int npb, int nph);

NPB, NPH are the width and height of rectangles in plot coordinates (> 0).
Default: (1,1).

A U T R E S

With a call to AUTRES, the size of coloured rectangles will be automatically calculated by GRAF3 or CRVMAT.

The call is: CALL AUTRES (IXDIM, IYDIM) level 1
or: void autres (int ixdim, int iydim);

IXDIM, IYDIM	are the number of data points in the X- and Y-direction.

S H D M O D

Normally, the routines CURVE3, CURVX3, CURVY3 and CRVMAT plot coloured rectangles, but a symbol mode can be enabled with the routine SHDMOD. The symbols used by the routines above and the size of the symbols can be set with the routines MARKER and HSYMBL.

The call is:	CALL SHDMOD (COPT, 'CURVE')	level 1, 2, 3
or:	void shdmod (char *copt, "CURVE");	

COPT — is a character string that can have the values 'RECT' and 'SYMB'.
Default: COPT = 'RECT'.

A X 3 L E N

The routine AX3LEN defines the axis lengths of a coloured axis system.

The call is:	CALL AX3LEN (NXL, NYL, NZL)	level 1, 2, 3
or:	void ax3len (int nxl, int nyl, int nzl);	

NXL, NYL, NZL — are the axis lengths in plot coordinates.

W I D B A R

The routine WIDBAR defines the width of a colour bar.

The call is:	CALL WIDBAR (NZB)	level 1, 2, 3
or:	void widbar (int nzb);	

NZB — is the width in plot coordinates. Default NZB = 85

V K X B A R

The routine VKXBAR defines horizontal shifting of colour bars. The distance between the colour bar and the axis system is, by default, 85 plot coordinates.

The call is:	CALL VKXBAR (NVFX)	level 1, 2, 3
or:	void vkxbar (int nvfx);	

NVFX — is an integer that defines the shifting. If NVFX is positive, the colour bar will be shifted right; if NVFX is negative the colour bar will be shifted left.
Default: NVFX = 0

V K Y B A R

The routine VKYBAR defines a vertical shifting of colour bars.

The call is:	CALL VKYBAR (NVFY)	level 1, 2, 3
or:	void vkybar (int nvfy);	

NVFY — is an integer that defines the shifting. If NVFY is positive, the colour bar will be shifted up; if NVFY is negative, the colour bar will be shifted down.
Default: NVFY = 0

N O B A R

The routine NOBAR instructs DISLIN to suppress the plotting of colour bars.

The call is:	CALL NOBAR	level 1, 2, 3
or:	void nobar ();	

C O L R A N

This routine defines the range of colours used for colour bars. By default, the range is 1 to 254.

The call is:	CALL COLRAN (NCA, NCE)	level 1, 2, 3
or:	void colran (int nca, int nce);	

NCA, NCE	are colour numbers in the range 1 to 254.	Default: (1, 254).

N O B G D

With a call to the routine NOBGD, the plotting of points with the colour 0 will be suppressed. This reduces plotting time and the size of plotfiles.

The call is:	CALL NOBGD	level 1, 2, 3
or:	void nobgd ();	

E X P Z L B

The routine EXPZLB expands the numbering of a logarithmically scaled Z-axis to the next order of magnitude that lies up or down the axis limits. The scaling of the colour bar will not be changed. This routine is useful if the range of the Z-axis scaling is smaller than 1 order of magnitude.

The call is:	CALL EXPZLB (CSTR)	level 1, 2, 3
or:	void expzlb (char *cstr);	

CSTR	is a character string defining the expansion of the Z-axis numbering.
= 'NONE'	means that the numbering will not be expanded.
= 'FIRST'	means that the numbering will be expanded downwards.
= 'BOTH'	means that the numbering will be expanded down- and upwards.

Default: CSTR = 'NONE'.

11.6 Elementary Plot Routines

The following routines plot coloured rectangles and pie sectors. They use the hardware features of a colour graphics system or PostScript printer.

R E C F L L

The routine RECFLL plots a coloured rectangle where the position is determined by the upper left corner.

The call is:	CALL RECFLL (NX, NY, NW, NH, NCOL)	level 1, 2, 3
or:	void recfll (int nx, int ny, int nw, int nh, int ncol);	

NX, NY	are the plot coordinates of the upper left corner.

NW, NH	are the width and height in plot coordinates.
NCOL	is a colour value.

P O I N T

The routine POINT plots a coloured rectangle where the position is determined by the centre.

The call is:	CALL POINT (NX, NY, NW, NH, NCOL)	level 1, 2, 3
or:	void point (int nx, int ny, int nw, int nh, int ncol);	

NX, NY	are the plot coordinates of the centre point.
NW, NH	are the width and height in plot coordinates.
NCOL	is a colour value.

R L P O I N

The routine RLPOIN plots a coloured rectangle where the position is specified in user coordinates.

The call is:	CALL RLPOIN (X, Y, NW, NH, NCOL)	level 2, 3
or:	void rlpoin (float x, float y, int nw, int nh, int ncol);	

Additional note: RLPOIN clips rectangles at the borders of an axis system.

S E C T O R

The routine SECTOR plots coloured pie sectors.

The call is:	CALL SECTOR (NX, NY, NR1, NR2, ALPHA, BETA, NCOL)	level 1, 2, 3
or:	void sector (int nx, int ny, int nr1, int nr2, float alpha, float beta, int ncol);	

NX, NY	are the plot coordinates of the centre point.
NR1	is the interior radius.
NR2	is the exterior radius.
ALPHA, BETA	are the start and end angles measured in degrees in a counter-clockwise direction.
NCOL	is a colour value.

Example: CALL SECTOR (100, 100, 0, 50, 0., 360., NCOL) plots a circle around the centre (100,100) with the radius 50 and the colour NCOL.

R L S E C

The routine RLSEC plots coloured pie sectors where the centre and the radii are specified in user coordinates.

The call is:	CALL RLSEC (X, Y, R1, R2, ALPHA, BETA, NCOL)	level 2, 3
or:	void rlsec (float x, float y, float r1, float r2, float alpha, float beta, int ncol);	

Additional Notes:

- For the conversion of the radii to plot coordinates, the scaling of the X-axis is used.
- Sectors plotted by RLSEC will not be cut off at the borders of an axis system.

11.7 Conversion of Coordinates

The function NZPOSN and the subroutine COLRAY convert user coordinates to colour values.

N Z P O S N

The function NZPOSN converts a Z-coordinate to a colour number.

The call is:	ICLR = NZPOSN (Z)	level 3
or:	int nzposn (float z);	

Additional note: If Z lies outside of the axis limits and Z is smaller than the lower limit, NZPOSN returns the value 0 and the routine returns the value 255 if Z is greater than the upper limit.

C O L R A Y

The routine COLRAY converts an array of Z-coordinates to colour values.

The call is:	CALL COLRAY (ZRAY, NRAY, N)	level 3
or:	void colray (float *zray, int *nray, int n);	

ZRAY	is an array of Z-coordinates.
NRAY	is an array of colour numbers calculated by COLRAY.
N	is the number of coordinates.

11.8 Example

```
      PROGRAM EX11_1
      PARAMETER (N=100)
      DIMENSION ZMAT(N,N)

      FPI=3.1415927/180.
      STEP=360./(N-1)
      DO I=1,N
        X=(I-1.)*STEP
        DO J=1,N
          Y=(J-1.)*STEP
          ZMAT(I,J)=2*SIN(X*FPI)*SIN(Y*FPI)
        END DO
      END DO

      CALL METAFL('POST')
      CALL DISINI
      CALL PAGERA
      CALL PSFONT('Times-Roman')

      CALL TITLIN('3-D Colour Plot of the Function',1)
      CALL TITLIN('F(X,Y) = 2 * SIN(X) * SIN(Y)',3)
      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')
      CALL NAME('Z-axis','Z')

      CALL INTAX
      CALL AUTRES(N,N)
      CALL AXSPOS(300,1850)
      CALL AX3LEN(2200,1400,1400)

      CALL GRAF3(0.,360.,0.,90.,0.,360.,0.,90.,
     *                           -2.,2.,-2.,1.)
      CALL CRVMAT(ZMAT,N,N,1,1)
      CALL HEIGHT(50)
      CALL PSFONT('Palatino-BoldItalic')
      CALL TITLE
      CALL DISFIN
      END
```

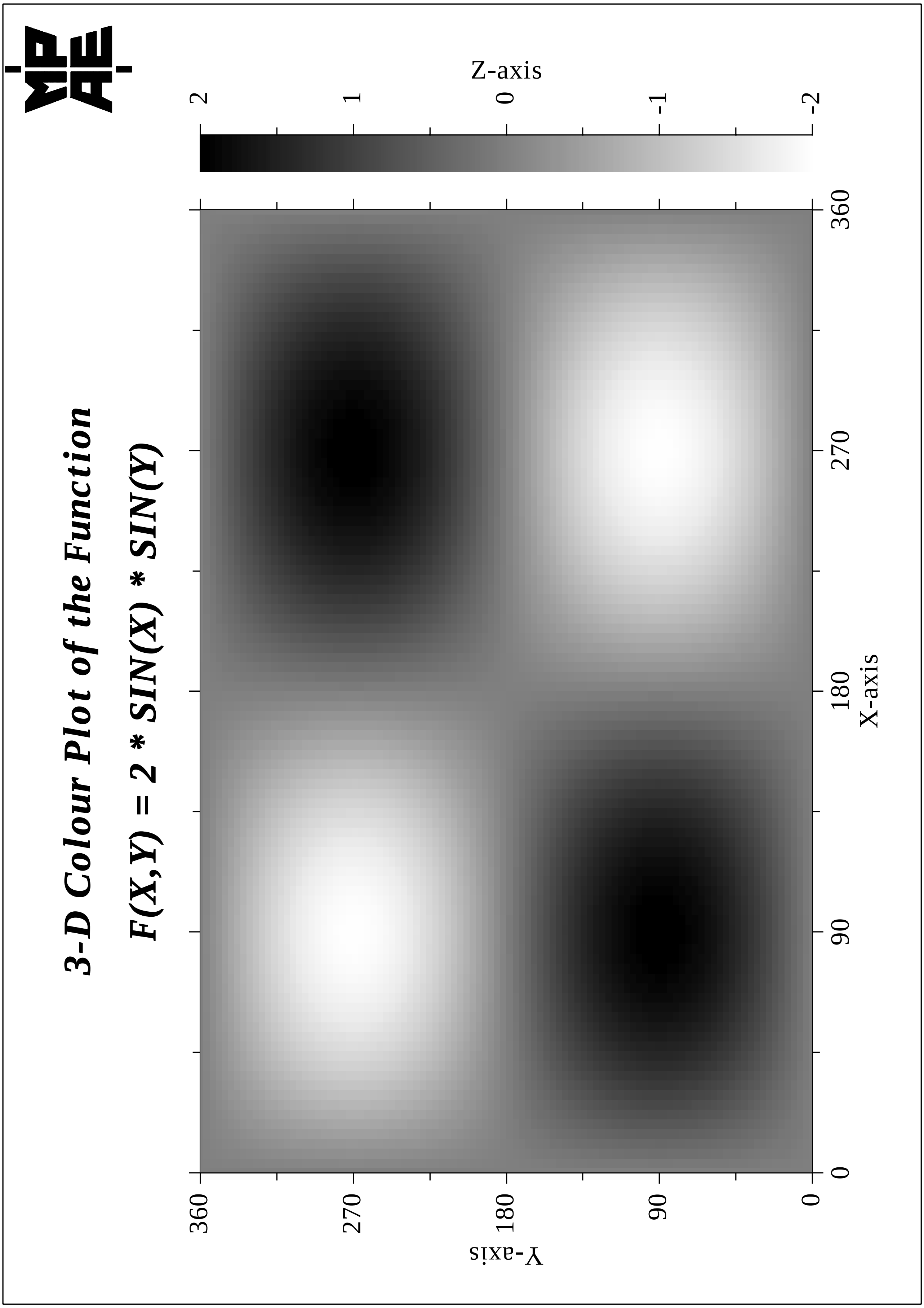


Figure 11.1: 3-D Colour Plot

Chapter 12

3-D Graphics

This chapter describes routines for 3-D coordinate systems. Axis systems, curves and surfaces can be drawn from various angular perspectives. All 2-D plotting routines can be used in a 3-D axis system.

12.1 Introduction

Three-dimensional objects must be plotted in a 3-D box which is projected onto a two-dimensional region on the page. The 3-D box contains an X-, Y- and Z-axis with the Z-axis lying in the vertical direction. The units of the axes are called absolute 3-D coordinates. They are abstract and have no relation to any physical units. An axis system is used to scale the 3-D box with user coordinates and to plot axis ticks, labels and names.

The position and size of a projected 3-D box depends upon the position and size of the region onto which the box is projected, and the point from which the box is viewed. The region is determined by the routines AXSPOS and AXSLEN where the centre of the 3-D box will be projected onto the centre of the region.

A X I S 3 D

The routine AXIS3D defines the lengths of the 3-D box. For the lengths, any positive values can be specified; DISLIN uses only the ratio of the values to calculate the axis lengths.

The call is:	CALL AXIS3D (X3AXIS, Y3AXIS, Z3AXIS)	level 1, 2, 3
or:	void axis3d (float x3axis, float y3axis, float z3axis);	

X3AXIS	is the length of the X-axis in absolute 3-D coordinates (> 0).
Y3AXIS	is the length of the Y-axis in absolute 3-D coordinates (> 0).
Z3AXIS	is the length of the Z-axis in absolute 3-D coordinates (> 0). Default: (2., 2., 2.)

Additional note: The lower left corner of the 3-D box is the point (-X3AXIS/2, -Y3AXIS/2, -Z3AXIS/2); the upper right corner is the point (X3AXIS/2, Y3AXIS/2, Z3AXIS/2). The centre point is (0., 0., 0.).

The following figure shows the default 3-D box:

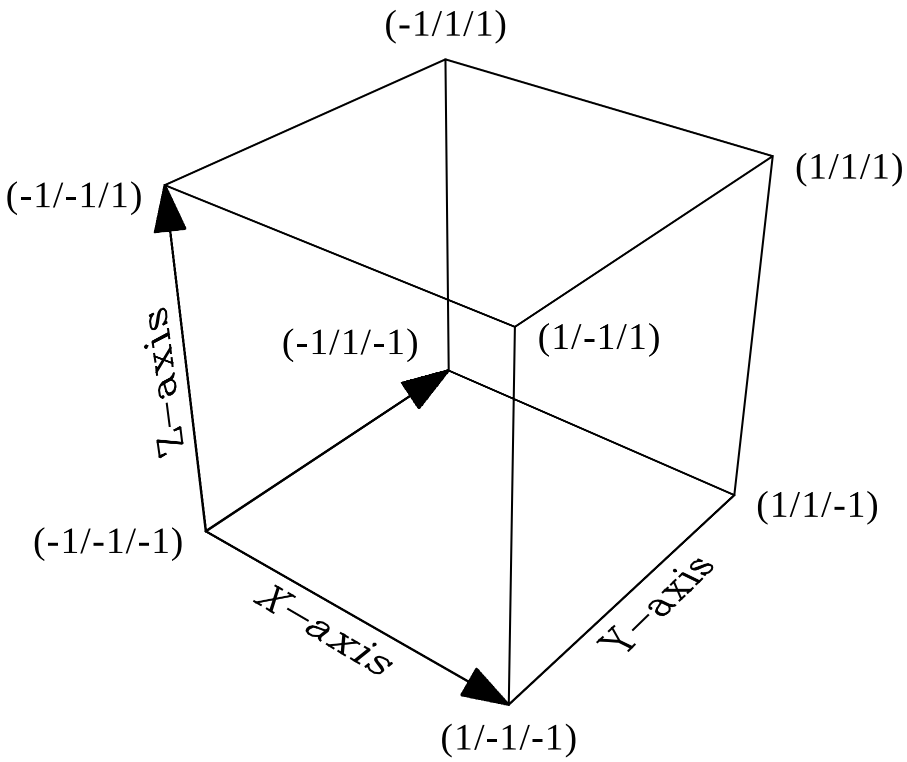


Figure 12.1: Default 3-D Box

12.2 Defining View Properties

The following routines define view properties such as viewpoint, target point, view angle and view orientation.

V I E W 3 D

The routine VIEW3D defines the viewpoint. The viewpoint is a point in space from which the 3-D box is observed and determines how objects are projected onto a 2-D plane. Objects will appear small if the viewpoint is far away. As the viewpoint is moved closer to the 3-D box, the objects will appear larger.

The call is:	CALL VIEW3D (XVU, YVU, ZVU, CVU)	level 1, 2, 3
or:	void view3d (float xvu, float yvu, float zvu, char *cvu);	

XVU, YVU, ZVU	define the position of the viewpoint. If CVU = 'ABS', the parameters must contain absolute 3-D coordinates, if CVU = 'USER', they must contain user coordinates and if CVU = 'ANGLE', the viewpoint must be specified by two angles and a radius. In the latter case, XVU is a rotation angle, YVU is the angle between the line from the viewpoint to the centre of the 3-D box and the horizontal direction and ZVU is the distance of the viewpoint from the centre of the 3-D box. XVU and YVU must be specified in degrees and ZVU in absolute 3-D coordinates.
CVU	is a character string defining the meaning of XVU, YVU and ZVU. Default: (2*X3AXIS, -2.5*Y3AXIS, 2*Z3AXIS, 'ABS').
Additional note:	The viewpoint must be placed outside the 3-D box. If the point lies inside, DISLIN will print a warning and use the default viewpoint.

V F O C 3 D

The routine VFOC3D defines the focus point. It specifies the location in the 3-D box that the camera points to.

The call is:	CALL VFOC3D (XFOC, YFOC, ZFOC, CVU)	level 1, 2, 3
or:	void vfoc3d (float xfoc, float yfoc, float zfoc, char *cvu);	

XFOC, YFOC, ZFOC	define the position of the focus point. If CVU = 'ABS', the parameters must contain absolute 3-D coordinates, if CVU = 'USER', they must contain user coordinates.
CVU	is a character string defining the meaning of XFOC, YFOC and ZFOC.

Default: (0., 0., 0., 'ABS').

V U P 3 D

The rotation of the camera around the viewing axis is defined by an angle.

The call is:	CALL VUP3D (ANG)	level 1, 2, 3
or:	void vup3d (float ang);	

ANG	defines the rotation angle in degrees. The camera is rotated in a clockwise direction.

Default: ANG = 0.

V A N G 3 D

VANG3D defines the view angle. It specifies the field of view of the lens.

The call is:	CALL VANG3D (ANG)	level 1, 2, 3
or:	void vang3d (float ang);	

ANG	defines the view angle in degrees.

Default: ANG = 28.

12.3 Plotting Axis Systems

G R A F 3 D

The routine GRAF3D plots a three-dimensional axis system. This routine must be called before any objects can be plotted in the 3-D box.

The call is:	CALL GRAF3D (XA, XE, XOR, XSTEP, YA, YE, YOR, YSTEP, ZA, ZE, ZOR, ZSTEP)	level 1
or:	void graf3d (float xa, float xe, float xor, float xstep, float ya, float ye, float yor, float ystep, float za, float ze, float zor, float zstep);	

XA, XE	are the lower and upper limits of the X-axis.
XOR, XSTEP	are the first X-axis label and the step between labels.
YA, YE	are the lower and upper limits of the Y-axis.
YOR, YSTEP	are the first Z-axis label and the step between labels.
ZA, ZE	are the lower and upper limits of the Z-axis.

ZOR, ZSTEP are the first Z-axis label and the step between labels.

Additional notes:

- GRAF3D must be called from level 1 and sets the level to 3.
- By default, the labels and axis titles on the 3-D box are also plotted with a perspective projection. This default mode does not allow the plotting of hardware fonts and switches automatically to the DISLIN vector font COMPLX if a hardware font is enabled. Other modes for plotting labels and axis titles that allow using of hardware fonts can be defined with the routine LABL3D.
- In default mode, GRAF3D suppresses the plotting of certain start labels to avoid overplotting of labels. This option can be disabled with the statement CALL FLAB3D.
- The user is referred to the notes on GRAF in chapter 4.

12.4 Plotting a Border around the 3-D Box

B O X 3 D

The routine BOX3D plots a border around the 3-D box.

The call is: CALL BOX3D level 3

or: void box3d ();

12.5 Plotting Grids

G R I D 3 D

The routine GRID3D plots a grid in the 3-D box.

The call is: CALL GRID3D (IGRID, JGRID, COPT) level 3

or: void grid3d (int igrid, int jgrid, char *copt);

IGRID is the number of grid lines between labels in the X-direction (or Y-direction for the YZ-plane).

JGRID is the number of grid lines between labels in the Z-direction (or Y-direction for the XY-plane).

COPT is a character string which defines where the grid will be plotted.

= 'ALL' will plot a grid in the XY-, XZ- and YZ-plane.

= 'BACK' will plot a grid in the XZ- and YZ-plane.

= 'BOTTOM' will plot a grid in the XY-plane.

12.6 Plotting Curves

C U R V 3 D

The routine CURV3D is similar to CURVE and connects data points with lines or marks them with symbols.

The call is: CALL CURV3D (XRAY, YRAY, ZRAY, N) level 3

or: void curv3d (float *xray, float *yray, float *zray, int n);

XRAY	is an array containing the X-coordinates of data points.
YRAY	is an array containing the Y-coordinates of data points.
ZRAY	is an array containing the Z-coordinates of data points.
N	is the number of data points.

Additional note: Data points will be interpolated linearly. The user is referred to the notes on CURVE in chapter 5.

12.7 Plotting a Surface Grid from a Function

S U R F U N

The routine SURFUN plots a surface grid of the three-dimensional function Z = F(X,Y).

The call is: CALL SURFUN (ZFUN, IXP, XDEL, IYP, YDEL) level 3

or: void surfun ((float) (*zfun()), int ixp, float xdel, int iyp, float ydel);

ZFUN	is the name of a FUNCTION subroutine that returns the function value for a given X- and Y-coordinate. ZFUN must be declared EXTERNAL in the calling program.
XDEL, YDEL	are the distances between grid lines in user coordinates. XDEL and YDEL determine the density of the surface plotted by SURFUN.
IXP, IYP	are the number of points between grid lines interpolated by SURFUN (≥ 0). If IXP = 0, surface lines in the X-direction will be suppressed; if IYP = 0, surface lines in the Y-direction will be suppressed.

12.8 Plotting a Surface Grid from a Matrix

The routines SURMAT and SURFCE plot surface grids of the three-dimensional function Z = F(X,Y) where the function values are given in the form of a matrix. SURMAT assumes that the function values correspond to a linear grid in the XY-plane while SURFCE can be used with non linear grids.

The calls are: CALL SURMAT (ZMAT, IXDIM, IYDIM, IXPTS, IYPTS) level 3

CALL SURFCE (XRAY, IXDIM, YRAY, IYDIM, ZMAT) level 3

or: void surmat (float *zmat, int ixdim, int iydim, int ixpts, int iypts);

void surfce (float *xray, int ixdim, float *yray, int iydim, float *zmat);

XRAY, YRAY	are arrays containing the X- and Y-user coordinates.
ZMAT	is a matrix with the dimension (IXDIM, IYDIM) containing the function values.
IXDIM, IYDIM	are the dimensions of ZMAT, XRAY and YRAY (≥ 2).
IXPTS, IYPTS	are the number of points interpolated between grid lines in the X- and Y-direction. These parameters determine the density of surfaces plotted by SURMAT. For positive values, the surface will be interpolated linearly. For a negative value, the absolute value will be used as a step for plotted surface lines. If IXPTS = 0, surface lines in the Y-direction will be suppressed; if IYPTS = 0, surface lines in the X-direction will be suppressed.

Additional notes: - The routines SURMAT and SURFCE suppress automatically hidden lines. The suppression can be disabled with the statement CALL NOHIDE.

- SURMAT and SURFCE use a horizon line algorithm for suppressing hidden lines. This algorithm is efficient but may fail for some complex data structures. An alternate method for suppressing hidden lines can be used with the routine SURSHD if only mesh lines are enabled with the statement CALL SURMSH ('ONLY').
- Surfaces can be protected from overwriting with CALL SHLSUR if the hidden-line algorithm is not disabled.
- The limits of the base grid are determined by the parameters in GRAF3D or can be altered with SURSZE (XA, XE, YA, YE). If XA, XE, YA and YE are the axis limits in GRAF3D or defined with SURSZE, the connection of grid points and matrix elements can be described by the formula:

  ZMAT(I,J) = F(X,Y)   where

  X = XA + (I - 1) * (XE - XA) / (IXDIM - 1)   I = 1,..,IXDIM   and
  Y = YA + (J - 1) * (YE - YA) / (IYDIM - 1)   J = 1,..,IYDIM.
- SURVIS (CVIS) determines the visible part of a surface where CVIS can have the values 'TOP', 'BOTTOM' and 'BOTH'. The default value is 'BOTH'.
- The statement CALL SURCLR (ICTOP, ICBOT) defines the colours of the upper and lower side of a surface where ICTOP and ICBOT contain colour values.

12.9 Plotting a Shaded Surface from a Matrix

S U R S H D

The routine SURSHD plots a shaded surface from a matrix where colour values are calculated from the Z-scaling in the routine GRAF3D or from the parameters of the routine ZSCALE.

The call is:	CALL SURSHD (XRAY, IXDIM, YRAY, IYDIM, ZMAT)	level 3
or:	void surshd (float *xray, int ixdim, float *yray, int iydim, float *zmat);	

XRAY, YRAY	are arrays containing the X- and Y-user coordinates.
ZMAT	is a matrix with the dimension (IXDIM, IYDIM) containing the function values.
IXDIM, IYDIM	are the dimensions of ZMAT, XRAY and YRAY (≥ 2).

Additional notes:

- The statement CALL ZSCALE (ZMIN, ZMAX) defines an alternate Z-scaling that will be used to calculate colour values in SURSHD. Normally, the Z-scaling in GRAF3D is used. For logarithmic scaling of the Z-axis, ZMIN and ZMAX must be exponents of base 10.
- A flat shading or a smooth shading can be selected with the routine SHDMOD. The default is flat shading and a depth sort is used for hidden-surface elimination. If smooth shading is selected, a Z-buffer is used for hidden-surface elimination. For that case, a raster format is needed for the graphics output format (for example METAFL ('XWIN') or METAFL ('TIFF')).
- Additional grid lines can be enabled with the routine SURMSH. SURSHD can generate only mesh lines if the keyword 'ONLY' is used in SURMSH.
- Lighting can be enabled for SURSHD with the routine LIGHT.

12.10 Plotting a Shaded Surface from a Parametric Function

S U R F C P

A three-dimensional parametric function is a function of the form (x(t,u), y(t,u), z(t,u)) where tmin $\leq$ t $\leq$ tmax and umin $\leq$ u $\leq$ umax. The routine SURFCP plots a shaded surface from a parametric function. The colours of the surface are calculated from the Z-scaling in the routine GRAF3D or from the parameters of the routine ZSCALE.

The call is:	CALL SURFCP (ZFUN, TMIN, TMAX, TSTEP, UMIN, UMAX, USTEP) level 3
or:	void surfcp ((float) (*zfun()), float tmin, float tmax, float tstep, float umin, float umax, float ustep);

ZFUN	is the name of a FUNCTION subroutine with the formal parameters X, Y and IOPT. If IOPT = 1, ZFUN should return the X-coordinate of the parametric function, if IOPT = 2, ZFUN should return the Y-coordinate and if IOPT = 3, ZFUN should return the Z-coordinate.
TMIN, TMAX, TSTEP	define the range and step size of the first parameter.
UMIN, UMAX, USTEP	define the range and step size of the second parameter.

Additional notes:

- SURFCP can plot a flat surface or a smooth surface defined by the routine SHDMOD. For a flat surface, a depth sort is used for hidden-surface elimination. For a smooth surface, a Z-buffer is used for hidden-surface elimination. In the latter case, a raster format is needed for the graphics output format (for example METAFL ('XWIN') or METAFL ('TIFF')).
- Lighting can be enabled for SURFCP with the routine LIGHT.
- Additional grid lines can be enabled with the routine SURMSH.

12.11 Plotting a Shaded Surface from Triangulated Data

S U R T R I

The routine SURTRI plots a shaded surface from triangulated data that can be calculated by the routine TRIANG from a set of irregularily distributed data points.

The call is:	CALL SURTRI (XRAY, YRAY, ZRAY, N, I1RAY, I2RAY, I3RAY, NTRI) level 3
or:	void surtri (float *xray, float *yray, float *zray, int n, int *i1ray, int *i2ray, int *i3ray, int ntri);

XRAY	is an array containing the X-coordinates of data points.
YRAY	is an array containing the Y-coordinates of data points.
ZRAY	is an array containing the Z-coordinates of data points.
N	is the number of data points.
I1RAY, I2RAY, I3RAY	is the Delaunay triangulation of the points (XRAY, YRAY) calculated by the routine TRIANG.
NTRI	is the number of triangles in I1RAY, I2RAY and I3RAY.

12.12 Plotting Isosurfaces

S U R I S O

The routine SURISO plots isosurfaces of the form f(x,y,z) = constant.

The call is:	CALL SURISO (XRAY, NX, YRAY, NY, ZRAY, NZ, WMAT, WLEV) level 3
or:	void suriso (float *xray, int nx, float *yray, int ny, float *zray, int nz, float *wmat, float wlev);

XRAY, YRAY, ZRAY	are arrays containing the X-, Y- and Z-user coordinates.
WMAT	is a matrix with the dimension (NX, NY, NZ) containing the function values.
NX, NY, NZ	are the dimensions of WMAT, XRAY, YRAY, and ZRAY (≥ 2).
WLEV	defines the level of the isosurface.

Additional notes:

- The algorithm used in SURISO is based on the Marching Cubes method. Reference: Lorensen, W.E. and Cline, H.E., Marching Cubes: a high resolution 3D surface reconstruction algorithm, Computer Graphics, Vol. 21, No. 4, pp 163-169 (Proc. of SIGGRAPH), 1987.
- SURISO can plot flat or smooth surface triangles defined by the routine SHDMOD. For smooth triangles, a Z-buffer is used for hidden-surface elimination. In that case a raster format is needed for the graphics output format.
- Lighting can be enabled for SURSIO with the routine LIGHT.
- Additional grid lines can be enabled with the routine SURMSH.

12.13 Plotting 3-D Bars

B A R S 3 D

BARS3D plots three-dimensional bars.

The call is:	CALL BARS3D (XRAY, YRAY, Z1RAY, Z2RAY, XWRAY, YWRAY, ICRAY, N) level 3
or:	void bars3d (float *xray, float *yray, float *z1ray, float *z2ray, float *xwray, float *ywray, int *icray, int n);

XRAY	is an array of user coordinates defining the position of the bars on the X-axis.
YRAY	is an array of user coordinates defining the position of the bars on the Y-axis.
Z1RAY	is an array of user coordinates containing the start points of the bars on the Z-axis.
Z2RAY	is an array of user coordinates containing the end points of the bars on the Z-axis.
XWRAY	is an array of user coordinates defining the width of the bars in X-direction.
YWRAY	is an array of user coordinates defining the width of the bars in Y-direction.
ICRAY	is an array of colour values used for the bars. The foreground colour is used for the colour value -1.
N	is the number of bars.

Additional note: Legends are supported for 3-D bar graphs. Legend entries are done for each new colour in ICRAY.

12.14 Additional Parameter Setting Routines

L A B L 3 D

The routine LABL3D modifies the appearance of labels and axis titles plotted on the 3-D box.

The call is:	CALL LABL3D (COPT)	level 1, 2, 3
or:	void labl3d (char *copt);	

COPT is a character string that can have the values 'STANDARD', 'HORIZONTAL', 'PARALLEL' and 'OTHER'. For the default mode 'STANDARD', hardware fonts cannot be used for plotting labels and axis titles. For that case, DISLIN will switch to the vector font COMPLX.

Default: COPT = 'STANDARD'.

N O H I D E

The suppression of hidden lines in the routines SURFUN, SURMAT and SURFCE can be disabled with a call to NOHIDE.

The call is:	CALL NOHIDE	level 1, 2, 3
or:	void nohide ();	

S H L S U R

The surfaces plotted by the routines SURFUN, SURMAT and SURFCE can be protected from overwriting with the routine SHLSUR.

The call is:	CALL SHLSUR	level 1, 2, 3
or:	void shlsur ();	

S U R O P T

Surface lines plotted with the routine SURFCE can be suppressed for the X- and Y-directions.

The call is:	CALL SUROPT (COPT)	level 1, 2, 3
or:	void suropt (char *copt);	

COPT is a character string that can have the values 'XISO', 'YISO' and 'BOTH'. If COPT = 'XISO', surface lines in the Y-direction will be suppressed by SURFCE. If COPT = 'YISO', surface lines in the X-direction will be suppressed.

Default: COPT = 'BOTH'.

S U R V I S

The routine SURVIS determines the visible part of the surfaces plotted by the routines SURFUN, SURMAT and SURFCE.

The call is:	CALL SURVIS (CVIS)	level 1, 2, 3
or:	void survis (char *cvis);	

CVIS is a character string that can have the values 'TOP', 'BOTTOM' and 'BOTH'.

Default: CVIS = 'BOTH'.

S U R C L R

The routine SURCLR defines the colours of the upper and lower side of surfaces plotted by the routines SURFUN, SURMAT and SURFCE.

The call is:	CALL SURCLR (ICTOP, ICBOT)	level 1, 2, 3
or:	void surclr (int ictop, int icbot);	

ICTOP, ICBOT — are colour values. The values -1 means that the current colour is used.
Default: (-1, -1).

S H D M O D

The routine SHDMOD defines flat or smooth shading for the routine SURSHD. If smooth shading is selected, DISLIN uses a Z-buffer for hidden-surface elimination. This means that the graphics output format must be set to a raster format (for example: METAFL ('XWIN') or METAFL ('TIFF').

The call is:	CALL SHDMOD (COPT, 'SURFACE')	level 1, 2, 3
or:	void shdmod (char *copt, "SURFACE");	

COPT — is a character string that can have the values 'FLAT' and 'SMOOTH'. If COPT = 'SMOOTH', a raster format is needed for the output graphics format (for example METAFL ('XWIN') or METAFL ('TIFF')).
Default: COPT = 'FLAT'.

S U R M S H

The routine SURMSH can enable additional grid lines for the routines SURSHD, SURFCP and SURISO.

The call is:	CALL SURMSH (COPT)	level 1, 2, 3
or:	void surmsh (char *copt);	

COPT — is a character string that can have the values 'ON', 'OFF' and 'ONLY'. For COPT = 'ONLY', the shading of the surfaces are suppressed and only mesh lines will be displayed.
Default: COPT = 'OFF'.

M S H C L R

The routine MSHCLR sets the colour for grid lines.

The call is:	CALL MSHCLR (ICLR)	level 1, 2, 3
or:	void mshclr (int iclr);	

ICLR — is a colour value where the value -1 means that the current colour is used.
Default: ICLR = -1.

Z S C A L E

The routine ZSCALE defines an alternate Z-scaling that will be used to calculate colour values in the routines SURTRI, SURSHD, SURFCP, CONSHD and CONTRI.

The call is:	CALL ZSCALE (ZMIN, ZMAX)	level 1, 2, 3
or:	void zscale (float zmin, float zmax);	

ZMIN,ZMAX	define the range of the Z-scaling. For logarithmic scaling of the Z-axis, ZMIN and ZMAX must be exponents of base 10.

C L I P 3 D

The routine CLIP3D defines 3-D clipping in the world coordinate system or in the eye coordinate system.

The call is:	CALL CLIP3D (COPT)	level 1, 2, 3
or:	void clip3d (char *copt);	

COPT	is a character string that can have the values 'WORLD' and 'EYE'. Default: COPT = 'WORLD'.

V C L P 3 D

If 3-D clipping is done in the eye coordinate system, front and back clipping planes can be defined with the routine VCLP3D.

The call is:	CALL VCLP3D (XFRONT, XBACK)	level 1, 2, 3
or:	void vclp3d (float xfront, float xback);	

XFRONT, XBACK	are the distances from the viewpoint in absolute 3-D coordinates. A negative value means infinity. Default: (1., -1.).

12.15 Lighting

Lighting can be enabled for some shading routines such as SURSHD, SURFCP, SURTRI and SURISO where up to 8 light sources can be defined. General lighting can be turned off or on in DISLIN with the routine LIGHT while single light sources can be turned off or on with the routine LITMOD. The routine LITPOS defines the position of light sources and the routines LITOP3 and MATOP3 modify lighting and material parameters. Finally, the routine GETLIT calculates the colour value for a specified point and normal.

L I G H T

The routine LIGHT enables lighting for shading routines such as SURSHD, SURFCP and SURISO.

The call is:	CALL LIGHT (CMODE)	level 1, 2, 3
or:	void light (char *cmode);	

CMODE	is a character string that can have the values ’ON’ and ’OFF’. Default: CMODE = ’OFF’.

L I T M O D

Up to 8 light sources can be defined in DISLIN. The routine LITMOD enables or disables single light sources.

The call is:	CALL LITMOD (ID, CMODE)	level 1, 2, 3
or:	void litmod (int id, char *cmode);	

ID	is the ID of the light source in the range 1 to 8.
CMODE	is a character string that can have the values ’ON’ and ’OFF’. The default values are CMODE = ’ON’ for light source 1 and CMODE = ’OFF’ for the other light sources.

L I T P O S

The routine LITPOS defines the position of light sources.

The call is:	CALL LITPOS (ID, XP, YP, ZP, COPT)	level 1, 2, 3
or:	void litpos (int id, float xp, float yp, float zp, char *copt);	

ID	is the ID of the light source in the range 1 to 8.
XP, YP, ZP	define the position of the light source. If COPT = ’ABS’, the parameters must contain absolute 3-D coordinates, if COPT = ’USER’, they must contain user coordinates and if COPT = ’ANGLE’, the position must be specified by two angles and a radius (see VIEW3D).
COPT	is a character string defining the meaning of XP, YP and ZP. Default: (2*X3AXIS, -2.5*Y3AXIS, 2*Z3AXIS, ’ABS’).

L I T O P T

The routine LITOPT modifies the constant, linear and quadratic attentuation factors of light sources.

The call is:	CALL LITOPT (ID, XVAL, COPT)	level 1, 2, 3
or:	void litopt (int id, float xval, char *copt);	

ID	is the ID of the light source in the range 1 to 8.
XVAL	is a floatingpoint number containing the new lighting parameter.
COPT	is a character string that can have the values ’CONSTANT’, ’LINEAR’ and ’QUADRATIC’.

Defaults: (1., ’CONSTANT’), (0., ’LINEAR’), (0., ’QUADRATIC’).

L I T O P 3

The routine LITOP3 modifies the ambient, diffuse and specular intensities of light sources.

The call is:	CALL LITOP3 (ID, XR, XG, XB, COPT)	level 1, 2, 3
or:	void litop3 (int id, float xr, float xg, float xb, char *copt);	

ID	is the ID of the light source in the range 1 to 8.
XR, XG, XB	are floatingpoint numbers in the range 0 to 1 for R, G and B.
COPT	is a character string that can have the values ’AMBIENT’, ’DIFFUSE’ and ’SPECULAR’.

Defaults: (0., ’AMBIENT’), (1., ’DIFFUSE’), (1., ’SPECULAR’).

M A T O P T

The routine MATOPT modifies material parameters.

The call is:	CALL MATOPT (XVAL, COPT)	level 1, 2, 3
or:	void matopt (float xval, char *copt);	

XVAL	is a floatingpoint number containing the new material parameter.
COPT	is a character string that can have the value ’EXPONENT’.

Default: (0., ’EXPONENT’).

M A T O P 3

The routine MATOP3 modifies material parameters such as ambient, diffuse and specular colour.

The call is:	CALL MATOP3 (XR, XG, XB, COPT)	level 1, 2, 3
or:	void matop3 (float xr, float xg, float xb, char *copt);	

XR, XG, XB	are floatingpoint numbers in the range 0 to 1 containing the new material parameters for R, G and B.
COPT	is a character string that can have the values ’AMBIENT’, ’DIFFUSE’ and ’SPECULAR’.

Defaults: (0.2, ’AMBIENT’), (0.8, ’DIFFUSE’),
(0., ’SPECULAR’).

G E T L I T

The routine GETLIT calculates colour values for given points and their normals specified in absolute coordinates.

The call is:	CALL GETLIT (XP, YP, ZP, XN, YN, ZN, ICLR)	level 1, 2, 3
or:	int getlit (float xp, float yp, float zp, float xn, float yn, float zn);	

XP, YP, ZP	are the X-, Y-and Z-coordinates of the point.
XN, YN, ZN	are the X-, Y- and Z-coordinates of the point normal.
ICLR	is the returned colour value. ICLR contains an explicit RGB value.

12.16 Surfaces from Randomly Distributed Points

The routine SURMAT assumes that function values are in the form of a matrix and correspond to a linear grid in the XY-plane. If three-dimensional data points are given as randomly distributed points of the form X(N), Y(N) and Z(N), the routine GETMAT can be used to calculate a function matrix.

G E T M A T

The routine GETMAT calculates a function matrix for randomly distributed data points.

The call is:	CALL GETMAT (XRAY, YRAY, ZRAY, N, ZMAT, NX, NY, ZVAL, IMAT, WMAT) level 2,3
or:	void getmat (float *xray, float *yray, float *zray, int n, float *zmat, int nx, int ny, float zval, int *imat, float *wmat);

XRAY, YRAY, ZRAY	are arrays containing the randomly distributed data points.
N	is the number of points.
ZMAT	is the function matrix of the dimension (NX, NY) calculated by GETMAT. The matrix elements correspond to a linear grid in the XY-plane whose limits are determined by the scaling values in GRAF3D or SURSZE.
NX, NY	are the dimensions of ZMAT, IMAT and WMAT.
ZVAL	will be used as a value for matrix elements when no data points can be found in an area around the corresponding grid points. In general, the start scaling of the Z-axis will be used for ZVAL.
IMAT	is a working matrix of the dimension (NX, NY). After a call to GETMAT, IMAT(I, J) contains the number of random data points found in an area around the grid points. The value -1 means that a random data value lies at a grid point.
WMAT	is a working matrix of the dimension (NX, NY).

The value ZMAT(J, K) of the corresponding grid point (J, K) is calculated by the formula:

$$ZMAT_{j,k} = \frac{\sum_{i=1}^{n} \frac{1}{D_i^w} Z_i}{\sum_{i=1}^{n} \frac{1}{D_i^w}}$$

where:	j, k	are indices from 1 to NX and 1 to NY, respectively.
	D_i	is the distance of the grid point (i, k) from the point P_i.
	w	is a weighting number (Default: 2.0).
	n	is the number of data points lying in the area around the grid point (j, k).

If P_i is a data point, the routine GETMAT finds the grid rectangle in the XY-plane in which the point lies. By default, P_i affects all grid points which lie up to 2 grid lines from P_i. A problem can arise when creating a large matrix from sparse data points because certain grid points may not lie near the actual random data points. Figure 12.2 shows the results of GETMAT using different values of IX and IY.

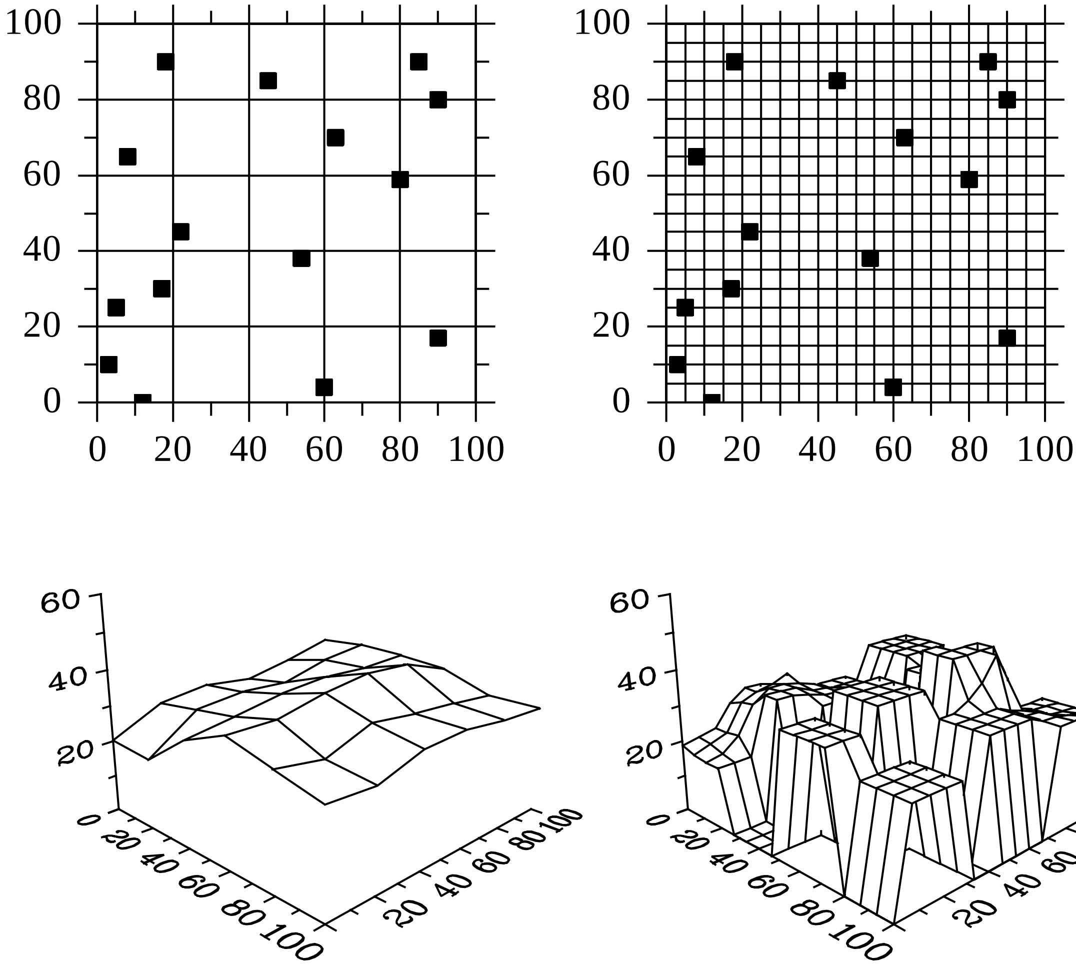


Figure 12.2: Results of GETMAT

An simple method to smooth surfaces from sparse data points is to enlarge the region around the randomly distributed data points where grid points are searched. This can be done using the routine MDFMAT.

MDFMAT

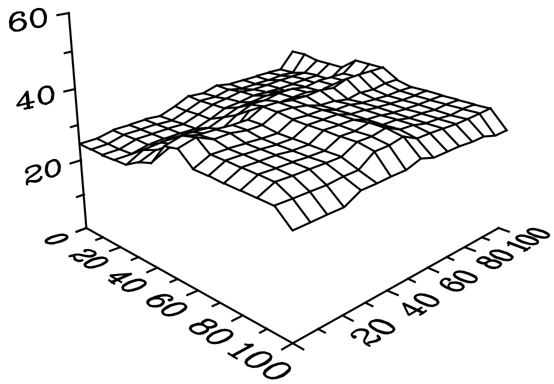
The routine MDFMAT modifies the algorithm in GETMAT.

The call is:	CALL MDFMAT (IX, IY, W)	level 1, 2, 3
or:	void mdfmat (int ix, int iy, float w);	

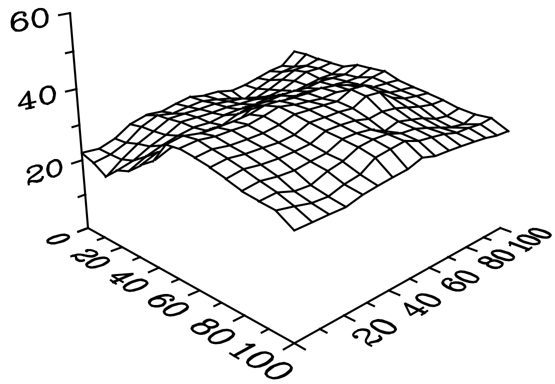
IX, IY	are the number of grid lines in the X- and Y-direction which determine the size of the region around data points.
W	is a weighting number.

Default: (2, 2, 2.0).

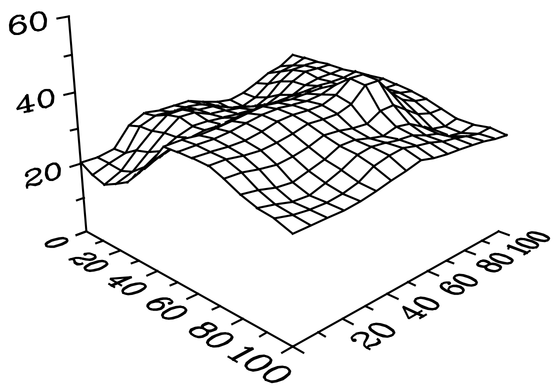
The following figure shows modifications of the above example:



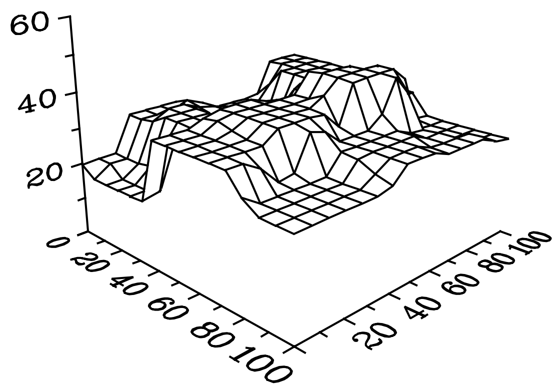
MDFMAT (5, 5, 0.1)



MDFMAT (5, 5, 1.0)



MDFMAT (5, 5, 2.0)



MDFMAT (5, 5, 15.0)

Figure 12.3: Modification of GETMAT

12.17 Projection of 2-D-Graphics into 3-D Space

Two-dimensional graphics in the XY-plane can be projected onto a plane in 3-D space. Therefore, all 2-D plot routines can be used in 3-D space.

G R F I N I

The routine GRFINI defines a plane in the 3-D box onto which all plot vectors will be projected. The plane in the 3-D box corresponds to a region in the XY-plane which is determined by AXSPOS and AXSLEN. GRFINI sets the level to 1.

The call is:	CALL GRFINI (X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3)	level 3
or:	void grfini (float x1, float y1, float z1, float x2, float y2, float z2, float x3, float y3, float z3);	

X1, Y1, Z1	are the absolute 3-D coordinates of the lower left corner of the 3-D plane.
X2, Y2, Z2	are the absolute 3-D coordinates of the lower right corner of the 3-D plane.
X3, Y3, Z3	are the absolute 3-D coordinates of the upper right corner of the 3-D plane.
Additional note:	If (NXA,NYA) is the lower left corner, NXL the width and NYL the height of the region determined by the routines AXSPOS and AXSLEN, the point (X1,Y1,Z1) corresponds to (NXA,NYA), (X2,Y2,Z2) to (NXA+NXL-1,NYA) and (X3,Y3,Z3) to (NXA+NXL-1,NYA-NYL+1), respectively.

G R F F I N

The routine GRFFIN terminates a projection into 3-D space. The level will be set back to 3.

The call is:	CALL GRFFIN	level 1, 2, 3
or:	void grffin ();	

12.18 Using the Z-Buffer

The DISLIN routines SURSHD and SURFCP use for smooth shading a 32-bit floating point Z-buffer for hidden-surface elimination. This Z-buffer can also be used by a programmer for creating shaded surfaces with elementary triangle routines.

Z B F I N I

The routine ZBFINI creates a Z-buffer. The graphics output format must be set to a raster format (for example METAFL ('XWIN') or METAFL ('TIFF')).

The call is:	CALL ZBFINI (IRET)	level 1,2,3
or:	int zbfini ();	

IRET	is the returned status (0: no errors).

Z B F F I N

The routine ZBFFIN terminates writing to a Z-buffer and frees the allocated space.

The call is:	CALL ZBFFIN	level 1,2,3
or:	void zbffin ();	

Z B F T R I

The routine ZBFTRI plots a smooth triangle where hidden-surface elimination is done with the Z-buffer.

The call is:	CALL ZBFTRI (XRAY, YRAY, ZRAY, IRAY)	level 3
or:	void zbftri (float *xray, float *yray, float *zray, int *iray);	

XRAY,YRAY,ZRAY	are the X-, Y-, and Z-coordinates of the three corners of the triangle in user coordinates.
IRAY	is an integer array containing the three colour values of the triangle corners.

Z B F L I N

The routine ZBFLIN plots a line in the current colour where the Z-buffer is used for hiddenline elimination. This routine is used by SURSHD and SURFCP for drawing surface grids.

The call is:	CALL ZBFLIN (X1, Y1, Z1, X2, Y2, Z2)	level 3
or:	void zbflin (float x1, float y1, float z1, float x2, float y2, float z2);	

X1, Y1, Z1	are the user coordinates of the start point.
X2, Y2, Z2	are the user coordinates of the end point.

12.19 Elementary Plot Routines

S T R T 3 D

The routine STRT3D moves the pen to a three-dimensional point.

The call is:	CALL STRT3D (X, Y, Z)	level 3
or:	void strt3d (float x, float y, float z);	

X, Y, Z	are the absolute 3-D coordinates of the point.

C O N N 3 D

The routine CONN3D plots a line from the current pen position to a three-dimensional point. The line will be cut off at the sides of the 3-D box. Different line styles can be used.

The call is:	CALL CONN3D (X, Y, Z)	level 3
or:	void conn3d (float x, float y, float z);	

X, Y, Z	are the absolute 3-D coordinates of the point.

V E C T R 3

The routine VECTR3 plots a vector in the 3-D box.

The call is:	CALL VECTR3 (X1, Y1 ,Z1, X2, Y2, Z2, IVEC)	level 3
or:	void vectr3 (float x1, float y1, float z1, float x2, float y2, float z2, int ivec);	

X1, Y1, Z1	are the absolute 3-D coordinates of the start point.
X2, Y2, Z2	are the absolute 3-D coordinates of the end point.
IVEC	defines the arrow head (see VECTOR).

S P H E 3 D

The routine SPHE3D plots a sphere.

The call is:	CALL SPHE3D (XM, YM ,ZM, R, N, M)	level 3
or:	void sphe3d (float xm, float ym, float zm, float r, int n, int m);	

XM, YM, ZM	are the user coordinates of the center point.
R	is the radius of the sphere in user coordinates.
N, M	defines the horizontal and vertical resolution of the sphere.
Additional notes:	- Lighting can be enabled for SPHE3D with the routine LIGHT.
	- Additional grid lines can be enabled with the routine SURMSH.

12.20 Transformation of Coordinates

P O S 3 P T

The routine POS3PT converts three-dimensional user coordinates to absolute 3-D coordinates.

The call is:	CALL POS3PT (X, Y, Z, XP, YP, ZP)	level 3
or:	void pos3pt (float x, float y, float z, float *xp, float *yp, float *zp);	

X, Y, Z	are the user coordinates.
XP, YP, ZP	are the absolute 3-D coordinates calculated by POS3PT.

The absolute 3-D coordinates can also be calculated with the following functions:

XP = X3DPOS (X, Y, Z)
YP = Y3DPOS (X, Y, Z)
ZP = Z3DPOS (X, Y, Z)

R E L 3 P T

The routine REL3PT converts user coordinates to plot coordinates.

The call is:	CALL REL3PT (X, Y, Z, XP, YP)	level 3
or:	void rel3pt (float x, float y, float z, float *xp, float *yp);	

X, Y, Z	are the user coordinates.
XP, YP	are the plot coordinates calculated by REL3PT.

The corresponding functions are:

XP = X3DREL (X, Y, Z)
YP = Y3DREL (X, Y, Z)

A B S 3 P T

The routine ABS3PT converts absolute 3-D coordinates to plot coordinates.

The call is:	CALL ABS3PT (X, Y, Z, XP, YP)	level 3

or: void abs3pt (float x, float y, float z, float *xp, float *yp);

X, Y, Z	are the absolute 3-D coordinates.
XP, YP	are the plot coordinates calculated by ABS3PT.

The corresponding functions are:

XP = X3DABS (X, Y, Z)

YP = Y3DABS (X, Y, Z)

12.21 Examples

```
      PROGRAM EXA12_1
      DIMENSION IXP(4),IYP(4)
      DATA IXP/200,1999,1999,200/ IYP/2600,2600,801,801/
      EXTERNAL ZFUN

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL AXSPOS(200,2600)
      CALL AXSLEN(1800,1800)
      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')
      CALL NAME('Z-axis','Z')
      CALL TITLIN('Surface Plot (SURFUN)',2)
      CALL TITLIN('F(X,Y) = 2*SIN(X)*SIN(Y)',4)

      CALL GRAF3D(0.,360.,0.,90.,0.,360.,0.,90.,
     *            -3.,3.,-3.,1.)
      CALL HEIGHT(50)
      CALL TITLE
      CALL SHLSUR
      CALL SURFUN(ZFUN,1,10.,1,10.)

C     Grid in the XY plane
      CALL GRFINI(-1.,-1.,-1.,1.,-1.,-1.,1.,1.,-1.)
      CALL NOGRAF
      CALL GRAF(0.,360.,0.,90.,0.,360.,0.,90.)
      CALL DASHL
      CALL GRID(1,1)
      CALL GRFFIN
C     Grid in the YZ plane
      CALL GRFINI(-1.,-1.,-1.,-1.,1.,-1.,-1.,1.,1.)
      CALL GRAF(0.,360.,0.,90.,-3.,3.,-3.,1.)
      CALL GRID(1,1)
      CALL GRFFIN
C     Shading in the XZ plane
      CALL GRFINI(-1.,1.,-1.,1.,1.,-1.,1.,1.,1.)
      CALL SHDPAT(7)
      CALL SOLID
      CALL AREAF(IXP,IYP,4)
      CALL GRFFIN
      CALL DISFIN
      END

      FUNCTION ZFUN(X,Y)
      FPI=3.14159/180.
      ZFUN=2*SIN(X*FPI)*SIN(Y*FPI)
      END
```

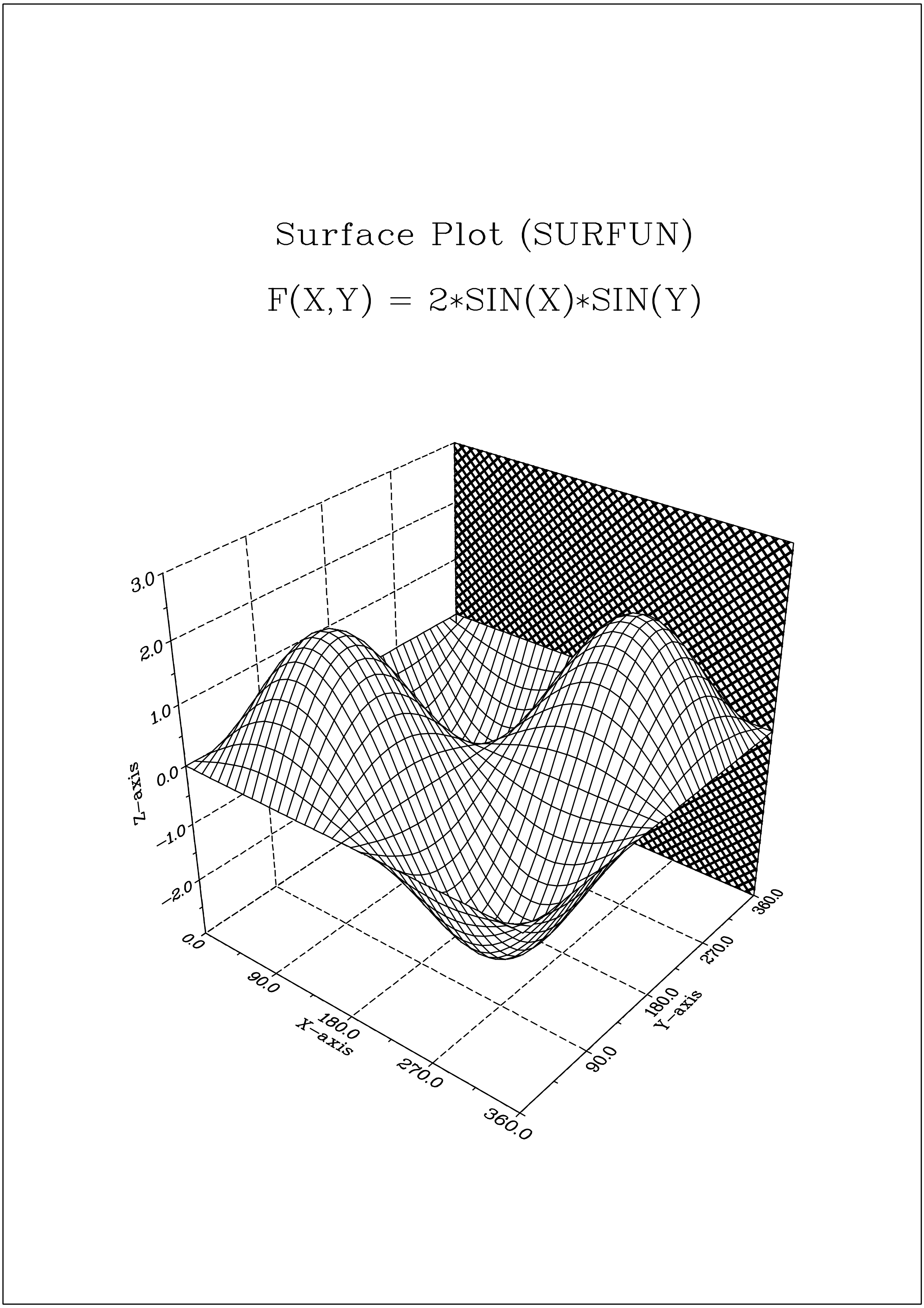


Figure 12.1: Surface Plot

```
PROGRAM EXA12_2
CHARACTER*60 CTIT1,CTIT2
EXTERNAL ZFUN

CTIT1='Surface Plot of the Parametric Function'
CTIT2='[COS(t)*(3+COS(u)), SIN(t)*(3+COS(u)), SIN(u)]'
PI=3.14159

CALL SETPAG('DA4P')
CALL METAFL('POST')
CALL DISINI
CALL HWFONT
CALL PAGERA
CALL AXSPOS(200,2400)
CALL AXSLEN(1800,1800)
CALL INTAX

CALL TITLIN(CTIT1,2)
CALL TITLIN(CTIT2,4)

CALL NAME('X-axis','X')
CALL NAME('Y-axis','Y')
CALL NAME('Z-axis','Z')

CALL VKYTIT(-300)
CALL GRAF3D(-4.,4.,-4.,1.,-4.,4.,-4.,1.,-3.,3.,-3.,1.)

CALL HEIGHT(40)
CALL TITLE

CALL SURMSH('ON')
STEP=2*PI/30.
CALL SURFCP(ZFUN,0.,2*PI,STEP,0.,2*PI,STEP)
CALL DISFIN
END

FUNCTION ZFUN(X,Y,IOPT)

IF(IOPT.EQ.1) THEN
  ZFUN=COS(X)*(3+COS(Y))
ELSE IF(IOPT.EQ.2) THEN
  ZFUN=SIN(X)*(3+COS(Y))
ELSE
  ZFUN=SIN(Y)
END IF
END
```

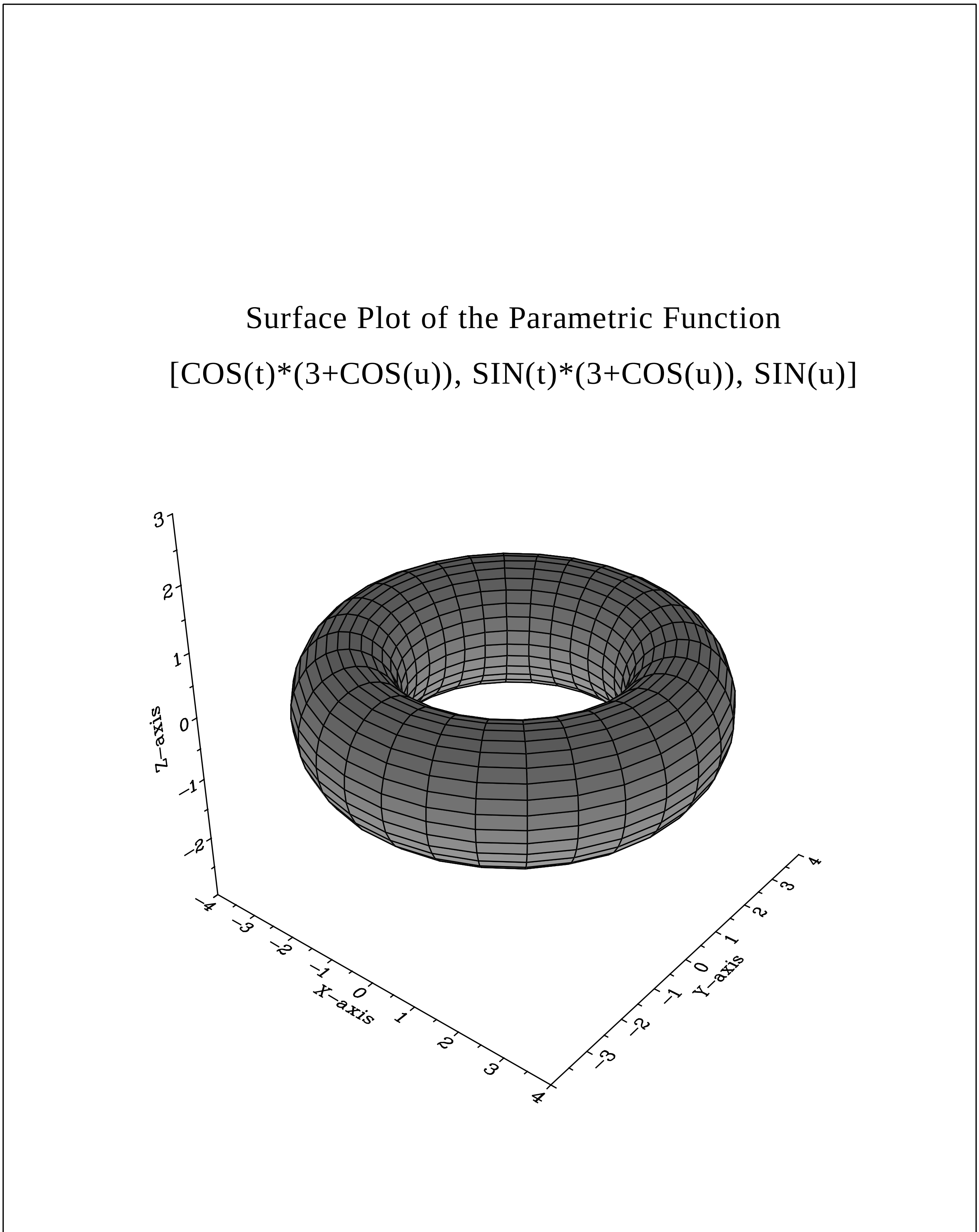


Figure 12.2: Surface Plot of a Parametric Function

```
      PROGRAM EXA12_3
      PARAMETER (N=18)
      DIMENSION XRAY(N),YRAY(N),Z1RAY(N),Z2RAY(N),XWRAY(N),
     *          YWRAY(N),ICRAY(N)
      CHARACTER*80 CBUF

      DATA XRAY/1., 3., 8., 1.5, 9., 6.3, 5.8, 2.3, 8.1, 3.5,
     *         2.2, 8.7, 9.2, 4.8, 3.4, 6.9, 7.5, 3.8/
      DATA YRAY/5., 8., 3.5, 2., 7., 1.,4.3, 7.2, 6.0, 8.5,
     *         4.1, 5.0, 7.3, 2.8, 1.6, 8.9, 9.5, 3.2/
      DATA Z1RAY/0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
     *           0., 0., 0., 0., 0., 0., 0., 0./
      DATA Z2RAY/4.,5.,3.,2.,3.5,4.5,2.,1.6,3.8,4.7,
     *           2.1, 3.5, 1.9, 4.2, 4.9, 2.8
      DATA ICRAY/30, 30, 30, 30, 30, 30, 100, 100, 100, 100,
     *           100, 100, 170, 170, 170, 170, 170, 170/

      DO I=1,N
        XWRAY(I)=0.5
        YWRAY(I)=0.5
      END DO

      CALL SETPAG('DA4P')
      CALL METAFL('PS')
      CALL DISINI
      CALL PAGERA
      CALL HWFONT
      CALL AXSPOS(200,2600)
      CALL AXSLEN(1800,1800)

      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')
      CALL NAME('Z-axis','Z')
      CALL TITLIN('3-D Bars / BARS3D',3)

      CALL LABL3D('HORI')
      CALL GRAF3D(0.,10.,0.,2.,0.,10.,0.,2.,0.,5.,0.,1.)
      CALL GRID3D(1,1,'BOTTOM')
      CALL BARS3D(XRAY,YRAY,Z1RAY,Z2RAY,XWRAY,YWRAY,ICRAY,N)

      CALL LEGINI(CBUF,3,20)
      CALL LEGTIT(' ')
      CALL LEGPOS(1300,1100)
      CALL LEGLIN(CBUF,'First',1)
      CALL LEGLIN(CBUF,'Second',2)
      CALL LEGLIN(CBUF,'Third',3)
      CALL LEGEND(CBUF,3)

      CALL HEIGHT(50)
      CALL TITLE
      CALL DISFIN
      END
```

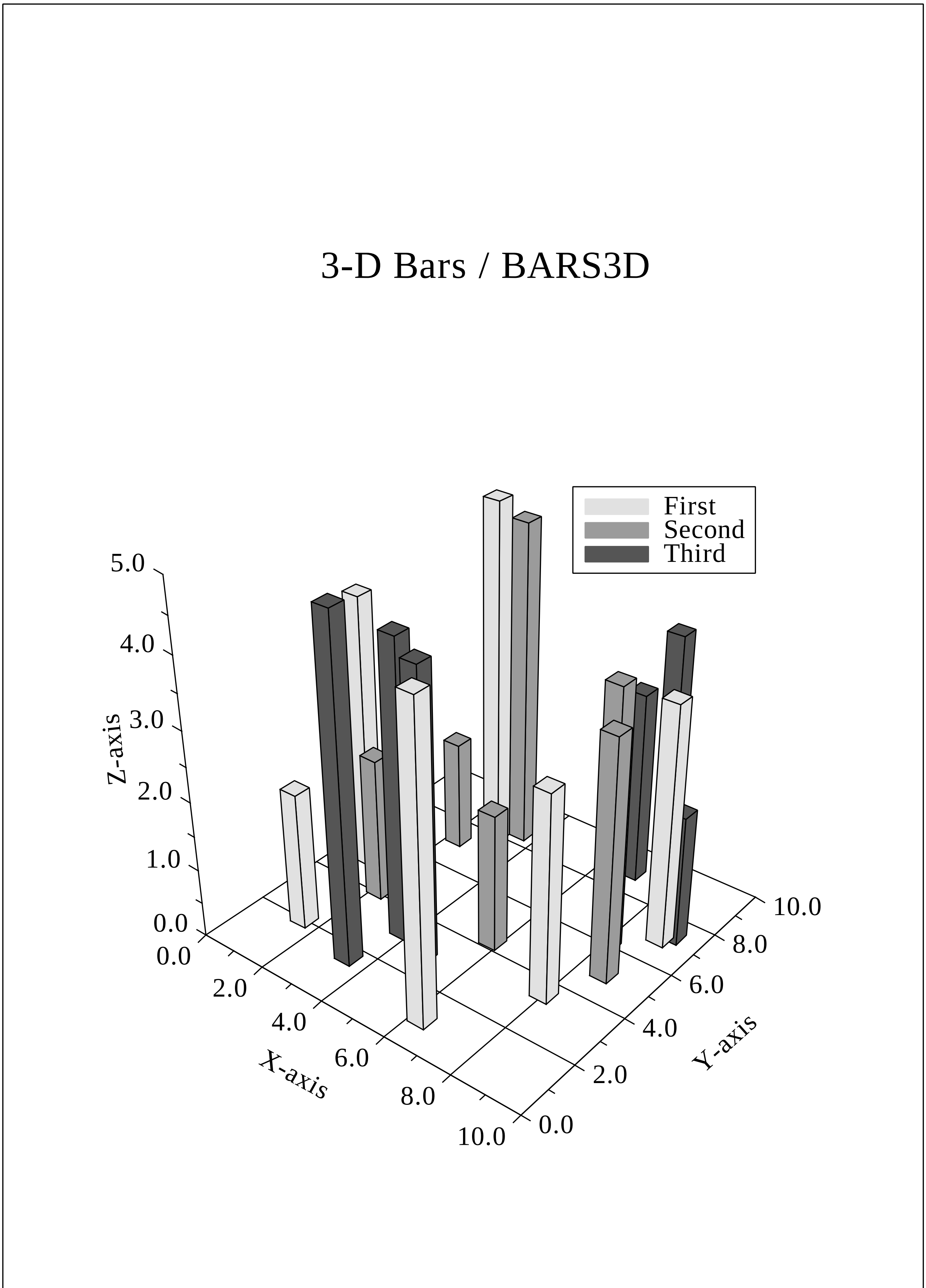


Figure 12.3: 3-D Bars / BARS3D

Chapter 13

Geographical Projections and Plotting Maps

This chapter presents different methods to project geographical coordinates onto a plane surface. Several base maps are stored in DISLIN for plotting maps.

13.1 Axis Systems and Secondary Axes

G R A F M P

The routine GRAFMP plots a geographical axis system.

The call is:	CALL GRAFMP (XA, XE, XOR, XSTP, YA, YE, YOR, YSTP)
or:	void grafmp (float xa, float xe, float xor, float xstp, float ya, float ye, float yor, float ystp);
XA, XE	are the lower and upper limits of the X-axis.
XOR, XSTP	are the first X-axis label and the step between labels.
YA, YE	are the lower and upper limits of the Y-axis.
YOR, YSTP	are the first Y-axis label and the step between labels.

Additional notes:

- GRAFMP must be called from level 1 and sets the level to 2.
- The axes must be linear and scaled in ascending order. In general, X-axes must be scaled between -180 and 180 degrees and Y-axes between -90 and 90 degrees.
- For elliptical projections, the plotting of an axis system will be suppressed. This will also be done for azimuthal projections with YE - YA $>$ 90.
- The statement CALL GRIDMP (I, J) overlays an axis system with a longitude and latitude grid where I and J are the number of grid lines between labels in the X- and Y-direction.

X A X M A P

The routine XAXMAP plots a secondary X-axis.

The call is:	CALL XAXMAP (A, B, OR, STEP, CSTR, NT, NY)	level 2
or:	void xaxmap (float a, float b, float or, float step, char *cstr, int nt, int ny);	

A, B	are the lower and upper limits of the X-axis.
OR, STEP	are the first label and the step between labels.
CSTR	is a character string containing the axis name.
NT	indicates how ticks, labels and the axis name are plotted. If NT = 0, they are plotted in a clockwise direction. If NT = 1, they are plotted in a counter-clockwise direction.
NY	defines the horizontal position of the X-axis. A secondary axis must be located inside the axis system.

Y A X M A P

The routine YAXMAP plots a secondary Y-axis.

The call is:	CALL YAXMAP (A, B, OR, STEP, CSTR, NT, NX)	level 2
or:	void yaxmap (float a, float b, float or, float step, char *cstr, int nt, int nx);	

A, B	are the lower and upper limits of the Y-axis.
OR, STEP	are the first label and the step between labels.
CSTR	is a character string containing the axis name.
NT	indicates how ticks, labels and the axis name are plotted. If NT = 0, they are plotted in a clockwise direction. If NT = 1, they are plotted in a counter-clockwise direction.
NX	defines the vertical position of the Y-axis. A secondary axis must be located inside the axis system.

13.2 Defining the Projection

Since a globe cannot be unfolded into a plane surface, many different methods have been developed to represent a globe on a plane surface. In cartography, there are 4 basic methods differentiated by attributes such as equal distance, area and angle.
The 4 basic methods are:

a) Cylindrical Projections

The surface of the globe is projected onto a cylinder which can be unfolded into a plane surface and touches the globe at the equator. The latitudes and longitudes of the globe are projected as straight lines.

b) Conical Projections

The surface of the globe is projected onto a cone which can also be unfolded into a plane surface. The cone touches or intersects the globe at two latitudes. The longitudes are projected as straight lines intersecting at the top of the cone and the latitudes are projected as concentric circles around the top of the cone.

c) Azimuthal Projections

For azimuthal projections, a hemisphere is projected onto a plane which touches the hemisphere at a point called the map pole. The longitudes and latitudes are projected as circles.

d) Elliptical Projections

Elliptical projections project the entire surface of the globe onto an elliptical region.

P R O J C T

The routine PROJCT selects the geographical projection.

The call is:	CALL PROJCT (CTYPE)	level 1
or:	void projct (char *ctype);	

CTYPE	is a character string defining the projection.
= 'CYLI'	defines a cylindrical equidistant projection.
= 'MERC'	selects the Mercator projection.
= 'EQUA'	defines a cylindrical equal-area projection.
= 'HAMM'	defines the elliptical projection of Hammer.
= 'AITO'	defines the elliptical projection of Aitoff.
= 'WINK'	defines the elliptical projection of Winkel.
= 'SANS'	defines the elliptical Mercator-Sanson projection.
= 'CONI'	defines a conical equidistant projection.
= 'ALBE'	defines a conical equal-area projection (Albers).
= 'CONF'	defines a conical conformal projection.
= 'AZIM'	defines an azimuthal equidistant projection.
= 'LAMB'	defines an azimuthal equal-area projection.
= 'STER'	defines an azimuthal stereographic projection.
= 'ORTH'	defines an azimuthal orthographic projection.
= 'MYPR'	defines an user-defined projection.

Default: CTYPE = 'CYLI'.

Additional notes: - For cylindrical equidistant projections, the scaling parameters in GRAFMP must be in the range:

$$-540 \le XA \le XE \le 540$$
$$-180 \le YA \le YE \le 180$$

For Mercator projections:

$$-540 \le XA \le XE \le 540$$
$$-85 \le YA \le YE \le 85$$

For cylindrical equal-area projections:

$$-540 \le XA \le XE \le 540$$
$$-90 \le YA \le YE \le 90$$

For elliptical projections:

$$-180 \le XA \le XE \le 180$$
$$-90 \le YA \le YE \le 90$$

For conical projections:

$$-180 \le XA \le XE \le 180$$
$$0 \le YA \le YE \le 90 \text{ or}$$
$$-90 \le YA \le YE \le 0$$

For azimuthal projections with YE - YA > 90, the hemisphere around the map pole is projected onto a circle. Therefore, the hemisphere can be selected with the map pole. The plotting of the axis system will be suppressed.

If YE - YA $\le$ 90, the part of the globe defined by the axis scaling is projected onto a rectangle. The map pole will be set by GRAFMP to ((XA+XE)/2, (YE+YA)/2). The scaling parameters must be in the range:

$$-180 \le XA \le XE \le 180 \text{ and}$$
$$XE - XA \le 180$$
$$-90 \le YA \le YE \le 90$$

- For all projections except the default projection, longitude and latitude coordinates will be projected with the same scaling factor for the X- and Y-axis. The scaling factor is determined by the scaling of the Y-axis while the scaling of the X-axis is used to centre the map. The longitude (XA+XE)/2 always lies in the centre of the axis system.
- Geographical projections can only be used for routines described in this chapter or routines that plot contours.

13.3 Plotting Maps

W O R L D

The routine WORLD plots coastlines and lakes.

The call is:	CALL WORLD	level 2
or:	void world ();	
Additional note:	The routine WORLD supports also some external map coordinates (see MAPBAS).	

S H D M A P

The routine SHDMAP plots shaded continents.

The call is:	CALL SHDMAP (CMAP)	level 2
or:	void shdmap (char *cmap);	

CMAP	is a character string defining the continent.
= ’AFRI’	means Africa.
= ’ANTA’	means the Antarctic.
= ’AUST’	means Australia.
= ’EURA’	means Europe and Asia.
= ’NORT’	means North America.
= ’SOUT’	means South America.

= 'LAKE'	means lakes.
= 'ALL'	means all continents and lakes.

Additional note: Shading patterns can be selected with SHDPAT and MYPAT. Colours can be defined with COLOR and SETCLR.

S H D A F R

The routine SHDAFR plots shaded African countries.

The call is: CALL SHDAFR (INRAY, IPRAY, ICRAY, N) level 2

or: void shdafr (int *inray, long *ipray, int *icray, int n);

INRAY is an integer array containing the countries to be shaded. INRAY can have the following values:

1: Algeria	19: Gabon	37:Nigeria
2: Angola	20: Gambia	38: Rwanda
3: Benin	21: Ghana	39: Senegal
4: Botswana	22: Guinea	40: Seychelles
5: Burkina Faso	23: Guinea Bissau	41: Sierra Leone
6: Burundi	24: Kenya	42: Somalia
7: Cameroon	25: Lesotho	43: South Africa
8: Central African Rep.	26: Liberia	44: Sudan
9: Chad	27: Libya	45: Swaziland
10: Comoros	28: Madagascar	46: Tanzania
11: Congo, Dem. Rep.	29: Malawi	47: Togo
12: Congo, Rep.	30: Mali	48: Tunesia
13: Cote d'Ivoire	31: Mauritania	49: Uganda
14: Djibouti	32: Mauritius	50: West Sahara
15: Egypt	33: Morocco	51: Zambia
16: Equatorial Guinea	34: Mozambique	52: Zimbawe
17: Eritrea	35: Namibia	
18: Ethiopia	36: Niger	0: Africa

IPRAY	is an integer array containing shading patterns.
ICRAY	is an integer array containing colour numbers.
N	is the number of countries to be shaded.

Additional note: - The plotting of outlines can be suppressed with CALL NOARLN.

S H D E U R

The routine SHDEUR plots shaded European countries.

The call is: CALL SHDEUR (INRAY, IPRAY, ICRAY, N) level 2

or: void shdeur (int *inray, long *ipray, int *icray, int n);

INRAY is an integer array containing the countries to be shaded. INRAY can have the following values:

1: Albania	17: Luxembourg	33: Belarus
2: Andorra	18: Malta	34: Bosnia
3: Belgium	19: Netherlands	35: Croatia
4: Bulgaria	20: North Ireland	36: Czech Republic
5: Germany	21: Norway	37: Estonia
6: Denmark	22: Austria	38: Latvia
7: Cyprus	23: Poland	39: Lithuania
8: United Kingdom	24: Portugal	40: Macedonia
9: Finland	25: Romania	41: Moldava
10: France	26: Sweden	42: Russia
11: Greece	27: Switzerland	43: Serbia
12: Ireland	28: Spain	44: Slovakia
13: Iceland	29: CSFR	45: Slowenia
14: Italy	30: Turkey	46: Ukraine
15: Yugoslavia	31: USSR	
16: Liechtenstein	32: Hungary	0: Europe

IPRAY is an integer array containing shading patterns.

ICRAY is an integer array containing colour numbers.

N is the number of countries to be shaded.

Additional notes:
- The plotting of outlines can be suppressed with CALL NOARLN.
- To stay compatible with older programs, the number 15 (Yugoslavia) plots Bosnia, Croatia, Macedonia, Serbia and Slowenia, the number 29 (CSFR) plots Czech Republic and Slovakia and the number 31 (USSR) plots Belarus, Estonia, Latvia, Lithuania, Moldava, Russia and Ukraine.

S H D U S A

The routine SHDUSA plots shaded USA states.

The call is: CALL SHDUSA (INRAY, IPRAY, ICRAY, N) level 2

or: void shdeur (int *inray, long *ipray, int *icray, int n);

INRAY is an integer array containing the states to be shaded. INRAY can have the following values:

1: Alabama	19: Maine	37: Oregon
2: Alaska	20: Maryland	38: Pennsylvania
3: Arizona	21: Massachusetts	39: Rhode Island
4: Arkansas	22: Michigan	40: South Carolina
5: California	23: Minnesota	41: South Dakota
6: Colorado	24: Mississippi	42: Tennessee
7: Connecticut	25: Missouri	43: Texas
8: Delaware	26: Montana	44: Utah
9: Floria	27: Nbraska	45: Vermont
10: Georgia	28: Nevada	46: Virginia
11: Hawaii	29: New Hampshire	47: Washington
12: Idaho	30: New Jersey	48: West Virginia
13: Ilinois	31: New Mexico	49: Wisconsin

14: Indiana	32: New York	50: Wyoming
15: Iowa	33: North Carolina	51: Washington DC
16: Kansas	34: North Dakota	
17: Kentucky	35: Ohio	
18: Louisiana	36: Oklahoma	0: USA

IPRAY is an integer array containing shading patterns.

ICRAY is an integer array containing colour numbers.

N is the number of states to be shaded.

13.4 Plotting Data Points

C U R V M P

The routine CURVMP plots curves through data points or marks them with symbols.

The call is: CALL CURVMP (XRAY, YRAY, N) level 2

or: void curvmp (float *xray, float *yray, int n);

XRAY, YRAY are real arrays containing the data points.

N is the number of data points.

Additional notes:

- CURVMP is similar to CURVE except that only a linear interpolation can be used.
- In general, a line between two points on the globe will not be projected as a straight line. Therefore, CURVMP interpolates lines linearly by small steps. An alternate plotting mode can be set with MAPMOD.

13.5 Parameter Setting Routines

M A P B A S

The routine MAPBAS defines the map data file used in the routine WORLD. An internal DISLIN map file, some external map files compiled by Paul Wessel and map files in Mapgen format can be used. The map files compiled by Paul Wessel can be copied via FTP anonymous from the servers

ftp://ftp.ngdc.noaa.gov/MGG/shorelines/
ftp://gmt.soest.hawaii.edu/pub/wessel/gshhs/.

The external map files 'gshhs_l.b', 'gshhs_i.b', 'gshhs_h.b' and 'gshhs_f.b' must be copied to the map subdirectory of the DISLIN directory, or the name of the map file must be specified with the routine MAPFIL.

Map files in Mapgen format are available from the Coastline Extractor:

http://rimmer.ngdc.noaa.gov/

The call is: CALL MAPBAS (CBAS) level 1, 2

or: void mapbas (char *cbas);

CBAS	is a character string defining the map data file.
= 'DISLIN'	defines the DISLIN base map.
= 'GSHL'	defines 'gshhs_l.b' as base map.
= 'GSHI'	defines 'gshhs_i.b' as base map.
= 'GSHH'	defines 'gshhs_h.b' as base map.
= 'GSHF'	defines 'gshhs_f.b' as base map.
= 'MAPFIL'	defines an external map file as base map that is specified with the routine MAPFIL.

Default: CBAS = 'DISLIN'.

M A P F I L

The routine MAPFIL defines an external map file. The map file can be used as base map if the routine MAPBAS is called with the parameter 'MAPFIL'.

The call is:	CALL MAPFIL (CFIL, COPT)	level 1, 2
or:	void mapfil (char *cfil, char *copt);	

CFIL	is a character string containing the filename of the external map file.
COPT	is a character string describing the format of the map coordinates. COPT can have the values 'GSHHS' and 'MAPGEN'.

M A P L E V

The routine MAPLEV defines land or lake coordinates for WORLD if the external map files from Paul Wessel are used.

The call is:	CALL MAPLEV (COPT)	level 1, 2
or:	void maplev (char *copt);	

COPT	is a character string that can have the values 'ALL', 'LAND' and 'LAKE'.

Default: COPT = 'ALL'.

M A P P O L

MAPPOL defines the map pole used for azimuthal projections.

The call is:	CALL MAPPOL (XPOL, YPOL)	level 1
or:	void mappol (float xpol, float ypol);	

XPOL, YPOL	are the longitude and latitude coordinates in degrees where:

$-180 \leq XPOL \leq 180$ and $-90 \leq YPOL \leq 90$.

Default: (0., 0.)

Additional note:	For an azimuthal projection with YE - YA $\leq$ 90, the map pole will be set by GRAFMP to ((XA+XE)/2, (YA+YE)/2).

M A P S P H

For an azimuthal projection with YE - YA $>$ 90, DISLIN automatically projects a hemisphere around the map pole onto a circle. The hemisphere can be reduced using MAPSPH.

The call is: CALL MAPSPH (XRAD) level 1

or: void mapsph (float xrad);

XRAD defines the region around the map pole that will be projected onto a circle ($0 < XRAD \leq 90$).

Default: XRAD = 90.

M A P R E F

The routine MAPREF defines, for conical projections, two latitudes where the cone intersects or touches the globe.

The call is: CALL MAPREF (YLW, YUP) level 1

or: void mapref (float ylw, float yup);

YLW, YUP are the lower and upper latitudes where:

$0 \leq YLW \leq YUP \leq 90$ or $-90 \leq YLW \leq YUP \leq 0$

Default: YLW = YA + 0.25 * (YE - YA)
YUP = YA + 0.75 * (YE - YA)

Additional note: YLW and YUP can have identical values and lie outside of the axis scaling.

M A P M O D

The routine MAPMOD determines how data points will be connected by CURVMP.

The call is: CALL MAPMOD (CMODE) level 1, 2

or: void mapmod (char *cmode);

CMODE is a character string defining the connection mode.

= 'STRAIGHT' defines straight lines.

= 'INTER' means that lines will be interpolated linearly.

Default: CMODE = 'INTER'.

13.6 Conversion of Coordinates

P O S 2 P T

The routine POS2PT converts map coordinates to plot coordinates.

The call is: CALL POS2PT (XLONG, YLAT, XP, YP) level 2

or: void pos2pt (float xlong, float ylat, float *xp, float *yp);

XLONG, YLAT are the map coordinates in degrees.

XP, YP are the plot coordinates calculated by POS2PT.

The corresponding functions are:

XP = X2DPOS (XLONG, YLAT)

YP = Y2DPOS (XLONG, YLAT)

13.7 User-defined Projections

An user-defined projection can be enabled with the keyword 'MYPR' in the routine PROJCT. For a user-defined projection, the user must write a routine that converts longitude and latitude coordinates to axis coordinates (plot coordinates relative to the origin of the axis system). The name of the user written routine must then passed to DISLIN with the routine SETCBK.

S E T C B K

The routine SETCBK defines a user written callback routine.

The call is:	CALL SETCBK (ROUTINE, 'MYPR')	level 0, 1, 2, 3
or:	void setcbk (void (*routine)(float *xp, float *yp), "MYPR");	

ROUTINE	is the name of a routine defined by the user. In Fortran, the routine must be declared as EXTERNAL.

In the following example, a cylindrical projection is implemented as an user-defined projection:

```
PROGRAM MYPR
EXTERNAL MYFUNC
COMMON /MYCOMM/ XA,XE,YA,YE,NXL,NYL

XA = -180.
XE = 180.
YA = -90.
YE = 90.

NXL = 2400
NYL = 1200

CALL METAFL ('cons')
CALL DISINI
CALL PAGERA
CALL COMPLX
CALL AXSLEN (NXL, NYL)

CALL PROJCT ('MYPR')
CALL SETCBK (MYFUNC, 'MYPR')

CALL GRAFMP (XA, XE, XA, 90., YA, YE, YA, 30.)
CALL GRIDMP (1,1)
CALL WORLD
CALL DISFIN
END

SUBROUTINE MYFUNC (XP, YP)
COMMON /MYCOMM/ XA,XE,YA,YE,NXL,NYL
XP = (XP - XA)/(XE - XA) * (NXL - 1)
YP = (YP - YA)/(YE - YA) * (NYL - 1)
END
```

13.8 Examples

```
PROGRAM EX13_1

CALL SETPAG('DA4L')
CALL DISINI
CALL PAGERA
CALL COMPLX

CALL FRAME(3)
CALL AXSPOS(400,1850)
CALL AXSLEN(2400,1400)

CALL NAME('Longitude','X')
CALL NAME('Latitude','Y')
CALL TITLIN('World Coastlines and Lakes',3)

CALL LABELS('MAP','XY')
CALL GRAFMP(-180.,180.,-180.,90.,-90.,90.,-90.,30.)

CALL GRIDMP(1,1)
CALL WORLD

CALL HEIGHT(50)
CALL TITLE
CALL DISFIN
END
```

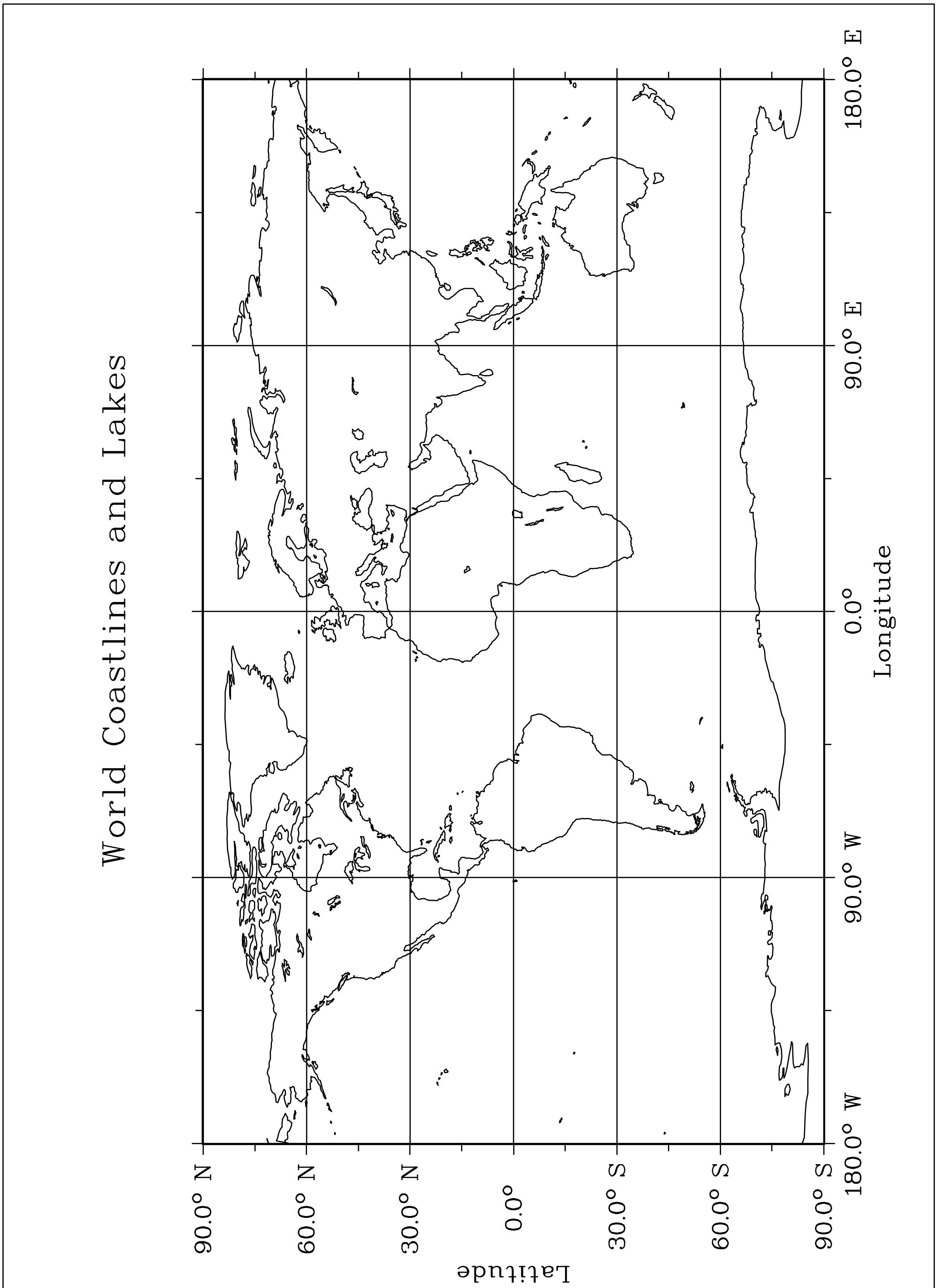


Figure 13.1: World Coastlines and Lakes

```
      PROGRAM EX13_2
      CHARACTER*6 CPROJ(3),CTIT*60
      DATA CPROJ/'Sanson','Winkel','Hammer'/

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL HEIGHT(40)
      CALL AXSLEN(1600,750)

      NYA=3850
      DO I=1,3
        NYA=NYA-950
        CALL AXSPOS(250,NYA)

        CALL PROJCT(CPROJ(I))
        CALL NOCLIP
        CALL GRAFMP(-180.,180.,-180.,30.,-90.,90.,-90.,15.)

        WRITE(CTIT,'(2A)') 'Elliptical Projection of ',
     *                      CPROJ(I)
        CALL TITLIN(CTIT,4)
        CALL TITLE

        CALL WORLD
        CALL GRIDMP(1,1)
        CALL ENDGRF
      END DO

      CALL DISFIN
      END
```

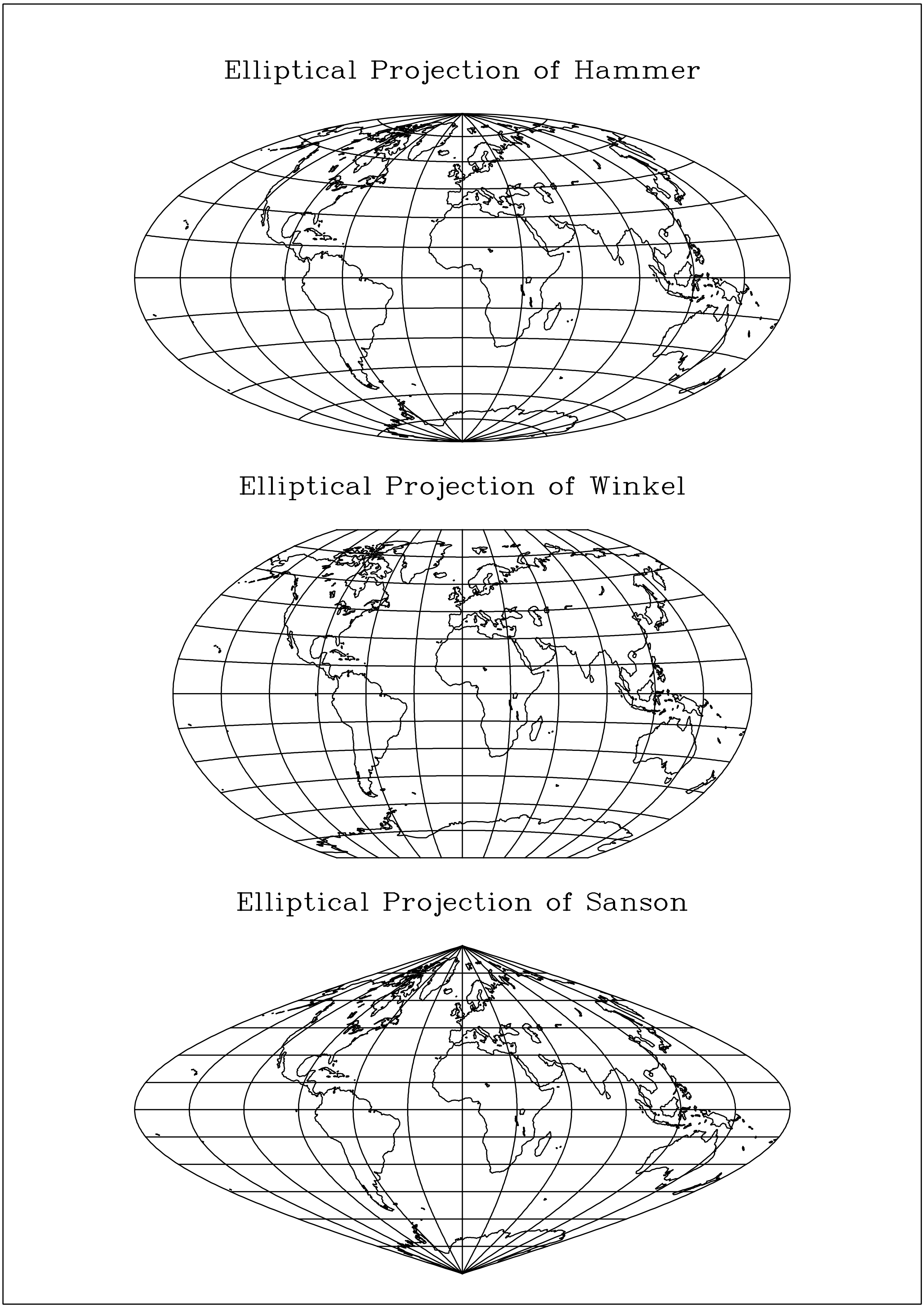


Figure 13.2: Elliptical Projections

```
PROGRAM EX13_3
DIMENSION NXA(4),NYA(4),XPOL(4),YPOL(4)
CHARACTER*60 CTIT
DATA NXA/200,1150,200,1150/NYA/1600,1600,2700,2700/
DATA XPOL/0.,0.,0.,0./YPOL/0.,45.,90.,-45./

CTIT='Azimuthal Lambert Projections'

CALL SETPAG('DA4P')
CALL DISINI
CALL PAGERA
CALL COMPLX

CALL HEIGHT(50)
NL=NLMESS(CTIT)
NX=(2250-NL)/2.
CALL MESSAG(CTIT,NX,300)

CALL AXSLEN(900,900)
CALL PROJCT('LAMBERT')

DO I=1,4
  CALL AXSPOS(NXA(I),NYA(I))
  CALL MAPPOL(XPOL(I),YPOL(I))
  CALL GRAFMP(-180.,180.,-180.,30.,-90.,90.,-90.,30.)

  CALL WORLD
  CALL GRIDMP(1,1)
  CALL ENDGRF
END DO

CALL DISFIN
END
```

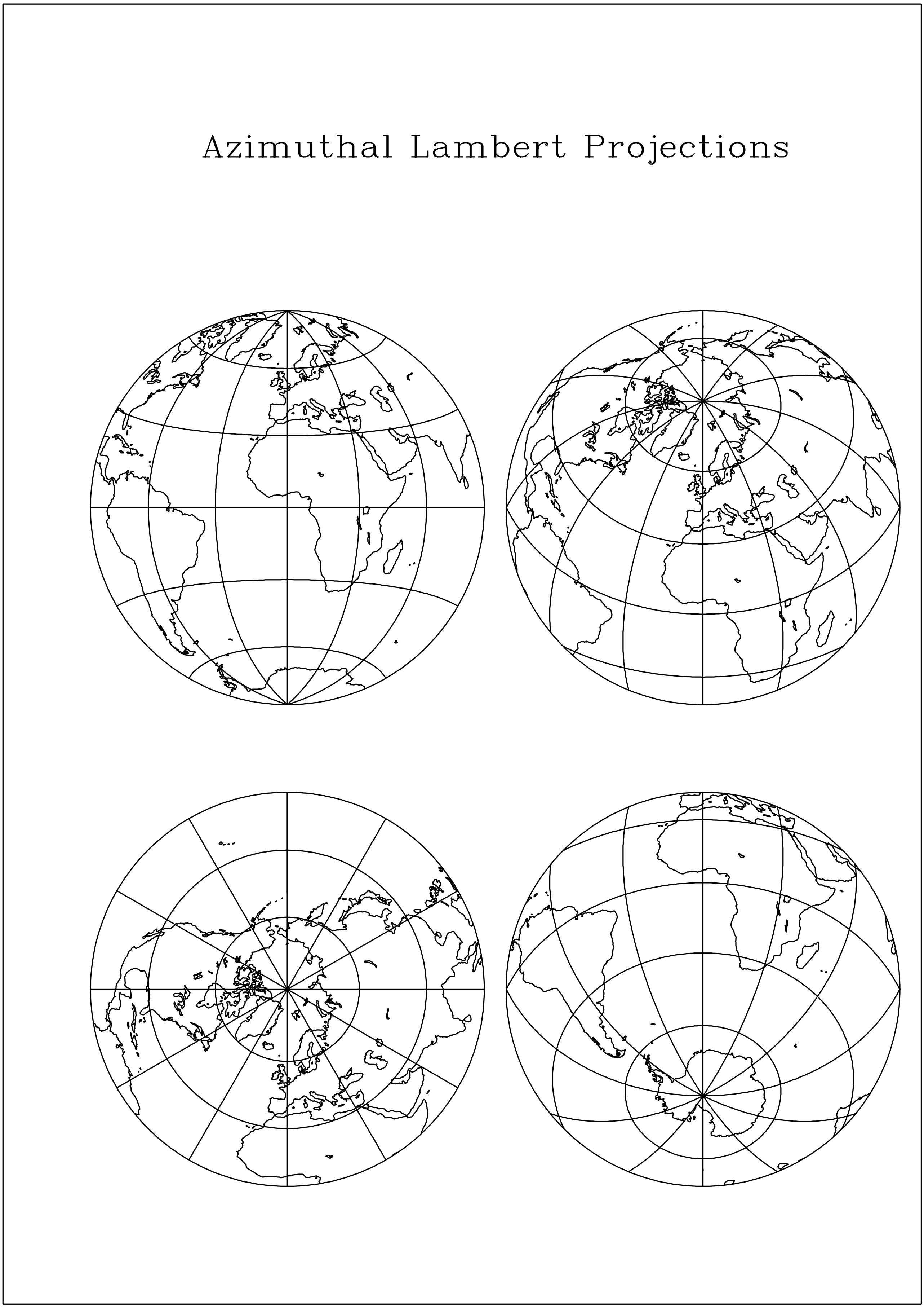


Figure 13.3: Azimuthal Lambert Projections

```
PROGRAM EX13_4
PARAMETER (N = 32)
DIMENSION INRAY(1),IPRAY(1),ICRAY(1)

INRAY(1)=0
IPRAY(I)=0
ICRAY(I)=255

CALL SETPAG('DA4P')
CALL DISINI
CALL PAGERA
CALL COMPLX

CALL INTAX
CALL TICKS(1,'XY')
CALL FRAME(3)
CALL AXSLEN(1600,2200)
CALL AXSPOS(400,2700)

CALL NAME('Longitude','X')
CALL NAME('Latitude','Y')
CALL TITLIN('Conformal Conic Projection',3)

CALL LABELS('MAP','XY')
CALL PROJCT('CONF')
CALL GRAFMP(-10.,30.,-10.,5.,35.,70.,35.,5.)

CALL GRIDMP(1,1)
CALL SHDEUR(INRAY,IPRAY,ICRAY,N)

CALL HEIGHT(50)
CALL TITLE
CALL DISFIN
END
```

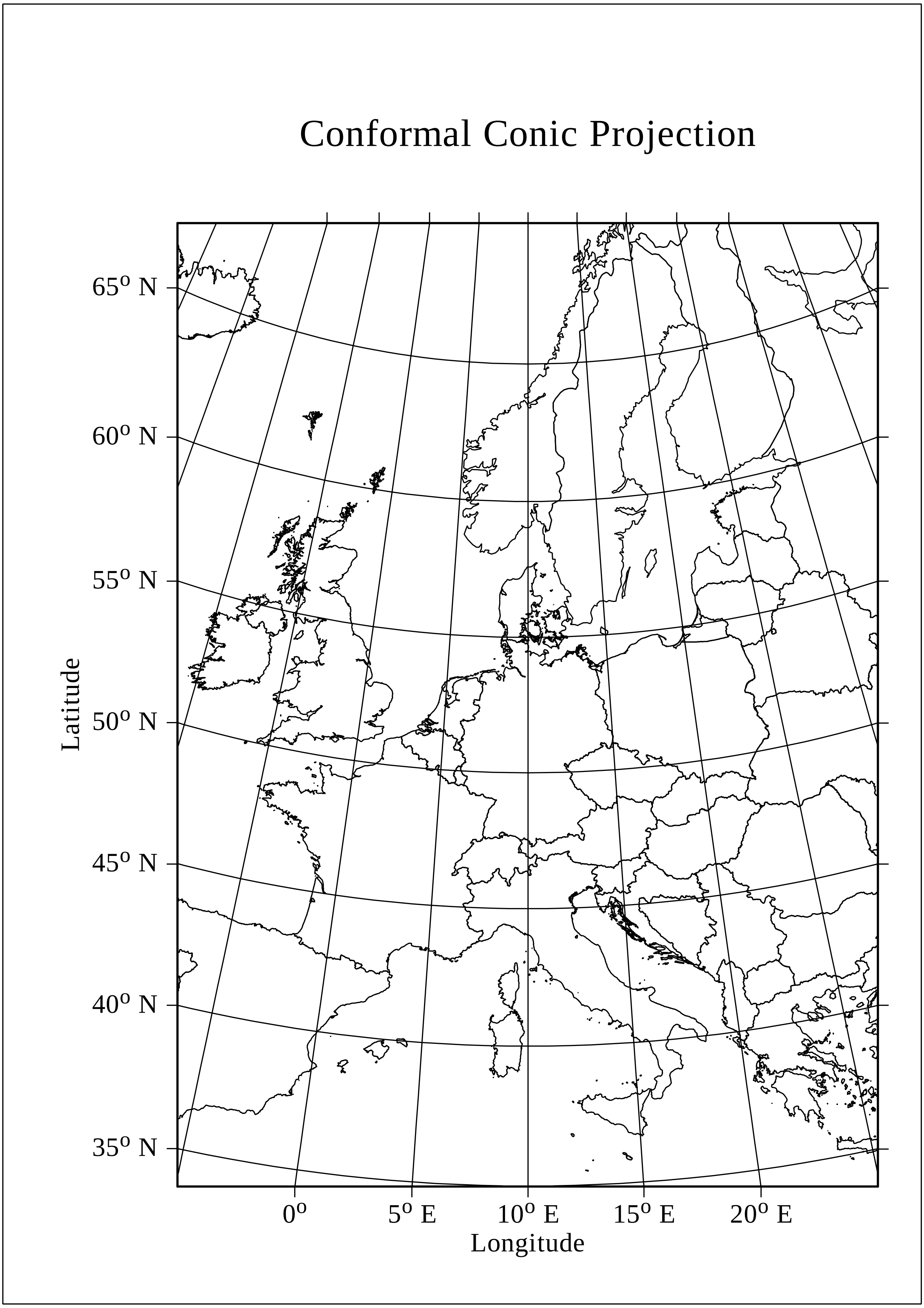


Figure 13.4: Conformal Conic Projection

Chapter 14

Contouring

This chapter describes routines for contouring three-dimensional functions of the form Z = F(X,Y). Contours can be generated with the routine CONPTS or with other software packages and plotted with the routine CONCRV or can be calculated and plotted by DISLIN with the routines CONMAT, CONTUR and CONSHD.

14.1 Plotting Contours

C O N C R V

CONCRV plots contours generated by other software packages.

The call is:	CALL CONCRV (XRAY, YRAY, N, ZLEV)	level 2, 3
or:	void concrv (float *xray, float *yray, int n, float zlev);	

XRAY, YRAY	are arrays containing the X- and Y-coordinates of a contour line.
N	is the number of points.
ZLEV	is a function value used for labels.

C O N T U R

The routine CONTUR calculates and plots contours of the function Z = F(X,Y).

The call is:	CALL CONTUR (XRAY, N, YRAY, M, ZMAT, ZLEV)	level 2, 3
or:	void contur (float *xray, int n, float *yray, int m, float *zmat, float zlev);	

XRAY	is an array containing X-coordinates.
N	is the dimension of XRAY.
YRAY	is an array containing Y-coordinates.
M	is the dimension of YRAY.
ZMAT	is a matrix of the dimension (N, M) containing function values.
ZLEV	is a function value that defines the contour line to be calculated. ZLEV can be used for labels.

C O N M A T

The routine CONMAT plots contours of the function Z = F(X,Y). The function values correspond to a linear grid in the XY-plane.

The call is:	CALL CONMAT (ZMAT, N, M, ZLEV)	level 2, 3
or:	void conmat (float *zmat, int n, int m, float zlev);	

ZMAT — is a matrix of the dimension (N, M) containing the function values. If XA, XE, YA and YE are the axis limits in GRAF or values defined with the routine SURSZE, the connection of grid points and matrix elements can be described by the formula:

ZMAT(I,J) = F(X,Y) where

X = XA + (I - 1) * (XE - XA) / (N - 1) , I = 1,..,N and
Y = YA + (J - 1) * (YE - YA) / (M - 1) , J = 1,..,M.

N, M — define the dimension of ZMAT.

ZLEV — is a function value that defines the contour line to be calculated. The value can be used for labels.

Additional notes:

- CONCRV, CONTUR and CONMAT use linear interpolation to connect contour points. The routine TRFMAT can be used to pass a matrix with a better resolution to CONTUR and CONMAT.
- Geographical projections can be defined for contouring.
- The thickness of contours can be set with THKCRV. Line styles and colours can also be defined. Legends are supported for filled and non-filled contours.
- The number of matrix points in CONTUR and CONMAT is limited to N * M $\leq$ 256000 in Fortran 77. There is no limit for the C and Fortran 90 libraries of DISLIN.
- To plot contours for randomly distributed points of the form X(N), Y(N) and Z(N), the routine GETMAT can be used to calculate a function matrix, or the data can be triangulated with the routine TRIANG and contours can be plotted from the triangulation with the routine CONTRI.

C O N T R I

The routine CONTRI plots contours from triangulated data that can be calculated by the routine TRIANG from a set of irregularily distributed data points.

The call is:	CALL CONTRI (XRAY, YRAY, ZRAY, N, I1RAY, I2RAY, I3RAY, NTRI, ZLEV)	level 2, 3
or:	void contri (float *xray, float *yray, float *zray, int n, int *i1ray, int *i2ray, int *i3ray, int ntri, float zlev);	

XRAY — is an array containing the X-coordinates of data points.

YRAY — is an array containing the Y-coordinates of data points.

ZRAY — is an array containing the Z-coordinates of data points.

N — is the number of data points.

I1RAY, I2RAY, I3RAY — is the Delaunay triangulation of the points (XRAY, YRAY) calculated by the routine TRIANG.

NTRI — is the number of triangles in I1RAY, I2RAY and I3RAY.

ZLEV — is a function value that defines the contour line to be calculated.

14.2 Plotting Filled Contours

C O N S H D

The routine CONSHD plots filled contours of the function Z = F(X,Y). Two algorithms can be selected for contour filling: a cell filling algorithm and a polygon filling algorithm. For a polygon filling, only closed contours can be filled. The algorithm can be defined with the routine SHDMOD.

The call is: CALL CONSHD (XRAY, N, YRAY, M, ZMAT, ZLVRAY, NLV) level 2, 3

or: void conshd (float *xray, int n, float *yray, int m, float *zmat, float *zlvray, int nlv);

XRAY is an array containing X-coordinates.

N is the dimension of XRAY.

YRAY is an array containing Y-coordinates.

M is the dimension of YRAY.

ZMAT is a matrix of the dimension (N, M) containing function values.

ZLVRAY is an array containing the levels. For polygon filling, the levels should be sorted in such a way that inner contours are plotted last.

NLV is the number of levels.

Additional notes:

- CONSHD may give better results for a higher resolution of ZMAT. A matrix with a higher resolution can be calculated with the routine TRFMAT.
- The colours of the filled contours are calculated from a fictive colour bar where the minimum and maximum of the contour levels are used for the lower and upper limits of the colour bar. The scaling of the colour bar can be modified with the routine ZSCALE while a colour bar can be displayed with the routine ZAXIS. If the colours of the filled contours should not be calculated from a colour bar, they can be defined directly with the routine CONCLR.
- For a cell filling, the calculation of contour colours are described in the following. The levels are sorted first in ascending order. By default, the colour of the region between two neighbouring levels is calculated from the lower value of the two levels. If you want to use the upper value, you can define it with the routine SHDMOD ('UPPER', 'COLOUR'). In default mode (SHDMOD ('LOWER', 'COLOUR'), the cells below the mimimum of the levels are filled with the lowermost colour of the colour bar, the cells above the maximum of the levels are filled with the uppermost colour of the colour bar. The plotting of this regions can be suppressed with the statement CALL SHDMOD ('NONE', 'CELL').

C O N F L L

The routine CONFLL plots filled contours from triangulated data that can be calculated by the routine TRIANG from a set of irregularily distributed data points.

The call is: CALL CONFLL (XRAY, YRAY, ZRAY, N, I1RAY, I2RAY, I3RAY, NTRI, ZLVRAY, NLV) level 2, 3

or: void confll (float *xray, float *yray, float *zray, int n, int *i1ray, int *i2ray, int *i3ray, int ntri, float *zlvray, int nlv);

XRAY	is an array containing the X-coordinates of data points.
YRAY	is an array containing the Y-coordinates of data points.
ZRAY	is an array containing the Z-coordinates of data points.
N	is the number of data points.
I1RAY, I2RAY, I3RAY	is the Delaunay triangulation of the points (XRAY, YRAY) calculated by the routine TRIANG.
NTRI	is the number of triangles in I1RAY, I2RAY and I3RAY.
ZLVRAY	is an array containing the levels.
NLV	is the number of levels.
Additional note:	See the notes for CONSHD.

14.3 Generating Contours

C O N P T S

The routine CONPTS generates contours without plotting. Multiple curves can be returned for one contour level.

The call is:	CALL CONPTS (XRAY, N, YRAY, M, ZMAT, ZLEV, XPTRAY, YPTRAY, MAXPTS, IRAY, MAXCRV, NCURVS) level 0, 1, 2, 3
or:	void conpts (float *xray, int n, float *yray, int m, float *zmat, float zlev, float *xptray, float *yptray, int maxpts, int *iray, int maxray, int *ncurvs);

XRAY	is an array containing X-coordinates.
N	is the dimension of XRAY.
YRAY	is an array containing Y-coordinates.
M	is the dimension of YRAY.
ZMAT	is a matrix of the dimension (N, M) containing function values.
ZLEV	is a function value that defines the contour line to be calculated.
XPTRAY, YPTRAY	are returned arrays containing the calculated contour. The arrays can contain several curves.
MAXPTS	is the maximal number of points that can be passed to XPTRAY and YPTRAY.
IRAY	is a returned integer array that contains the number of points for each generated contour curve.
MAXCRV	is the maximal number of entries that can be passed to IRAY.
NCURVS	is the returned number of generated curves.

Example:

The following statements generate from some arrays XRAY, YRAY and ZMAT contours and plot them with the routine CURVE.

```
PARAMETER (N=100, MAXPTS=1000,MAXCRV=10)
REAL ZMAT(N,N),XRAY(N),YRAY(N),XPTS(MAXPTS),YPTS(MAXPTS)
INTEGER IRAY(MAXCRV)
.....
```

```
      DO I=1,12
        ZLEV=0.1+(I-1)*0.1
        CALL CONPTS(XRAY,N,YRAY,N,ZMAT,ZLEV,XPTS,YPTS,MAXPTS,
     *                IRAY,MAXCRV,NCURVS)
        K=1
        DO J=1,NCURVS
          CALL CURVE(XPTS(K),YPTS(K),IRAY(J))
          K=K+IRAY(J)
        END do
      END DO
```

T R I P T S

The routine TRIPTS generates contours from triangulated data without plotting. Multiple curves can be returned for one contour level.

The call is:	CALL TRIPTS (XRAY, YRAY, ZRAY, N, I1RAY, I2RAY, I3RAY, NTRI, ZLEV, XPTRAY, YPTRAY, MAXPTS, IRAY, MAXCRV, NCURVS) level 0, 1, 2, 3
or:	void tripts (float *xray, float *yray, float *zray, int n, int *i1ray, int *i2ray, int *i3ray, int ntri, float zlev, float *xptray, float *yptray, int maxpts, int *iray, int maxray, int *ncurvs);

XRAY	is an array containing the X-coordinates of data points.
YRAY	is an array containing the Y-coordinates of data points.
ZRAY	is an array containing the Z-coordinates of data points.
N	is the number of data points.
I1RAY, I2RAY, I3RAY	is the Delaunay triangulation of the points (XRAY, YRAY) calculated by the routine TRIANG.
NTRI	is the number of triangles in I1RAY, I2RAY and I3RAY.
ZLEV	is a function value that defines the contour line to be calculated.
XPTRAY, YPTRAY	are returned arrays containing the calculated contour. The arrays can contain several curves.
MAXPTS	is the maximal number of points that can be passed to XPTRAY and YPTRAY.
IRAY	is a returned integer array that contains the number of points for each generated contour curve.
MAXCRV	is the maximal number of entries that can be passed to IRAY.
NCURVS	is the returned number of generated curves.

14.4 Parameter Setting Routines

L A B E L S

The routine LABELS defines contour labels.

The call is:	CALL LABELS (COPT, 'CONTUR') level 1, 2, 3
or:	void labels (char *copt, "CONTUR");

COPT	is a character string defining the labels.
= ’NONE’	means that no labels will be plotted.
= ’FLOAT’	means that the level value will be used for labels.
= ’CONLAB’	means that labels defined with the routine CONLAB will be plotted.
	Default: COPT = ’NONE’.

Additional note: The number of decimal places in contour labels can be defined with CALL LABDIG (NDIG, ’CONTUR’). The default value for NDIG is 1.

L A B D I S

The routine LABDIS defines the distance between contour labels.

The call is: CALL LABDIS (NDIS, ’CONTUR’) level 1, 2, 3

or: void labdis (int ndis, ”CONTUR”);

NDIS is the distance between labels in plot coordinates.

Default: NDIS = 500

L A B C L R

The routine LABCLR defines the colour of contour labels.

The call is: CALL LABCLR (NCLR, ’CONTUR’) level 1, 2, 3

or: void labclr (int nclr, ”CONTUR”);

NCLR is a colour value. If NCLR = -1, the contour labels will be plotted with the current colour.

Default: NCLR = -1

C O N L A B

The routine CONLAB defines a character string which will be used for labels if the routine LABELS is called with the parameter ’CONLAB’.

The call is: CALL CONLAB (CLAB) level 1, 2, 3

or: void conlab (char *clab);

CLAB is a character string containing the label.

Default: CLAB = ’ ’.

C O N M O D

The routine CONMOD modifies the appearance of contour labels. By default, DISLIN suppresses the plotting of labels at a position where the contour is very curved. To measure the curvature of a contour for an interval, DISLIN calculates the ratio between the arc length and the length of the straight line between the interval limits. If the quotient is much greater than 1, the contour line is very curved in that interval.

The call is: CALL CONMOD (XFAC, XQUOT) level 1, 2, 3

or: void conmod (float xfac, float xquot);

XFAC	defines the length of intervals (≥ 0). The curvature of contours will be tested in intervals of the length (1 + XFAC) * W where W is the label length.
XQUOT	defines an upper limit for the quotient of arc length and length of the straight line (> 1). If the quotient is greater than XQUOT, the plotting of labels will be suppressed in the tested interval.

Default: (0.5, 1.5).

C O N G A P

The routine CONGAP defines the distance between contour lines and labels.

The call is:	CALL CONGAP (XFAC)	level 1, 2, 3
or:	void congap (float xfac);	

XFAC	is a real number used as a scaling factor. The distance between contour lines and labels is set to XFAC * NH where NH is the current character height.

Default: XFAC = 0.5.

S H D M O D

The routine SHDMOD selects the algorithm used for contour filling, or modifies options for cell filling.

The call is:	CALL SHDMOD (COPT, CKEY)	level 1, 2, 3
or:	void shdmod (char *copt, char *ckey);	

CKEY	is a character string containing one of the following keywords:
= 'CONTUR'	defines the algorithm used for contour filling. COPT can have the values 'CELL' and 'POLY'. The default value is COPT = 'CELL'.
= 'CELL'	modifies the cell filling algorithm. COPT can have the values 'BOTH', 'UPPER', 'LOWER' and 'NONE'. If COPT = 'NONE', the filling of the regions below and above the level limits are suppressed. If COPT = 'UPPER', the filling of the region below the lowermost level is suppressed. COPT = 'LOWER' means that the filling of the region above the uppermost level is suppressed. By default, both regions described above are filled.
= 'COLOR'	modifies the calculation of colours for cell filling. COPT can have the values 'LOWER', 'MIDDLE' and 'UPPER'. For COPT = 'LOWER', the lower value of two neighbouring levels is used for colour calculation, for COPT = 'UPPER', the upper value of two neighbouring levels is used, and for COPT = 'MIDDLE', the mean value of the two levels is used for colour calculation. The default value is COPT = 'LOWER'.

C O N C L R

The routine CONCLR defines colours for filled contour lines.

The call is:	CALL CONCLR (NCRAY, N)	level 1, 2, 3
or:	void conclr (int *ncray, int n);	

NCRAY	is an integer array containing colour numbers.
N	is the number of entries in NCRAY.

14.5 Examples

```
PROGRAM EX14_1
PARAMETER (N=100)
DIMENSION X(N),Y(N),Z(N,N)

FPI=3.14159/180.
STEP=360./(N-1)
DO I=1,N
  X(I)=(I-1.)*STEP
  Y(I)=(I-1.)*STEP
END DO

DO I=1,N
  DO J=1,N
    Z(I,J)=2*SIN(X(I)*FPI)*SIN(Y(J)*FPI)
  END DO
END DO

CALL SETPAG('DA4P')
CALL DISINI
CALL PAGERA
CALL COMPLX

CALL TITLIN('Contour Plot',1)
CALL TITLIN('F(X,Y) = 2 * SIN(X) * SIN(Y)',3)
CALL NAME('X-axis','X')
CALL NAME('Y-axis','Y')

CALL INTAX
CALL AXSPOS(450,2670)
CALL GRAF(0.,360.,0.,90.,0.,360.,0.,90.)

CALL HEIGHT(30)
DO I=1,9
  ZLEV=-2.+(I-1)*0.5
  IF(I.EQ.5) THEN
    CALL LABELS('NONE','CONTUR')
  ELSE
     CALL LABELS('FLOAT','CONTUR')
  END IF
CALL CONTUR(X,N,Y,N,Z,ZLEV)
END DO
CALL HEIGHT(50)
CALL TITLE
CALL DISFIN
END
```

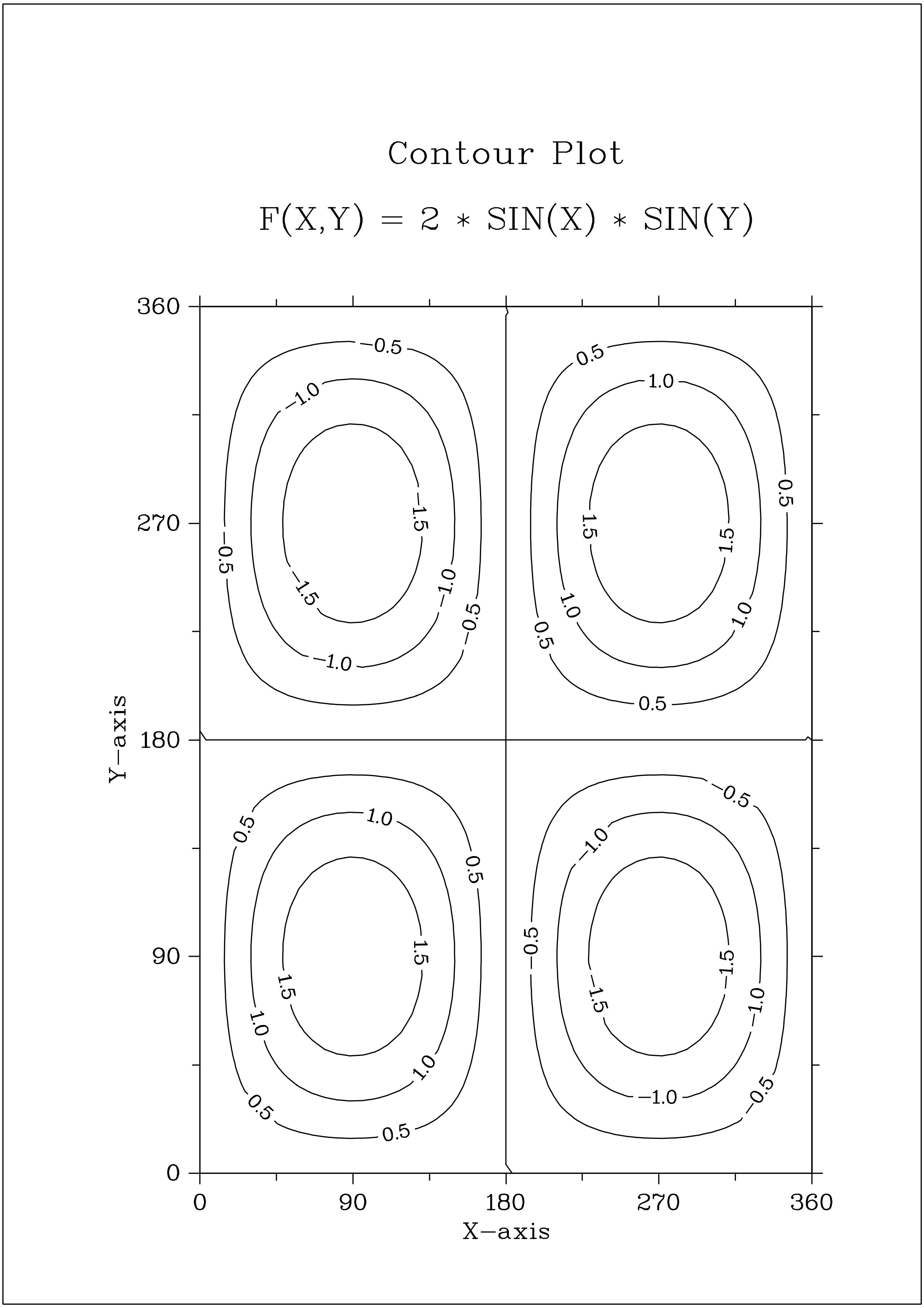


Figure 14.1: Contour Plot

```
PROGRAM EX14_2
PARAMETER (N=100)
DIMENSION ZMAT(N,N)

STEP=1.2/(N-1)
DO I=1,N
  X=0.4+(I-1)*STEP
  DO J=1,N
    Y=0.4+(J-1)*STEP
    ZMAT(I,J)=(X**2.-1.)**2. + (Y**2.-1.)**2.
  END DO
END DO

CALL SETPAG('DA4P')
CALL DISINI
CALL PAGERA
CALL COMPLX
 CALL MIXALF
CALL TITLIN('Contour Plot',1)
CALL TITLIN('F(X,Y) = (X[2$ - 1)[2$ + (Y[2$ - 1)[2$',3)
CALL NAME('X-axis','X')
CALL NAME('Y-axis','Y')

CALL AXSPOS(450,2670)
CALL GRAF(0.4,1.6,0.4,0.2,0.4,1.6,0.4,0.2)

DO I=1,12
  ZLEV=0.1+(I-1)*0.1
  IF(MOD(I,3).EQ.1) THEN
    CALL SOLID
    CALL THKCRV(3)
  ELSE IF(MOD(I,3).EQ.2) THEN
    CALL DASH
    CALL THKCRV(1)
  ELSE
    CALL DOT
    CALL THKCRV(1)
  END IF

  CALL CONMAT(ZMAT,N,N,ZLEV)
END DO

CALL HEIGHT(50)
CALL TITLE
CALL DISFIN
END
```

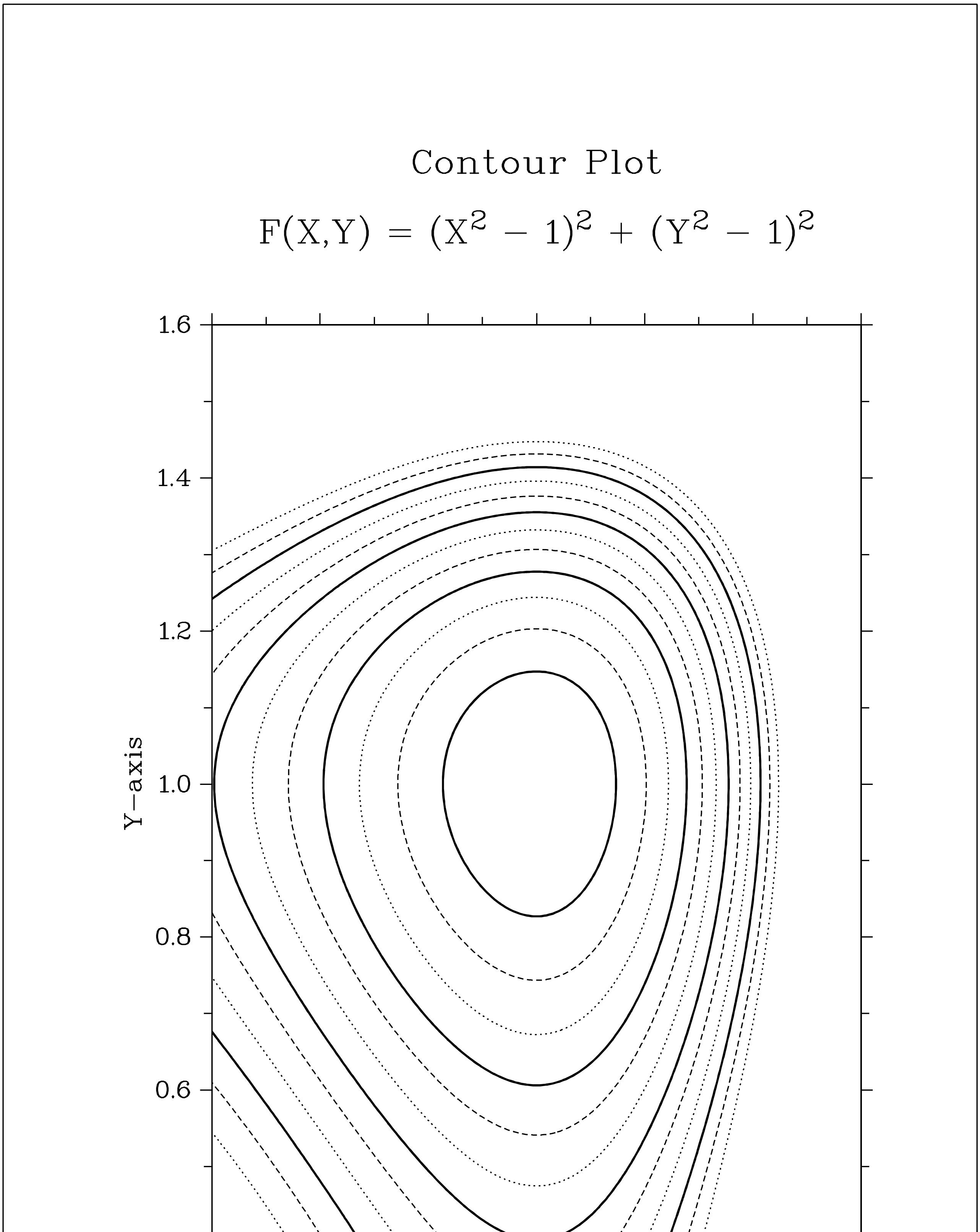


Figure 14.2: Contour Plot

```
      PROGRAM EX14_3
      PARAMETER (N=100)
      DIMENSION ZMAT(N,N),XRAY(N),YRAY(N),ZLEV(12)

      STEP=1.6/(N-1)
      DO I=1,N
        XRAY(I)=0.0+(I-1)*STEP
        DO J=1,N
          YRAY(J)=0.0+(J-1)*STEP
          ZMAT(I,J)=(XRAY(I)**2.-1.)**2. +
     *              (YRAY(J)**2.-1.)**2.
        END DO
      END DO

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL MIXALF
      CALL TITLIN('Shaded Contour Plot',1)
      CALL TITLIN('F(X,Y) = (X[2$ - 1)[2$ + (Y[2$ - 1)[2$',3)
      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')

      CALL SHDMOD('POLY','CONTUR')
      CALL AXSPOS(450,2670)
      CALL GRAF(0.0,1.6,0.0,0.2,0.0,1.6,0.0,0.2)

      DO I=1,12
        ZLEV(13-I)=0.1+(I-1)*0.1
      END DO

      CALL CONSHD(XRAY,N,YRAY,N,ZMAT,ZLEV,12)

      CALL HEIGHT(50)
      CALL TITLE
      CALL DISFIN
      END
```

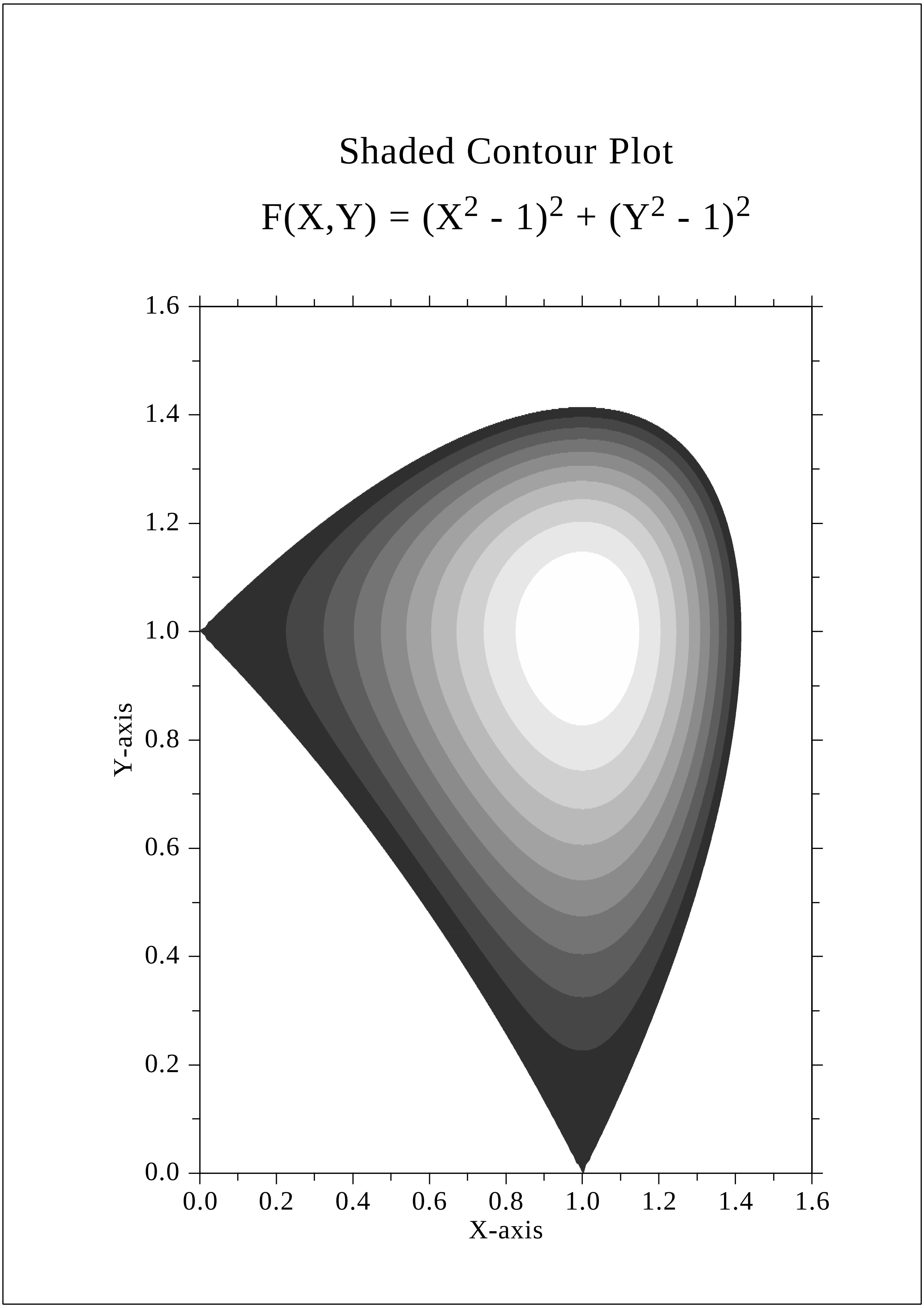


Figure 14.3: Shaded Contour Plot

Chapter 15

Widget Routines

DISLIN offers some routines for creating graphical user interfaces in Fortran and C programs. The routines are called widget routines and use the Motif widget libraries on X11 and the API functions on Windows systems.

There are sets of routines in DISLIN for creating single widgets, for setting parameters, for requesting current widget values selected by the user and for creating dialogs.

Routines for creating single widgets begin with the characters 'WG', parameter setting routines with the characters 'SWG', requesting routines with the characters 'GWG' and dialog routines with the characters 'DWG'.

Normally, creating widget and parameter setting routines should be used between the routines WGINI and WGFIN while requesting routines can be called after WGFIN, or in a callback routine. Dialog routines can be used independently from the routines WGINI and WGFIN.

15.1 Widget Routines

W G I N I

The routine WGINI initializes the widget routines and creates a main widget.

The call is:	CALL WGINI (COPT, ID)
or:	int wgini (char *copt);

COPT	is a character string that defines how children widgets are laid out in the main widget:
= 'VERT'	means that children widgets are laid out in columns from top to bottom.
= 'HORI'	means that children widgets are laid out in rows from left to right.
= 'FORM'	means that the position and size of children widgets is defined by the user with the routines SWGPOS, SWGSIZ and SWGWIN.
ID	is the returned widget index. It can be used as a parent widget index in other widget calls.

W G F I N

WGFIN terminates the widget routines. The widgets will be displayed on the screen. After choosing OK in the Exit menu, all widgets are deleted and the program is continued after WGFIN. After choosing Quit in the Exit menu, the program is terminated.

The call is:	CALL WGFIN

or: void wgfin ();

W G B A S

The routine WGBAS creates a container widget. It can be used as a parent widget for other widgets.

The call is: CALL WGBAS (IP, COPT, ID)

or: int wgbas (int ip, char *copt);

IP is the index of the parent widget.

COPT is a character string that can have the values 'HORI', 'VERT' and 'FORM'. It determines how children widgets are laid out in the container widget (s. WGINI).

ID is the returned widget index. It can be used as a parent widget index in other widget calls.

W G P O P

The routine WGPOP creates a popup menu in the menubar of the main widget, or a popup submenu of a pop umenu. Entries in the popup menu must be created with WGAPP.

The call is: CALL WGPOP (IP, CLAB, ID)

or: int wgpop (int ip, char *clab);

IP is the index of a widget created by WGINI, or the index of another popup widget.

CLAB is a character string containing the title of the popup menu.

ID is the returned widget index. It can be used as a parent widget index for WGAPP and WGPOP.

W G A P P

The routine WGAPP creates an entry in a popup menu. The popup menu must be created with the routine WGPOP.

The call is: CALL WGAPP (IP, CLAB, ID)

or: int wgapp (int ip, char *clab);

IP is the index of a popup menu created with WGPOP.

CLAB is a character string containing a label.

ID is the returned widget index. It should be connected with a callback routine (see SWGCBK).

W G L A B

The routine WGLAB creates a label widget. The widget can be used to display a character string.

The call is: CALL WGLAB (IP, CSTR, ID)

or: int wglab (int ip, char *cstr);

IP is the index of the parent widget.

CSTR is a character string that should be displayed.

ID	is the returned widget index.

W G B U T

The routine WGBUT creates a button widget. The widget represents a labeled button that the user can turn on or off by clicking.

The call is:	CALL WGBUT (IP, CLAB, IVAL, ID)
or:	int wgbut (int ip, char *clab, int ival);

IP	is the index of the parent widget.
CLAB	is a character string that will be used as a label.
IVAL	can have the values 0 (off) and 1 (on) and is used to initialize the button.
ID	is the returned widget index.

W G S T X T

The routine WGSTXT creates a scrolled widget that can be used for text output. The text cannot not be modified. Text entries in the widget can be made with the routine SWGTXT.

The call is:	CALL WGSTXT (IP, NSIZE, NMAX, ID)
or:	int wgtxt (int ip, int nsize, int nmax);

IP	is the index of the parent widget.
NSIZE	defines the vertical size of the widget in text rows.
NMAX	defines the maximal number of displayed entries in the scrolled widget. If this number is reached and a new entry is made, the first entry in the widget is deleted.
ID	is the returned widget index.

W G T X T

The routine WGTXT creates a text widget. The widget can be used to get text from the keyboard.

The call is:	CALL WGTXT (IP, CSTR, ID)
or:	int wgtxt (int ip, char *cstr);

IP	is the index of the parent widget.
CSTR	is a character string that will be displayed in the text widget ($\leq$ 256 characters).
ID	is the returned widget index.

W G L T X T

The routine WGLTXT creates a labeled text widget. The widget can be used to get text from the keyboard.

The call is:	CALL WGLTXT (IP, CLAB, CSTR, NWTH, ID)
or:	int wgltxt (int ip, char *clab, char *cstr, int nwth);

IP	is the index of the parent widget.

CLAB	is a character string containing a label. It will be displayed on the left side of the widget.
CSTR	is a character string that will be displayed in the text widget ($\leq$ 256 characters).
NWTH	defines the width of the text field (0 $\leq$ NWTH $\leq$ 100). For example, NWTH = 30 means that the width of the text field is: 0.3 * widget width.
ID	is the returned widget index.

W G F I L

The routine WGFIL creates a file widget. The widget can be used to get a filename from the keyboard. The filename can be typed directly into the file field or can be selected from a file selection box if an entry in the File menu is chosen.

The call is:	CALL WGFIL (IP, CLAB, CFIL, CMASK, ID)
or:	int wgfil (int ip, char *clab, char *cfil, char *cmask);

IP	is the index of the parent widget.
CLAB	is a character string used for an entry in the File menu.
CFIL	is a character string that will be displayed in the file widget ($\leq$ 256 characters).
CMASK	specifies the search pattern used in determining the files to be displayed in the file selection box.
ID	is the returned widget index.

W G L I S

The routine WGLIS creates a list widget. This widget is used whenever an application must present a list of names from which the user can choose.

The call is:	CALL WGLIS (IP, CLIS, ISEL, ID)
or:	int wglis (int ip, char *clis, int isel);

IP	is the index of the parent widget.
CLIS	is a character string that contains the list elements. Elements must be separated by the character '|'.
ISEL	defines the pre-selected element ($\geq$ 1).
ID	is the returned widget index.

W G D L I S

The routine WGDLIS creates a dropping list widget. This list widget can be used to save space in the parent widget.

The call is:	CALL WGDLIS (IP, CLIS, ISEL, ID)
or:	int wgdlis (int ip, char *clis, int isel);

IP	is the index of the parent widget.
CLIS	is a character string that contains the list elements. Elements must be separated by the character '|'.
ISEL	defines the pre-selected element ($\geq$ 1).

ID	is the returned widget index.
Additional note:	This widget may not be supported on all X11 workstations since it is a feature of Motif 1.2. If WGDLIS is not supported, WGLIS will be used instead.

WGBOX

The routine WGBOX creates a list widget where the list elements are displayed as toggle buttons.

The call is:	CALL WGBOX (IP, CLIS, ISEL, ID)
or:	int wgbox (int ip, char *clis, int isel);

IP	is the index of the parent widget.
CLIS	is a character string that contains the list elements. Elements must be separated by the character '|'.
ISEL	defines the pre-selected element (≥ 1).
ID	is the returned widget index.

WGSCL

The routine WGSCL creates a scale widget. The widget can be displayed in horizontal or vertical direction.

The call is:	CALL WGSCL (IP, CLAB, XMIN, XMAX, XVAL, NDEZ, ID)
or:	int wgscl (int ip, char *clab, float xmin, float xmax, float xval, int ndez);

IP	is the index of the parent widget.
CLAB	is a character string used for a label.
XMIN	is a floating-point value that defines the minimal value of the scale widget.
XMAX	is a floating-point value that defines the maximal value of the scale widget.
XVAL	defines the value of the scale widget.
NDEZ	is the number of digits used in the scale widget.
ID	is the returned widget index.
Additional note:	A step parameter for scale widgets can be defined with the routine SWGSTP.

WGDRAW

The routine WGDRAW creates a draw widget that can be used for graphical output from DISLIN plotting routines.

The call is:	CALL WGDRAW (IP, ID)
or:	int wgdraw (int ip);

IP	is the index of the parent widget.
ID	is the returned widget index.
Additional notes:	- The returned widget ID of a draw widget can be used in the routine SETXID for setting the graphical output of DISLIN routines to the draw widget. For X11, SETXID should be called if the widgets are already realized. Normally, SETXID should be called in a callback routine.

- By default, the height of a draw widget is identical width the width of the widget. The height of draw widgets can be modified with the routine SWGDRW.

W G O K

The routine WGOK creates a push button widget where the button has the same meaning as the OK entry in the Exit menu. If the button is pressed, all widgets are deleted and the program is continued after WGFIN.

The call is:	CALL WGOK (IP, ID)
or:	int wgok (int ip);
IP	is the index of the parent widget.
ID	is the returned widget index.

W G Q U I T

The routine WGQUIT creates a push button widget where the button has the same meaning as the QUIT entry in the Exit menu. If the button is pressed, the program is terminated.

The call is:	CALL WGQUIT (IP, ID)
or:	int wgquit (int ip);
IP	is the index of the parent widget.
ID	is the returned widget index.

W G P B U T

The routine WGPBUT creates a push button widget.

The call is:	CALL WGPBUT (IP, CLAB, ID)
or:	int wgpbut (int ip, char *clab);
IP	is the index of the parent widget.
CLAB	is a character string that will be used as a label.
ID	is the returned widget index. It should be connected with a callback routine.

W G C M D

The routine WGCMD creates a push button widget. A corresponding system command will be executed if the button is pressed.

The call is:	CALL WGCMD (IP, CLAB, CMD, ID)
or:	int wgcmd (int ip, char *clab, char *cmd);
IP	is the index of the parent widget.
CLAB	is a character string that will be used as a label.
CMD	is a character string containing a system command.
ID	is the returned widget index. It should be connected with a callback routine.

15.2 Parameter Setting Routines

S W G W T H

The routine SWGWTH sets the default width of horizontal and parent/base widgets.

The call is:	CALL SWGWTH (NWTH)
or:	void swgwth (int nwth);

NWTH — is an integer containing a positive number of characters or a negative number between -1 and -100. If NWTH $<$ 0, the widget width is set to ABS(NWTH) * NWIDTH / 100 where NWIDTH is the screen width.

Default: NWTH = 20.

S W G D R W

The routine SWGDRW modifies the height of draw widgets.

The call is:	CALL SWGDRW (XF)
or:	void swgdrw (float xf);

XF — is a positive floatingpoint number. The height of a draw widget is set to XF * NW where NW is the widget width.

Default: XF = 1.

S W G C L R

The routine SWGCLR defines colours for widgets.

The call is:	CALL SWGCLR (XR, XG, XB, COPT)
or:	void swgclr (float xr, float xg, float xb, char *copt);

XR, XG, XB — are RGB values between 0 and 1.

COPT — is a character string that can have the values ’BACK’, ’FORE’, ’SCROLL’ and ’LTEXT’. The keywords ’BACK’ and ’FORE’ define background and foreground colours, ’SCROLL’ defines the colour of the slider in scale widgets, and ’LTEXT’ sets the background colour of the edit window in labeled text widgets.

Additional notes:
- Colours can not be defined for push button widgets. This is a restriction in the Windows API.
- Multiple draw widgets must have the same background colour since they belong to the same widget class. The same is valid for multiple main widgets created by WGINI.

S W G F N T

The routine SWGFNT defines fonts for widgets.

The call is:	CALL SWGFNT (CFNT, NPTS)
or:	void swgfnt (char *cfnt, int npts);

CFNT	is a character string containing the font. For Windows, CFNT can contain a TrueType font (see WINFNT), or one of the Windows raster fonts such as System, FixedSys, Terminal, Courier, MS Serif and MS Sans Serif. For X11, CFNT can contain an X11 font. CNFT = 'STANDARD' resets the font to the default value.
NPTS	is the font size in points (72 points = 1 inch). Note that only a few different font sizes are available for Windows raster fonts. For X11, the parameter NPTS will be ignored since the font size is already part of the font name.

S W G F O C

The routine SWGFOC sets the keyboard focus to the specified widget.

The call is:	CALL SWGFOC (ID)
or:	void swgfoc (int id);
ID	is the widget index.

S W G O P T

The routine SWGOPT sets widget options.

The call is:	CALL SWGOPT (COPT, CKEY)
or:	void swgopt (char *copt, char *ckey);
COPT	is a character string containing an option.
CKEY	is a character string containing a keyword:
= 'POSITION'	If CKEY = 'POSITION', COPT can have the values 'STANDARD' and 'CENTER'. For COPT = 'CENTER', the main widget will be centered on the screen. The default position of the main widget is the upper left corner of the screen.
= 'MASK'	If CKEY = 'MASK', COPT can have the values 'STANDARD' and 'USER'. For COPT = 'USER', the mask entry in the routines WGFIL and DWGFIL can be controlled completely by the user. For that case, the mask parameter in WGFIL and DWGFIL can have the following syntax: it contains of a pair of strings separated by a '+' sign. The first string contains the label, the second string the search filter. For example: 'Data (*.dat)+*.dat'. 'Data (*.dat)' is the label while '*.dat' the filter. Multiple pairs of strings for the mask are also possible.
= 'CALLBACK'	The behaviour of callback routines for text widgets can be modified with this keyword. COPT can have the values 'RETURN', 'CHANGE' and 'BOTH'. For 'RETURN', the callback routine is only called if a return is given in the text field, for 'CHANGE, the callback routine is called for each change in the text field. The default value is 'RETURN'.
= 'CLOSE'	This keyword changes the behaviour of the close button of the main widget. For COPT = 'QUIT', the program will be terminated. For COPT = 'OK', the main widget is deleted and the program is continued after WGFIN. Default: ('STANDARD', 'POSITION'), ('STANDARD', 'MASK'), ('RETURN', 'CALLBACK'), ('QUIT', 'CLOSE').
Additional note:	Some X11 Window managers ignore the position of the main widget.

S W G P O P

The routine SWGPOP modifies the appearance of the popup menubar.

The call is:	CALL SWGPOP (COPT)
or:	void swgpop (char *copt);
COPT	is a character string containing an option:
= 'NOOK'	suppresses the 'OK' entry in the 'EXIT' menu.
= 'NOQUIT'	suppresses the 'QUIT' entry in the 'EXIT' menu.
= 'NOHELP'	suppresses the 'HELP' button in the menubar.
= 'OK'	enables the 'OK' entry in the 'EXIT' menu (default).
= 'QUIT'	enables the 'QUIT' entry in the 'EXIT' menu (default).
= 'HELP'	enables the 'HELP' button in the menubar (default).

S W G T I T

The routine SWGTIT defines a title displayed in the main widget.

The call is:	CALL SWGTIT (CTIT)
or:	void swgtit (char *ctit);
CTIT	is a character string containing the title.

S W G H L P

The routine SWGHLP sets a character string that will be displayed if the Help menu is clicked by the user.

The call is:	CALL SWGHLP (CSTR)
or:	void swghlp (char *cstr);
CSTR	is a character string that will be displayed in the help box. The character '|' can be used as a newline character.

S W G S I Z

The routine SWGSIZ defines the size of widgets.

The call is:	CALL SWGSIZ (NW, NH)
or:	void swgsiz (int nw, int nh);
NW, NH	are the width and height of the widget in pixels.

S W G P O S

The routine SWGPOS defines the position of widgets.

The call is:	CALL SWGPOS (NX, NY)
or:	void swgpos (int nx, int ny);
NX, NY	are the upper left corner of the widget in pixels. The point is relative to the upper left corner of the parent widget.

S W G W I N

The routine SWGWIN defines the position and size of widgets.

The call is:	CALL SWGWIN (NX, NY, NW, NH)
or:	void swgwin (int nx, int ny, int nw, int nh);

NX, NY	are the upper left corner of the widget in pixels. The point is relative to the upper left corner of the parent widget.
NW, NH	are the width and height of the widget in pixels.

S W G T Y P

The routine SWGTYP modifies the appearance of certain widgets.

The call is:	CALL SWGTYP (CTYPE, CLASS)
or:	void swgtyp (char *ctype, char *class);

CTYPE	is a character string containing a keyword:
= ’VERT’	means that list elements in box widgets or scale widgets will be displayed in vertical direction.
= ’HORI’	means that list elements in box widgets or scale widgets will be displayed in horizontal direction.
= ’SCROLL’	means that scrollbars will be created in list widgets.
= ’NOSCROLL’	means that no scrollbars will be created in list widgets.
= ’VSCROLL’	means that just a vertical scrollbar is created in list widgets.
= ’AUTO’	means that scrollbars will be created in list widgets if the number of elements is greater that 8.
CLASS	is a character string containing the widget class where CLASS can have the values ’LIST’, ’BOX’ and ’SCALE’. If CLASS = ’LIST’, CTYPE can have the values ’AUTO’, ’SCROLL’ and ’NOSCROLL’. If CLASS = ’BOX’ or CLASS = ’SCALE’, CTYPE can have the values ’VERT’ and ’HORI’.

Defaults: (’VERT’, ’BOX’), (’HORI’, ’SCALE’), (’AUTO’, ’LIST’).

S W G J U S

The routine SWGJUS defines the alignment of labels in label and button widgets.

The call is:	CALL SWGJUS (CJUS, CLASS)
or:	void swgjus (char *cjus, char *class);

CJUS	is a character string defining the alignment:
= ’LEFT’	means that labels will be displayed on the left side of label and button widgets.
= ’CENTER’	means that labels will be displayed in the center of label and button widgets.
= ’RIGHT’	means that labels will be displayed on the right side of label and button widgets.
CLASS	is a character string defining the widget class. CLASS can have the values ’LABEL’ and ’BUTTON’.

Defaults: (’LEFT’, ’LABEL’), (’CENTER’, ’BUTTON’).

S W G S P C

The routine SWGSPC defines horizontal and vertical space between widgets.

The call is:	CALL SWGSPC (XSPC, YSPC)
or:	void swgspc (float xspc, float yspc);

XSPC, YSPC — are floatingpoint numbers defining the space between widgets. For non negative values, the spaces XSPC * NWCHAR and YSPC * NHCHAR are used where NWCHAR and NHCHAR are the current character width and height. For negative values, the horizontal and vertical spaces are set to ABS(XSPC) * NWIDTH / 100 and ABS (YSPC) * NHEIGHT where NWIDTH and NHEIGHT are the width and height of the screen.

Default: (4.0, 0.5).

S W G S T P

The routine SWGSTP defines a step value for scale widgets.

The call is:	CALL SWGSTP (XSTP)
or:	void swgstp (float xstp);

XSTP — is a positive floatingpoint number defining the step value. The default value is (MAX - MIN) / 100.

S W G M R G

The routine SWGMRG defines margins for widgets.

The call is:	CALL SWGMRG (IVAL, CMRG)
or:	void swgmrg (int ival, char *cmrg);

IVAL — is the margin value in pixels.

CMRG — is a character string that can have the values ’LEFT’, ’TOP’, ’RIGHT’ and ’BOTTOM’. By default, all margins are zero.

S W G M I X

The routine SWGMIX defines control characters for separating elements in list strings.

The call is:	CALL SWGMIX (CHAR, CMIX)
or:	void swgmix (char *char, char *cmix);

CHAR — is a new control character.

CMIX — is a character string that defines the function of the control character. CMIX can have the value ’SEP’.

S W G C B K

The routine SWGCBK connects a widget with a callback routine. The callback routine is called if the status of the widget is changed. Callback routines can be defined for button, pushbutton, file, list, scale, box and text widgets, and for popup menu entries.

The call is:	CALL SWGCBK (ID, ROUTINE)

or:		void swgcbk (int id, void (*routine)(int id));

ID	is a widget ID.
ROUTINE	is the name of a routine defined by the user. In Fortran, the routine must be declared as EXTERNAL.

Additional notes:
- SWGCBK is a new version of the old DISLIN routine SWGCB (ID, ROUTINE, IRAY) that is still in the library.
- See section 15.6 for examples.

S W G A T T

The routine SWGATT sets widget attributes.

The call is:	CALL SWGATT (ID, CATT, COPT)
or:	void swgatt (int id, char *catt, char *copt);

ID	is a widget ID.
CATT	is a character string containing an attribute. If COPT = ’STATUS’, CATT can have the values ’ACTIVE’, ’INACTIVE’ and ’INVISIBLE’. If COPT = ’LIST’, CATT can have new list elements for a list widget. In that case, CATT has the same meaning as the parameter CLIS in WGLIS.
COPT	is a character string that can have the values ’STATUS’ and ’LIST’.

S W G B U T

The routine SWGBUT sets the status of a button widget. If the widget is a push button widget, the connected callback routine will be executed if the status 1 is passed to SWGBUT.

The call is:	CALL SWGBUT (ID, IVAL)
or:	void swgbut (int id, int ival);

ID	is a widget ID of a button widget.
IVAL	can have the values 0 and 1.

S W G L I S

The routine SWGLIS changes the selection in a list widget.

The call is:	CALL SWGLIS (ID, ISEL)
or:	void swglis (int id, int isel);

ID	is a widget ID of a list widget.
ISEL	defines the selected element (≥ 1).

S W G B O X

The routine SWGBOX changes the selection in a box widget.

The call is:	CALL SWGBOX (ID, ISEL)
or:	void swgbox (int id, int isel);

ID	is a widget ID of a box widget.
ISEL	defines the selected element (≥ 1).

S W G T X T

The routine SWGTXT changes the value of a text widget.

The call is:	CALL SWGTXT (ID, CVAL)
ID	is a widget ID of a text widget.
CVAL	is a character string containing the new text.

S W G F I L

The routine SWGFIL changes the value of a file widget.

The call is:	CALL SWGFIL (ID, CFIL)
or:	void swgfil (int id, char *cfil);
ID	is a widget ID of a file widget.
CFIL	is a character string containing the new filename.

S W G S C L

The routine SWGSCL changes the value of a scale widget.

The call is:	CALL SWGSCL (ID, XVAL)
or:	void swgscl (int id, float xval);
ID	is a widget ID of a scale widget.
XVAL	is a floatingpoint number containing the new value of the scale widget.

15.3 Requesting Routines

Requesting routines can be used to request the current widget values selected by the user. The routines should be called after WGFIN, or in a callback routine.

G W G B U T

The routine GWGBUT returns the status of a button widget.

The call is:	CALL GWGBUT (ID, IVAL)
or:	int gwgbut (int id);
ID	is the index of the button widget.
IVAL	is the returned status where IVAL = 0 means off and IVAL = 1 means on.

G W G T X T

The routine GWGTXT returns the input of a text widget.

The call is:	CALL GWGTXT (ID, CSTR)
or:	void gwgtxt (int id, char *cstr);
ID	is the index of the text widget.
CSTR	is the returned character string that can have up to 256 characters.

G W G F I L

The routine GWGFIL returns the input of a file widget.

The call is:	CALL GWGFIL (ID, CFIL)
or:	void gwgfil (int id, char *cfil);

ID	is the index of the file widget.
CFIL	is the returned filename that can have up to 256 characters.

G W G L I S

The routine GWGLIS returns the selected element of a list widget.

The call is:	CALL GWGLIS (ID, ISEL)
or:	int gwglis (int id);

ID	is the index of the list widget.
ISEL	is the selected list element returned by GWGLIS.

G W G B O X

The routine GWGBOX returns the selected element of a box widget.

The call is:	CALL GWGBOX (ID, ISEL)
or:	int gwgbox (int id);

ID	is the index of the box widget.
ISEL	is the selected element returned by GWGBOX.

G W G S C L

The routine GWGSCL returns the value of a scale widget.

The call is:	CALL GWGSCL (ID, XVAL)
or:	float gwgscl (int id);

ID	is the index of the scale widget.
XVAL	is the returned value.

G W G A T T

The routine GWGATT returns a widget attribute.

The call is:	CALL GWGATT (ID, IATT, COPT)
or:	int gwgatt (int id, char *copt);

ID	is a widget ID.
IATT	is a returned attribute. If COPT = ’STATUS’, IATT can have the values 0 for ’ACTIVE’, 1 for ’INACTIVE’ and 2 for ’INVISIBLE’.
COPT	is a character string that can have the value ’STATUS’.

G W G X I D

The routine GWGXID returns the window ID for a specified widget ID.

The call is:	CALL GWGXID (ID, IWINID)
or:	int gwgxid (int id);

ID	is the widget ID.
IWINID	is the returned window ID.
Additional note:	For X11, the window ID of a widget can only be calculated if the widget is already realized. This means that GWGXID should be called in a callback routine and not directly behind a widget. For X11, widgets are realized in the routine WGFIN.

15.4 Utility Routines

I T M S T R

The routine ITMSTR extracts a list element from a list string.

The call is:	CALL ITMSTR (CLIS, IDX, CITEM)
or:	char *itmstr (char *clis, int idx);

CLIS	is a character string that contains the list elements (s. WGLIS).
IDX	is the index of the element that should be extracted from CLIS (beginning with 1).
CITEM	is a character string containing the extracted list element.

I T M C N T

The routine ITMCNT returns the number of elements in a list string.

The call is:	N = ITMCNT (CLIS)
or:	int itmcnt (char *clis);

CLIS	is a character string that contains the list elements (s. WGLIS).
N	is the calculated number of elements in CLIS.

I T M C A T

The routine ITMCAT concatenates an element to a list string.

The call is:	CALL ITMCAT (CLIS, CITEM)
or:	void itmcat (char *clis, char *item);

CLIS	is a character string that contains the list elements (s. WGLIS).
CITEM	is a character string that will be concatenated to CLIS. If CLIS is blank, CITEM will be the first element in CLIS.
Additional note:	Trailing blanks in CLIS and CITEM will be ignored.

M S G B O X

The routine MSGBOX displays a message in form of a dialog widget. It can be used to display messages in callback routines.

The call is: CALL MSGBOX (CSTR)

or: void msgbox (char *cstr);

CSTR is a character string containing a message.

R E A W G T

The routine REAWGT realizes a widget tree. Since the windows ID of a widget can only be calculated for X11 if the widget is already realized, this routine is useful if the windows ID of a widget is needed before WGFIN. Normally, the widget tree is realized in WGFIN.

The call is: CALL REAWGT

or: void reawgt ();

S E N D O K

The routine SENDOK has the same meaning as when the OK entry in the Exit menu is pressed. All widgets are deleted and the program is continued after WGFIN.

The call is: CALL SENDOK

or: void sendok ();

S E N D M B

The routine SENDMB sends a mouse button 2 event to the DISLIN routine DISFIN. It can be used for closing the graphics window.

The call is: CALL SENDMB

or: void sendmb ();

15.5 Dialog Routines

Dialog routines are collections of widgets that can be used to display messages, to get text strings, to get filenames from a file selection box and to get selections from a list of items. Dialog routines can be used independently from the routines WGINI and WGFIN.

D W G M S G

The routine DWGMSG displays a message.

The call is: CALL DWGMSG (CSTR)

or: void dwgmsg (char *cstr);

CSTR is a character string that will be displayed in a message box. Multiple lines can be separated by the character '|'.

D W G B U T

The routine DWGBUT displays a message that can be answered by the user with 'Yes' or 'No'.

The call is:	CALL DWGBUT (CSTR, IVAL)
or:	int dwgbut (char *cstr, int ival);

CSTR	is a character string that will be displayed in a message box. Multiple lines can be separated by the character '|'.
IVAL	is the returned answer of the user. IVAL = 1 means 'Yes', IVAL = 0 means 'No'. IVAL is also used to initialize the button.

D W G T X T

The routine DWGTXT creates a dialog widget that can be used to prompt the user for input.

The call is:	CALL DWGTXT (CLAB, CSTR)
or:	char *dwgtxt (char *clab, char *cstr);

CLAB	is a character string that will be displayed in the dialog widget as a label.
CSTR	is a character string that is used to initialize the text field. After the call to DWGTXT, CSTR returns the user input. For the C Routine, the user input is returned as function value.

D W G F I L

The routine DWGFIL creates a file selection box that can be used to get a filename.

The call is:	CALL DWGFIL (CLAB, CFIL, CMASK)
or:	char *dwgfil (char *clab, char *cfil, char *cmask);

CLAB	is a character string that will be displayed in the dialog widget.
CFIL	is the returned filename selected by the user.
CMASK	specifies the search pattern used in determining the files to be displayed in the file selection box.

D W G L I S

The routine DWGLIS creates a dialog widget that can be used to to get a selection from a list of items.

The call is:	CALL DWGLIS (CLAB, CLIS, ISEL)
or:	int dwglis (char *clab, char *clis, int isel);

CLAB	is a character string that will be displayed in the dialog widget.
CLIS	is a character string that contains the list elements. Elements must be separated by the character '|'.
ISEL	defines the pre-selected element and contains the selected element after return. Element numbering begins with the number 1.

15.6 Examples

The following short program creates some widgets and requests the values of the widgets.

```
PROGRAM EXA1
CHARACTER*80 CL1,CFIL

CL1='Item1|Item2|Item3|Item4|Item5'
CFIL=' '

CALL SWGTIT ('EXAMPLE 1')
CALL WGINI  ('VERT', IP)

CALL WGLAB  (IP, 'File Widget:', ID)
CALL WGFIL  (IP, 'Open File', CFIL, '*.c', ID_FIL)

CALL WGLAB  (IP, 'List Widget:', ID)
CALL WGLIS  (IP, CL1, 1, ID_LIS)

CALL WGLAB  (IP, 'Button Widgets:', ID)
CALL WGBUT  (IP, 'This is Button 1', 0, ID_BUT1)
CALL WGBUT  (IP, 'This is Button 2', 1, ID_BUT2)

CALL WGLAB  (IP, 'Scale Widget:', ID)
CALL WGSCL  (IP, ' ', 0., 10., 5., 1, ID_SCL)

CALL WGOK   (IP, ID_OK)
CALL WGFIN

CALL GWGFIL (ID_FIL, CFIL)
CALL GWGLIS (ID_LIS, ILIS)
CALL GWGBUT (ID_BUT1, IB1)
CALL GWGBUT (ID_BUT2, IB2)
CALL GWGSCL (ID_SCL, XSCL)
END
```

Figure 15.1: Widgets

The next example displays some widgets packed in two columns.

```
PROGRAM EXA2
CHARACTER*80 CL1,CSTR

CL1='Item1|Item2|Item3|Item4|Item5'
CSTR=' '

CALL SWGTIT ('EXAMPLE 2')
CALL WGINI  ('HORI', IP)
CALL WGBAS  (IP, 'VERT', IPL)
CALL WGBAS  (IP, 'VERT', IPR)

CALL WGLAB  (IPL, 'Text Widget:', ID)
CALL WGTXT  (IPL, CSTR, ID_TXT1)
CALL WGLAB  (IPL, 'List Widget:', ID)
CALL WGLIS  (IPL, CL1, 1, ID_LIS)
CALL WGLAB  (IPR, 'Labeled Text Widget:', ID)
CALL WGLTXT (IPR, 'Give Text:', CSTR, 40, ID_TXT2)
CALL WGLAB  (IPR, 'Box Widget:', ID)
CALL WGBOX  (IPR, CL1, 1, ID_BOX)

CALL WGQUIT (IPL, ID_OK)
CALL WGOK   (IPL, ID_OK)
CALL WGFIN
END
```

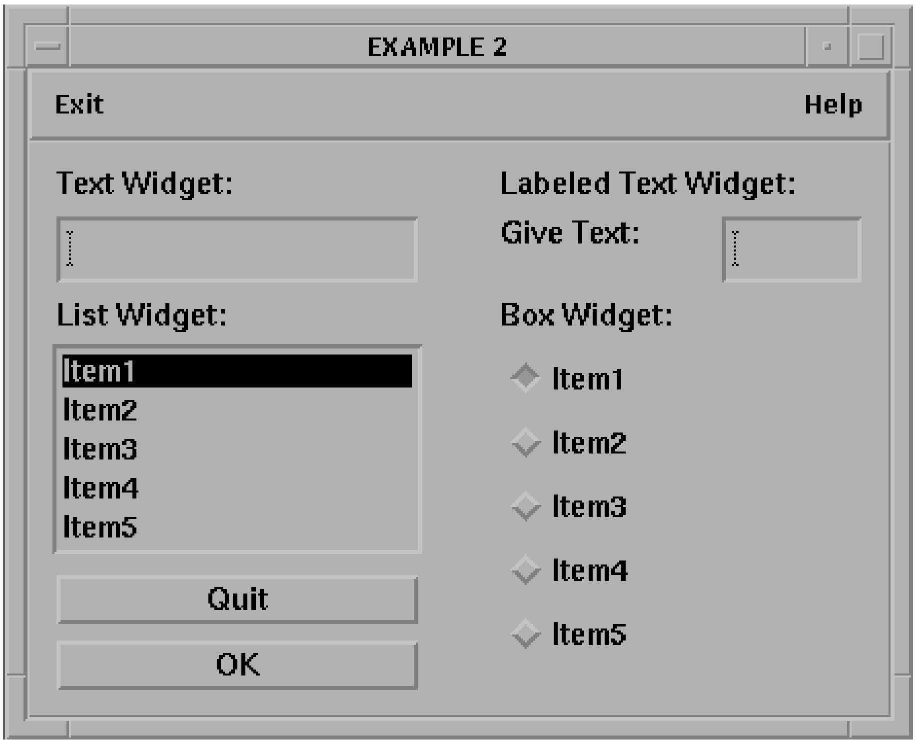


Figure 15.2: Widgets

The following example explains the use of callback routines. A list widget is created and the selected list element is displayed in a text widget.

```
      PROGRAM EXA3
      COMMON /MYCOM1/ ID_LIS,ID_TXT
      COMMON /MYCOM2/ CLIS
      CHARACTER*80 CLIS
      EXTERNAL MYSUB

      CLIS = 'Item 1|Item 2|Item 3|Item 4|Item 5'

      CALL WGINI  ('VERT', IP)
      CALL WGLIS  (IP, CLIS, 1, ID_LIS)
      CALL SWGCBK (ID_LIS, MYSUB)
      CALL WGTXT  (IP, ' ', ID_TXT)
      CALL WGFIN
      END

      SUBROUTINE MYSUB (ID)
C     ID   is the widget ID of WGLIS ( = ID_LIS)

      COMMON /MYCOM1/ ID_LIS,ID_TXT
      COMMON /MYCOM2/ CLIS
      CHARACTER*80 CLIS, CITEM

      CALL GWGLIS (ID_LIS, ISEL)
      CALL ITMSTR (CLIS, ISEL, CITEM)
      CALL SWGTXT (ID_TXT, CITEM)
      END
```

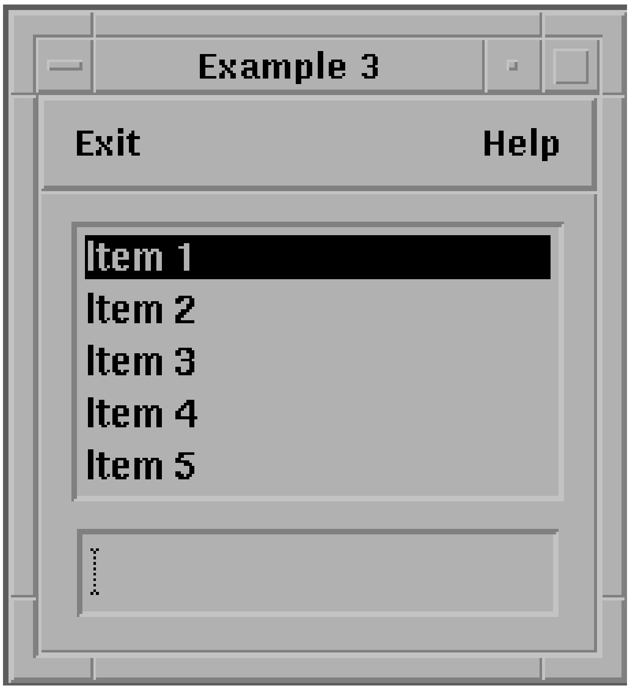


Figure 15.3: Widgets

The C coding of example 3 is given below:

```
#include <stdio.h>
#include "dislin.h"

void mysub (int ip);

static int id_lis, id_txt;
static char clis[] = "Item 1|Item 2|Item 3|Item 4|Item 5";

main()
{ int  ip;

  swgtit ("Example 3");

  ip      = wgini  ("VERT");
  id_lis  = wglis (ip, clis, 1);
  swgcbk (id_lis, mysub);

  id_txt = wgtxt (ip, " ");
  wgfin ();
}

void mysub (int id)
{ int isel;
  char *citem;

  isel = gwglis (id_lis);
  citem = itmstr (clis, isel);
  swgtxt (id_txt, citem);
}
```

Chapter 16

Quickplots

This chapter presents some quickplots that are collections of DISLIN routines for displaying data with one statement. Axis scaling is done automatically by the quickplots. By default, graphical output is sent to the screen.

16.1 Plotting Curves

Q P L O T

QPLOT connects data points with lines.

The call is:	CALL QPLOT (XRAY, YRAY, N)	level 0, 1
or:	void qplot (float *xray, float *yray, int n);	

XRAY, YRAY	are arrays that contain X- and Y-coordinates.
N	is the number of data points.

16.2 Scatter Plots

Q P L S C A

QPLSCA marks data points with symbols.

The call is:	CALL QPLSCA (XRAY, YRAY, N)	level 0, 1
or:	void qplsca (float *xray, float *yray, int n);	

XRAY, YRAY	are arrays that contain X- and Y-coordinates.
N	is the number of data points.

16.3 Bar Graphs

Q P L B A R

QPLBAR plots a bar graph.

The call is:	CALL QPLBAR (XRAY, N)	level 0, 1
or:	void qplbar (float *xray, int n);	

XRAY	is an array containing data points.
N	is the number of data points.

16.4 Pie Charts

Q P L P I E

QPLPIE plots a pie chart.

The call is:	CALL QPLPIE (XRAY, N)	level 0, 1
or:	void qppie (float *xray, int n);	

XRAY	is an array containing data points.
N	is the number of data points.

16.5 3-D Colour Plots

Q P L C L R

QPLCLR makes a 3-D colour plot of a matrix.

The call is:	CALL QPLCLR (ZMAT, IXDIM, IYDIM)	level 0, 1
or:	void qplclr (float *zmat, int ixdim, int iydim);	

ZMAT	is a matrix with the dimension (IXDIM, IYDIM) containing the function values.
IXDIM, IYDIM	are the dimensions of ZMAT.

16.6 Surface Plots

Q P L S U R

QPLSUR makes a surface plot of a matrix.

The call is:	CALL QPLSUR (ZMAT, IXDIM, IYDIM)	level 0, 1
or:	void qplsur (float *zmat, int ixdim, int iydim);	

ZMAT	is a matrix with the dimension (IXDIM, IYDIM) containing the function values.
IXDIM, IYDIM	are the dimensions of ZMAT.

16.7 Contour Plots

Q P L C O N

QPLCON makes a contour plot of a matrix.

The call is:	CALL QPLCON (ZMAT, IXDIM, IYDIM, NLV)	level 0, 1
or:	void qplcon (float *zmat, int ixdim, int iydim, int nlv);	

ZMAT	is a matrix with the dimension (IXDIM, IYDIM) containing the function values.
IXDIM, IYDIM	are the dimensions of ZMAT.
NLV	is the number of contour levels that should be generated.

16.8 Setting Parameters for Quickplots

Quickplots can be called in level 0 and in level 1 of DISLIN. If they are called in level 0, the statements CALL METAFL ('CONS') and CALL DISINI are executed by quickplots. If they are called in level 1, these statements will be suppressed. This means that programs can change the output device of quickplots and define axis names and titles if they call quickplots in level 1 after a call to DISINI.

The following example defines axis names and a title for QPLOT:

```
CALL METAFL ('CONS')
CALL DISINI

CALL NAME ('X-axis', 'X')
CALL NAME ('Y-axis', 'Y')
CALL TITLIN ('This is a Title', 2)
CALL QPLOT (XRAY, YRAY, N)
END
```

Appendix A

Using DISLIN from Interpreting Languages

The most DISLIN distributions contain plotting extensions for the interpreting languages Perl, Python and Java. This appendix gives a short description of how DISLIN can be called from this languages. For a complete description, the user is referred to the Perl, Python, Java and DISGCL manuals of DISLIN.

A.1 The DISLIN Interpreter DISGCL

The DISLIN utility program DISGCL is an interpreter for DISLIN. All DISLIN statements can be written to a script file and then be executed with DISGCL, or can be entered in an interactive mode.

Similar to programming languages such as Fortran and C, high-level language elements can be used within DISGCL. These are variables, operators, expressions, array operations, loops, if and switch statements, user-defined subroutines and functions, and file I/O routines.

An easy to use interface for data input is given to include data into DISGCL jobs. The format of data files is very simple and useful for most DISLIN plotting routines.

Several quickplots are offered by DISGCL which are collections of DISLIN statements to display data with one command.

The DISGCL command has the following syntax:

Command:	DISGCL [filename[.gcl]] [args] [options]
filename	is the name of a DISGCL script file. The extension '.gcl' is optional.
args	are optional arguments that can be passed to DISGCL scripts (see DISGCL).
options	is an optional field of keywords separated by blanks (see DISGCL).

Notes:
- If no parameters are specified, DISGCL runs in interactive mode.
- DISGCL searches the current working directory for the DISGCL script file. If the search fails, DISGCL searches the directory defined by the environment variable GCL_PATH.
- On UNIX systems, an DISGCL script file can be executed directly if the following line is included at the beginning of the script file:

  #! /path/disgcl -f

  where path is the directory containing the disgcl executable.

DISGCL script files must have the following syntax:

- A DISGCL script file must begin with the indentifier ’%GCL’.
- Each line may contain up to 132 characters.
- The current statement can be continued on the next line if a masterspace (@) is used at the end of the line.
- Lines are allowed to carry trailing comment fields, following a double slash (//) or the ’#’ character. Empty lines are also be interpreted as comment lines.
- Keywords and routine names can be in upper and lowercase letters.
- String constants must be enclosed in a pair of either apostrophes or quotation marks.

Here is the example C.1 of appendix C coded as GCL script file:

```
%GCL
// Demonstration of CURVE

N=101
PI    = 3.1415926

XRAY  = FALLOC (N)
XRAY  = (XRAY - 1.) * 3.6
YRAY1 = SIN (XRAY * PI / 180.)
YRAY2 = COS (XRAY * PI / 180.)

METAFL (’CONS’)
DISINI ()
COMPLX ()
PAGERA ()

AXSPOS (450, 1800)
AXSLEN (2200, 1200)

NAME   (’X-axis’, ’X’)
NAME   (’Y-axis’, ’Y’)
TITLIN (’Demonstration of CURVE’, 1)
TITLIN (’SIN(X), COS(X)’, 3)
TICKS  (10, ’X’)
LABDIG (-1, ’X’)

GRAF   (0.,360.,0.,90.,-1.,1.,-1.,0.5)
TITLE  ()

COLOR  (’RED’)
CURVE  (xray, yray1, n)
COLOR  (’GREEN’)
CURVE  (xray, yray2, n)

COLOR (’FORE’)
DASH   ()
XAXGIT ()
DISFIN ()
```

A.2 Using DISLIN from Perl

The Practical Extraction and Report Language is supported by DISLIN. Pre-compiled DISLIN modules for Perl are available for the most operating systems.

For passing parameters from Perl to DISLIN, the following rules are applied:

- Parameters can be passed from Perl to DISLIN routines as variables, constants and expressions.
- String constants must be enclosed in a pair of either apostrophes or quotation marks.
- Floatingpoint parameters can be passed from Perl as integer and floatingpoint numbers.
- Arrays can be passed from Perl to DISLIN with the starting characters '\ @'.

Note: Normally, the number and meaning of parameters passed to DISLIN routines are identical with the syntax description of the routines in the DISLIN manual. DISLIN routines that return one scalar are implemented for Perl as functions. A description of all DISLIN routines that can be called from Perl is presented in the DISLIN manual for Perl.

Here is the example C.1 of appendix C in Perl coding:

```
#!/usr/bin/perl
use Dislin;

$n = 101;
$pi = 3.1415926;
$f = $pi / 180.;
$step = 360. / ($n - 1);
for ($i = 0; $i < $n; $i++) {
    $xray[$i] = $i * $step;
    $x = $xray[$i] * $f;
    $y1ray[$i] = sin ($x);
    $y2ray[$i] = cos ($x);
}

Dislin::metafl ('xwin');
Dislin::disini ();
Dislin::complx ();
Dislin::pagera ();

Dislin::axspos (450, 1800);
Dislin::axslen (2200, 1200);

Dislin::name   ('X-axis', 'X');
Dislin::name   ('Y-axis', 'Y');

Dislin::labdig (-1, 'X');
Dislin::ticks  (10, 'XY');

Dislin::titlin ('Demonstration of CURVE', 1);
Dislin::titlin ('SIN (X), COS (X)', 3);

Dislin::graf   (0., 360., 0., 90., -1., 1., -1., 0.5);
```

```
Dislin::title  ();

Dislin::color  ('red');
Dislin::curve  (\@xray, \@y1ray, $n);
Dislin::color  ('green');
Dislin::curve  (\@xray, \@y2ray, $n);

Dislin::color  ('foreground');
Dislin::dash   ();
Dislin::xaxgit ();
Dislin::disfin ();
```

A.3 Using DISLIN from Python

The programming language is also a popular interpreting language that is supported by DISLIN. The passing of parameters from Python to DISLIN routines is not so strict as in other programming languages. The following rules are applied:

- Parameters can be passed from Python to DISLIN routines as variables, constants and expressions.
- String constants must be enclosed in a pair of either apostrophes or quotation marks.
- Floatingpoint parameters can be passed from Python as integer and floatingpoint numbers.
- Integer parameters can be passed from Python as integer and floatingpoint numbers. If a floatingpoint number is passed for an integer parameter, the fractional part of the floatingpoint number will be truncated.
- Floatingpoint arrays can be passed from Python as floatingpoint and integer lists. They were copied to 32 bit C arrays before they are passed to DISLIN routines.
- Integer arrays must be passed as integer lists.
- Memory must be allocated for Arrays that are used from DISLIN routines as output parameters. For example, they can be created with the Python command 'range'.

Note: Normally, the number and meaning of parameters passed to DISLIN routines are identical with the syntax description of the routines in the DISLIN manual. DISLIN routines that return one ore more scalars are implemented for Python as functions that return a tuple of scalars. For example, the statement 'nw,nh = getpag ()' returns the page size.

The example C.1 of appendix C has in Python the following coding:

```
#! /usr/bin/env python
import math
import dislin

n = 101
f = 3.1415926 / 180.
x = range (n)
y1 = range (n)
y2 = range (n)
for i in range (0,n):
  x[i] = i * 3.6
  v = i * 3.6 * f
```

```
  y1[i] = math.sin (v)
  y2[i] = math.cos (v)

dislin.metafl ('xwin')
dislin.disini ()
dislin.complx ()
dislin.pagera ()

dislin.axspos (450, 1800)
dislin.axslen (2200, 1200)

dislin.name   ('X-axis', 'X')
dislin.name   ('Y-axis', 'Y')

dislin.labdig (-1, 'X')
dislin.ticks  (10, 'XY')

dislin.titlin ('Demonstration of CURVE', 1)
dislin.titlin ('SIN (X), COS (X)', 3)

dislin.graf   (0., 360., 0., 90., -1., 1., -1., 0.5)
dislin.title  ()

dislin.color  ('red')
dislin.curve  (x, y1, n)
dislin.color  ('green')
dislin.curve  (x, y2, n)

dislin.color  ('foreground')
dislin.dash   ()
dislin.xaxgit ()
dislin.disfin ()
```

A.4 Using DISLIN from Java

Pre-compiled interfaces for calling DISLIN from Java are available for the most operating systems. The following rules are applied for calling DISLIN routines from Java:

- Parameters can be passed from Java to DISLIN routines as variables, constants and expressions.
- String constants must be enclosed in a pair quotation marks.
- Floatingpoint parameters must be passed as float variables, constants and expressions. Floatingpoint constants are specified with an appending f or F.
- Integer parameters must be of type int.
- Two-dimensional arrays must be passed as one-dimensional arrays from Java to DISLIN. For example, if you have the two-dimensional array XMAT[N][M] in Java, you have to pass the one-dimensional array XRAY[N*M] to DISLIN where XRAY[i*M+j] corresponds to XMAT[i][j].
- The number and meaning of parameters passed to DISLIN routines are identical with the syntax description of the routines in the DISLIN manual except for routines that change parameters.

These routines are implemented in Java as functions with a return value. For example, the function getpag (&nw, &nh) returns in DISLIN the page width. In Java, this routine is implemented as nw = getpag (1) and nh = getpag (2).

Example C.1 of appendix C coded in Java has the following form:

```
import de.dislin.Dislin;
public class curve {
  public static void main (String args []) {
     int n = 100, i;
     double x, fpi = 3.1415926/180., step = 360. / (n-1);

     float xray []  = new float [n];
     float y1ray [] = new float [n];
     float y2ray [] = new float [n];

     for (i = 0; i < n; i++) {
        xray[i] = (float) (i * step);
        x = xray[i] * fpi;
        y1ray[i] = (float) Math.sin (x);
        y2ray[i] = (float) Math.cos (x);
     }

     Dislin.metafl ("cons");
     Dislin.disini ();
     Dislin.pagera ();
     Dislin.complx ();

     Dislin.axspos (450, 1800);
     Dislin.axslen (2200, 1200);
     Dislin.name   ("X-axis", "x");
     Dislin.name   ("Y-axis",  "y");

     Dislin.labdig (-1, "x");
     Dislin.ticks  (10, "xy");
     Dislin.titlin ("Demonstration of CURVE", 1);
     Dislin.titlin ("SIN(X), COS(X)", 3);

     Dislin.graf   (0.f, 360.f, 0.f, 90.f,
                      -1.f, 1.f, -1.f, 0.5f);
     Dislin.title  ();

     Dislin.color  ("red");
     Dislin.curve  (xray, y1ray, n);
     Dislin.color  ("green");
     Dislin.curve  (xray, y2ray, n);

     Dislin.color  ("fore");
     Dislin.dash   ();
     Dislin.xaxgit ();
     Dislin.disfin ();
  }
}
```

Appendix B

Short Description of Routines

Initialization and Introductory Routines

CGMBGD	defines the background colour for CGM files.
CGMPIC	sets the picture ID for CGM files.
DISINI	initializes DISLIN.
ERASE	clears the screen.
ERRDEV	defines the error device.
ERRFIL	sets the name of the error file.
ERRMOD	modifies the printing of error messages.
FILBOX	defines the position and size of included metafiles.
GIFMOD	enables transparency for GIF files.
HWORIG	defines the origin of the PostScript hardware page.
HWPAGE	defines the size of the PostScript hardware page.
IMGFMT	defines the format of image files.
INCFIL	includes GKSLIN, CGM and BMP files into a graphics.
METAFL	defines the plotfile format.
NEWPAG	creates a new page.
ORIGIN	defines the origin.
PAGE	sets the page size.
PAGERA	plots a page border.
PAGFLL	fills the page with a colour.
PAGHDR	plots a page header.
PAGMOD	selects a page rotation.
PAGORG	defines the origin of the page.
PDFBUF	copies a PDF file to a buffer.
PDFMOD	defines compression mode for PDF files.
PDFMRK	defines bookmarks for PDF files.
PNGMOD	enables transparency for PNG files.
SCLFAC	defines a scaling factor for the entire plot.
SCLMOD	defines a scaling mode.
SCRMOD	swaps back- and foreground colours.
SETFIL	sets the plotfile name.
SETPAG	selects a predefined page format.
SETXID	defines an external X window or pixmap.
SYMFIL	sends a plotfile to a device.
TIFMOD	defines the physical resolution of TIFF files.
UNITS	defines the plot units.
WMFMOD	modifies the format of WMF files.

Termination and Parameter Resetting

DISFIN	terminates DISLIN.
ENDGRF	terminates an axis system and sets the level to 1.
RESET	resets parameters to default values.

Plotting Text and Numbers

ANGLE	defines the character angle.
CHAANG	defines an inclination angle for characters.
CHASPC	affects character spacing.
CHAWTH	affects the width of characters.
FIXSPC	sets a constant character width.
FRMESS	defines the thickness of text frames.
HEIGHT	defines the character height.
MESSAG	plots text.
MIXALF	enables control signs in character strings for plotting indices and exponents.
NEWMIX	defines an alternate set of control characters for plotting indices and exponents.
NLMESS	returns the length of character strings in plot coordinates.
NUMBER	plots floating-point numbers.
NUMFMT	determines the format of numbers.
NUMODE	modifies the appearance of numbers.
RLMESS	plots text where the position is specified in user coordinates.
RLNUMB	plots numbers where the position is specified in user coordinates.
SETBAS	determines the position of indices and exponents.
SETEXP	determines the character height of indices and exponents.
SETMIX	defines global control signs for plotting indices and exponents.
TEXMOD	enables TeX mode for plotting mathematical formulas.
TEXOPT	defines TeX options.
TEXVAL	modifies the character height of indices and exponents in TeX mode.
TXTJUS	defines the alignment of text and numbers.

Colours

COLOR	defines the colour used for text and lines.
HSVRGB	converts HSV to RGB coordinates.
INDRGB	calculates an colour index.
INTRGB	calculates an explicit colour value.
MYVLT	changes the current colour table.
RGBHSV	converts RGB to HSV coordinates.
SETCLR	defines colours.
SETIND	changes the current colour table.
SETRGB	defines colours.
SETVLT	selects a colour table.
VLTFIL	stores or loads a colour table.

Fonts

BASALF	defines the base alphabet.
BMPFNT	defines a bitmap font.
CHACOD	defines the character coding.
COMPLX	sets a complex font.
DUPLX	sets a double-stroke font.
DISALF	sets the default font.
EUSHFT	defines a shift character for special European characters.
GOTHIC	sets a gothic font.
HELVE	sets a shaded font.
HELVES	sets a shaded font with small characters.
HWFONT	sets a standard hardware font.
PSFONT	sets a PostScript font.
PSMODE	enables Greek and Italic PostScript characters.
SERIF	sets a complex shaded font.
SIMPLX	sets a single-stroke font.
SMXALF	defines shift characters for alternate alphabets.
TRIPLX	sets a triple-stroke font.
WINFNT	sets a TrueType font for screen output on Windows.
X11FNT	sets an X11 font for screen output on X11 systems.

Symbols

HSYMBL	defines the height of symbols.
MYSYMB	defines an user-defined symbol.
RLSYMB	plots symbols where the centre is specified in user coordinates.
SYMBOL	plots symbols.
SYMROT	defines a rotation angle for symbols.

Axis Systems

ADDLAB	plots additional single labels.
AX2GRF	suppresses the plotting of the upper X- and the left Y-axis.
AX3LEN	defines axis lengths for a coloured 3-D axis system.
AXGIT	plots the lines X = 0 and Y = 0.
AXSBGD	defines the background colour.
AXSLEN	defines axis lengths for a 2-D axis system.
AXSORG	determines the position of crossed axis systems.
AXSPOS	determines the position of axis systems.
AXSTYP	selects rectangular or crossed axis systems.
BOX2D	plots a border around an axis system.
CENTER	centres axis systems.
CROSS	plots the lines X = 0 and Y = 0 and marks them with ticks.
ENDGRF	terminates an axis system.
FRMCLR	defines the colour of frames.
FRAME	defines the frame thickness of axis systems.
GRACE	affects the clipping margin of axis systems.
GRAF	plots a two-dimensional axis system.
GRAF3	plots an axis system for colour graphics.
GRDPOL	plos a ploar grid.
GRID	overlays a grid on an axis system.
NOCLIP	suppresses clipping of user coordinates.
NOGRAF	suppresses the plotting of an axis system.

POLAR	plots a polar axis system.
POLMOD	modifies the appearance of polar labels.
SETGRF	suppresses parts of an axis system.
SETSCL	sets automatic scaling.
TITLE	plots a title over an axis system.
XAXGIT	plots the line Y = 0.
XCROSS	plots the line Y = 0 and marks it with ticks.
YAXGIT	plots the line X = 0.
YCROSS	plots the line X = 0 and marks it with ticks.

Secondary Axes

XAXIS	plots a linear X-axis.
XAXLG	plots a logarithmic X-axis.
YAXIS	plots a linear Y-axis.
YAXLG	plots a logarithmic Y-axis.
ZAXIS	plots a linearly scaled colour bar.
ZAXLG	plots a logarithmically scaled colour bar.

Modification of Axes

AXCLRS	defines colours for axis elements.
AXENDS	suppresses certain labels.
AXSSCL	defines the axis scaling.
HNAME	defines the character height of axis names.
INTAX	defines integer numbering for all axes.
LABDIG	sets the number of decimal places for labels.
LABDIS	sets the distance between labels and ticks.
LABELS	selects labels.
LABJUS	defines the alignment of axis labels.
LABMOD	modifies date labels.
LABPOS	determines the position of labels.
LABTYP	defines vertical or horizontal labels.
LOGTIC	modifies the appearance of logarithmic ticks.
MYLAB	sets user-defined labels.
NAMDIS	sets the distance between axis names and labels.
NAME	defines axis titles.
NAMJUS	defines the alignment of axis titles.
NOLINE	suppresses the plotting of axis lines.
RGTLAB	right-justifies labels.
RVYNAM	defines an angle for Y-axis names.
TICKS	sets the number of ticks.
TICLEN	sets the length of ticks.
TICMOD	modifies the plotting of ticks at calendar axes.
TICPOS	determines the position of ticks.
TIMOPT	modifies time labels.

Axis System Titles

HTITLE	defines the character height of titles.
LFTTIT	left-justifies title lines.
LINESP	defines line spacing.
TITJUS	defines the alignment of titles.
TITLE	plots axis system titles.
TITLIN	defines text lines for titles.
TITPOS	defines the position of titles.
VKYTIT	shifts titles in the vertical direction.

Plotting Data Points

BARS	plots a bar graph.
BARS3D	plots 3-D bars.
CHNATT	changes curve attributes.
CHNCRV	defines attributes changed automatically by CURVE.
CRVMAT	plots a coloured surface.
CRVTRI	plots a coloured surface from triangulated data.
CURVE	plots curves.
CURVE3	plots coloured rectangles.
CURVX3	plots rows of coloured rectangles.
CURVY3	plots columns of coloured rectangles.
ERRBAR	plots error bars.
FIELD	plots a vector field.
GAPCRV	defines gaps plotted by CURVE.
INCCRV	defines the number of curves plotted with equal attributes.
INCMRK	selects symbols or lines for CURVE.
MARKER	sets the symbols plotted by CURVE.
NOCHEK	suppresses listing of data points that lie outside of the axis scaling.
PIEGRF	plots a pie chart.
POLCRV	defines the interpolation method used by CURVE.
RESATT	resets curve attributes.
SETRES	sets the size of coloured rectangles.
SHDCRV	plots shaded areas between curves.
SPLMOD	modifies spline interpolation.
THKCRV	defines the thickness of curves.

Legends

FRAME	sets the frame thickness of legends.
LEGEND	plots legends.
LEGINI	initializes legends.
LEGLIN	defines text for legend lines.
LEGOPT	modifies the appearance of legends.
LEGPAT	stores curve attributes.
LEGPOS	determines the position of legends.
LEGTIT	defines the legend title.
LINESP	affects line spacing.
MIXLEG	enables multiple text lines in legends.
NXLEGN	returns the width of legends in plot coordinates.
NYLEGN	returns the height of legends in plot coordinates.

Line Styles and Shading Patterns

CHNDOT	sets a dotted-dashed line style.
CHNDSH	sets a dashed-dotted line style.
COLOR	sets a colour.
DASH	sets a dashed line style.
DASHL	sets a long-dashed line style.
DASHM	sets a medium-dashed line style.
DOT	sets a dotted line style.
DOTL	sets a long-dotted line style.
LINTYP	defines a line style.
LINWID	sets the line width.

LNCAP	sets the line cap parameter.
LNJOIN	sets the line join parameter.
LNMLT	sets the miter limit parameter.
MYLINE	sets a user-defined line style.
MYPAT	defines a global shading pattern.
PENWID	sets the pen width.
SHDPAT	selects a shading pattern.
SOLID	sets a solid line style.

Cycles

CLRCYC	modifies the colour cycle.
LINCYC	modifies the line style cycle.
PATCYC	modifies the pattern cycle.

Base Transformations

TRFRES	resets base transformations.
TRFROT	affects the rotation of plot vectors.
TRFSCL	affects the scaling of plot vectors.
TRFSHF	affects the shifting of plot vectors.

Shielding

SHIELD	defines automatic shielding.
SHLCIR	defines circles as shielded areas.
SHLDEL	deletes shielded areas.
SHLELL	defines ellipses as shielded areas.
SHLIND	returns the index of a shielded area.
SHLPIE	defines pie segments as shielded areas.
SHLPOL	defines polygons as shielded areas.
SHLRCT	defines rotated rectangles as shielded areas.
SHLREC	defines rectangles as shielded areas.
SHLRES	deletes shielded areas.
SHLVIS	enables or disables shielded areas.

Parameter Requesting Routines

GETALF	returns the base alphabet.
GETANG	returns the current angle used for text and numbers.
GETCLP	returns the currents clipping window.
GETCLR	returns the current colour number.
GETDIG	returns the number of decimal places used in labels.
GETDSP	returns the terminal type.
GETFIL	returns the current plotfile name.
GETGRF	returns the scaling of the current axis system.
GETHGT	returns the current character height.
GETHNM	returns the character height of axis titles.
GETIND	returns the RGB coordinates for a colour index.
GETLAB	returns the current labels.
GETLEN	returns the current axis lengths.
GETLEV	returns the current level.
GETLIN	returns the current line width.
GETMFL	returns the current file format.

GETMIX	returns shift characters defined for indices and exponents.
GETOR	returns the current origin.
GETPAG	returns the current page size.
GETPAT	returns the current shading pattern.
GETPLV	returns the patchlevel of the current DISLIN library.
GETPOS	returns the position of the axis system.
GETRAN	returns the range of colour bars.
GETRES	returns the size of points used in 3-D colour graphics.
GETRGB	returns the RGB coordinates of the current colour.
GETSCL	returns the current axis scaling.
GETSCR	returns the screen size in pixels.
GETSHF	returns the control character used for European characters.
GETSP1	returns the distance between axis ticks and labels.
GETSP2	returns the distance between axis labels and names.
GETSYM	returns the current symbol number and height.
GETTCL	returns the current tick lengths.
GETTIC	returns the number of ticks plotted between labels.
GETTYP	returns the current line style.
GETUNI	returns the current unit used for messages.
GETVER	returns the version number of the currently used DISLIN library.
GETVK	returns the current lengths used for shifting.
GETVLT	returns the current colour table.
GETWID	returns the width of colour bars.
GETWIN	returns the position and size of the graphics window.
GETXID	returns the X window ID.
GMXALF	returns shift characters for alphabets.

Elementary Plot Routines

ARCELL	plots elliptical arcs.
AREAF	plots polygons.
CIRCLE	plots circles.
CONNPT	plots a line to a point.
ELLIPS	plots ellipses.
LINE	plots lines.
NOARLN	suppresses the outline of geometric figures.
PIE	plots pie segments.
POINT	plots coloured rectangles where the position is defined by the centre point.
RECFLL	plots coloured rectangles.
RECTAN	plots rectangles.
RNDREC	plots a rectangle with rounded corners.
RLARC	plots elliptical arcs for user coordinates.
RLAREA	plots polygons for user coordinates.
RLCIRC	plots circles for user coordinates.
RLCONN	plots a line to a point (user coordinates).
RLELL	plots ellipses for user coordinates.
RLINE	plots lines for user coordinates.
RLPIE	plots pie segments for user coordinates.
RLPOIN	plots coloured rectangles for user coordinates.
RLREC	plots rectangles for user coordinates.
RLRND	plots for user coordinates a rectangle with rounded corners.
RLSEC	plots coloured pie sectors for user coordinates.

RLSTRT	moves the pen to a point (user coordinates).
RLVEC	plots vectors for user coordinates.
RLWIND	plots wind speed symbols for user coordinates.
SECTOR	plots coloured pie sectors.
STRTPT	moves the pen to a point.
TRIFLL	plots filled triangles.
VECTOR	plots vectors.
WINDBR	plots wind speed symbols.
XMOVE	moves the pen to a point.
XDRAW	plots a line to a point.

Conversion of Coordinates

COLRAY	converts Z-coordinates to colour numbers.
NXPOSN	converts X-coordinates to plot coordinates.
NYPOSN	converts Y-coordinates to plot coordinates.
NZPOSN	converts Z-coordinates to colour numbers.
TRFCO1	converts one-dimensional coordinates.
TRFCO2	converts two-dimensional coordinates.
TRFCO3	converts three-dimensional coordinates.
TRFREL	converts X- and Y-coordinates to plot coordinates.
XINVRS	converts X plot coordinates to user coordinates.
XPOSN	converts X-coordinates to real plot coordinates.
YINVRS	converts Y plot coordinates to user coordinates.
YPOSN	converts Y-coordinates to real plot coordinates.

Utility Routines

BEZIER	calculates a Bezier interpolation.
BITSI2	allows bit manipulation on 16 bit variables.
BITSI4	allows bit manipulation on 32 bit variables.
CIRC3P	calculates a circle specified by three points.
FCHA	converts floating-point numbers to character strings.
FLEN	calculates the number of digits for floating-point numbers.
HISTOG	calculates a histogram.
INTCHA	converts integers to character strings.
INTLEN	calculates the number of digits for integers.
NLMESS	returns the length of character strings in plot coordinates.
NLNUMB	returns the length of numbers in plot coordinates.
SORTR1	sorts floating-point numbers.
SORTR2	sorts points in the X-direction.
SPLINE	returns splined points as calculated in CURVE.
SWAPI2	swaps the bytes of 16 bit variables.
SWAPI4	swaps the bytes of 32 bit variables.
TRFMAT	converts matrices.
TRIANG	calculates the Delaunay triangulation.
TRMLEN	calculates the number of characters in character strings.
UPSTR	converts a character string to uppercase letters.

Binary File I/O

CLOSFL	closes a file.
OPENFL	opens a file for binary I/O.
POSIFL	skips to a certain position relative to the start.
READFL	reads a given number of bytes.
SKIPFL	skips a number of bytes from the current position.
TELLFL	returns the file position.
WRITFL	writes a given number of bytes.

Date Routines

BASDAT	defines the base date.
INCDAT	calculates incremented days.
NWKDAY	returns the weekday for a date.
TRFDAT	converts incremented days to a date.

Window Routines

CLSWIN	closes a window.
OPNWIN	opens a window for graphics output.
SELWIN	selects a window for graphics output.
WINAPP	defines a window or console application.
WINDOW	defines the position and size of windows.
WINID	returns the ID of the currently selected window.
WINKEY	defines a key that can be used for program continuation in DISFIN.
WINMOD	affects the handling of windows in the termination routine DISFIN.
WINSIZ	defines the size of windows.
WINTIT	sets the title of the currently selected window.

Cursor Routines

CSRMOD	modifies the behavior of CSRPOS.
CSRMOV	returns collected cursor movements.
CSRPOS	sets and returns the cursor position.
CSRPT1	returns a pressed cursor position.
CSRPTS	returns collected cursor positions.
CSRTYP	defines the cursor type.
CSRUNI	defines the unit returned cursor routines.
SETCSR	defines the cursor type of the graphics window.

Image Routines

IMGBOX	defines a rectangle for PostScript/PDF output.
IMGCLP	defines a clipping rectangle for RBMP, RTIFF, RPNG, RGIF, and RPPM.
IMGINI	initializes transfering of image data.
IMGFIN	terminates transfering of image data.
IMGMOD	selects index or RGB mode.
IMGSIZ	defines an image size for PostScript/PDF output.
RBFPNG	stores an image as PNG file in a buffer.
RBMP	stores an image as a BMP file.
RGIF	stores an image as a GIF file.
RIMAGE	copies an image from memory to a file.
RPIXEL	reads a pixel from memory.
RPIXLS	reads image data from memory.
RPNG	stores an image as a PNG file.
RPPM	stores an image as a PPM file.
RPXROW	reads a row of image data from memory.
RTIFF	stores an image as a TIFF file.
TIFORG	defines the position of TIFF files copied with WTIFF.
TIFWIN	defines a clipping window for TIFF files copied with WTIFF.
WIMAGE	copies an image from file to memory.
WPIXEL	writes a pixel to memory.
WPIXLS	writes image data to memory.
WPXROW	write a row of image data to memory.
WTIFF	copies a TIFF file created by DISLIN to memory.

Bar Graphs

BARBOR	defines the colour of bar borders.
BARCLR	defines the colours of bars.
BARGRP	affects clustered bars.
BARMOD	selects fixed or variable bars.
BAROPT	sets parameters for 3-D bars.
BARPOS	selects predefined positions for bars.
BARS	plots bar graphs.
BARTYP	selects vertical or horizontal bars.
CHNBAR	modifies the appearance of bars.
LABCLR	defines the colour of bar labels.
LABDIG	defines the number of decimal places in bar labels.
LABELS	defines bar labels.
LABPOS	defines the position of bar labels.

Pie Charts

CHNPIE	defines colour and pattern attributes for pie segments.
LABCLR	defines the colour of segment labels.
LABDIG	defines the number of decimal places in segment labels.
LABELS	defines pie labels.
LABPOS	defines the position of segment labels.
LABTYP	modifies the appearance of segment labels.
PIEBOR	defines the colour of pie borders.
PIECLR	defines pie colours.
PIEEXP	defines exploded pie segments.
PIEGRF	plots pie charts.
PIELAB	sets additional character strings plotted in segment labels.
PIEOPT	sets parameters for 3-D pie charts.
PIETYP	selects 2-D or 3-D pie charts.
PIEVEC	modifies the arrow plotted between labels and segments.
USRPIE	is a user-defined subroutine to modify pie charts.

Coloured 3-D Graphics

AX3LEN	defines axis lengths.
COLOR	defines colours.
COLRAN	defines the range of colour bars.
CRVMAT	plots a coloured surface.
CRVTRI	plots a coloured surface from triangulated data..
CURVE3	plots coloured rectangles.
CURVX3	plots rows of coloured rectangles.
CURVY3	plots columns of coloured rectangles.
ERASE	erases the screen.
GRAF3	plots a coloured axis system.
NOBAR	suppresses the plotting of colour bars.
NOBGD	suppresses the plotting of points which have the same colour as the background.
NZPOSN	converts a Z-coordinate to a colour number.
POINT	plots coloured rectangles.
RECFLL	plots coloured rectangles.
RLPOIN	plots coloured rectangles for user coordinates where the position is defined by the centre point.
RLSEC	plots coloured pie sectors for user coordinates.

SECTOR	plots coloured pie sectors.
SETRES	defines the size of coloured rectangles.
VKXBAR	shifts colour bars in the X-direction.
VKYBAR	shifts colour bars in the Y-direction.
WIDBAR	defines the width of colour bars.
ZAXIS	plots linearly scaled colour bars.
ZAXLG	plots logarithmically scaled colour bars.

3-D Graphics

ABS3PT	converts absolute 3-D coordinates to plot coordinates.
AXIS3D	defines the lengths of the 3-D box.
BARS3D	plots 3-D bars.
BOX3D	plots a border around the 3-D box.
CONN3D	plots a line to a point in 3-D space.
CURV3D	plots curves or symbols.
FLAB3D	disables the suppression of axis labels.
GETLIT	calculates colour values.
GETMAT	calculates a function matrix from randomly distributed data points.
GRAF3D	plots an axis system.
GRFFIN	terminates a projection into 3-D space.
GRFINI	initializes projections in 3-D space.
GRID3D	plots a grid.
LABL3D	modifies the appearance of labels on the 3-D box.
LIGHT	turns lighting on or off.
LITMOD	turns single light sources on or off.
LITOP3	modifies light parameters.
LITOPT	modifies light parameters.
LITPOS	sets the position of light sources.
MATOP3	modifies material parameters.
MATOPT	modifies material parameters.
MDFMAT	modifies the algorithm used in GETMAT.
MSHCLR	defines the colour of surface meshes.
NOHIDE	disables the hidden-line algorithm.
POS3PT	converts user coordinates to absolute 3-D coordinates.
REL3PT	converts user coordinates to plot coordinates.
SHDMOD	defines flat or smooth shading for surfaces.
SHLSUR	protects surfaces from overwriting.
SPHE3D	plots a sphere.
STRT3D	moves the pen to a point.
SURCLR	selects surface colours.
SURFCE	plots the surface of a function matrix.
SURFCP	plots a shaded surface of a parametric function.
SURFUN	plots the surface of the function Z = F(X,Y).
SURISO	plots isosurfaces.
SURMAT	plots the surface of a function matrix.
SURMSH	enables grid lines.
SUROPT	suppresses surfaces lines plotted by SURFCE.
SURSHD	plots a coloured surface.
SURTRI	plots a coloured surface from triangulated data.
SURVIS	determines the visible part of surfaces.
VANG3D	defines the field of view.

VECTR3	plots vectors in 3-D space.
VFOC3D	defines the focus point.
VIEW3D	defines the viewpoint.
VUP3D	defines the camera orientation.
ZBFFIN	terminates the Z-buffer.
ZBFINI	allocates space for a Z-buffer.
ZBFLIN	plots lines.
ZBFTRI	plots triangles.
ZSCALE	defines a Z-scaling for coloured surfaces.

Geographical Projections

CURVMP	plots curves or symbols.
GRAFMP	plots a geographical axis system.
GRIDMP	plots a grid.
MAPBAS	defines a base map.
MAPFIL	defines an external map file.
MAPLEV	specifies land or lake plotting.
MAPMOD	modifies the connection of points used in CURVMP.
MAPPOL	defines the map pole used for azimuthal projections.
MAPREF	defines two latitudes used for conical projections.
POS2PT	converts user coordinates to plot coordinates.
PROJCT	selects a projection.
SETCBK	sets a callback routine for a user-defined projection.
SHDAFR	shades African countries.
SHDEUR	shades European countries.
SHDMAP	shades continents.
SHDUSA	shades USA states.
WORLD	plots coastlines and lakes.
XAXMAP	plots a secondary X-axis.
YAXMAP	plots a secondary Y-axis.

Contouring

CONCLR	defines colours fro filled contours.
CONCRV	plots generated contours.
CONFLL	plots filled contours from triangulated data.
CONGAP	affects the spacing between contour lines and labels.
CONLAB	defines a character string used for contour labels.
CONMAT	plots contours.
CONMOD	affects the position of contour labels.
CONPTS	generates contours.
CONSHD	plots shaded contours.
CONTRI	plots contours from triangulated data.
CONTUR	plots contours.
LABCLR	defines the colour of contour labels.
LABDIS	defines the distance between labels.
LABELS	defines contour labels.
SHDMOD	sets the algorithm for shaded contours.
TRIPTS	generates contours from triangulated data.

Widget Routines

DWGBUT	displays a message that can be answered with ’Yes’ or ’No’.
DWGFIL	creates a file selection box.
DWGLIS	gets a selection from a list of items.
DWGMSG	displays a message.
DWGTXT	prompts an user for input.
GWGATT	requests a widget attribute.
GWGBOX	requests the value of a box widget.
GWGBUT	requests the status of a button widget.
GWGFIL	requests the value of a file widget.
GWGLIS	requests the value of a list widget.
GWGSCL	requests the value of a scale widget.
GWGTXT	requests the value of a text widget.
GWGXID	returns the window ID for a widget.
ITMCAT	concatenates an element to a list string.
ITMCNT	calculates the number of elements in a list string.
ITMSTR	extracts an element from a list string.
MSGBOX	prints a message.
SWGATT	sets widget attributes.
SWGBOX	changes the selection of a box widget.
SWGBUT	changes the status of a button widget.
SWGCBK	connects a widget with a callback routine.
SWGCLR	defines colours for widgets.
SWGDRW	defines the height of draw widgets.
SWGFIL	changes the value of a file widget.
SWGFNT	sets fonts for widgets.
SWGFOC	sets the keyboard focus.
SWGHLP	sets a character string that will be displayed if the Help menu is clicked.
SWGJUS	defines the alignment of label widgets.
SWGLIS	changes the selection of a list widget.
SWGMIX	defines control characters.
SWGMRG	defines widget margins.
SWGOPT	sets a center option for the parent widget.
SWGPOP	modifies the appearance of the popup menubar.
SWGPOS	defines the position of widgets.
SWGSCL	changes the value of a scale widget.
SWGSIZ	defines the size of widgets.
SWGSPC	modifies the space between widgets.
SWGSTP	defines a step value for scale widgets.
SWGTIT	sets a title for the main widget.
SWGTXT	changes the value of a text widget.
SWGTYP	modifies the appearance of widgets.
SWGWIN	defines the position and size of widgets.
SWGWTH	sets the default width of widgets.
WGAPP	creates an entry in a popup menu.
WGBAS	creates a container wdiget.
WGBOX	creates a list widget where the list elements are displayed as toggle buttons.
WGBUT	creates a button widget.
WGCMD	creates a command widget.
WGDLIS	creates a dropping list widget.
WGDRAW	creates a draw widget.

WGFIL	creates a file widget.
WGFIN	terminates widget routines.
WGINI	creates a main widget and initalizes widget routines.
WGLAB	creates a label widget.
WGLIS	creates a list widget.
WGLTXT	creates a labeled text widget.
WGOK	creates a OK push button widget.
WGPBUT	creates a push button widget.
WGPOP	creates a popup menu.
WGQUIT	creates a QUIT push button widget.
WGSCL	creates a scale widget.
WGSTXT	creates a scrolled text widget.
WGTXT	creates a text widget.

Quickplots

QPLBAR	plots a bar graph.
QPLCLR	plots a colour surface of a matrix.
QPLCON	plots a contour lines of a matrix.
QPLPIE	plots a pie chart.
QPLOT	makes a curve plot.
QPLSCA	makes a scatter plot.
QPLSUR	plots a surface of a matrix.

MPS Logo

MPSLOGO	plots the MPS logo.

Appendix C

Examples

C.1 Demonstration of CURVE

```
      PROGRAM EXA_1
C     USE DISLIN        for Fortran 90!
      PARAMETER (N=301)
      DIMENSION XRAY(N),Y1RAY(N),Y2RAY(N)

      PI=3.1415926
      FPI=PI/180.
      STEP=360./(N-1)

      DO I=1,N
        XRAY(I)=(I-1)*STEP
        X=XRAY(I)*FPI
        Y1RAY(I)=SIN(X)
        Y2RAY(I)=COS(X)
      END DO

      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL AXSPOS(450,1800)
      CALL AXSLEN(2200,1200)

      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')

      CALL LABDIG(-1,'X')
      CALL TICKS(10,'XY')

      CALL TITLIN('Demonstration of CURVE',1)
      CALL TITLIN('SIN(X), COS(X)',3)

      CALL GRAF(0.,360.,0.,90.,-1.,1.,-1.,0.5)
      CALL TITLE

      CALL CURVE(XRAY,Y1RAY,N)
      CALL CURVE(XRAY,Y2RAY,N)

      CALL DASH
      CALL XAXGIT

      CALL DISFIN
      END
```

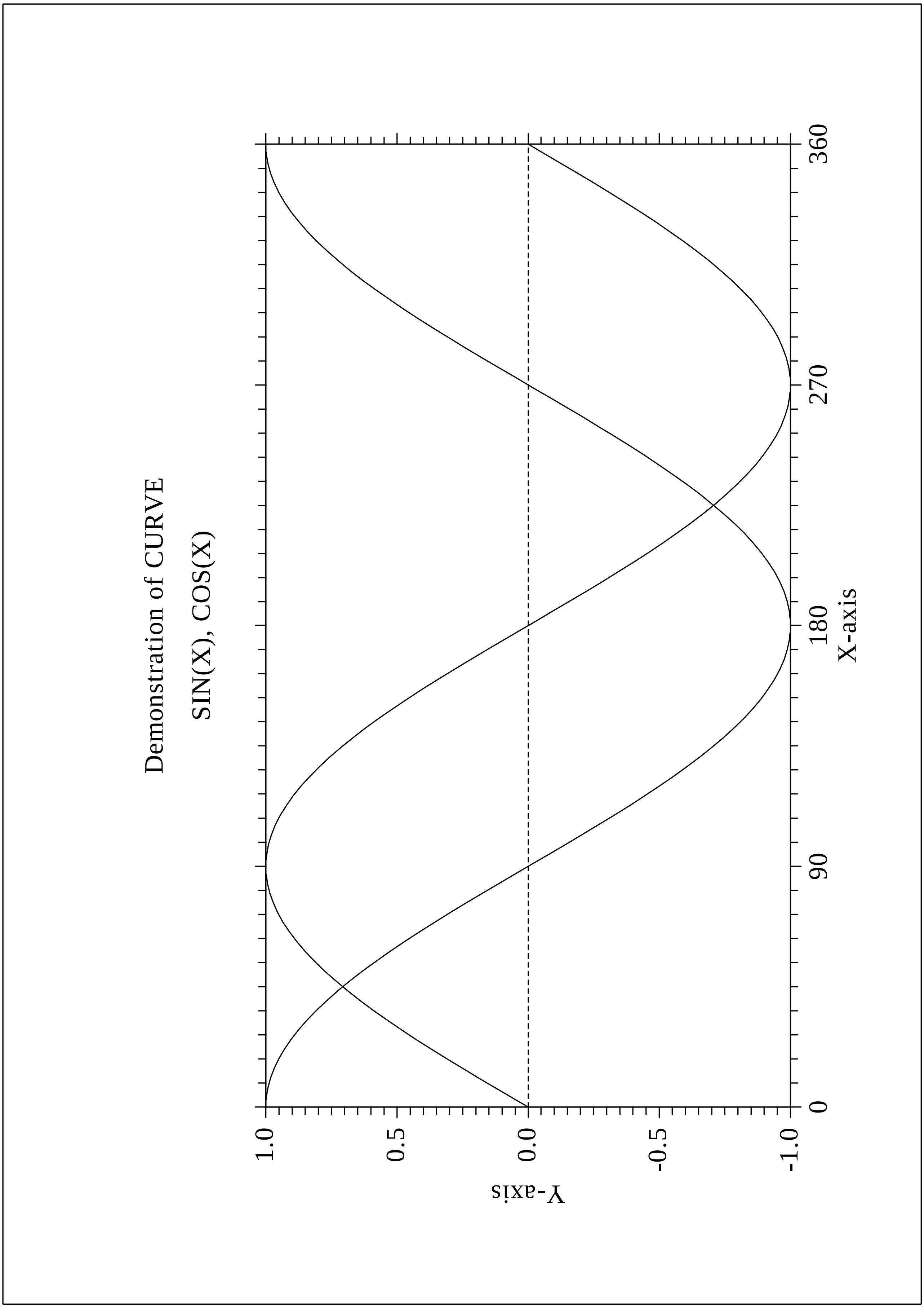


Figure B.1: Demonstration of CURVE

C.2 Polar Plots

```
      PROGRAM EXA_2
C     USE DISLIN         for Fortran 90!
      PARAMETER (N=300, M=10)
      REAL XRAY1(N),YRAY1(N),XRAY2(M),YRAY2(M)

      XPI=3.1415927
      STEP=360./(N-1)
      DO I=1,N
        A=(I-1)*STEP
        A=A*XPI/180
        YRAY1(I)=A
        XRAY1(I)=SIN(5*A)
      END DO

      DO I=1,M
        XRAY2(I)=I
        YRAY2(I)=I
      END DO

      CALL SETPAG('DA4P')
      CALL METAFL('CONS')
      CALL DISINI
      CALL PAGERA
      CALL HWFONT

      CALL TITLIN ('Polar Plots', 2)
      CALL TICKS(3,'Y')
      CALL AXENDS('NOENDS','X')
      CALL LABDIG(-1,'Y')
      CALL AXSLEN(1000,1000)
      CALL AXSORG(1050,900)

      CALL POLAR(1.,0., 0.2, 0., 30.)
      CALL CURVE(XRAY1,YRAY1,N)
      CALL HTITLE(50)
      CALL TITLE
      CALL ENDGRF

      CALL LABDIG(-1,'X')
      CALL AXSORG(1050,2250)
      CALL LABTYP('VERT','Y')
      CALL POLAR(10.,0.,2.,0.,30.)
      CALL BARWTH(-5.)
      CALL POLCRV('FBARS')
      CALL CURVE(XRAY2,YRAY2,M)
      CALL DISFIN
      END
```

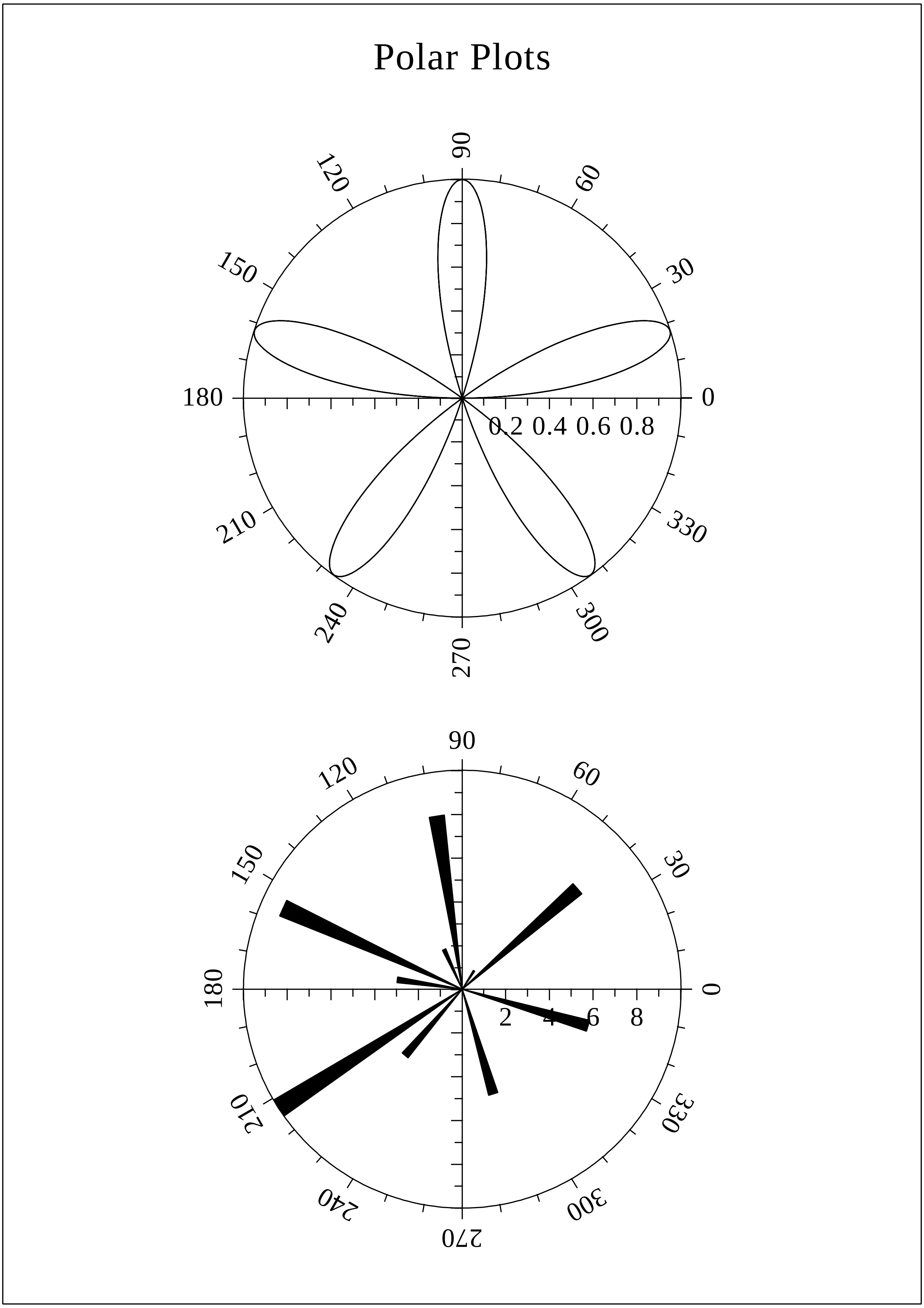


Figure B.2: Polar Plots

C.3 Symbols

```
      PROGRAM EXA_3
C     USE DISLIN         for Fortran 90!
      CHARACTER*20 CTIT,CSTR*2
      CTIT='Symbols'

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL COMPLX
      CALL PAGERA
      CALL PAGHDR('H. Michels  (',')',2,0)

      CALL HEIGHT(60)

      NL=NLMESS(CTIT)
      CALL MESSAG(CTIT,(2100-NL)/2,200)

      CALL HEIGHT(50)
      CALL HSYMBL(120)

      NY=150

      DO I=0,21
        IF(MOD(I,4).EQ.0) THEN
          NY=NY+400
          NXP=550
        ELSE
          NXP=NXP+350
        END IF

        IF(I.LT.10) THEN
          WRITE(CSTR,'(I1)') I
        ELSE
          WRITE(CSTR,'(I2)') I
        END IF

        NL=NLMESS(CSTR)/2
        CALL MESSAG(CSTR,NXP-NL,NY+150)
        CALL SYMBOL(I,NXP,NY)
      END DO

      CALL DISFIN
      END
```

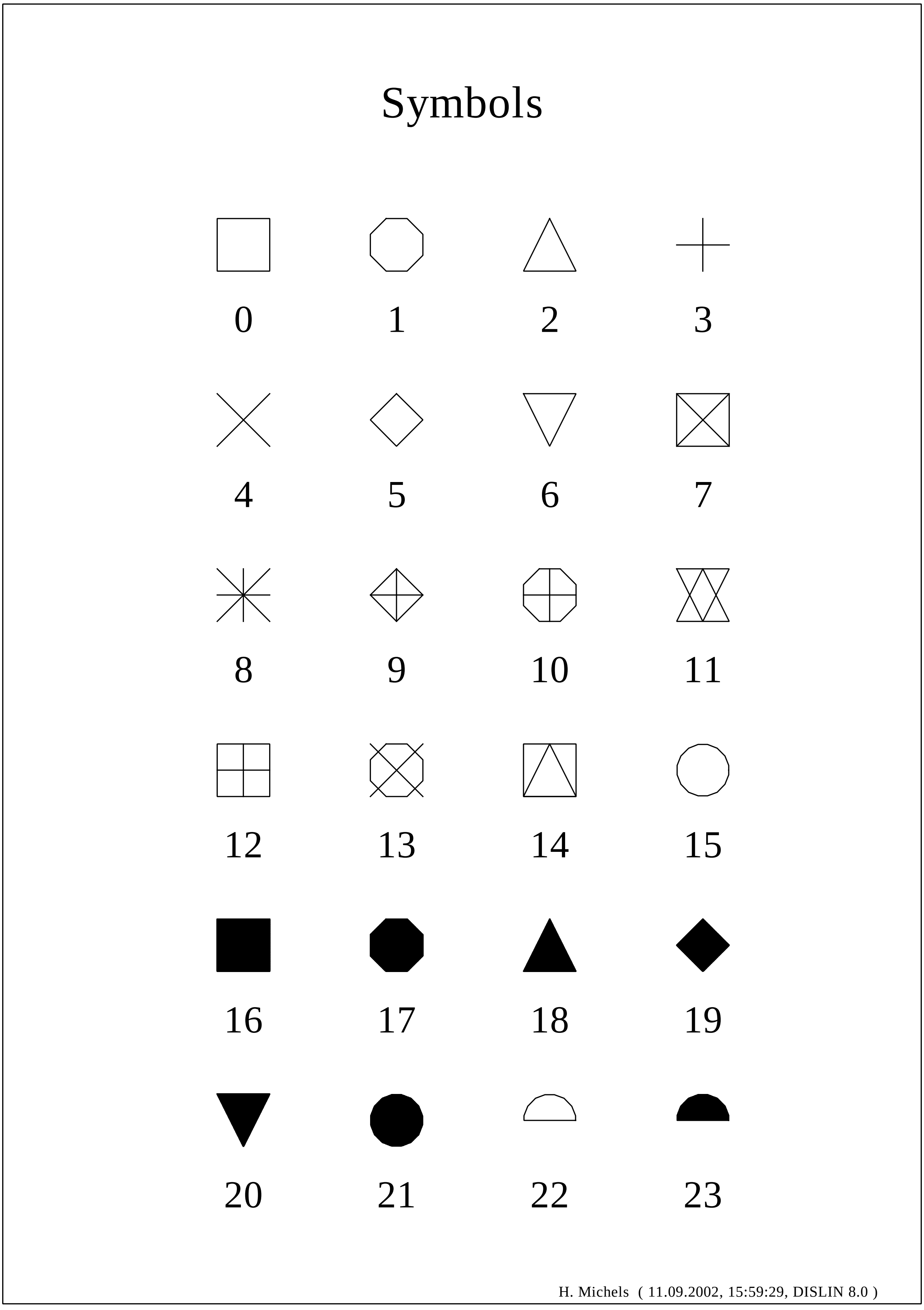


Figure B.3: Symbols

C.4 Logarithmic Scaling

```
      PROGRAM EXA_4
C     USE DISLIN         for Fortran 90!
      CHARACTER*60 CTIT,CLAB(3)*5
      DATA CLAB/'LOG','FLOAT','ELOG '/

      CTIT='Logarithmic Scaling'

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX
      CALL AXSLEN(1400,500)

      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')
      CALL AXSSCL('LOG','XY')

      CALL TITLIN(CTIT,2)

      DO I=1,3
        NYA=2650-(I-1)*800
        CALL LABDIG(-1,'XY')
        IF(I.EQ.2)THEN
          CALL LABDIG(1,'Y')
          CALL NAME(' ','X')
        END IF

        CALL AXSPOS(500,NYA)
        CALL MESSAG('Labels: '//CLAB(I),600,NYA-400)
        CALL LABELS(CLAB(I),'XY')
        CALL GRAF(0.,3.,0.,1.,-1.,2.,-1.,1.)

        IF(I.EQ.3) THEN
          CALL HEIGHT(50)
          CALL TITLE
        END IF

        CALL ENDGRF
      END DO

      CALL DISFIN
      END
```

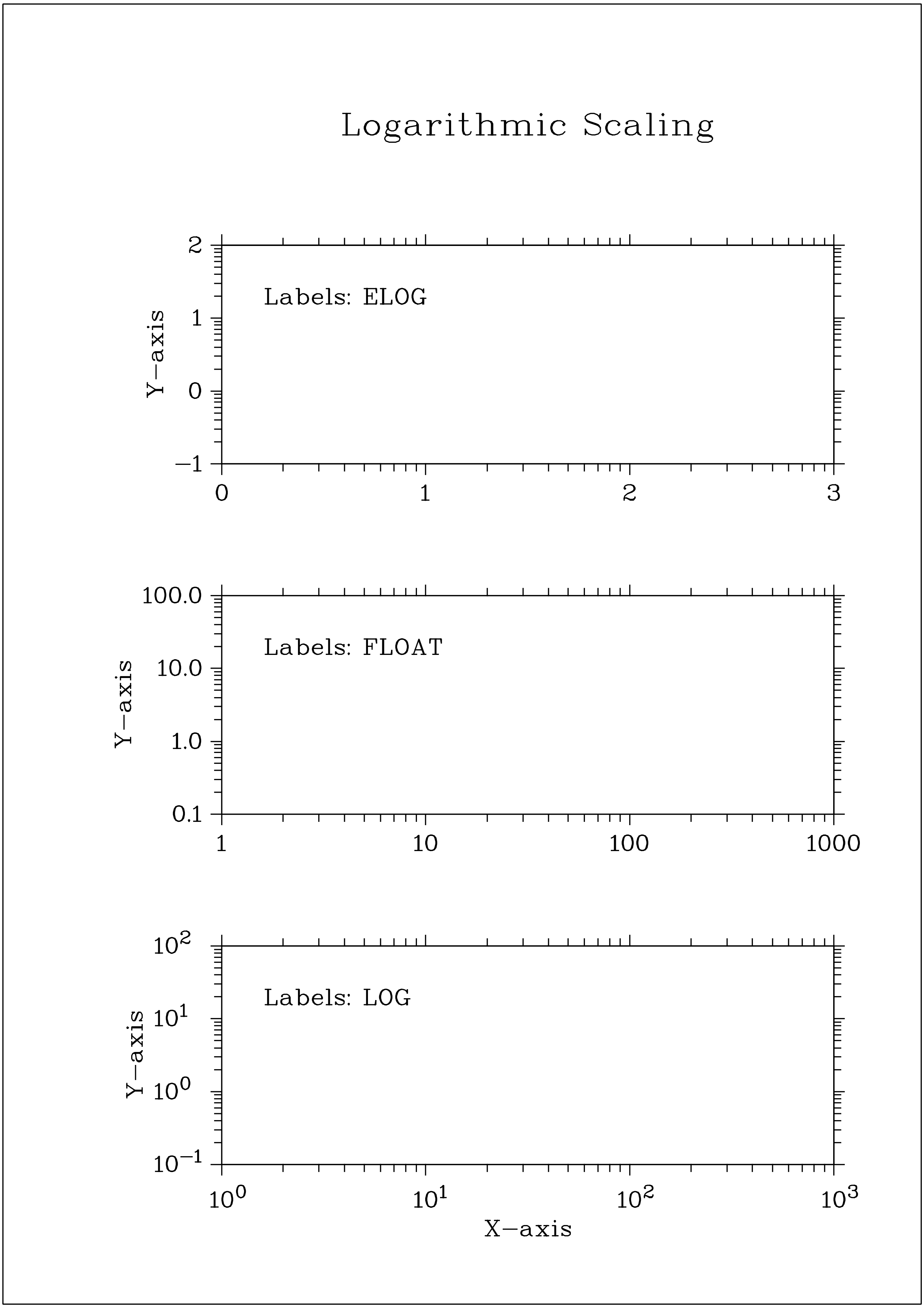


Figure B.4: Logarithmic Scaling

C.5 Interpolation Methods

```
      PROGRAM EXA_5
C     USE DISLIN         for Fortran 90!
      DIMENSION X(16), Y(16)
      CHARACTER*8 CPOL(6),CTIT*60

      DATA X/0.,1.,3.,4.5,6.,8.,9.,11.,12.,12.5,13.,
     *       15.,16.,17.,19.,20./,
     * Y/2.,4.,4.5,3.,1.,7.,2.,3.,5.,2.,2.5,2.,4.,6.,
     *   5.5,4./,
     * CPOL/'SPLINE','STEM','BARS','STAIRS','STEP,'LINEAR'/
     * NYA/2700/

      CTIT='Interpolation Methods'

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX
      CALL INCMRK(1)
      CALL HSYMBL(25)
      CALL TITLIN(CTIT,1)
      CALL AXSLEN(1500,350)
      CALL SETGRF('LINE','LINE','LINE','LINE')

      DO I=1,6
        CALL AXSPOS(350,NYA-(I-1)*350)
        CALL POLCRV(CPOL(I))
        CALL MARKER(0)

        CALL GRAF(0.,20.,0.,5.,0.,10.,0.,5.)
        NX=NXPOSN(1.)
        NY=NYPOSN(8.)
        CALL MESSAG(CPOL(I),NX,NY)
        CALL CURVE(X,Y,16)

        IF(I.EQ.6) THEN
          CALL HEIGHT(50)
          CALL TITLE
        END IF
        CALL ENDGRF
      END DO

      CALL DISFIN
      END
```

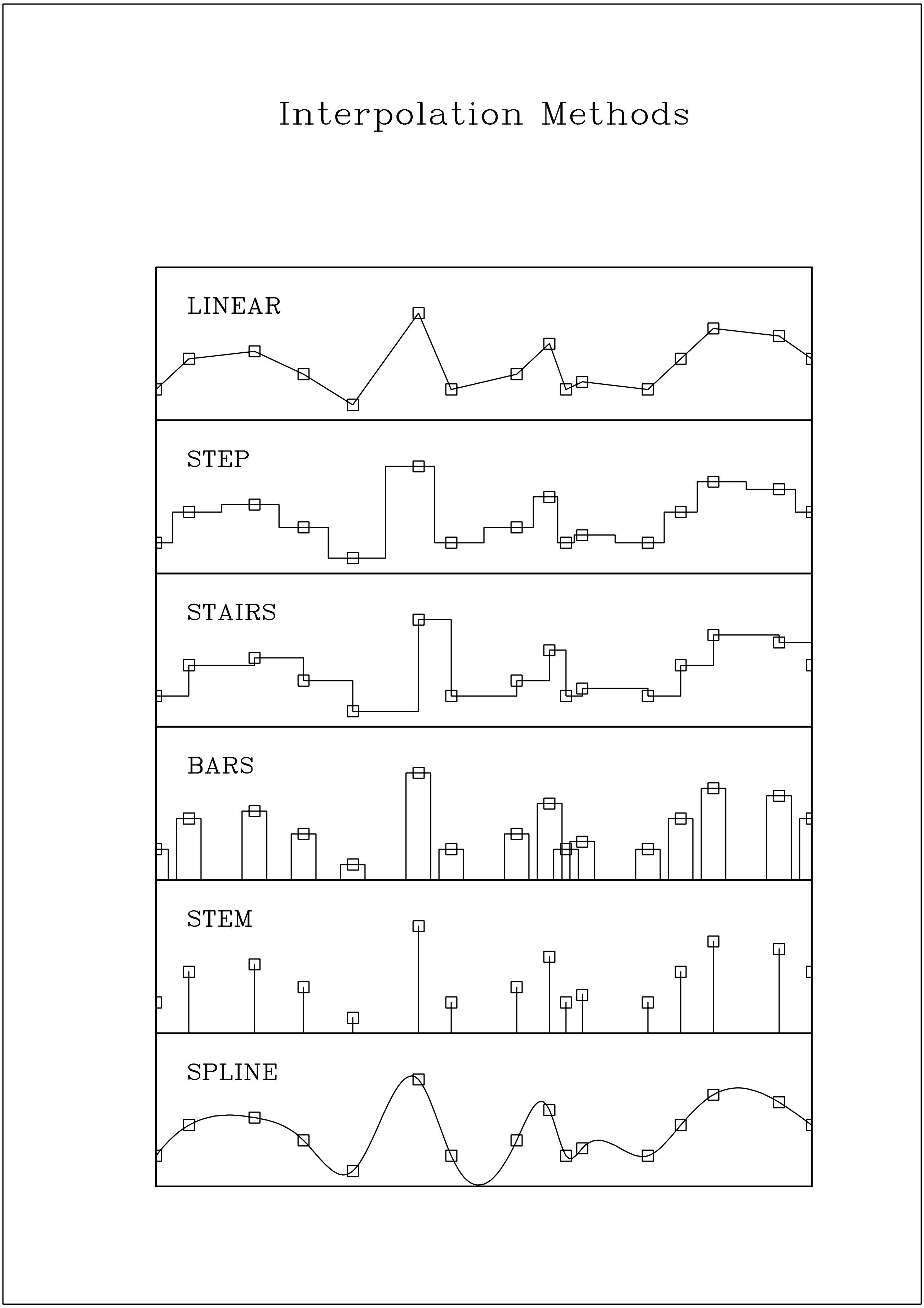


Figure B.5: Interpolation Methods

C.6 Line Styles

```
      PROGRAM EXA_6
C     USE DISLIN         for Fortran 90!
      DIMENSION X(2),Y(2)
      CHARACTER*6 CTYP(8)
      DATA X/3.,9./CTYP/'SOLID','DOT','DASH','CHNDSH',
     *                  'CHNDOT','DASHM','DOTL','DASHL'/

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX
      CALL CENTER
      CALL CHNCRV('LINE')

      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')

      CALL TITLIN('Demonstration of CURVE',1)
      CALL TITLIN('Line Styles',3)

      CALL GRAF(0.,10.,0.,2.,0.,10.,0.,2.)
      CALL TITLE

      DO I=1,8
        Y(1)=9.5-I
        Y(2)=9.5-I
        NY=NYPOSN(Y(1))
        NX=NXPOSN(1.0)
        CALL MESSAG(CTYP(I),NX,NY-20)
        CALL CURVE(X,Y,2)
      END DO

      CALL DISFIN
      END
```

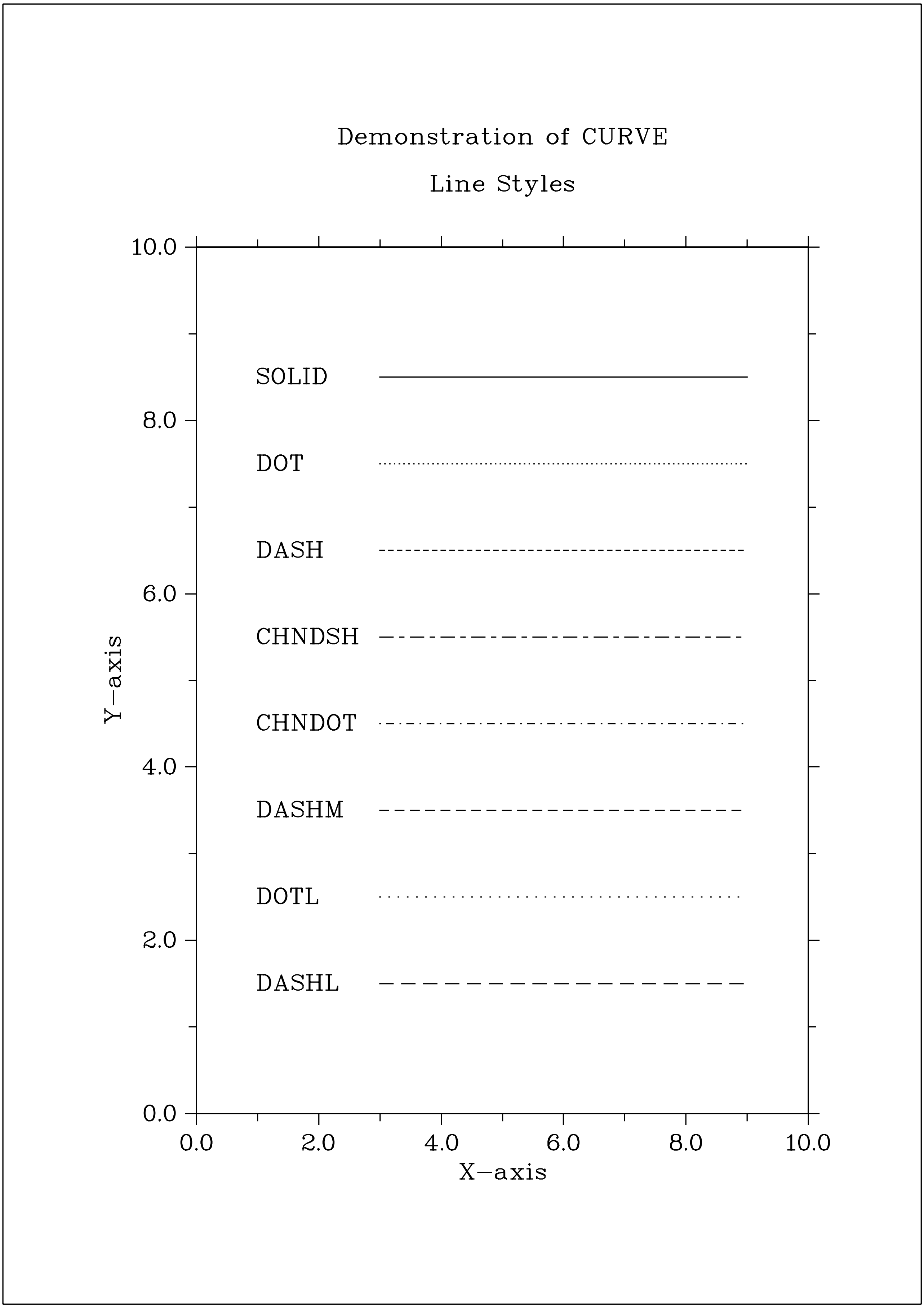


Figure B.6: Line Styles

C.7 Legends

```
      PROGRAM EXA_7
C     USE DISLIN         for Fortran 90!
      PARAMETER(N=301)
      DIMENSION XRAY(N),Y1RAY(N),Y2RAY(N)
      CHARACTER*14 CBUF

      FPI=3.1415926/180.
      STEP=360./(N-1)
      DO I=1,N
        XRAY(I)=(I-1)*STEP
        X=XRAY(I)*FPI
        Y1RAY(I)=SIN(X)
        Y2RAY(I)=COS(X)
      END DO

      CALL DISINI
      CALL PAGERA
      CALL COMPLX
      CALL AXSPOS(450,1800)
      CALL AXSLEN(2200,1200)

      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')
      CALL TITLIN('Demonstration of CURVE',1)
      CALL TITLIN('Legend',3)
      CALL LABDIG(-1,'X')
      CALL TICKS(10,'XY')

      CALL GRAF(0.,360.,0.,90.,-1.,1.,-1.,0.5)
      CALL TITLE
      CALL XAXGIT

      CALL CHNCRV('LINE')
      CALL CURVE(XRAY,Y1RAY,N)
      CALL CURVE(XRAY,Y2RAY,N)

      CALL LEGINI(CBUF,2,7)     ! Legend statements
      NX=NXPOSN(190.)
      NY=NYPOSN(0.75)
      CALL LEGPOS(NX,NY)
      CALL LEGLIN(CBUF,'sin (x)',1)
      CALL LEGLIN(CBUF,'cos (x)',2)
      CALL LEGTIT('Legend')
      CALL LEGEND(CBUF,3)

      CALL DISFIN
      END
```

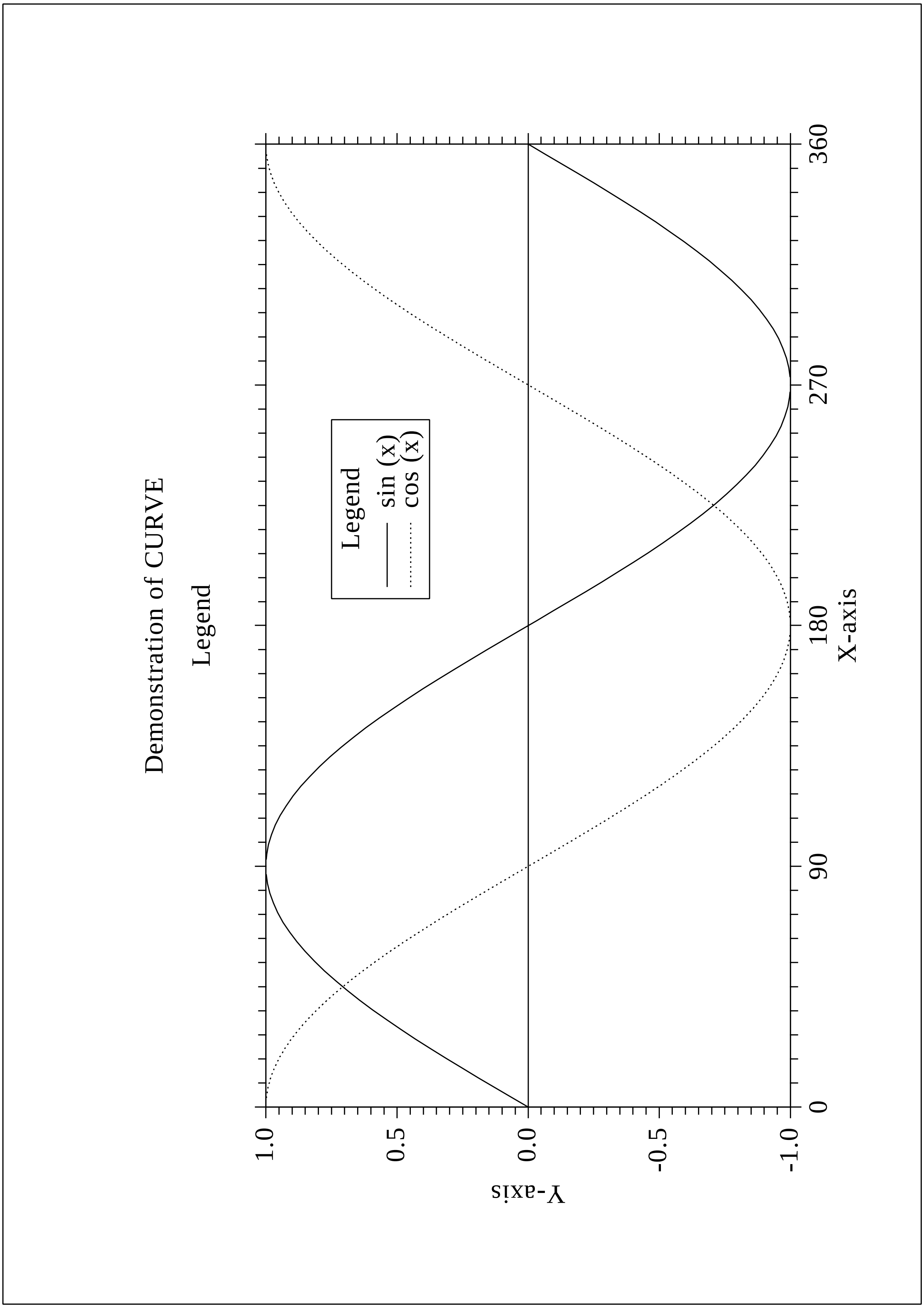


Figure B.7: Legends

C.8 Shading Patterns (AREAF)

```
      PROGRAM EXA_8
C     USE DISLIN         for Fortran 90!
      DIMENSION IXP(4),IYP(4),IX(4),IY(4)
      CHARACTER*60 CTIT,CSTR*2
      DATA IX/0,300,300,0/IY/0,0,400,400/

      CTIT='Shading Patterns (AREAF)'

      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL HEIGHT(50)
      NL=NLMESS(CTIT)
      NX=(2970-NL)/2
      CALL MESSAG(CTIT,NX,200)

      NX0=335
      NY0=350

      DO I=1,3
        NY=NY0+(I-1)*600
        DO J=1,6
          NX=NX0+(J-1)*400
          II=(I-1)*6+J-1
          CALL SHDPAT(II)
          WRITE(CSTR,'(I2)') II

          DO K=1,4
            IXP(K)=IX(K)+NX
            IYP(K)=IY(K)+NY
          END DO
          CALL AREAF(IXP,IYP,4)

          NL=NLMESS(CSTR)
          NX=NX+(300-NL)/2
          CALL MESSAG(CSTR,NX,NY+460)
        END DO
      END DO

      CALL DISFIN
      END
```

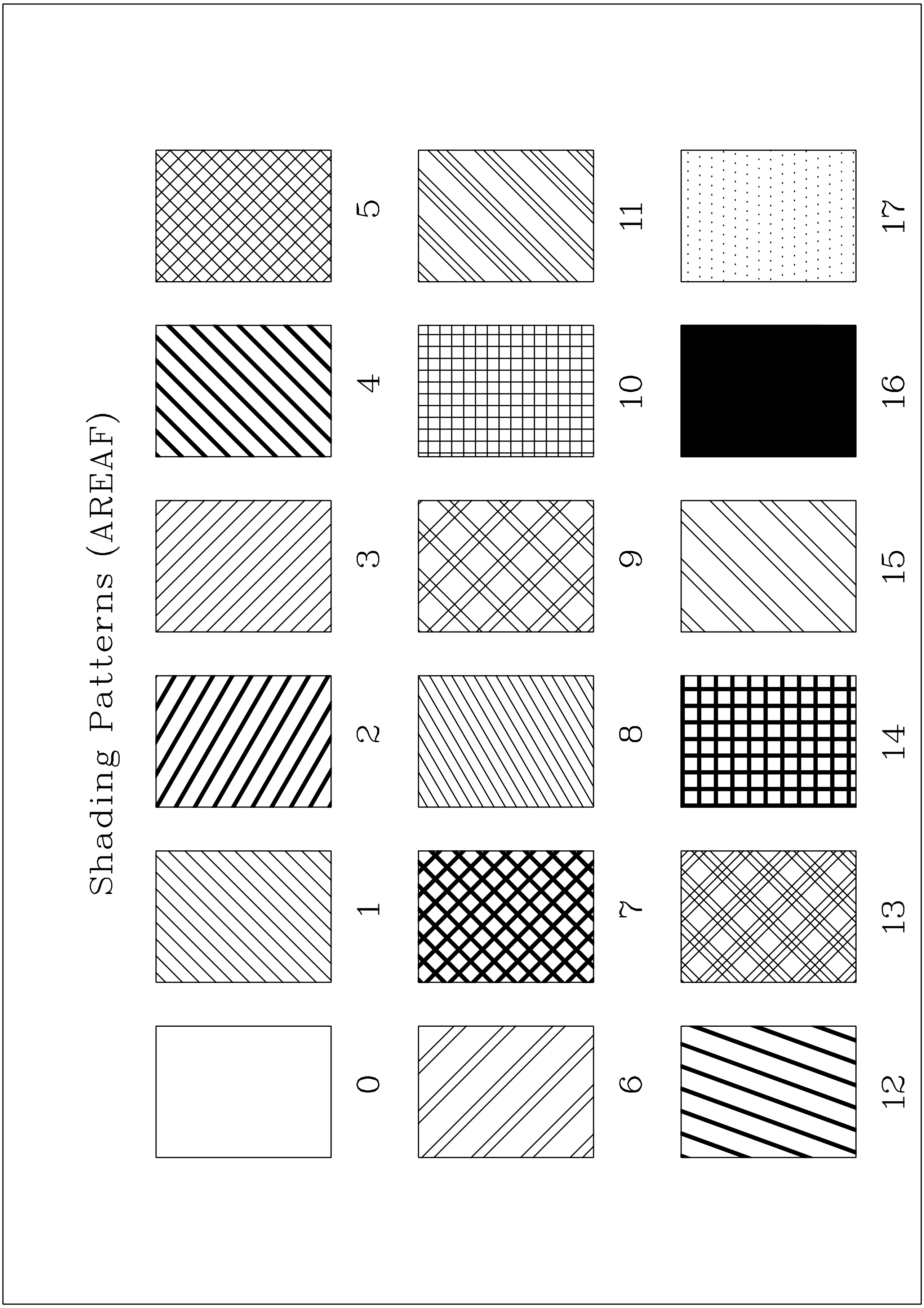


Figure B.8: Shading Patterns

C.9 Vectors

```
      PROGRAM EXA_8
C     USE DISLIN         for Fortran 90!
      DIMENSION IVEC(20)
      CHARACTER*60 CTIT,CNUM*4
      DATA IVEC/0,1111,1311,1421,1531,1701,1911,
     *          3111,3311,3421,3531,3703,4221,4302,
     *          4413,4522,4701,5312,5502,5703/

      CTIT='Vectors'

      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL HEIGHT(60)
      NL=NLMESS(CTIT)
      NX=(2970-NL)/2
      CALL MESSAG(CTIT,NX,200)

      CALL HEIGHT(50)
      NX=300
      NY=400

      DO I=1,20
        IF(I.EQ.11) THEN
          NX=NX+2970/2
          NY=400
        END IF

        WRITE(CNUM,'(I4)') IVEC(I)
        NL=NLMESS(CNUM)
        CALL MESSAG(CNUM,NX-NL,NY-25 )

        CALL VECTOR(NX+100,NY,NX+1000,NY,IVEC(I))
        NY=NY+160
      END DO

      CALL DISFIN
      END
```

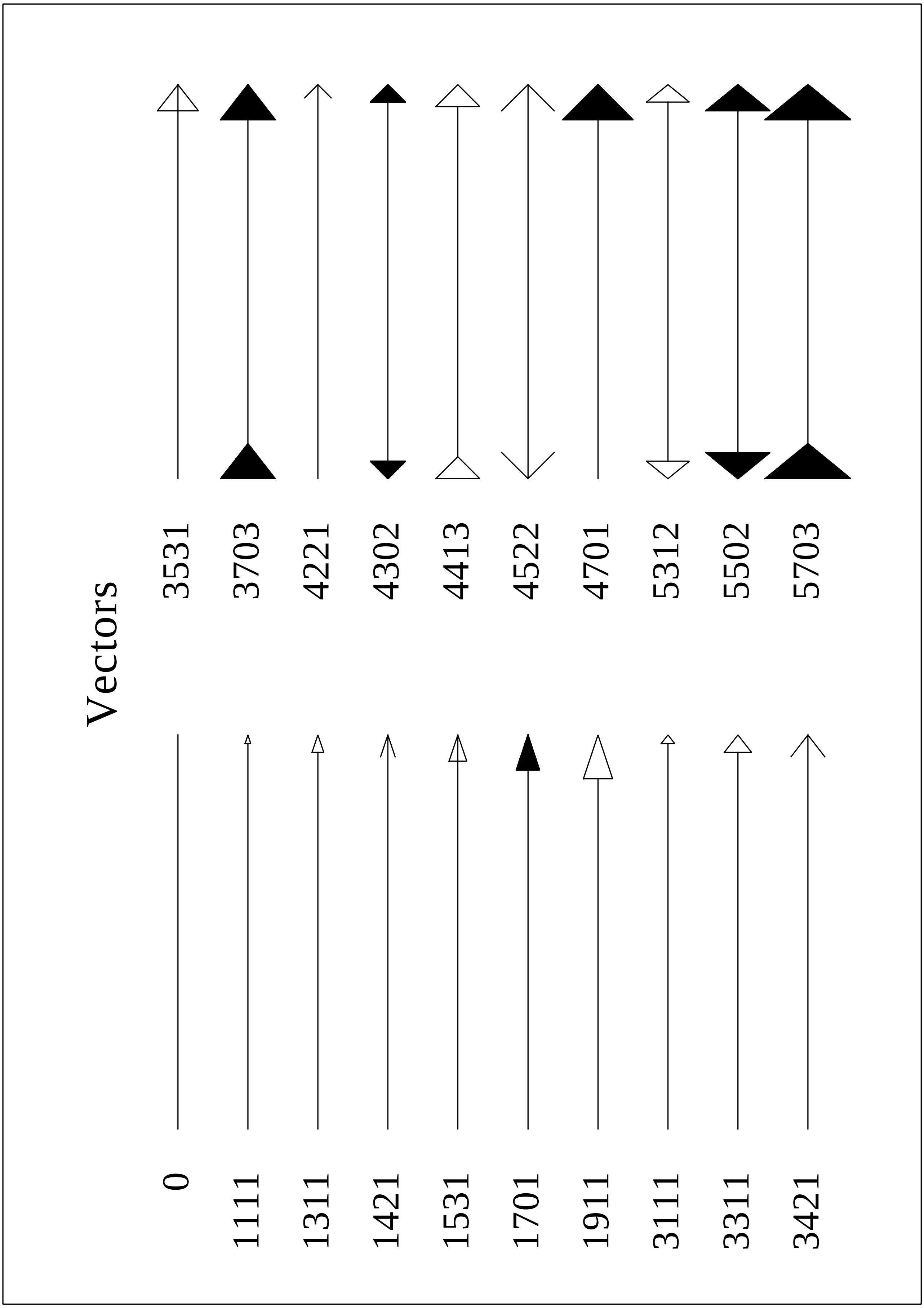


Figure B.9: Vectors

C.10 Shading Patterns (PIEGRF)

```
      PROGRAM EXA_10
C     USE DISLIN         for Fortran 90!
      DIMENSION XRAY(18)
      CHARACTER*60 CTIT,CBUF*36,CSTR*2
      DATA XRAY/18*1./

      CTIT='Shading Patterns (PIEGRF)'

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL AXSPOS(250,2700)
      CALL AXSLEN(1600,2200)
      CALL TITLIN(CTIT,3)
      CALL HEIGHT(50)

      CALL LEGINI(CBUF,18,2)

      DO I=1,18
        WRITE(CSTR,'(I2)') I-1
        CALL LEGLIN(CBUF,CSTR,I)
      END DO

      CALL LABELS('NONE','PIE')
      CALL PIEGRF(CBUF,1,XRAY,18)
      CALL TITLE

      CALL DISFIN
      END
```

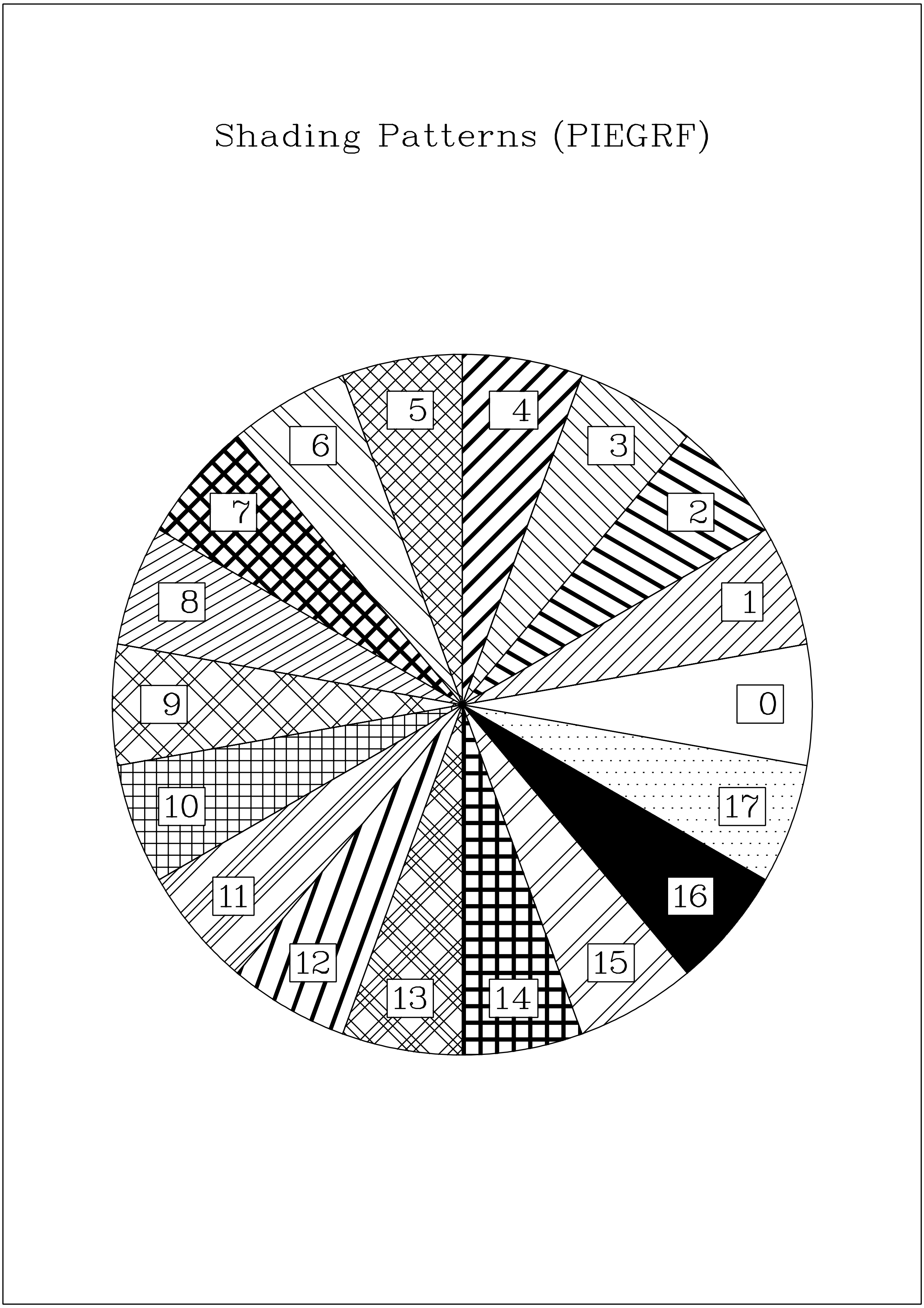


Figure B.10: Shading Patterns

C.11 3-D Bar Graph / 3-D Pie Chart

```
      PROGRAM EXA_11
C     USE DISLIN        for Fortran 90!
      CHARACTER*80 CBUF
      REAL XRAY(5),Y1RAY(5),Y2RAY(5)
      INTEGER IC1RAY(5),IC2RAY(5)
      DATA XRAY/2.,4.,6.,8.,10./,Y1RAY/0.,0.,0.,0.,0./,
     *     Y2RAY/3.2,1.5,2.0,1.0,3.0/
      DATA IC1RAY/50,150,100,200,175/,
     *     IC2RAY/50,150,100,200,175/

      CALL METAFL('POST')
      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL HWFONT

      CALL TITLIN('3-D Bar Graph / 3-D Pie Chart', 2)
      CALL HTITLE(40)

      CALL SHDPAT(16)
      CALL AXSLEN(1500,1000)
      CALL AXSPOS(300,1400)

      CALL BARWTH(0.5)
      CALL BARTYP('3DVERT')
      CALL LABELS('SECOND','BARS')
      CALL LABPOS('OUTSIDE','BARS')
      CALL LABCLR(255,'BARS')
      CALL GRAF(0.,12.,0.,2.,0.,5.,0.,1.)
      CALL TITLE
      CALL COLOR('RED')
      CALL BARS(XRAY,Y1RAY,Y2RAY,5)
      CALL ENDGRF

      CALL SHDPAT(16)
      CALL LABELS('DATA','PIE')
      CALL LABCLR(255,'PIE')
      CALL CHNPIE('NONE')
      CALL PIECLR(IC1RAY,IC2RAY,5)
      CALL PIETYP('3D')
      CALL AXSPOS(300,2700)
      CALL PIEGRF(CBUF,0,Y2RAY,5)
      CALL DISFIN
      END
```

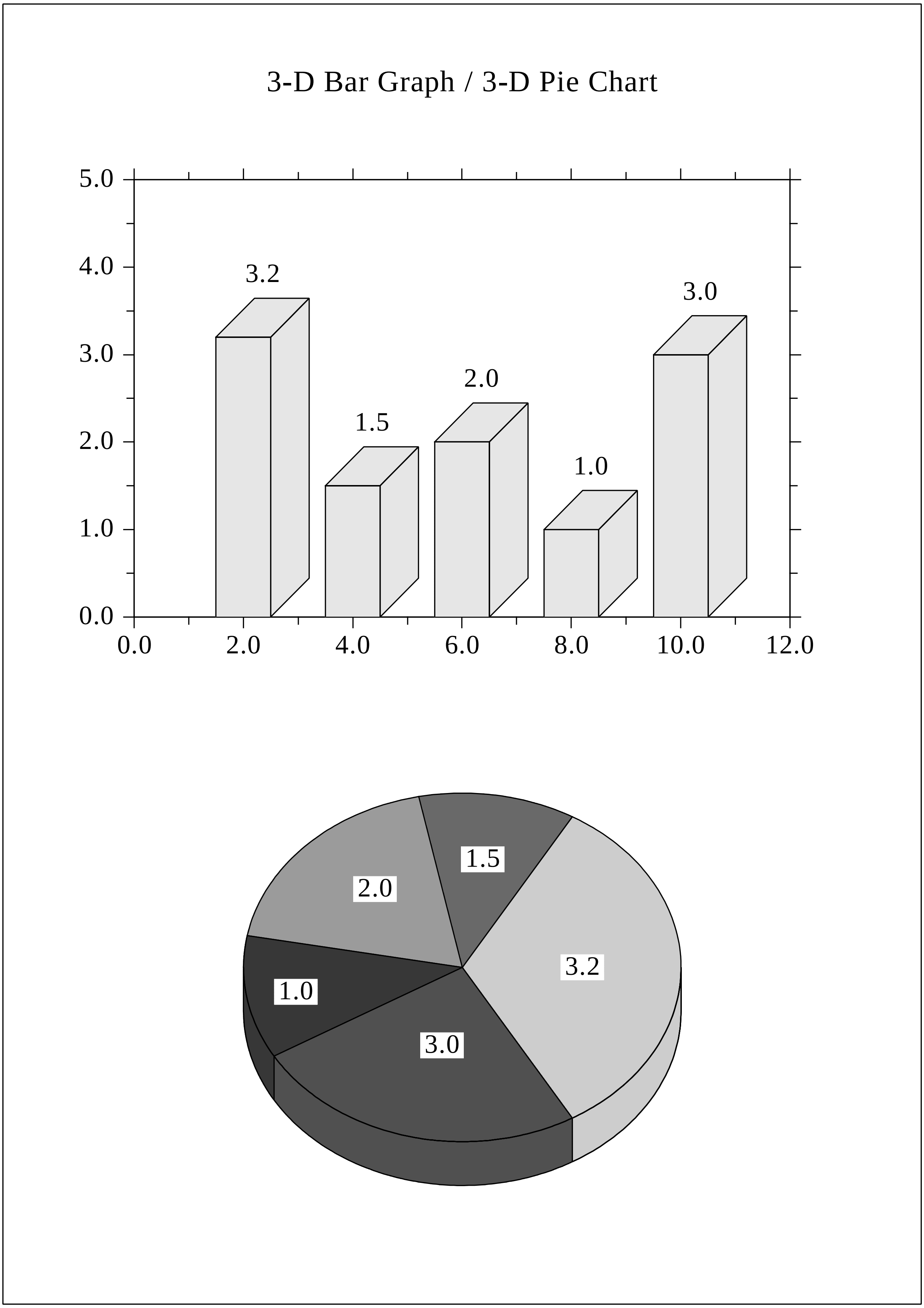


Figure B.11: 3-D Bar Graph / 3-D Pie Chart

C.12 Surface Plot (SURFUN)

```
      PROGRAM EXA_12
C     USE DISLIN         for Fortran 90!
      CHARACTER*60 CTIT1,CTIT2
      EXTERNAL ZFUN

      CTIT1='Surface Plot (SURFUN)'
      CTIT2='F(X,Y) = 2*SIN(X)*SIN(Y)'

      CALL SETPAG('DA4P')
      CALL DISINI
      CALL PAGERA
      CALL COMPLX

      CALL AXSPOS(200,2600)
      CALL AXSLEN(1800,1800)

      CALL NAME('X-axis','X')
      CALL NAME('Y-axis','Y')
      CALL NAME('Z-axis','Z')

      CALL TITLIN(CTIT1,2)
      CALL TITLIN(CTIT2,4)

      CALL VIEW3D(-5.,-5.,4.,'ABS')
      CALL GRAF3D(0.,360.,0.,90.,0.,360.,0.,90.,
     *            -3.,3.,-3.,1.)
      CALL HEIGHT(50)
      CALL TITLE

      CALL SURFUN(ZFUN,1,10.,1,10.)

      CALL DISFIN
      END

      FUNCTION ZFUN(X,Y)
      FPI=3.14159/180.
      ZFUN=2*SIN(X*FPI)*SIN(Y*FPI)
      END
```

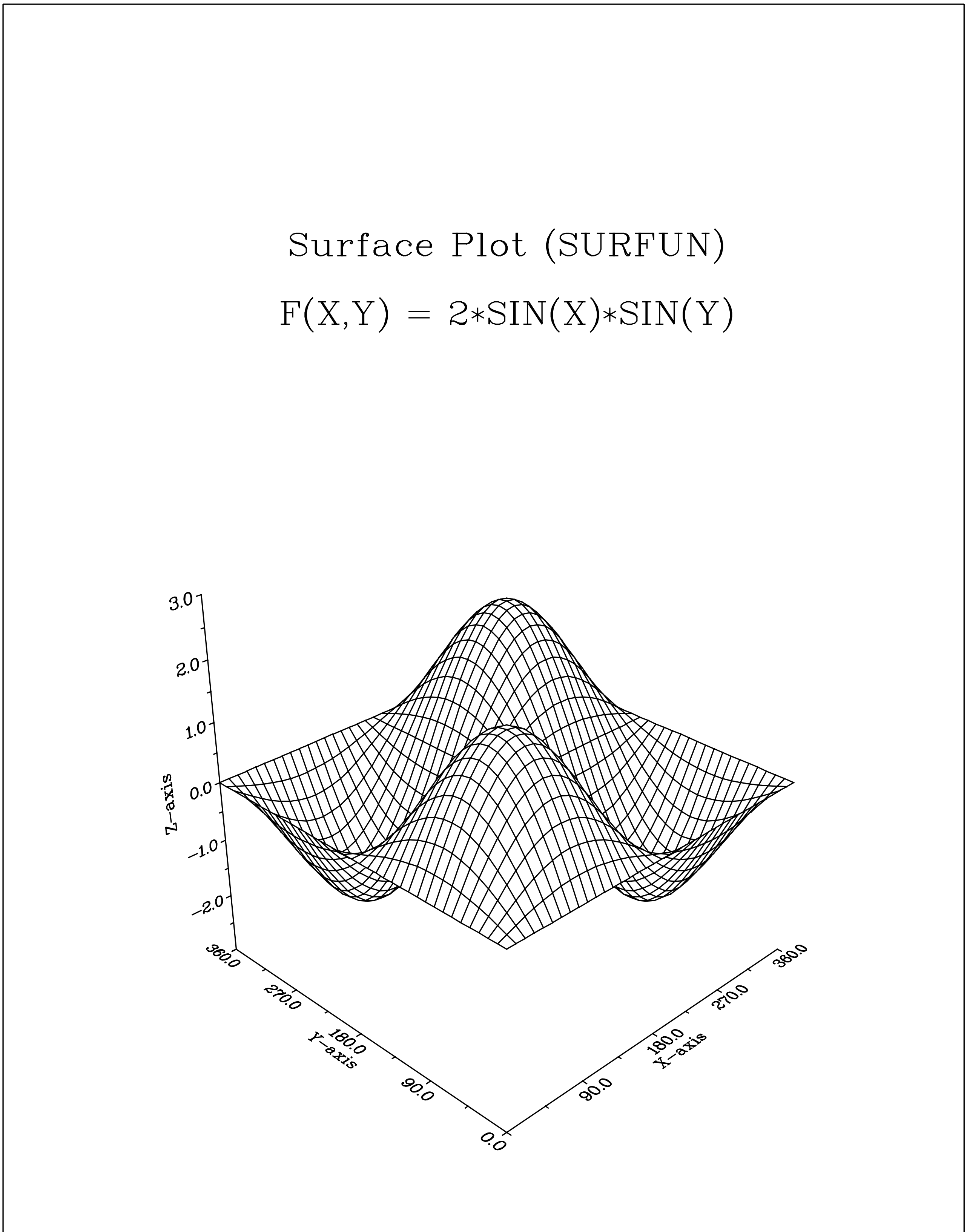


Figure B.12: Surface Plot

C.13 Map Plot

```
      PROGRAM EXA_13
C     USE DISLIN         for Fortran 90!
      DIMENSION XC(9),YC(9)
      CHARACTER*12 CSTR(9)

      DATA XC/-22.,18.,37.5,0.,2.5,12.5,23.5,-3.75,14.25/
     *     YC/64.,59.6,56.,51.5,48.5,42.,38.,40.3,50.1/
     *   CSTR/'Reykjavik','Stockholm','Moskau','London',
     *        'Paris','Rom','Athen','Madrid','Prag'/

      CALL METAFL('POST')
      CALL DISINI
      CALL PAGERA
      CALL HWFONT

      CALL AXSPOS(500,1850)
      CALL AXSLEN(2200,1400)

      CALL LABDIG(-1,'xy')
      CALL TICKS(1,'xy')
      CALL NAME('Longitude','x')
      CALL NAME('Latitude','y')

      CALL TITLIN('Map Plot',3)
      CALL INCMRK(-1)

      CALL LABELS('MAP','xy')
      CALL PROJCT('LAMBERT')
      CALL FRAME(3)
      CALL GRAFMP(-40.,60.,-40.,20.,35.,70.,40.,10.)

      CALL WORLD
      CALL CURVMP(XC,YC,9)

      DO I=1,9
        CALL POS2PT(XC(I),YC(I),XP,YP)
        NXP=XP+30
        NYP=YP
        CALL MESSAG(CSTR(I),NXP,NYP)
      END DO

      CALL GRIDMP(1,1)

      CALL HEIGHT(50)
      CALL TITLE
      CALL DISFIN
      END
```

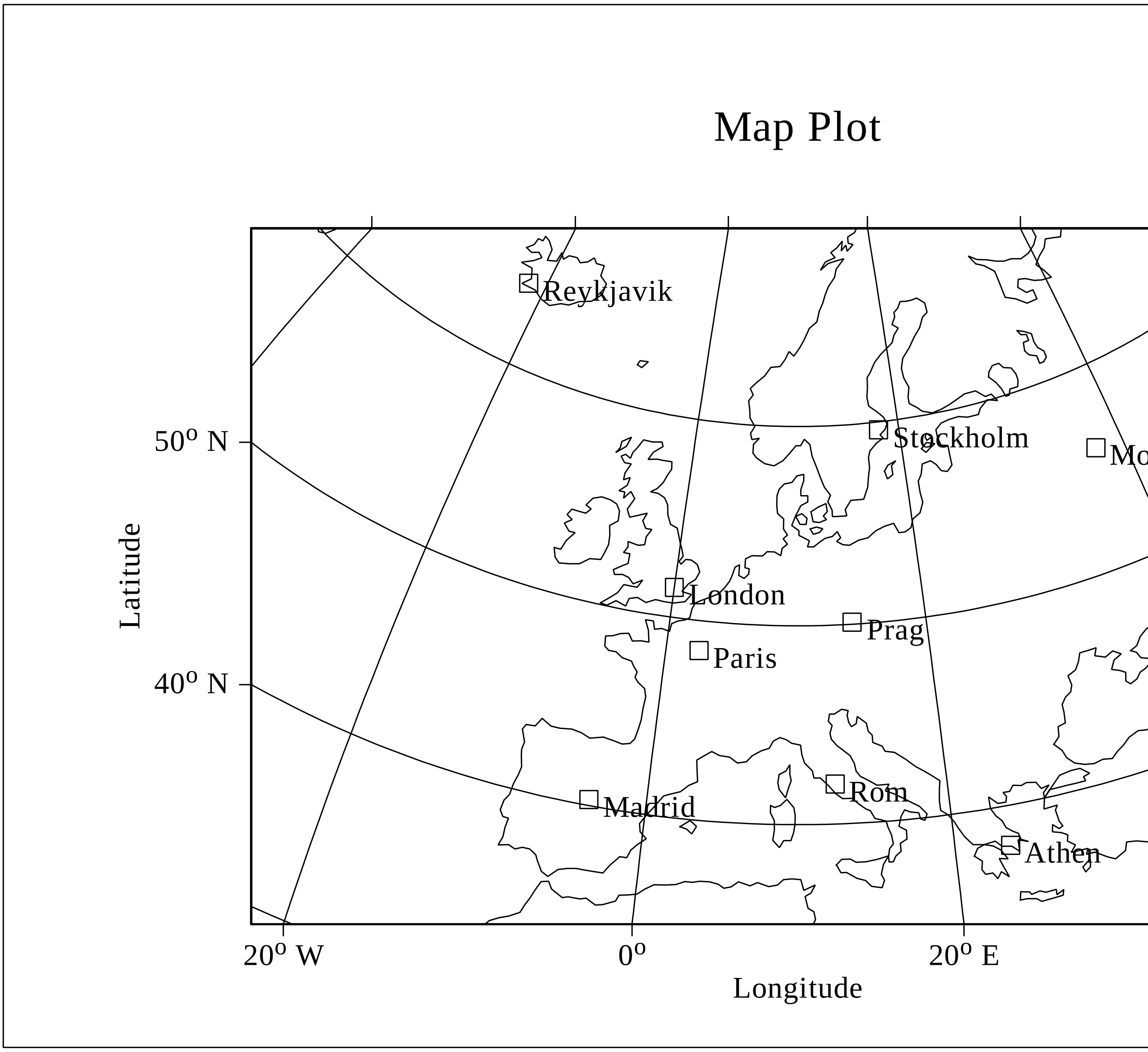


Figure B.13: Map Plot

Appendix D

Index

This appendix presents all routines in the graphics library in alphabetical order. For parameters, the following conventions are used:

- INTEGER variables begin with the character N or I
- CHARACTER variables begin with the character C
- other variables are REAL
- arrays end with the keyword 'RAY'.

The abbreviations have the meaning:

ps denotes a parameter setting routine
rq denotes a parameter requesting routine
p denotes a plot routine.
w denotes a widget routine.

Routine	Parameter	Level		Page
BARBOR	(ICLR)	1,2,3	ps	131
BARCLR	(IC1, IC2, IC3)	1,2,3	ps	131
BARGRP	(NGRP, GAP)	1,2,3	ps	131
BARMOD	(CMOD,COPT)	1,2,3	ps	130
BAROPT	(XF, ANG)	1,2,3	ps	132
BARPOS	(COPT)	1,2,3	ps	131
BARS	(XRAY, Y1RAY, Y2RAY, N)	2,3	p	129
BARS3D	(XRAY, YRAY, Z1RAY, Y2RAY, XWRAY, YWRAY, ICRAY, N)	2,3	p	160
BARTYP	(CTYP)	1,2,3	ps	129
BARWTH	(FACTOR)	1,2,3	ps	86, 130
BASALF	(CALPH)	1,2,3	ps	60
BASDAT	(ID, IM, IY)	0,1,2,3	ps	114
BEZIER	(XRAY, YRAY, N, XPRAY, YPRAY, NP)	0,1,2,3		113
BITSI2	(NBITS, NINP2, IINP, NOUT2, IOUT, IOPT)	0,1,2,3		115
BITSI4	(NBITS, NINP, IINP, NOUT, IOUT, IOPT)	0,1,2,3		116
BMPFNT	(CFONT)	1,2,3	ps	59
BOX2D	none	1,2,3	p	16
BOX3D	none	3	p	156
CENTER	none	1,2,3	ps	40
CGMBGD	(XR, XG, XB)	0,1,2,3	ps	30
CGMPIC	(CSTR)	0,1,2,3	ps	30
CHAANG	(ANGLE)	1,2,3	ps	57
CHACOD	(COPT)	1,2,3	ps	60
CHASPC	(XSPC)	1,2,3	ps	57
CHAWTH	(XWTH)	1,2,3	ps	57
CHNATT	none	1,2,3	ps	83
CHNBAR	(COPT)	1,2,3	ps	130
CHNCRV	(CATT)	1,2,3	ps	83
CHNDOT	none	1,2,3	ps	86
CHNDSH	none	1,2,3	ps	86
CHNPIE	(CATT)	1,2,3	ps	134
CIRC3P	(X1, Y1, X2, Y2, X3, Y3, XM, YM, R)	0,1,2,3		114
CIRCLE	(NX, NY, NR)	1,2,3	p	106
CIRCSP	(NSPC)	1,2,3	ps	107
CLIP3D	(COPT)	1,2,3	ps	163
CLOSFL	(NLU)	0,1,2,3		117
CLPBOR	(COPT)	2,3	ps	49
CLPWIN	(NX, NY, NW, NH)	1,2,3	ps	49
CLRCYC	(INDEX, ICLR)	1,2,3	ps	89
CLRMOD	(CMOD)	0	ps	36
CLSWIN	(ID)	1,2,3	ps	119
COLOR	(CCOL)	1,2,3	ps	52
COLRAN	(NCA, NCE)	1,2,3	ps	147
COLRAY	(ZRAY, NRAY, N)	3		149
COMPLX	none	1,2,3	ps	58
CONCLR	(NCRAY,N)	1,2,3	ps	203
CONCRV	(XRAY, YRAY, N, ZLEV)	2,3	p	197

Routine	Parameter	Level		Page
ERRBAR	(XRAY, YRAY, E1RAY, E2RAY, N)	2,3	p	24
ERRDEV	(CDEV)	0	ps	35
ERRFIL	(CFIL)	0	ps	35
ERRMOD	(CKEY,CMOD)	0	ps	35
EUSHFT	(CNAT, CHAR)	1,2,3	ps	61
EXPZLB	(CSTR)	1,2,3	ps	147
FCHA	(X, NDIG, NL, CSTR)	0,1,2,3		112
FIELD	(X1RAY, Y1RAY, X2RAY, Y2RAY, N, IVEC)	2, 3	p	25
FILBOX	(NX, NY, NW, NH)	1,2,3	ps	14
FILCLR	(CMOD)	1,2,3	ps	14
FILMOD	(CMOD)	0,1,2,3	ps	29
FIXSPC	(XFAC)	1,2,3	ps	57
FLAB3D	none	1,2,3	ps	156
FLEN	(X, NDIG, NL)	0,1,2,3		112
FRAME	(NFRM)	1,2,3	ps	50
FRMCLR	(NCLR)	1,2,3	ps	50
FRMESS	(NFRM)	1,2,3	ps	56
GAPCRV	(XGAP)	1,2,3	ps	89
GETALF	(CALPH)	1,2,3	rq	96
GETANG	(NANG)	1,2,3	rq	96
GETBPP	(NBPP)	0,1,2,3	rq	100
GETCLP	(NX, NY, NW, NH)	1,2,3	rq	100
GETCLR	(NCOL)	1,2,3	rq	98
GETDIG	(NXDIG, NYDIG, NZDIG)	1,2,3	rq	97
GETDSP	(CDSP)	0,1,2,3	rq	100
GETFIL	(CFIL)	1,2,3	rq	95
GETGRF	(XA, XE, XOR, XSTP, CAX)	2,3	rq	97
GETHGT	(NHCHAR)	1,2,3	rq	96
GETHNM	(NHNAME)	1,2,3	rq	96
GETIND	(INDEX, XR, XG, XB)	1,2,3	rq	99
GETLAB	(CXLAB, CYLAB, CZLAB)	1,2,3	rq	98
GETLEN	(NXL, NYL, NZL)	1,2,3	rq	96
GETLEV	(NLEV)	1,2,3	rq	98
GETLIN	(NWIDTH)	1,2,3	rq	99
GETLIT	(XP, YP, ZP, XN, YN, ZN, ICLR)	1,2,3	rq	165
GETMAT	(XRAY, YRAY, ZRAY, N, ZMAT, NX, NY, ZVAL, IMAT, WMAT)	2,3		166
GETMFL	(CFMT)	1,2,3	rq	95
GETMIX	(CHAR, CMIX)	1,2,3	rq	96
GETOR	(NX0, NY0)	1,2,3	rq	95
GETPAG	(NXPAG, NYPAG)	1,2,3	rq	95
GETPAT	(NPAT)	1,2,3	rq	99
GETPLV	(NPLV)	1,2,3	rq	98
GETPOS	(NXA, NYA)	1,2,3	rq	95
GETRAN	(NCA, NCE)	1,2,3	rq	100
GETRES	(NPB, NPH)	1,2,3	rq	99
GETRGB	(XR, XG, XB)	1,2,3	rq	99
GETSCL	(NXSCL, NYSCL, NZSCL)	1,2,3	rq	98
GETSCR	(NWPIX, NHPIX)	0,1,2,3	rq	100

Routine	Parameter	Level		Page
GETSHF	(CNAT, CHAR)	1,2,3	rq	96
GETSP1	(NXDIS, NYDIS, NZDIS)	1,2,3	rq	97
GETSP2	(NXDIS, NYDIS, NZDIS)	1,2,3	rq	98
GETSYM	(NSYM, NHSYM)	1,2,3	rq	99
GETTCL	(NMAJ, NMIN)	1,2,3	rq	97
GETTIC	(NXTIC, NYTIC, NZTIC)	1,2,3	rq	97
GETTYP	(NTYP)	1,2,3	rq	99
GETUNI	(NU)	1,2,3	rq	98
GETVER	(XVER)	0,1,2,3	rq	98
GETVK	(NV, NVFX, NVFY)	1,2,3	rq	100
GETVLT	(CVLT)	1,2,3	rq	99
GETWID	(NZB)	1,2,3	rq	100
GETWIN	(NX, NY, NW, NH)	1,2,3	rq	100
GETXID	(IVAL, CTYPE)	1,2,3	rq	101
GIFMOD	(CMODE, CKEY)	0	ps	31
GMXALF	(CALPH, C1, C2, N)	1,2,3	rq	97
GOTHIC	none	1,2,3	ps	58
GRACE	(NGRACE)	1,2,3	ps	49
GRAF	(XA, XE, XOR, XSTP, YA, YE, YOR, YSTP)	1	p	15
GRAF3	(XA, XE, XOR, XSTP, YA, YE, YOR, YSTP, ZA, ZE, ZOR, ZSTP)	1	p	143
GRAF3D	(XA, XE, XOR, XSTP, YA, YE, YOR, YSTP, ZA, ZE, ZOR, ZSTP)	1	p	155
GRAFMP	(XA, XE, XOR, XSTP, YA, YE, YOR, YSTP)	1	p	179
GRDPOL	(IXGRID, IYGRID)	2,3	p	17
GRFFIN	none	1,2,3	ps	169
GRFINI	(X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3)	3	ps	169
GRID	(IXGRID, IYGRID)	2,3	p	17
GRID3D	(IXGRID, IYGRID, COPT)	2,3	p	156
GRIDMP	(IXGRID, IYGRID)	2	p	179
GWGATT	(ID, IATT, COPT)	0	rq	224
GWGBOX	(ID, ISEL)	0	rq	224
GWGBUT	(ID, IVAL)	0	rq	223
GWGFIL	(ID, CFIL)	0	rq	224
GWGLIS	(ID, ISEL)	0	rq	224
GWGSCL	(ID, XVAL)	0	rq	224
GWGTXT	(ID, CSTR)	0	rq	223
GWGXID	(ID, IWINID)	0	rq	225
HEIGHT	(NHCHAR)	1,2,3	ps	55
HELVE	none	1,2,3	ps	58
HELVES	none	1,2,3	ps	58
HISTOG	(XRAY, N, XHRAY, YHRAY, NH)	0,1,2,3	ps	113
HNAME	(NHNAME)	1,2,3	ps	47
HSVRGB	(XH, XS, XV, XR, XG, XB)	0,1,2,3		55
HSYMBL	(NHSYM)	1,2,3	ps	84
HTITLE	(NHTIT)	1,2,3	ps	52
HWFONT	none	1,2,3	ps	60
HWORIG	(NX, NY)	0	ps	34
HWPAGE	(NW, NH)	0	ps	34

Routine	Parameter	Level		Page
IMGBOX	(NX, NY, NW, NH)	1,2,3	ps	124
IMGCLP	(NX, NY, NW, NH)	1,2,3	ps	127
IMGFIN	none	1,2,3	ps	122
IMGFMT	(COPT)	0	ps	32
IMGINI	none	1,2,3	ps	122
IMGMOD	(COPT)	1,2,3	ps	124
IMGSIZ	(NW, NH)	1,2,3	ps	124
INCCRV	(NCRV)	1,2,3	ps	83
= INCDAT	(ID, IM, IY)	0,1,2,3		115
INCFIL	(CFIL)	1,2,3	p	14
INCMRK	(NMRK)	1,2,3	ps	84
= INDRGB	(XR, XG, XB)	1,2,3		54
INTAX	none	1,2,3	ps	45
INTCHA	(NX, NL, CSTR)	0,1,2,3		112
INTLEN	(NX, NL)	0,1,2,3		111
= INTRGB	(XR, XG, XB)	0,1,2,3		54
ITMCAT	(CLIS, CITEM)	0,1,2,3		225
= ITMCNT	(CLIS)	0,1,2,3		225
ITMSTR	(CLIS, IDX, CITEM)	0,1,2,3		225
LABCLR	(ICLR, COPT)	1,2,3	ps	133,202
LABDIG	(NDIG, CAX)	1,2,3	ps	45
LABDIS	(NDIS, CAX)	1,2,3	ps	45,202
LABELS	(CLAB, CAX)	1,2,3	ps	43,201
LABJUS	(CJUS, CAX)	1,2,3	ps	44
LABL3D	(COPT)	1,2,3	ps	161
LABMOD	(CKEY, CVAL, CAX)	1,2,3	ps	45
LABPOS	(CPOS, CAX)	1,2,3	ps	44
LABTYP	(CTYP, CAX)	1,2,3	ps	44
LEGCLR	none	1,2,3	ps	23
LEGEND	(CBUF, NCOR)	2,3	p	22
LEGINI	(CBUF, NLIN, NMAXLN)	1,2,3	ps	22
LEGLIN	(CBUF, CSTR, ILIN)	1,2,3	ps	22
LEGOPT	(XF1, XF2, XF3)	1,2,3	ps	23
LEGPAT	(ITYP, ITHK, ISYM, ICLR, IPAT, ILIN)	1,2,3	ps	23
LEGPOS	(NX, NY)	1,2,3	ps	23
LEGTIT	(CSTR)	1,2,3	ps	23
LFTTIT	none	1,2,3	ps	51
LIGHT	(CMODE)	1,2,3	ps	164
LINCYC	(INDEX, ITYP)	1,2,3	ps	89
LINE	(NX, NY, NU, NV)	1,2,3	p	104
LINESP	(XFAC)	1,2,3	ps	51
LINTYP	(NTYP)	1,2,3	ps	86
LINWID	(NWIDTH)	1,2,3	ps	87
LITMOD	(ID, CMODE)	1,2,3	ps	164
LITOP3	(ID, XR, XG, XB, CTYPE)	1,2,3	ps	165
LITOPT	(ID, XVAL, CTYPE)	1,2,3	ps	164
LITPOS	(ID, XP, YP, ZP, COPT)	1,2,3	ps	164
LNCAP	(CAP)	1,2,3	ps	87
LNJOIN	(CJOIN)	1,2,3	ps	87

Routine	Parameter	Level		Page
LNMLT	(XFC)	1,2,3	ps	88
LOGTIC	(CMOD)	1,2,3	ps	42
MAPBAS	(COPT)	1,2,3	ps	185
MAPFIL	(CFIL, COPT)	1,2,3	ps	186
MAPLEV	(COPT)	1,2,3	ps	186
MAPMOD	(CMODE)	1,2,3	ps	187
MAPPOL	(XPOL, YPOL)	1	ps	186
MAPREF	(YLOWER, YUPPER)	1	ps	187
MAPSPH	(XRAD)	1	ps	186
MARKER	(NSYM)	1,2,3	ps	84
MATOP3	(XR, XG, XB, CTYPE)	1,2,3	ps	165
MATOPT	(XVAL, CTYPE)	1,2,3	ps	165
MDFMAT	(IX, IY, WEIGHT)	1,2,3	ps	167
MESSAG	(CSTR, NX, NY)	1,2,3	p	11
METAFL	(CFMT)	0	ps	28
MIXALF	none	1,2,3	ps	71
MIXLEG	none	1,2,3	ps	23
MPSLOGO	(NX, NY, NSIZE, COPT)	1,2,3	p	127
MSGBOX	(CSTR)	0	w	226
MSHCLR	(ICLR)	1,2,3	ps	162
MYLAB	(CSTR, ITICK, CAX)	1,2,3	ps	43
MYLINE	(NRAY, N)	1,2,3	ps	86
MYPAT	(IANG, ITYPE, IDENS, ICROSS)	1,2,3	ps	88
MYSYMB	(XRAY, YRAY, N, ISYM, IFLAG)	1,2,3	ps	84
MYVLT	(XRRAY, XGRAY, XBRAY, N)	0,1,2,3	ps	53
NAMDIS	(NDIS, CAX)	1,2,3	ps	47
NAME	(CSTR, CAX)	1,2,3	ps	47
NAMJUS	(CJUS, CAX)	1,2,3	ps	47
NEGLOG	(EPS)	1,2,3	ps	21
NEWMIX	none	1,2,3	ps	71
NEWPAG	none	1	ps	34
= NLMESS	(CSTR)	1,2,3		111
= NLNUMB	(X, NDIG)	1,2,3		111
NOARLN	none	1,2,3	ps	89
NOBAR	none	1,2,3	ps	147
NOBGD	none	1,2,3	ps	147
NOCHEK	none	1,2,3	ps	86
NOCLIP	none	1,2,3	ps	49
NOGRAF	none	1	ps	48
NOHIDE	none	1,2,3	ps	161
NOLINE	(CAX)	1,2,3	ps	48
NUMBER	(X, NDIG, NX, NY)	1,2,3	p	11
NUMFMT	(COPT)	1,2,3	ps	56
NUMODE	(CDEC, CGRP, CPOS, CFIX)	1,2,3	ps	56
=NWKDAY	(ID, IM, IY)	0,1,2,3		115
=NXLEGN	(CBUF)	2,3		23
=NXPOSN	(X)	2,3		109
=NYLEGN	(CBUF)	2,3		23
=NYPOSN	(Y)	2,3		109

Routine	Parameter	Level		Page
=NZPOSN	(Z)	3		149
OPENFL	(CFIL, NLU, IRW, ISTAT)	0,1,2,3		117
OPNWIN	(ID)	1,2,3		119
ORIGIN	(NX0, NY0)	1	ps	28
PAGE	(NWPAGE, NHPAGE)	0	ps	32
PAGERA	none	1,2,3	p	13
PAGFLL	(NCLR)	1,2,3	p	13
PAGHDR	(CSTR1, CSTR2, IOPT, IDIR)	1,2,3	p	13
PAGMOD	(CMOD)	0	ps	33
PAGORG	(COPT)	1,2,3	ps	28
PATCYC	(INDEX, IPAT)	1,2,3	ps	90
PDFBUF	(CBUF, NMAX, N)	0		127
PDFMOD	(CMOD, CKEY)	0	ps	31
PDFMRK	(CSTR, COPT)	1,2,3	ps	31
PENWID	(XWIDTH)	1,2,3	ps	87
PIE	(NXM, NYM, NR, ALPHA, BETA)	1,2,3	p	106
PIEBOR	(ICLR)	1,2,3	ps	136
PIECLR	(IC1RAY, IC2RAY, N)	1,2,3	ps	136
PIEEXP	none	1,2,3	ps	137
PIEGRF	(CBUF, NLIN, XRAY, NSEG)	1	p	134
PIELAB	(CLAB, CPOS)	1,2,3	ps	137
PIEOPT	(XF, ANG)	1,2,3	ps	137
PIETYP	(CTYP)	1,2,3	ps	134
PIEVEC	(IVEC, COPT)	1,2,3	ps	137
PNGMOD	(CMOD, CKEY)	0	ps	31
POINT	(NX, NY, NB, NH, NCOL)	1,2,3	p	148
POLAR	(XE, XOR, XSTP, YOR, YSTP)	1	p	16
POLCRV	(CPOL)	1,2,3	ps	85
POLMOD	(CPOS, CDIR)	1,2,3	ps	46
POS2PT	(X, Y, XP, YP)	2		187
POS3PT	(X, Y, Z, XABS, YABS, ZABS)	3		171
POSIFL	(NLU, NBYTE, ISTAT)	0,1,2,3		118
PROJCT	(CPROJ)	1	ps	181
PSFONT	(CFONT)	1,2,3	ps	58
PSMODE	(CMODE)	1,2,3	ps	61
QPLBAR	(XRAY, N)	0,1	p	233
QPLCLR	(ZMAT, IXDIM, IYDIM)	0,1	p	234
QPLCON	(ZMAT, IXDIM, IYDIM, NLEV)	0,1	p	234
QPLOT	(XRAY, YRAY, N)	0,1	p	233
QPLPIE	(XRAY, N)	0,1	p	234
QPLSCA	(XRAY, YRAY, N)	0,1	p	233
QPLSUR	(ZMAT, IXDIM, IYDIM)	0,1	p	234
RBFPNG	(CBUF, NMAX, N)	1,2,3		126
RBMP	(CFIL)	1,2,3		127
READFL	(NLU, IRAY, NBYTE, ISTAT)	0,1,2,3		117
REAWGT	none	0	w	226
RECFLL	(NX, NY, NW, NH, NCOL)	1,2,3	p	147
RECTAN	(NX, NY, NW, NH)	1,2,3	p	105
REL3PT	(X, Y, Z, XP, YP)	3		171

Routine	Parameter	Level		Page
RESATT	none	1,2,3	ps	83
RESET	(CNAME)	1,2,3	ps	27
RGBHSV	(XR, XG, XB, XH, XS, XV)	0,1,2,3		55
RGIF	(CFIL)	1,2,3		124
RGTLAB	none	1,2,3	ps	46
RIMAGE	(CFIL)	1,2,3		124
RLARC	(XM, YM, XA, XB, A, B, T)	2,3	p	107
RLAREA	(XRAY, YRAY, N)	2,3	p	107
RLCIRC	(XM, YM, R)	2,3	p	107
RLCONN	(X, Y)	2,3	p	103
RLELL	(XM, YM, A, B)	2,3	p	107
RLINE	(X, Y, U, V)	2,3	p	104
RLMESS	(CSTR, X, Y)	2,3	p	12
RLNUMB	(X, NDIG, XP, YP)	2,3	p	12
RLPIE	(XM, YM, R, ALPHA, BETA)	2,3	p	107
RLPOIN	(X, Y, NB, NH, NCOL)	2,3	p	148
RLREC	(X, Y, WIDTH, HEIGHT)	2,3	p	107
RLRND	(X, Y, WIDTH, HEIGHT, IOPT)	2,3	p	107
RLSEC	(XM, YM, R1, R2, ALPHA, BETA, NCOL)	2,3	p	148
RLSTRT	(X, Y)	2,3	p	103
RLSYMB	(NSYM,X,Y)	2,3	p	12
RLVEC	(X1, Y1, X2, Y2, IVEC)	2,3	p	105
RLWIND	(X,XP,YP,NW,A)	2,3	p	105
RNDREC	(NX, NY, NW, NH, IOPT)	1,2,3	p	106
RPIXEL	(IX, IY, NCLR)	1,2,3		122
RPIXLS	(IRAY, IX, IY, NW, NH)	1,2,3		123
RPNG	(CFIL)	1,2,3		126
RPPM	(CFIL)	1,2,3		126
RPXROW	(IRAY, IX, IY, N)	1,2,3		123
RTIFF	(CFIL)	1,2,3		125
RVYNAM	none	1,2,3	ps	47
SCALE	(CSCL,CAX)	1,2,3	ps	41
SCLFAC	(XFAC)	0	ps	33
SCLMOD	(CMODE)	0	ps	33
SCRMOD	(CMODE)	0	ps	29
SECTOR	(NX, NY, NR1, NR2, ALPHA, BETA, NCOL)	1,2,3	p	148
SELWIN	(ID)	1,2,3	ps	119
SENDBF	none	1,2,3		118
SENDMB	none	1,2,3		226
SENDOK	none	0		226
SERIF	none	1,2,3	ps	58
SETBAS	(XFAC)	1,2,3	ps	71
SETCBK	(ROUTINE, COPT)	0,1,2,3	ps	188
SETCLR	(NCOL)	1,2,3	ps	53
SETCSR	(COPT)	1,2,3	ps	122
SETEXP	(FEXP)	1,2,3	ps	71
SETFIL	(CFIL)	0		29
SETGRF	(C1, C2, C3, C4)	1	ps	48
SETIND	(INDEX, XR, XG, XB)	1,2,3	ps	54

Routine	Parameter	Level		Page
SETMIX	(CHAR, CMIX)	1,2,3	ps	72
SETPAG	(CPAG)	0		32
SETRES	(NPB, NPH)	1,2,3	ps	145
SETRGB	(XR, XG, XB)	1,2,3	ps	53
SETSCL	(XRAY, N, CAX)	1,2,3	ps	41
SETVLT	(CVLT)	1,2,3	ps	53
SETXID	(ID,COPT)	0	ps	38
SHDAFR	(INRAY, IPRAY, ICRAY, N)	2	p	183
SHDCHA	none	1,2,3	ps	58
SHDCRV	(X1RAY, Y1RAY, N1, X2RAY, Y2RAY, N2)	2,3	p	24
SHDEUR	(INRAY, IPRAY, ICRAY, N)	2	p	183
SHDMAP	(CMAP)	2,3	p	182
SHDMOD	(CMOD, CTYPE)	1,2,3	ps	162, 203
SHDPAT	(IPAT)	1,2,3	ps	88
SHDUSA	(INRAY, IPRAY, ICRAY, N)	2	p	184
SHIELD	(CAREA, CMODE)	1,2,3	ps	91
SHLCIR	(NX, NY, NR)	1,2,3	ps	91
SHLDEL	(ID)	1,2,3	ps	81
SHLELL	(NX, NY, NA, NB, THETA)	1,2,3	ps	91
SHLIND	(ID)	1,2,3	ps	92
SHLPIE	(NX, NY, NR, A, B)	1,2,3	ps	91
SHLPOL	(NXRAY, NYRAY, N)	1,2,3	ps	91
SHLRCT	(NX, NY, NW, NH, THETA)	1,2,3	ps	91
SHLREC	(NX, NY, NW, NH)	1,2,3	ps	91
SHLRES	(N)	1,2,3	ps	92
SHLSUR	none	1,2,3	ps	161
SHLVIS	(ID, CMODE)	1,2,3	ps	92
SIMPLX	none	1,2,3	ps	58
SKIPFL	(NLU, NBYTE, ISTAT)	0,1,2,3		118
SMXALF	(CALPH, C1, C2, N)	1,2,3	ps	60
SOLID	none	1,2,3	ps	86
SORTR1	(XRAY, N, COPT)	0,1,2,3		112
SORTR2	(XRAY, YRAY, N, COPT)	0,1,2,3		112
SPHE3D	(XM, YM, ZM, R, N, M)	3	p	171
SPLINE	(XRAY, YRAY, N, XSRAY, YSRAY, NSPL)	1,2,3		113
SPLMOD	(NGRAD, NPTS)	1,2,3	p	85
STRT3D	(X, Y, Z)	3	p	170
STRTPT	(X, Y)	1,2,3	p	103
SURCLR	(ICTOP, ICBOT)	1,2,3	ps	162
SURFCE	(XRAY, N, YRAY, M, ZMAT)	3	p	157
SURFCP	(ZFUN, T1, T2, TSTP, U1, U2, USTP)	3	p	159
SURFUN	(ZFUN, IXPTS, XD, IYPTS, YD)	3	p	157
SURISO	(XRAY, NX, YRAY, NY, ZRAY, NZ, WMAT, WLEV)	3	p	160
SURMAT	(ZMAT, NX, NY, IXPTS, IYPTS)	3	p	157
SURMSH	(COPT)	1,2,3	ps	162
SUROPT	(COPT)	1,2,3	ps	161
SURSHD	(XRAY, N, YRAY, M, ZMAT)	3	p	158
SURSZE	(XMIN, XMAX, YMIN, YMAX)	1,2,3	ps	158

Routine	Parameter	Level		Page
SURTRI	(XRAY, YRAY, ZRAY, N, I1RAY, I2RAY, I3RAY, NTRI)	3	p	159
SURVIS	(CVIS)	1,2,3	ps	161
SWAPI2	(IRAY2, N)	0,1,2,3		116
SWAPI4	(IRAY, N)	0,1,2,3		116
SWGATT	(ID, CATT, COPT)	0	ps	222
SWGBOX	(ID, ISEL)	0	ps	222
SWGBUT	(ID, IVAL)	0	ps	222
SWGCBK	(IP, ROUTINE)	0	ps	221
SWGCLR	(XR, XG, XB, COPT)	0	ps	217
SWGDRW	(XF)	0	ps	217
SWGFIL	(ID, CFIL)	0	ps	223
SWGFNT	(CFONT, NPTS)	0	ps	217
SWGFOC	(ID)	0	ps	218
SWGHLP	(CSTR)	0	ps	219
SWGJUS	(CJUS, CLASS)	0	ps	220
SWGLIS	(ID, ISEL)	0	ps	222
SWGMIX	(CHAR, CMIX)	0	ps	221
SWGMRG	(IVAL, CMRG)	0	ps	221
SWGOPT	(COPT, CKEY)	0	ps	218
SWGPOP	(COPT)	0	ps	219
SWGPOS	(NX, NY)	0	ps	219
SWGSCL	(ID, XVAL)	0	ps	223
SWGSIZ	(NW, NH)	0	ps	219
SWGSPC	(XSPC, YSPC)	0	ps	221
SWGSTP	(XSTP)	0	ps	221
SWGTIT	(CTIT)	0	ps	219
SWGTXT	(ID, CSTR)	0	ps	222
SWGTYP	(CTYPE, CLASS)	0	ps	220
SWGWIN	(NX, NY, NW, NH)	0	ps	220
SWGWTH	(NWTH)	0	ps	217
SYMBOL	(NSYM, NX, NY)	1,2,3	p	12
SYMFIL	(CDEV, CSTAT)	0		13
SYMROT	(ANGLE)	1,2,3	ps	12
TELLFL	(NLU, NBYTE)	0,1,2,3		118
TEXMOD	(CMODE)	1,2,3	ps	76
TEXOPT	(COPT, CTYPE)	1,2,3	ps	76
TEXVAL	(X, COPT)	1,2,3	ps	76
THKCRV	(NTHK)	1,2,3	ps	85
THRINI	(N)	0		127
THRFIN	none	0		127
TICKS	(ITICK, CAX)	1,2,3	ps	41
TICLEN	(NMAJ, NMIN)	1,2,3	ps	42
TICMOD	(CMOD, CAX)	1,2,3	ps	42
TICPOS	(CPOS, CAX)	1,2,3	ps	41
TIFMOD	(N, CVAL, COPT)	0	ps	30
TIFORG	(NX, NY)	1,2,3	ps	125
TIFWIN	(NX, NY, NW, NH)	1,2,3	ps	126
TIMOPT	none	1,2,3	ps	46

Routine	Parameter	Level		Page
TITJUS	(CJUS)	1,2,3	ps	51
TITLE	none	2,3	p	16
TITLIN	(CSTR, N)	1,2,3	ps	51
TITPOS	(CPOS)	1,2,3	ps	51
TRFCO1	(XRAY, N, CFROM, CTO)	0,1,2,3		110
TRFCO2	(XRAY, YRAY, N, CFROM, CTO)	0,1,2,3		110
TRFCO3	(XRAY, YRAY, ZRAY,N, CFROM, CTO)	0,1,2,3		110
TRFDAT	(N, ID, IM, IY)	0,1,2,3		115
TRFMAT	(ZMAT, NX, NY, ZMAT2, NX2, NY2)	0,1,2,3		110
TRFREL	(XRAY, YRAY, N)	2,3		109
TRFRES	none	1,2,3	ps	90
TRFROT	(XANG, NX, NY)	1,2,3	ps	90
TRFSCL	(XSCL, YSCL)	1,2,3	ps	90
TRFSHF	(NXSHFT, NYSHFT)	1,2,3	ps	90
TRIANG	(XRAY, YRAY, N, I1RAY, I2RAY, I3RAY, NMAX, NTRI)	0,1,2,3		114
TRIFLL	(XRAY, YRAY)	1,2,3	p	105
TRIPLX	none	1,2,3	ps	58
TRIPTS	(XRAY, YRAY, ZVRAY, N, I1RAY, I2RAY, I3RAY, NTRI, ZLEV, XPTS, YPTS, MAXPTS, IRAY, MAXCRV, NCRV)	0,1,2,3		201
= TRMLEN	(CSTR)	0,1,2,3		111
TXTJUS	(CJUS)	1,2,3	ps	55
UNIT	(NU)	1,2,3	ps	35
UNITS	(COPT)	0	ps	27
UPSTR	(CSTR)	0,1,2,3		111
USRPIE	(ISEG, XD, XP, NR, NOFF, ANG, NVY, IDRW, IANN)	1,2,3	ps	138
VANG3D	(ANG)	1,2,3	ps	155
VCLP3D	(XCLP1, XCLP2)	1,2,3	ps	163
VECTOR	(NX1, NY1, NX2, NY2, IVEC)	1,2,3	p	104
VECTR3	(X1, Y1, Z1, X2, Y2, Z2, IVEC)	3	p	170
VFOC3D	(X, Y, Z, COPT)	1,2,3	ps	155
VIEW3D	(XVU, YVU, ZVU, CVU)	1,2,3	ps	154
VKXBAR	(NVFX)	1,2,3	ps	146
VKYBAR	(NVFY)	1,2,3	ps	146
VKYTIT	(NV)	1,2,3	ps	52
VLTFIL	(CFIL, CMODE)	1,2,3		54
VUP3D	(ANG)	1,2,3	ps	155
WGAPP	(IP, CLAB, ID)	0	w	212
WGBAS	(IP, COPT, ID)	0	w	212
WGBOX	(IP, CLIS, ISEL, ID)	0	w	215
WGBUT	(IP, CLAB, IVAL, ID)	0	w	213
WGCMD	(IP, CLAB, CMD, ID)	0	w	216
WGDLIS	(IP, CLIS, ISEL, ID)	0	w	214
WGDRAW	(IP)	0	w	215
WGFIL	(IP, CLAB, CFIL, CMASK, ID)	0	w	214
WGFIN	none)	0	w	211

Routine	Parameter	Level		Page
WGINI	(COPT, ID)	0	w	211
WGLAB	(IP, CSTR, ID)	0	w	212
WGLIS	(IP, CLIS, ISEL, ID)	0	w	214
WGLTXT	(IP, CLAB, CSTR, NWTH, ID)	0	w	213
WGOK	(IP, ID)	0	w	216
WGPBUT	(IP, CLAB, ID)	0	w	216
WGPOP	(IP, CLAB, ID)	0	w	212
WGQUIT	(IP, ID)	0	w	216
WGSCL	(IP, CLAB, XMIN, XMAX, XVAL, NDEZ, ID)	0	w	215
WGSTXT	(IP, NSIZE, NMAX, ID)	0	w	213
WGTXT	(IP, CSTR, ID)	0	w	213
WIDBAR	(NZB)	1,2,3	ps	146
WIMAGE	(CFIL)	1,2,3	p	125
WINAPP	(CAPP)	0	ps	35
WINDBR	(X,NXP,NYP,NW,A)	1,2,3	p	105
WINDOW	(NX, NY, NW, NH)	0	ps	36
WINFNT	(CFONT)	1,2,3	ps	59
WINID	(ID)	1,2,3	rq	119
WINKEY	(COPT)	1,2,3	ps	37
WINMOD	(CMOD)	1,2,3	ps	37
WINOPT	(IOPT, CKEY)	0	ps	37
WINSIZ	(NW, NH)	0,1,2,3	ps	36
WINTIT	(CTIT)	1,2,3	ps	120
WMFMOD	(CMOD, CKEY)	0	ps	30
WORLD	none	2,3	p	182
WPIXEL	(IX, IY, NCLR)	1,2,3	p	123
WPIXLS	(IRAY, IX, IY, NW, NH)	1,2,3	p	123
WPXROW	(IRAY, IX, IY, N)	1,2,3	p	123
WRITFL	(NLU, IRAY, NBYTE, ISTAT)	0,1,2,3		117
WTIFF	(CFIL)	1,2,3	p	125
X11FNT	(CFONT,COPT)	1,2,3	ps	59
X11MOD	(CMOD)	0	ps	36
= X2DPOS	(X, Y)	2		187
= X3DABS	(X, Y, Z)	3		172
= X3DPOS	(X, Y, Z)	3		171
= X3DREL	(X, Y, Z)	3		171
XAXGIT	none	2,3	p	18
XAXIS	(XA, XE, XOR, XSTP, NL, CX, IT, NX, NY)	1,2,3	p	18
XAXLG	(XA, XE, XOR, XSTP, NL, CX, IT, NX, NY)	1,2,3	p	18
XAXMAP	(XA, XE, XOR, XSTP, CX, IT, NY)	2	p	179
XCROSS	none	2,3	p	18
XDRAW	(X, Y)	1,2,3	p	103
=XINVRS	(NXP)	2,3		109
XMOVE	(X, Y)	1,2,3	p	103
=XPOSN	(X)	2,3		109
= Y2DPOS	(X, Y)	2		187
= Y3DABS	(X, Y, Z)	3		172
= Y3DPOS	(X, Y, Z)	3		171