

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.

GETPAG

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);

GETFIL

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.

GETMFL

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.

GETOR

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);

GETPOS

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

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

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

GETLEN

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);

GETHGT

GETHGT returns the character height (see HEIGHT).

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

or: int gethgt ();

GETANG

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 ();

GETALF

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.

GETMIX

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.

GETSHF

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

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

or: int *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).

GETSCL

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 SCALE).

The call is: CALL GETSCL (NXLOG, NYLOG, NZLOG) level 1, 2, 3
or: void getscl (int *nxlog, int *nylog, int *nzlog);

GETLAB

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);

GETCLR

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 ();

GETUNI

GETUNI returns the logical unit used for error messages.

The call is: CALL GETUNI (NU) level 1, 2, 3
or: FILE *getuni ();

GETVER

GETVER returns the version number of the currently used DISLIN library.

The call is: CALL GETVER (XVER) level 1, 2, 3
or: float getver ();

GETPLV

GETPLV returns the patch level of the currently used DISLIN library.

The call is: CALL GETPLV (IPLV) level 1, 2, 3
or: int getplv ();

GETLEV

GETLEV returns the level.

The call is: CALL GETLEV (NLEV) level 1, 2, 3
or: int getlev ();

GETSYM

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);

GETTYP

GETTYP returns the current line style (see LINTYP).

The call is: CALL GETTYP (NTYP) level 1, 2, 3
or: int gettyp ();

GETLIN

The routine GETLIN returns the current line width (see LINWID).

The call is: CALL GETLIN (NWIDTH) level 1, 2, 3
or: int getlin ();

GETPAT

The routine GETPAT returns the current shading pattern (see SHDPAT).

The call is: CALL GETPAT (NPAT) level 1, 2, 3
or: long getpat ();

GETRES

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);

GETVLT

GETVLT returns the current colour table (see SETVLT).

The call is: CALL GETVLT (CVLT) level 1, 2, 3
or: char *getvlt ();

GETIND

For a colour index, the routine GETIND returns the corresponding RGB coordinates stored in the current colour table (see SETIND).

The call is: CALL GETIND (I, XR, XG, XB) level 1, 2, 3
or: void getind (int i, float *xr, float *xg, float *xb);

GETRGB

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);

GETSCR

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);

GETBPP

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 ();

GETDSP

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.

GETRAN

GETRAN returns the colour range of colour bars (see SETRAN).

The call is: CALL GETRAN (NCA, NCE) level 1, 2, 3
or: void getran (int *nca, int *nce);

GETWID

GETWID returns the width of the colour bar plotted in 3-D colour graphics (see BARWID).

The call is: CALL GETWID (NZB) level 1, 2, 3
or: int getwid ();

GETVK

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);

GETWIN

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);

GETCLP

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);

GETXID

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'.

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.

LINE

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.

RLINE

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 NOCHECK is not used, points lying outside the axis scaling will be listed.

8.2 Vectors

VECTOR

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.

RLVEC

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 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.

RECTAN

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.

RNDREC

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 ($0 \leq \text{IOPT} \leq 9$). For IOPT = 0, rounding is disabled.

CIRCLE

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.

ELLIPS

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.

PIE

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.

ARCELL

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.

AREAF

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 2000.
 - 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 xinvs (int npx);	
	float yinvs (int npy);	

TRFREL

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.

TRFCO1

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'.

TRFCO2

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'.

TRFCO3

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 trfco2 (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 $0 \leq \text{longitude} \leq 360$ and $-90 \leq \text{latitude} \leq 90$. 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'.

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.

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 incdat (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 nwkdav (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 BITS12 allows bit manipulation on 16 bit variables.

The call is: CALL BITS12 (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.

BITSI4

The routine BITS4 allows bit manipulation on 32 bit variables.

The call is: CALL BITS4 (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.

9.6 Byte Swapping

SWAPI2

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.

SWAPI4

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.

9.8 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.

OPENFL

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 ($0 \leq \text{NLU} \leq 99$). 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).

CLOSFL

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.

READFL

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).

WRITFL

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).

SKIPFL

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).

TELLFL

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.

POSIFL

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).

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'.

BARWTH

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

BARMOD

BARMOD modifies the width of bars.

The call is: CALL BARMOD (CMOD, COPT) level 1, 2, 3
or: void barmod (char *cmode, 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 similar way as described in BARWTH.

COPT is a character string that must contain the value 'WIDTH'. Default: ('FIXED', 'WIDTH').

BARPOS

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.

BARGRP

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.

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'.

LABPOS

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'.

LABDIG

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 (≥ -1).

Default: NDIG = 1

LABCLR

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 number between -1 and 255. If NCLR = -1, the bar labels will be plotted with the current colour.

Default: NCLR = -1

10.2 Pie Charts

PIEGRF

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.

PIETYP

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'.

CHNPIE

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'.

LABDIG

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 (≥ -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').

LABCLR

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 number between -1 and 255. If NCLR = -1, the pie labels will be plotted with the current colour.

Default: NCLR = -1

PIECLR

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 numbers between -1 and 255 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.

PIEBOR

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 number between -1 and 255.

Default: IC = -1

PIE OPT

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.).

PIE LAB

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.

PIE EXP

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.

PIEVEC

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').


```

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

```

Bar Graphs (BARS)

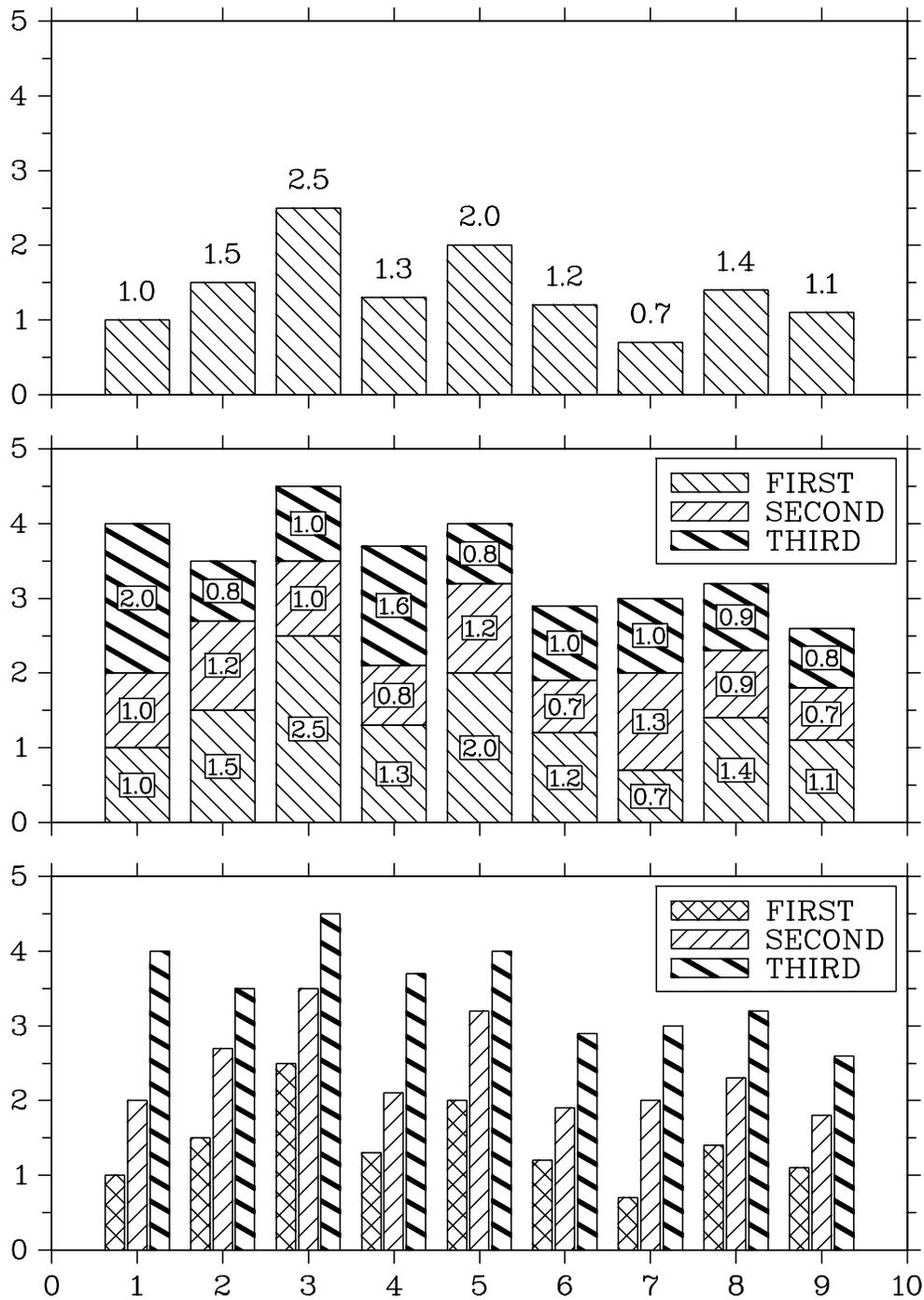


Figure 10.1: Bar Graphs

```

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

```

Pie Charts (PIEGRF)

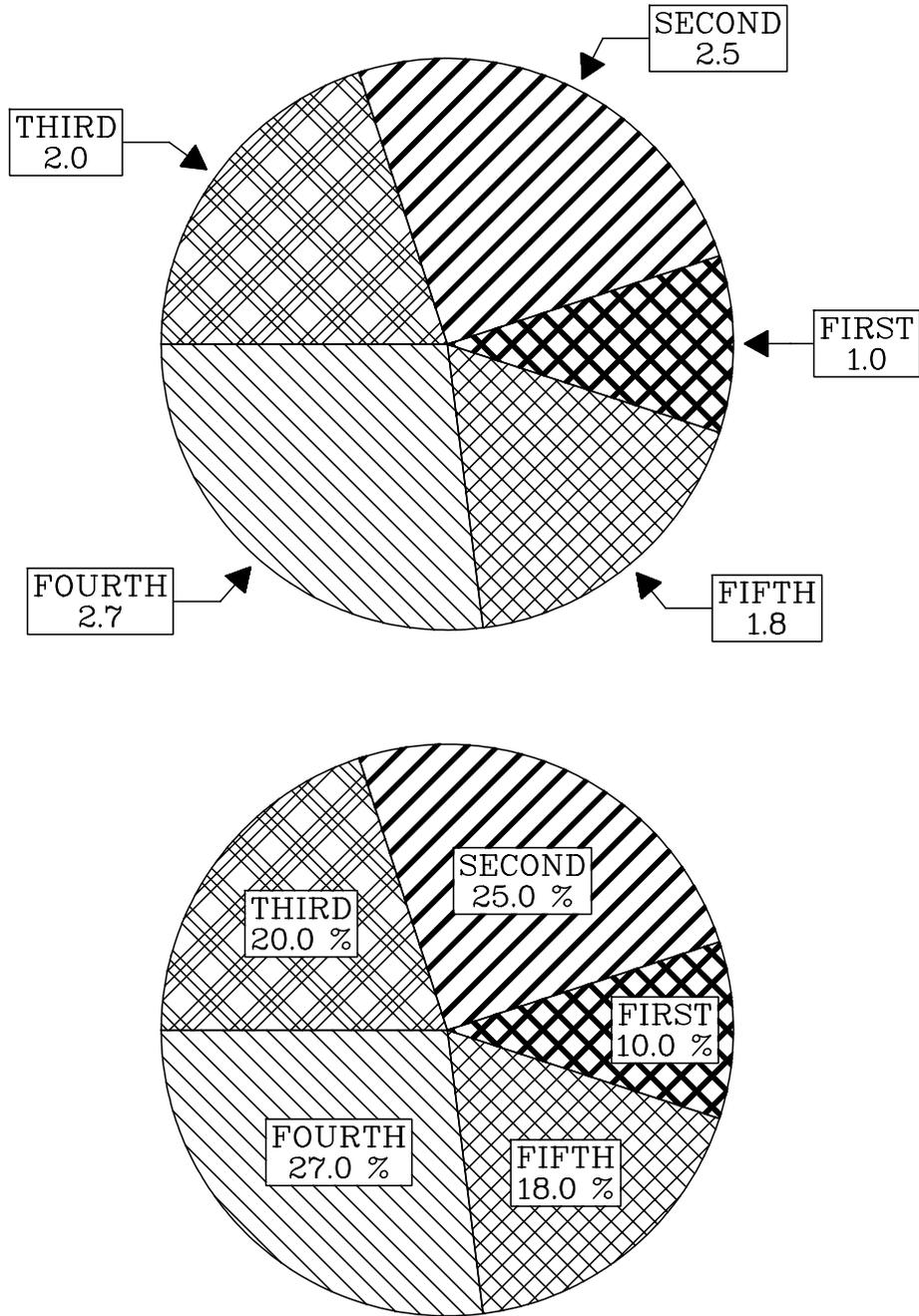


Figure 10.2: Pie Charts

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.

The following plotting routines require special graphics devices that can display a wide range of colours, such as a workstation or a PC with a VGA card. These devices will be called colour graphics systems throughout this chapter. A laserprinter with a PostScript emulation can also be used for 3-D colour graphics.

- Additional notes:
- A colour graphics system can be chosen with the parameters 'CONS', 'XWIN' and 'XWi' in the routine METAFL where i is a window number between 1 and 5. With the parameters 'POST' and 'PSCL', PostScript files can be created.
 - The output of plot commands to the screen may be buffered. To send the buffer to the screen, the statement CALL SENDBF must be used.
 - Up to 256 colours can be displayed at the same time. Colours are numbered in the range 0 to 255 where the number is an index in the current colour table. DISLIN provides several colour tables.

11.2 Window Terminals

Window terminals are the usual graphics displays on workstations and PCs. In DISLIN, graphical output to windows can be activated with the keywords 'CONS', 'XWIN' and 'XWi' in the routine METAFL where i is a window number between 1 and 5. For the keyword 'CONS', the entire screen will be used for graphics output while for the other keywords, smaller windows are created. The keywords 'XWIN' and 'XWi' have the same meaning. By default, the position of windows depends upon the window number. Window 1 is situated in the lower right corner, window 2 in the upper right corner, window 3 in the upper left corner, window 4 in the lower left corner and window 5 in the centre of the screen. The position and size of windows can be changed with the routines WINDOW and WINSIZ.

- Additional notes:
- Normally, a program using X11 or Windows API emulation is blocked in the termination routine DISFIN and waiting for a mouse button 2 event to continue program execution. This behaviour of DISLIN can be modified with the routine WINMOD.
 - To conserve current screen and window colours, the number of colours on an X Window screen is reduced in DISLIN from 256 to 129 colours. This can be changed with the routine CLRMOD.

11.6 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.

11.7 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.

- = '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 'RGREY' for PostScript files created with the keyword 'POST' in the routine METAFL, and otherwise 'RAIN'.

MYVLT

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.

SETIND

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.

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.

HSVRGB

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.

RGBHSV

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);

IX, IY is the position of the pixel in screen coordinates.
ICLR is the new colour value of the pixel.

RPIXLS

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.

WPIXLS

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.

RPXROW

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 image data.

WPXROW

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 image data.

Additional note: IMGINI and IMGFIN must be used with the routines RPIXEL, WPIXEL, RPIXLS, WPIXLS, RPXROW and WPXROW.

ID is the returned window number.

WINTIT

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.

11.11 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.

RECFL

The routine RECFL plots a coloured rectangle where the position is determined by the upper left corner.

The call is: CALL RECFL (NX, NY, NW, NH, NCOL) level 1, 2, 3
or: void recfl (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 index in the range 0 to 255.

POINT

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 index in the range 0 to 255.

RLPOIN

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 rlpoint (float x, float y, int nw, int nh, int ncol);

Additional note: RLPOIN clips rectangles at the borders of an axis system.

SECTOR

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 index between 0 and 255.

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.

RLSEC

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.

11.12 Conversion of Coordinates

The function NZPOSN and the subroutine COLRAY convert user coordinates to colour values.

NZPOSN

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.

COLRAY

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.

11.13 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
```



3-D Colour Plot of the Function

$$F(X,Y) = 2 * SIN(X) * SIN(Y)$$

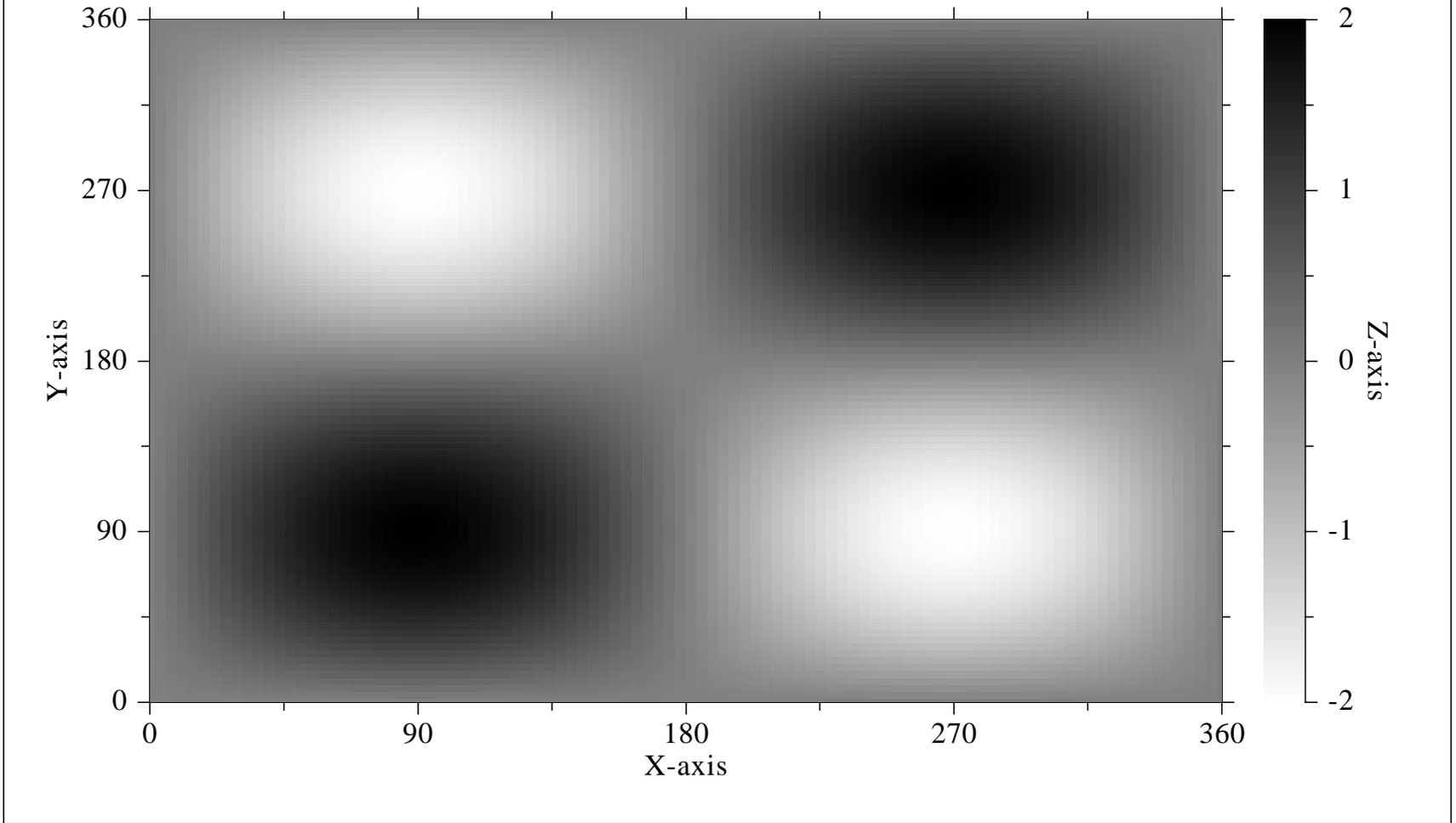


Figure 11.1: 3-D Colour Plot

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. The box itself will be cut off at the borders of the region unless suppressed with the routine NOCLIP.

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:

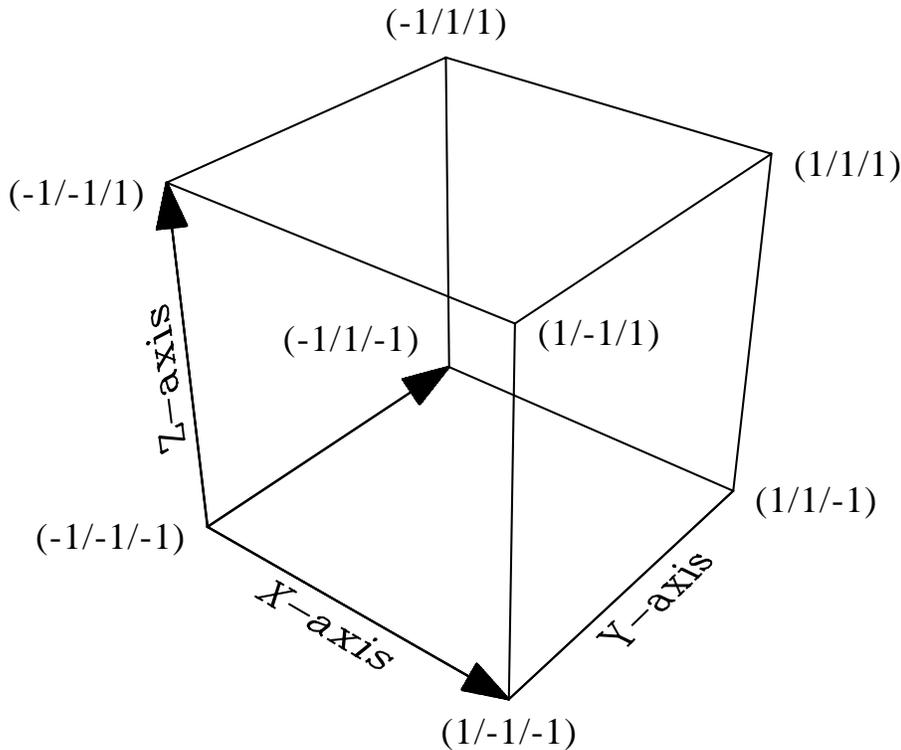


Figure 12.1: Default 3-D Box

12.2 Defining View Properties

The following routines define view properties such as viewpoint, target point, view angle and view orientation.

VIEW3D

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.

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.
 - To avoid overwriting labels, GRAF3D suppresses the plotting of certain start 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

BOX3D

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

GRID3D

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 igrd, 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

CURV3D

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

SURFUN

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 SURSIZE (XA, XE, YA, YE). If XA, XE, YA and YE are the axis limits in GRAF3D or defined with SURSIZE, the connection of grid points and matrix elements can be described by the formula:

$ZMAT(I,J) = F(X,Y)$ where

$$X = XA + (I - 1) * (XE - XA) / (IXDIM - 1) \quad I = 1, \dots, IXDIM \quad \text{and}$$

$$Y = YA + (J - 1) * (YE - YA) / (IYDIM - 1) \quad J = 1, \dots, IYDIM.$$

- 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. The parameters must be in the range -1 to 255 where the default value -1 means that the current colour is used.

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

SURFCP

A three-dimensional parametric function is a function of the form $(x(t,u), y(t,u), z(t,u))$ where $t_{min} \leq t \leq t_{max}$ and $u_{min} \leq u \leq u_{max}$. 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

SURTRI

The routine SURTRI plots a shaded surface from triangulated data that can be calculated by the routine TRIANG from a set of irregularly 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.

or: void mshclr (int iclr);

ICLR is a colour value in the range -1 to 255 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.).

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 'AMBIENT', 'DIFFUSE', 'SPECULAR', 'CONSTANT', 'LINEAR' and 'QUADRATIC'.
 Defaults: (0., 'AMBIENT'), (1., 'DIFFUSE'), (1., 'SPECULAR'), (1., 'CONSTANT'), (0., 'LINEAR'), (0., 'QUADRATIC').

M A T O P T

The routine MATOPT modifies material parameters such as ambient, diffuse and specular colour. The specular exponent can also be modified.

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 values 'AMBIENT', 'DIFFUSE', 'SPECULAR' and 'EXPONENT'.
 Defaults: (0.2, 'AMBIENT'), (0.8, 'DIFFUSE'), (0., 'SPECULAR'), (0., 'EXPONENT').

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.

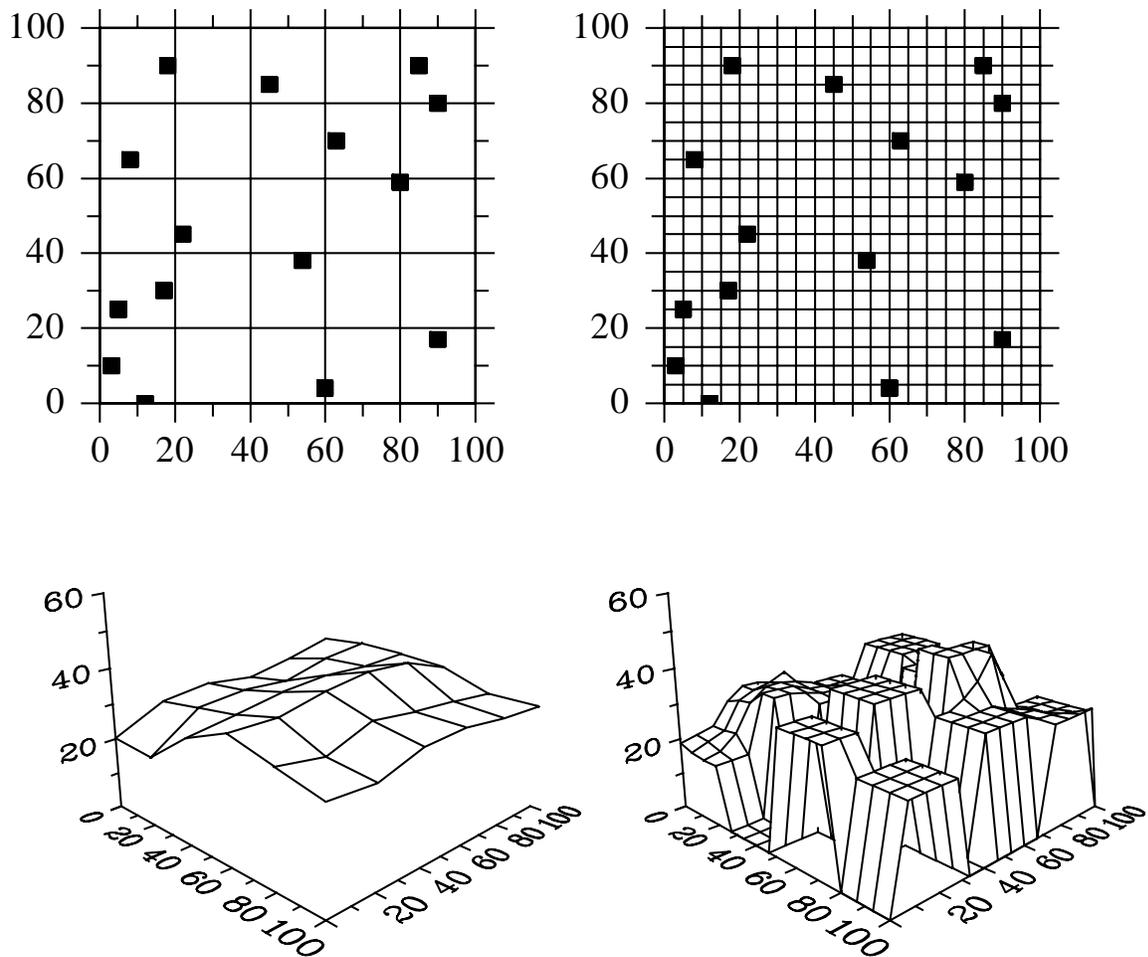


Figure 12.2: Results of GETMAT

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

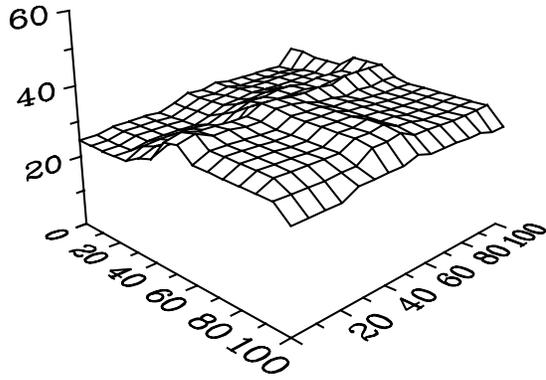
or: `void mdformat (int ix, int iy, float w);`

IX, IY are the number of grid lines in the X- and Y-direction which determine the size of the region around data points.

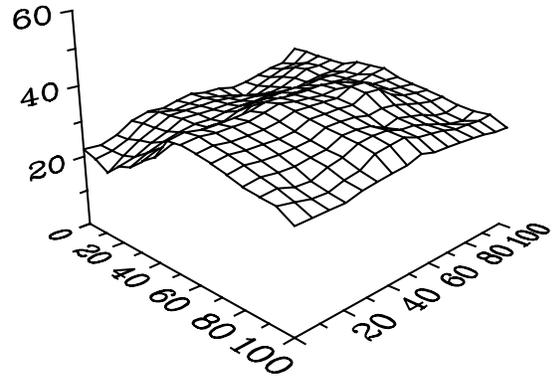
W is a weighting number.

Default: (2, 2, 2.0).

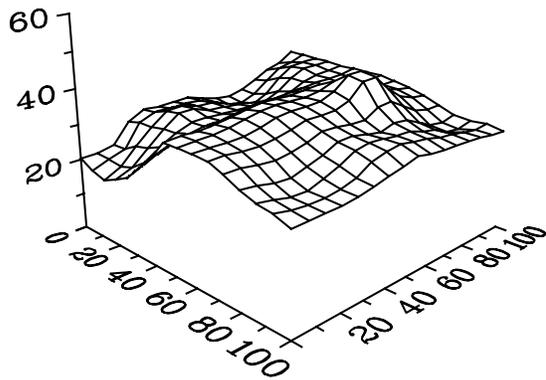
The following figure shows modifications of the above example:



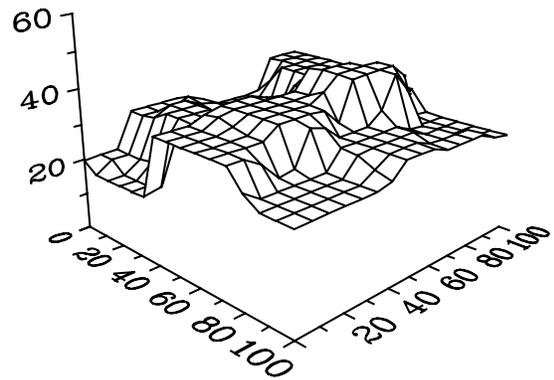
MDFMAT (5, 5, 0.1)



MDFMAT (5, 5, 1.0)



MDFMAT (5, 5, 2.0)



MDFMAT (5, 5, 15.0)

Figure 12.3: Modification of GETMAT

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.

12.18 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).

SPHE3D

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.

12.19 Transformation of Coordinates

POS3PT

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)

REL3PT

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)

ABS3PT

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)`

12.20 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
```

Surface Plot (SURFUN)

$$F(X,Y) = 2*\text{SIN}(X)*\text{SIN}(Y)$$

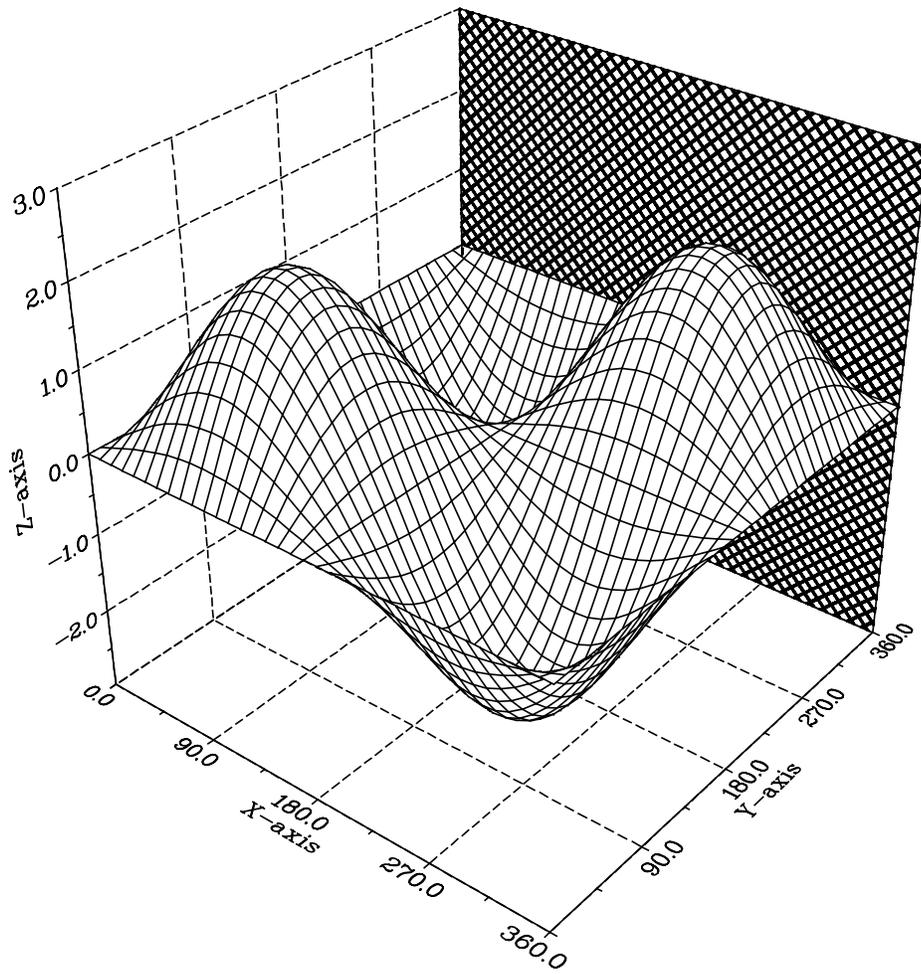


Figure 12.1: Surface Plot

```

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

```

Surface Plot of the Parametric Function
 $[\cos(t)(3+\cos(u)), \sin(t)(3+\cos(u)), \sin(u)]$

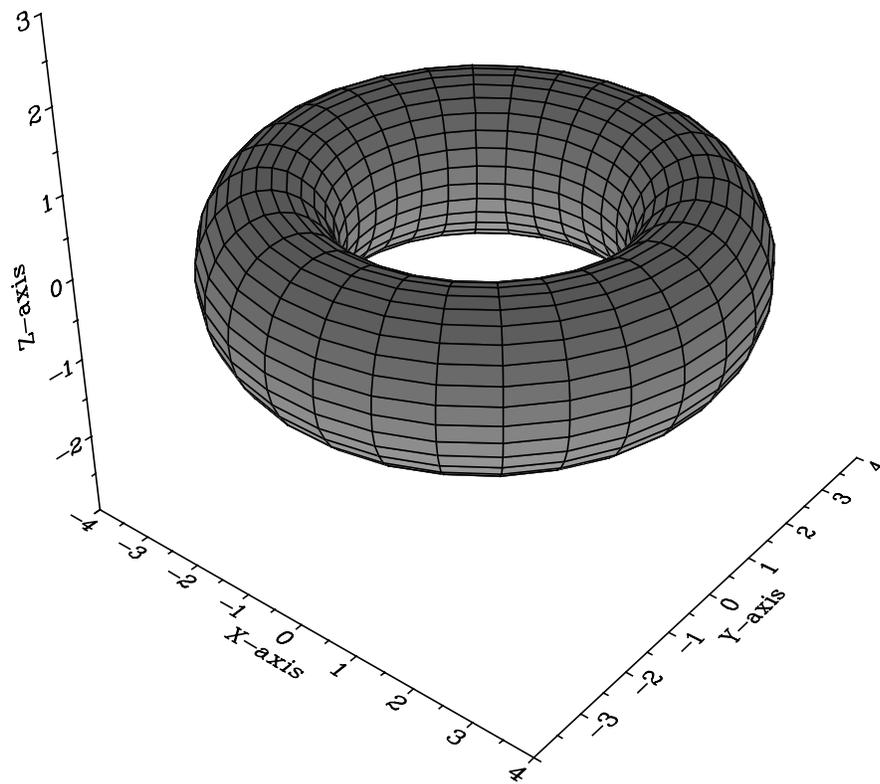


Figure 12.2: Surface Plot of a Parametric Function

