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 (Albert).

= '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.

Default: CTYPE = 'CYLI'.

Additional notes: - For cylindrical equidistant projections, the scaling parameters in GRAFMP must be in the range:

$$-540 \leq XA \leq XE \leq 540$$
$$-180 \leq YA \leq YE \leq 180$$

For Mercator projections:

$$-540 \leq XA \leq XE \leq 540$$
$$-\ 85 \leq YA \leq YE \leq\ 85$$

For cylindrical equal-area projections:

$$-540 \leq XA \leq XE \leq 540$$
$$-\ 90 \leq YA \leq YE \leq\ 90$$

For elliptical projections:

$$-180 \leq XA \leq XE \leq 180$$
$$-\ 90 \leq YA \leq YE \leq\ 90$$

For conical projections:

$$-180 \leq XA \leq XE \leq 180$$
$$0 \leq YA \leq YE \leq 90 \text{ or}$$
$$-90 \leq YA \leq YE \leq 0$$

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 $\leq$ 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:

$$-180 \leq XA \leq XE \leq 180 \text{ and}$$
$$XE - XA \leq 180$$
$$-90 \leq YA \leq YE \leq 90$$

- 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 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	13: Iceland	24: Portugal
2: Andorra	14: Italy	25: Romania
3: Belgium	15: Yugoslavia	26: Sweden
4: Bulgaria	16: Liechtenstein	27: Switzerland
5: Germany	17: Luxembourg	28: Spain
6: Denmark	18: Malta	29: CSFR
8: England/GB	19: Netherlands	30: Turkey
9: Finland	20: North Ireland	31: USSR
10: France	21: Norway	32: Hungary
11: Greece	22: Austria	
12: Ireland	23: Poland	

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.

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. A DISLIN map file and some external map files compiled by Paul Wessel can be used. The external map files can be copied via FTP anonymous from the servers

ftp://ftp.ngdc.noaa.gov/MGG/shorelines/
ftp://kiawe.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.

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.

Default: CBAS = 'DISLIN'.

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 'BOTH', 'LAND' and 'LAKE'.
Default: COPT = 'BOTH'.

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 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
```

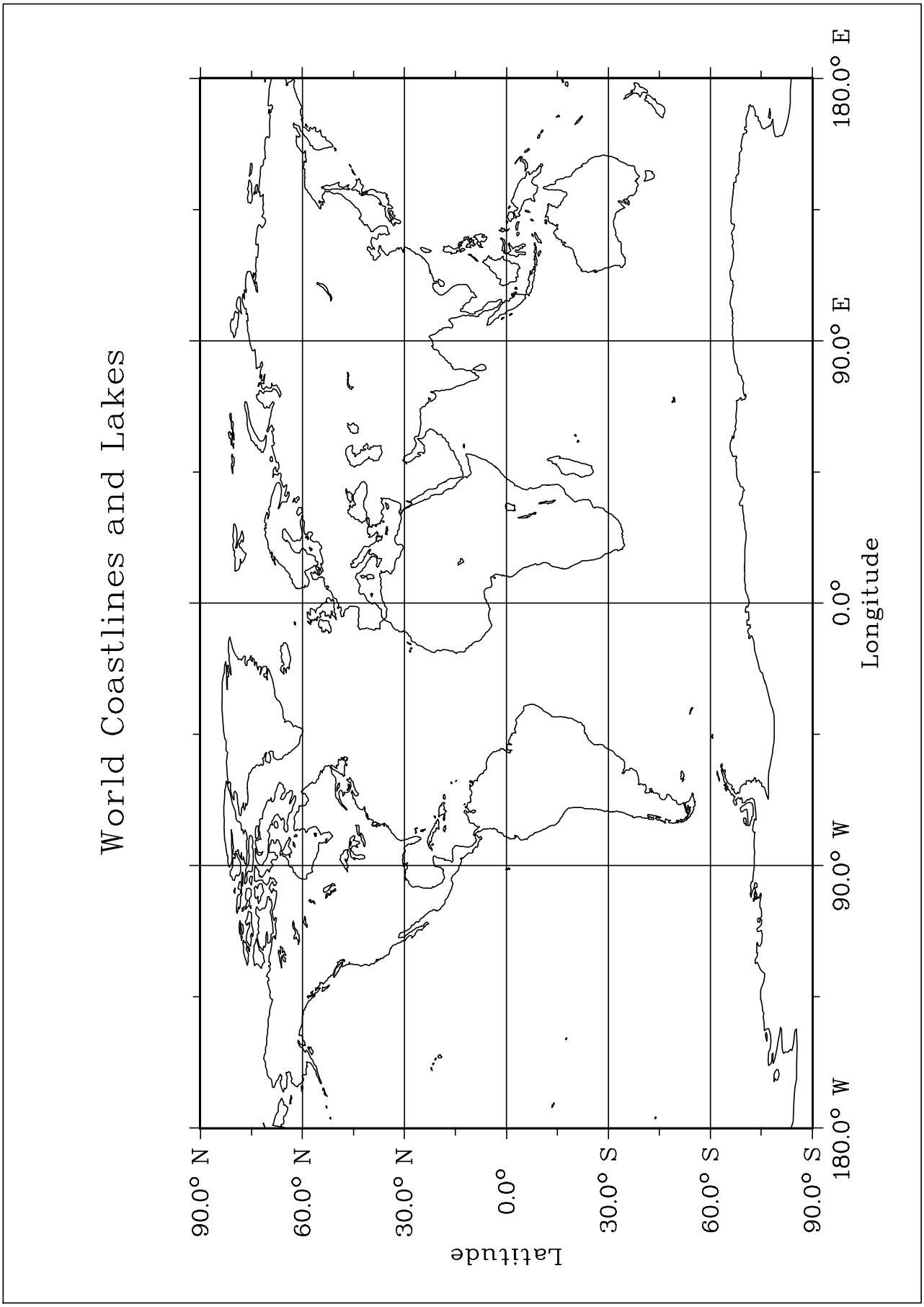

Figure 13.1: World Coastlines and Lakes

```
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
```

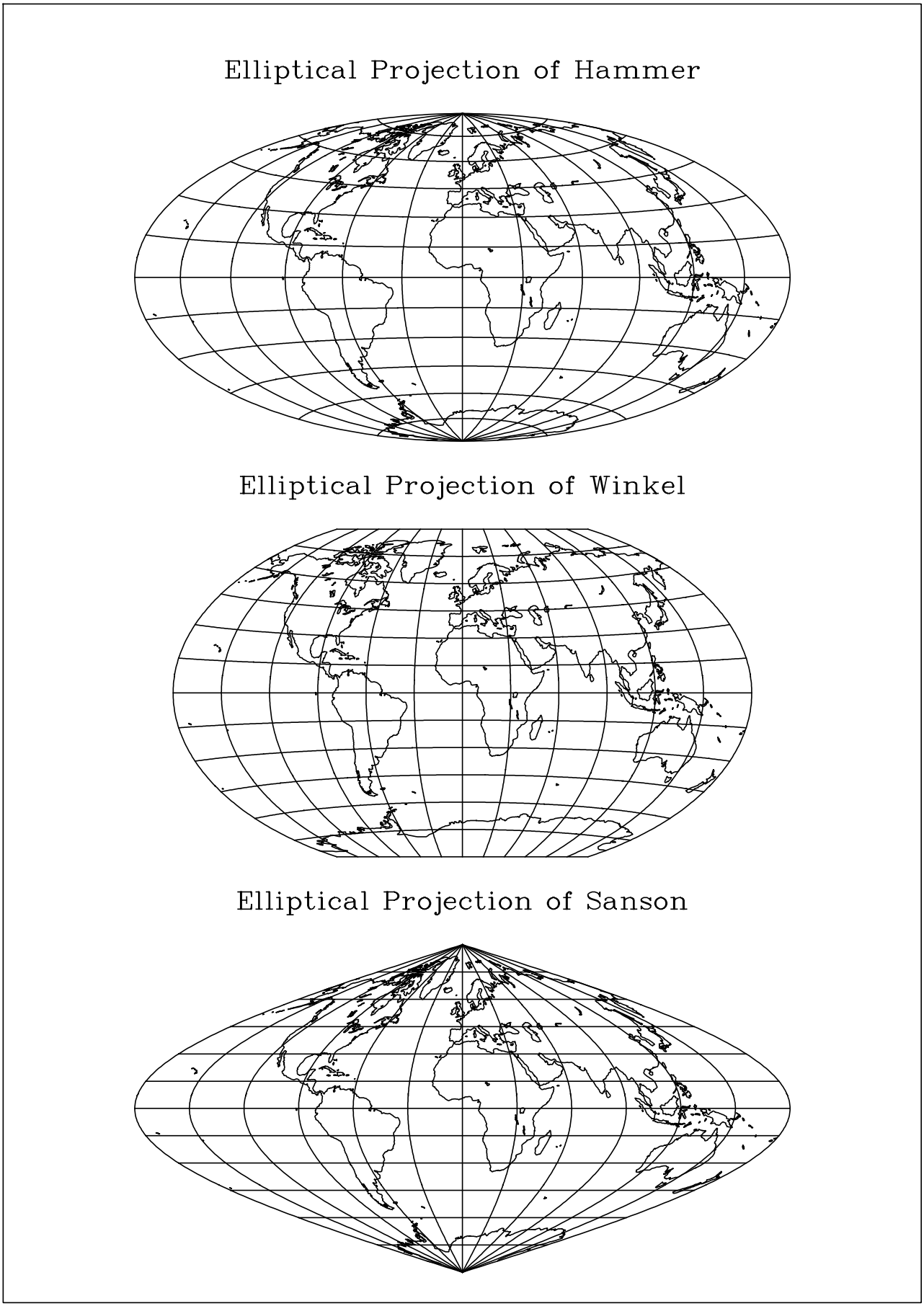

Figure 13.2: Elliptical Projections

```
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
```

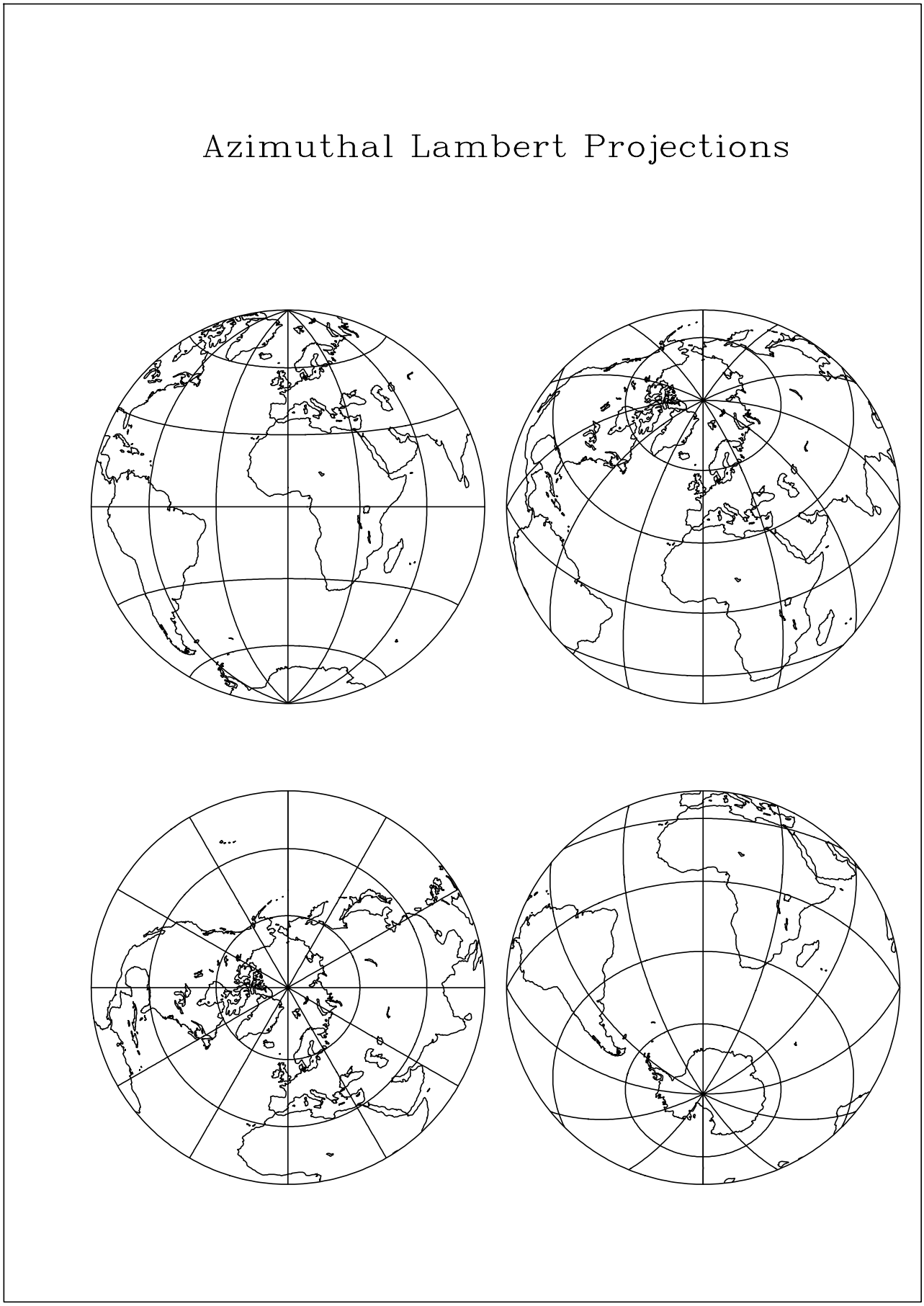

Figure 13.3: Azimuthal Lambert Projections

```
PROGRAM EX13_4
PARAMETER (N = 32)
DIMENSION INRAY(N),IPRAY(N),ICRAY(N)

DO I=1,N
  INRAY(I)=I
  IPRAY(I)=0
  ICRAY(I)=1
END DO

CALL SETPAG('DA4P')
CALL DISINI
CALL SETVLT ('SMALL')
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
```

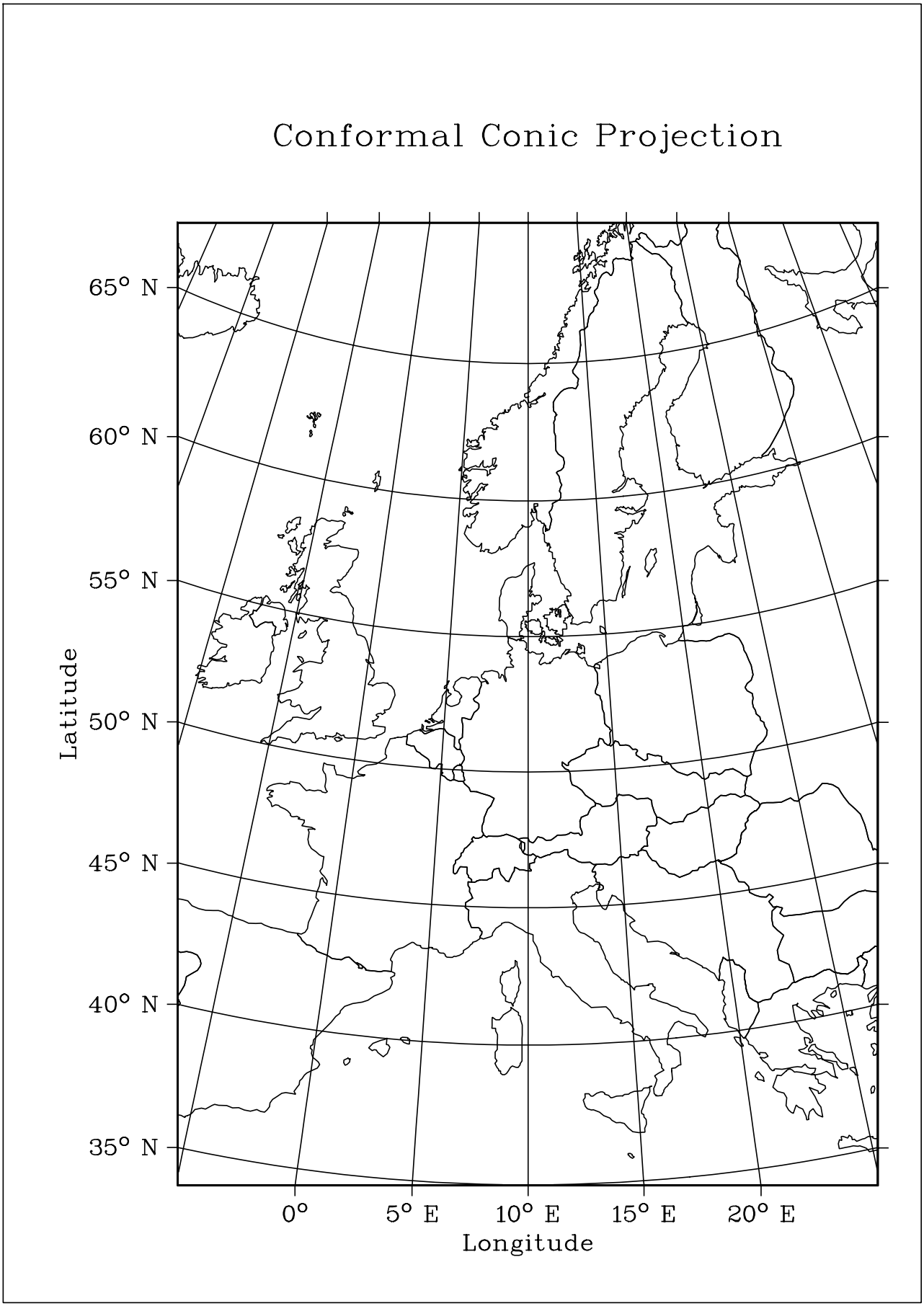

Figure 13.4: Conformal Conic Projection

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.
- 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.

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, NLEV) level 2, 3

or: void conshd (float *xray, int n, float *yray, int m, float *zmat, float *zlvray, int nlev);

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. The levels must be sorted in ascending order, if cell filling is selected. For polygon filling, the levels should be sorted in such a way that inner contours are plotted last.
NLEV	is the number of levels.

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
```

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 DIGITS (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 number between -1 and 255. 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 defines an algorithm used for contour filling.

The call is:	CALL SHDMOD (COPT, 'CONTUR')	level 1, 2, 3
or:	void shdmod (char *copt, "CONTUR");	

COPT — is a character string defining the algorithm.

= 'CELL' — defines cell filling.

= 'POLY' — defines polygon filling.

Default: COPT = 'CELL'.

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
```

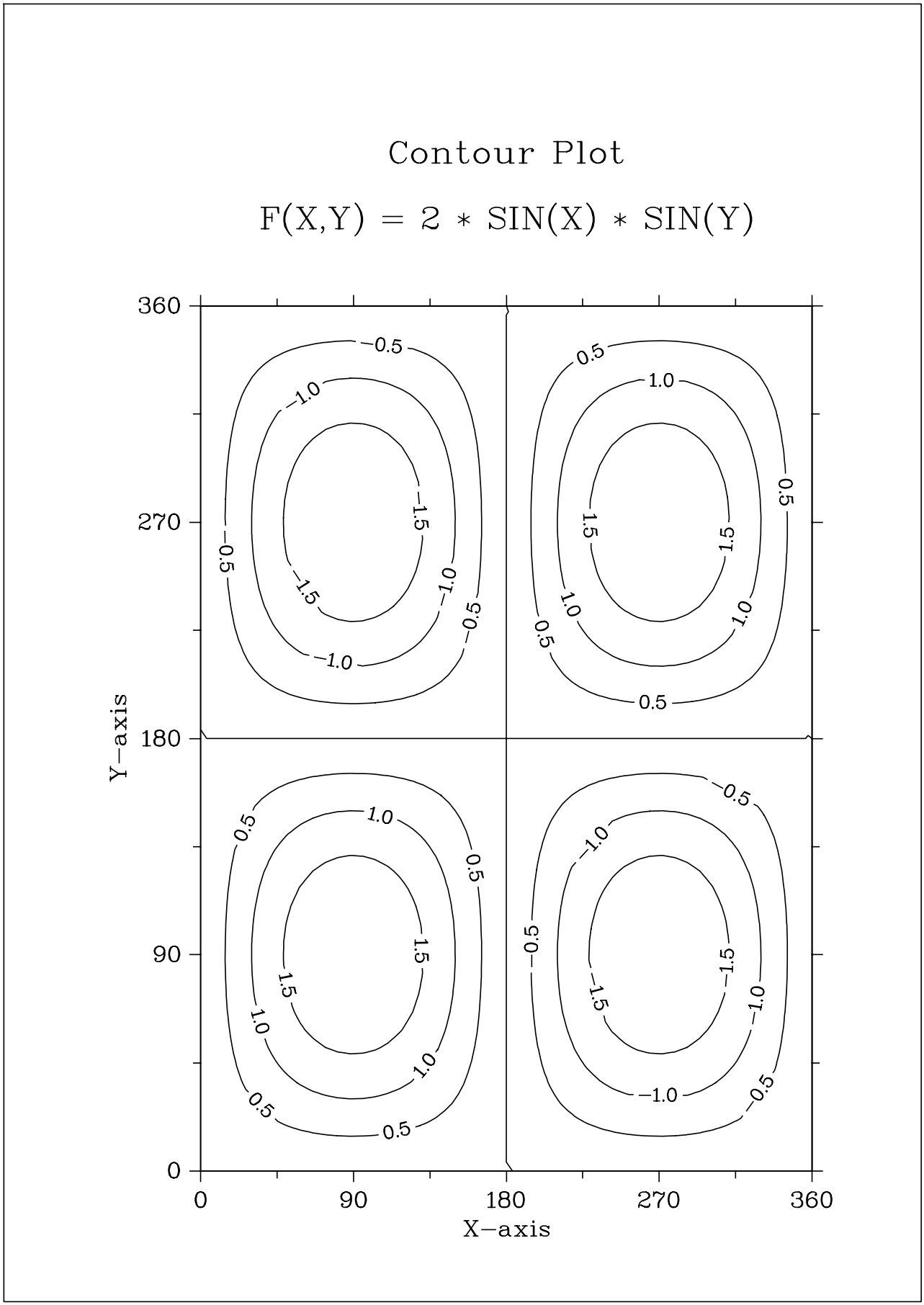

Figure 14.1: Contour Plot

```
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
```

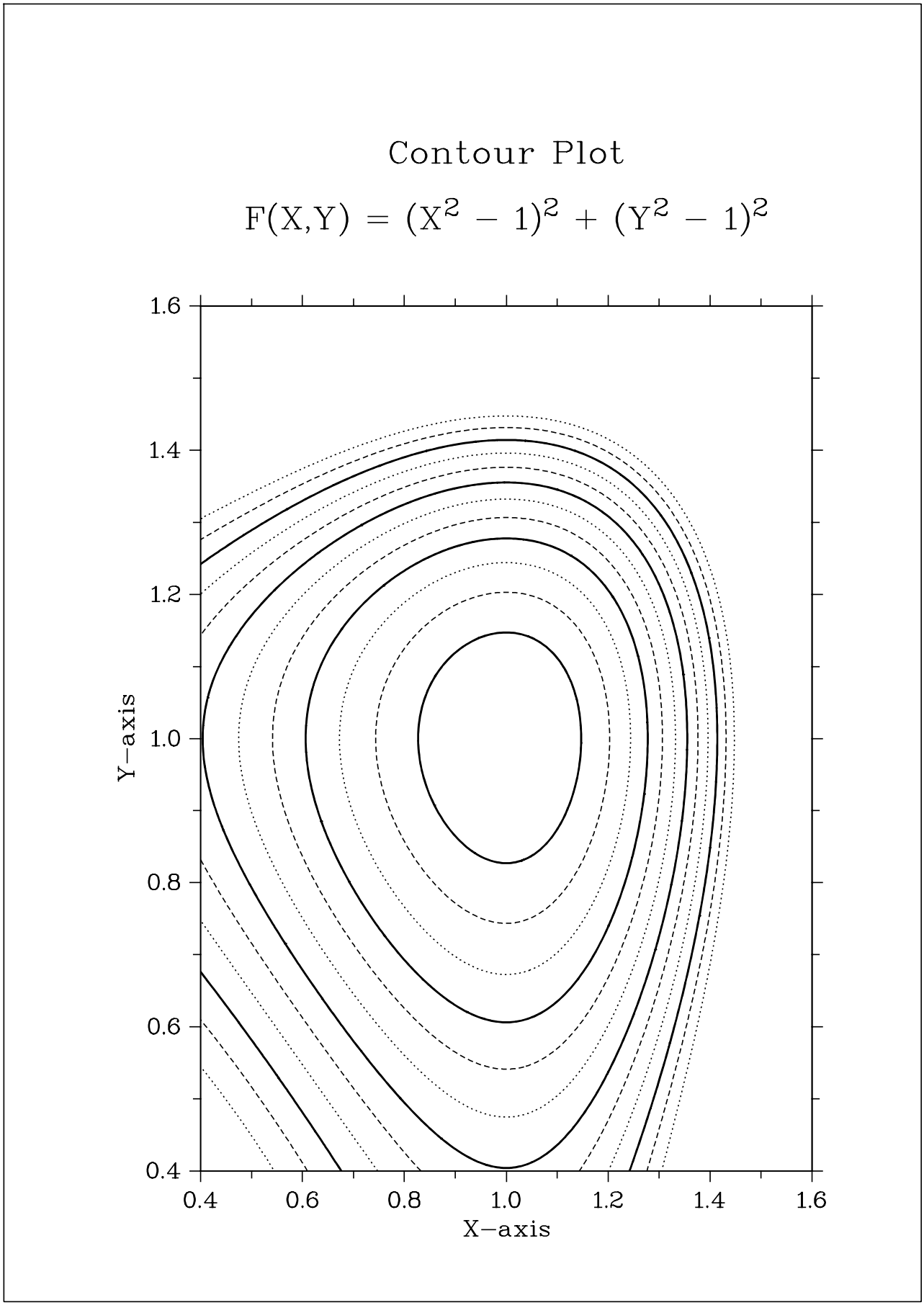

Figure 14.2: Contour Plot

```
 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
```

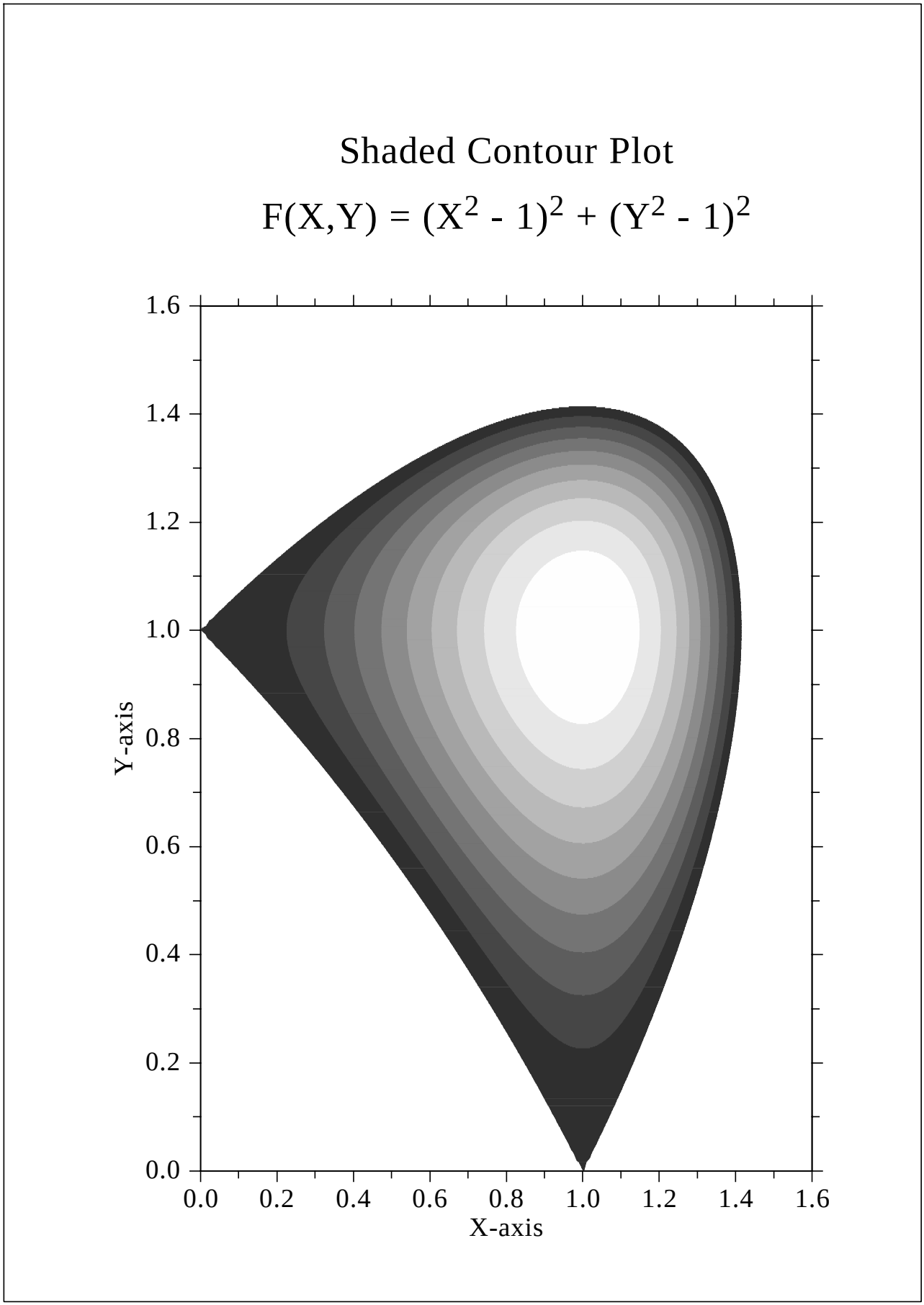

Figure 14.3: Shaded Contour Plot

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 95/98/NT 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. 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 the parent widget where the parent widget must be created with WGINI.

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.

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 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.
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.
NWTH	defines the width of the text field ($0 \leq$ NWTH ≤ 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.
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.

W G B O X

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 ($\geq$ 1).
ID	is the returned widget index.

W G S C L

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.

W G D R A W

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 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. 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.

Default: ('STANDARD', 'POSITION').

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.

= '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'), ('LEFT', '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 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, box and text widgets, and for popup menu entries.

The call is: CALL SWGCBK (ID, ROUTINE)

or: void swgcbk (int id, void (*routine)());

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'.

COPT is a character string that can have the value 'STATUS'.

S W G B U T

The routine SWGBUT sets the status of a button widget.

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.

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.

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

G E T D S P

The routine GETDSP returns the terminal type.

The call is:	CALL GETDSP (CDSP)
or:	char *getdsp ();
CDSP	is a returned character string that can have the values 'XWIN' for X Window terminals and 'NOXW' for non X Window terminals.

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.

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. At the moment, SENDOK is just available in the Windows 95/NT versions of DISLIN.

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);

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'.

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.

CSTR — is the returned input of the user.

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
```

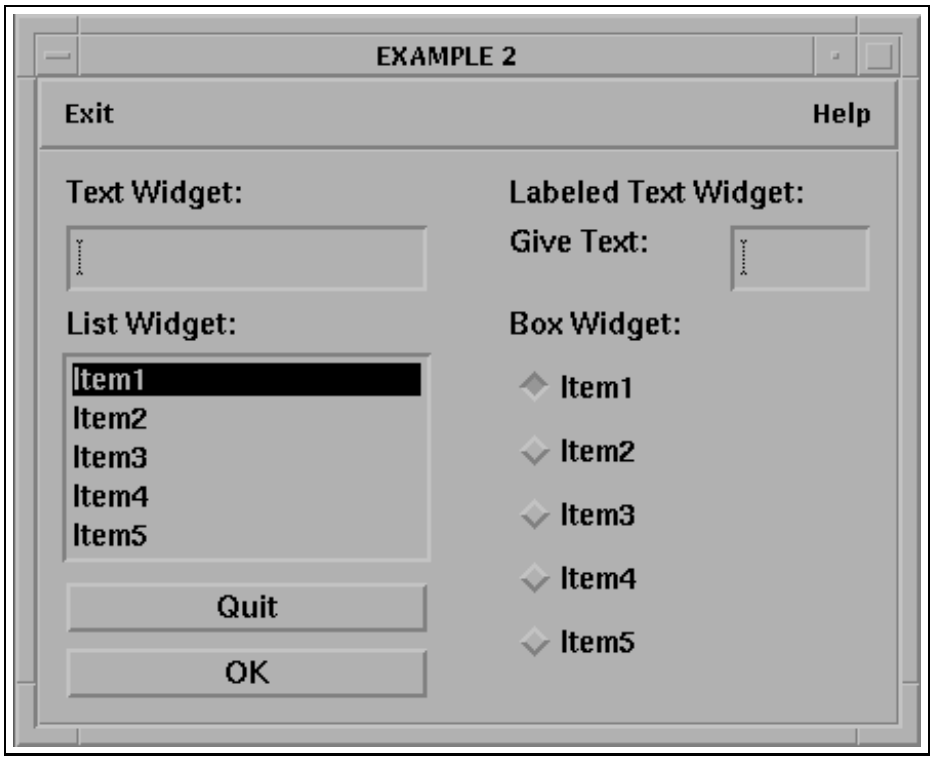

Figure 15.2: Widgets

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
```

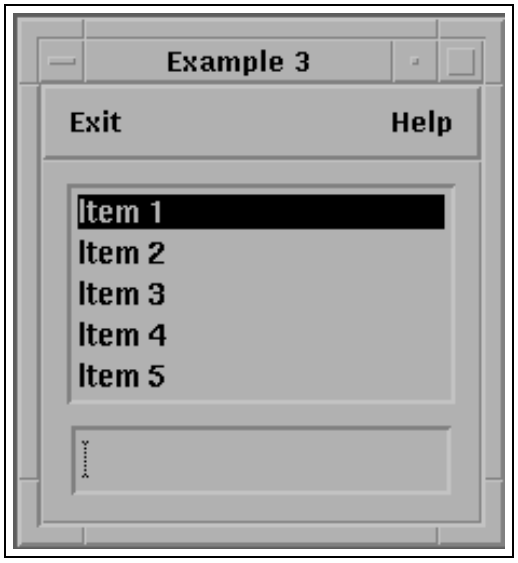

Figure 15.3: Widgets

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 send 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
```

Chapter 17

MPAe Emblem

This chapter describes routines for plotting and modifying the MPAe emblem.

17.1 Plotting the MPAe Emblem

M P A E P L

The routine MPAEPL plots the MPAe emblem.

The call is:	CALL MPAEPL (IOPT)
or:	void mpaepl (int iopt);
IOPT	defines the position of the MPAe emblem:
= 1	defines the lower left corner of the page.
= 2	defines the lower right corner of the page.
= 3	defines the upper right corner of the page.
= 4	defines the upper left corner of the page.

17.2 Parameter Setting Routines

M P L P O S

The routine MPLPOS defines a global position of the MPAe emblem. The parameter in MPAEPL will be ignored.

The call is:	CALL MPLPOS (NX, NY)
or:	void mplpos (int nx, int ny);
NX, NY	are the plot coordinates of the upper left corner.

M P L C L R

The routine MPLCLR defines the fore- and background colours of the MPAe emblem.

The call is:	CALL MPLCLR (NBG, NFG)
or:	void mplclr (int nbg, int nfg);

NBG, NFG are the back- and foreground colours.

Default: (192/132).

M P L S I Z

MPLSIZ defines the size of the MPAe emblem.

The call is: CALL MPLSIZ (NSIZE)

or: void mplsiz (int nsize);

NSIZE is the size in plot coordinates.

Default: 300.

M P L A N G

MPLANG defines a rotation angle for the MPAe emblem.

The call is: CALL MPLANG (XANG)

or: void mplang (float xang);

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

Default: XANG = 0.

N O F I L L

A call to NOFILL suppresses the shading of the MPAe emblem.

The call is: CALL NOFILL

or: void nofill ();