SIMPLEPLOT 3-D

BUSS Ltd.

SIMPLEPLOT 3-D

BUSS Ltd.
Business and Innovation Centre
Angel Way
BRADFORD
West Yorkshire
BD7 1BX
United Kingdom

Telephone	01274 841396
	+44 1274 841396
Fax	01274 841388
	+44 1274 841388

Web Site	http://www.buss.co.uk/buss
Email	info@buss.co.uk

Fourth Edition, October 2000

Contents

List of Tables

List of Example Programs

List of Figures

Preface

SIMPLEPLOT is a library of FORTRAN subroutines for plotting graphs. A wide variety of graphs can be drawn as well as more general pictures and diagrams. Facilities are biased towards the graphical representation of data; in particular, scientific data.

SIMPLEPLOT was originally designed for programmers who wanted to draw pictures of their data with minimum programming effort. Although it still achieves this goal, SIMPLEPLOT has developed into a much more powerful tool for professional software engineers.

Six separate sections constitute the complete SIMPLEPLOT Mark 2:

- The basic package for conventional graph plotting – x-y plots and polar plots.
- Additional subroutines for 3-dimensional plotting – contour maps and surface pictures of 3-D data.
- Additional subroutines for presentation graphics – bar charts, histograms and pie charts.
- *SIMPLEPLOT Volumes* – perspective pictures of 4-dimensional data.
- *SIMPLEPLOT Maps* – for representing data based on geographical coordinate systems.
- *SIMPLEPLOT ViSualization* – for full colour modelling of functions of two, three and four variables.

SIMPLEPLOT-PLUS refers to a SIMPLEPLOT library which is made up from the first three sections and contains many additional facilities.

This manual refers only to selected subroutines available from Sections *1*, *2*, *4* and *+* which are relevant for plotting 3-dimensional data. Additional facilities for 2-D plotting and for plotting representations of functions of three variables are described in the *SIMPLEPLOT Reference manual*.

Graphics device interface

The SIMPLEPLOT library is independent of any single graphics system and is device *sensitive* with a device independent interface for the user. This means that the user is protected from having to know about the features of the target output device, but SIMPLEPLOT makes as much use of these features as possible.

SIMPLEPLOT has already been interfaced to a large number of graphics devices, and the range of validated device drivers is continuously being extended. It can address graphics devices directly, or through separate low-level graphics systems (*eg.* GKS, X Window System) or graphics languages (*eg.* PostScript, CGM).

What sort of pictures can you draw ?

This manual describes the following facilities for plotting 3-dimensional data:

- Surface pictures
- Line contour maps
- Single contour curves
- Shaded contour maps
- Single shaded contour ranges
- Waterfall charts
- Cross-sectional curves of 3-D data
- Interpolation of 3-D data

and details of some general facilities which are useful with 3-D plotting:

- Labelled curves
- Bundled attributes
- Sequences of attributes
- Number formatting

The full SIMPLEPLOT library provides a much wider range of facilities but only a subset is described in this manual.

1. Introduction

SIMPLEPLOT 3-D describes the facilities in SIMPLEPLOT for plotting 3-D data. It can be used as a manual in its own right, but detailed explanations of all SIMPLEPLOT subroutines are found in the *SIMPLEPLOT Reference manual* (8th edition).

1.1 Overview

SIMPLEPLOT 3-D is organized as a tutorial, followed by a cookbook, technical appendices and reference material:

- Introduction to *SIMPLEPLOT 3-D*.
- Tutorial – Chapters 2–7
  2. Getting started
  3. Three-dimensional data
  4. Contour plotting
  5. Surface pictures
  6. Waterfall charts
  7. Additional 3-D facilities
- Cookbook – Chapter 8
- Technical Appendices
  - A. Subroutine Specifications – brief descriptions with argument lists.
  - D. Graphic Details – shading patterns, broken line styles, marker symbols and character sets.
  - E. Diagnostics
  - G. Glossary
  - S. Subroutine Summary
- Index

1.2 Software version

Software version]

This manual is based on SIMPLEPLOT Mark 2, version 2-14.

1.3 Target audience

Target audience]

How you choose to use *SIMPLEPLOT 3-D* will depend on your experience of SIMPLEPLOT. *SIMPLEPLOT 3-D* has been written with the following readers in mind:

- Existing SIMPLEPLOT users who want to plot 3-D data
- Long-term 'Section 2' users who want to refresh out-of-date or forgotten information.

All readers should be familiar with programming in FORTRAN on their host computer system, and using basic SIMPLEPLOT facilities as described in the *SIMPLEPLOT Primer*.

1.4 Related documents

Related documents]

Related documents include

- The *SIMPLEPLOT Primer* which provides a general introduction to SIMPLEPLOT, especially those facilities available for plotting 2-D data.
- The *SIMPLEPLOT Reference manual* (8th edition) which contains full specifications for all SIMPLEPLOT-PLUS subroutines.
- The *SIMPLEPLOT 2-13 Supplement* which gives details of the facilities introduced in SIMPLEPLOT version 2-13.
- The *SIMPLEPLOT 2-14 Supplement* which gives details of the facilities introduced in SIMPLEPLOT version 2-14.

1.5 How to report problems

How to report problems]

If you have any problems with SIMPLEPLOT software or its associated products and services please notify BUSS Ltd on a Software Performance Report (SPR) form. One of these should be sent out with every software kit – please photocopy it or contact BUSS Ltd if you would like extra copies.

1.6 How to use this manual

How to use this manual]

Newcomers to plotting 3-D data with SIMPLEPLOT, may find it easier to get started using this manual in the following way:

1. Read Chapter 2 and decide whether your requirements are covered.
2. Look through Chapter 3 for data structures that SIMPLEPLOT can accept and decide how best to present your data for speed and efficiency. Then, either:
   - Convert an existing program into a plotting program by adding calls to subroutines from one of these chapters alone, or
   - Find the example program in Chapter 8 which is the closest to what you want to produce and adapt it to your data.
3. Execute your program according to your host computer's requirements for a SIMPLEPLOT program.
4. When the program works and produces basic graphs, it can be enhanced by using the subroutines described in Chapter 7.

Having gained confidence in using the subroutines you should then be able to use any combination of subroutines for which you have an application.

1.7 Conventions

Conventions]

The following conventions are used in this manual for example programs and subroutine specifications.

Example programs: The example programs are collected together in Chapter 8. They are written in standard FORTRAN and have been designed to be as brief as possible. The example programs at the beginning of Chapter 8 concentrate on how to draw simple representations of data. Examples later in the chapter are more realistic, combining features of Chapter 7 with the data plotting facilities described in Chapters 3.

Standard intrinsic FORTRAN type conventions are used throughout the example programs and subroutine specifications. In all example programs, small data sets are included in the program; longer data sets are read from files which are included as part of the software distribution kit.

Each example program is accompanied by an explanation of those subroutines which have not occurred in any previous example or which are being used in a new context. Diagnostics from example programs are not included but a full explanation of all diagnostic messages issued by the subroutines described in this manual is given in Appendix E, *Diagnostics*.

Subroutine specifications: All specifications are given in a similar format whether they are classified as graphics, specification or auxiliary subroutines. Appendix A, *Subroutine Specifications*, gives an alphabetical list of brief specifications of all subroutines used in this manual including details of arguments. Full details of these and other SIMPLEPLOT subroutines are found in the *SIMPLEPLOT Reference manual* (8th edition).

Illustrations: All figures and output from example programs are produced using SIMPLEPLOT version 2-14 using the PostScript or CPS (Colour PostScript) device driver; in monochrome, the different colours encoded by the CPS driver are represented by different levels of grey scale.

2. Getting Started

2.1 Facilities

SIMPLEPLOT provides facilities for representing 3-D data as surface pictures, contour maps and waterfall charts. Cross sections of 3-D data can be drawn on 2-D pictures and interpolation functions are available. Figure 2.1 illustrates some of the pictures which can be drawn.

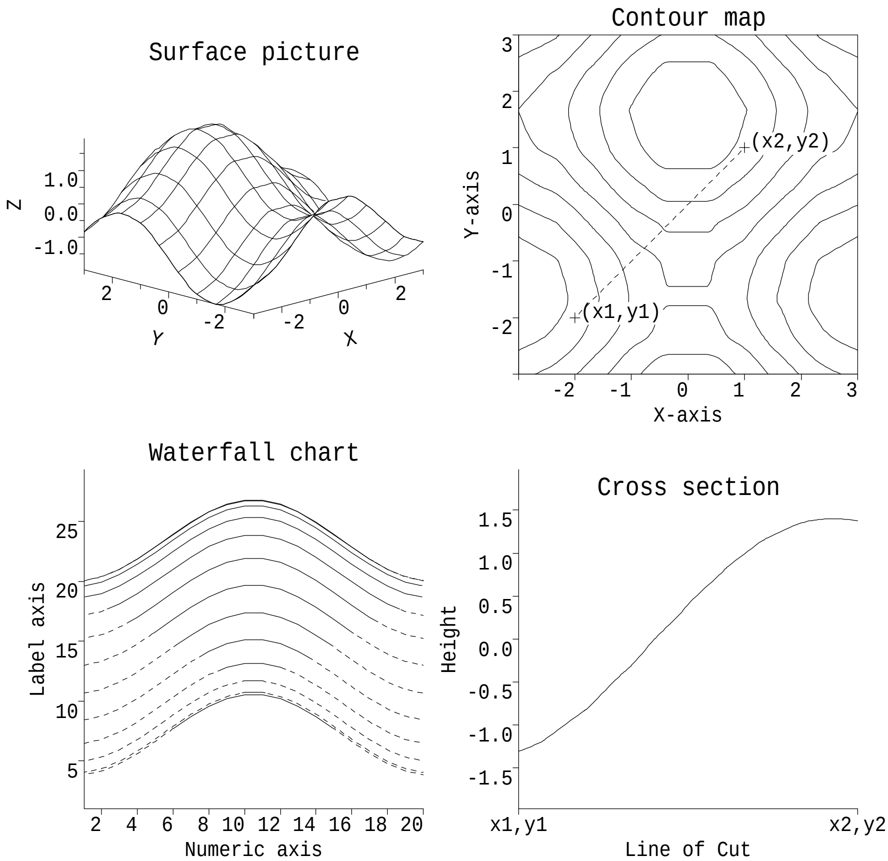


Figure 2.1 Pictorial representations of 3-D data

2.1.1 Surface picture

Surface picture]

A single-valued continuous function of two variables can be represented in 3-D space by a surface over a fixed plane. The height of the surface above a base plane is proportional to the z values. The function is represented graphically by drawing the surface with hidden lines eliminated. The surface may be viewed from any of the corners, and the x-y plane may be rotated.

2.1.2 Contour map

Contour map]

Surface heights can be represented by contours on a 2-D Cartesian or polar picture. The following operations are available separately:

- Draw the curve(s) representing a single z value.
- Draw a complete map of contour curves from a data set.
- Shade the region between two z values.
- Shade a complete contour map of a data set.

Contour lines may also be added to surface pictures.

2.1.3 Waterfall chart

Waterfall chart]

In waterfall charts, rows of data are plotted as a series of staggered line plots on a Cartesian graph. This gives a good impression of the configuration of the data and can reveal subtle variations which are less evident on a surface picture.

2.1.4 Cross section

Cross section]

A cross section through a surface is a curve showing the variation of z along the straight path between two specified (x, y) points.

3. Three-dimensional Data

SIMPLEPLOT 3-D operates on three-dimensional data representing a single-valued continuous function of two variables, $z = f(x, y)$ or $z = f(r, \theta)$. Several different forms of such three-dimensional data can be interpreted directly. This chapter covers the following topics:

3.1 Missing data

3.2 Gridded 3-D data

- Equally-spaced grids
- Specified grids
- Gridded data structures
- Finding the limits of gridded data
- Plotting gridded data

3.3 User-defined functions of two variables:

- Scales and function limits
- Finding the limits of user-defined functions
- Plotting user-defined functions

3.4 Ungridded 3-D data

- Ungridded data structure
- Subroutine names
- Scales and limits
- Converting ungridded data to a regular grid

3.5 Polar data, $z = f(r, \theta)$

- Coordinate interpretation
- Gridded polar data
- Polar functions, $z = f(r, \theta)$
- Ungridded polar data

The SIMPLEPLOT coordinate system is Cartesian by default, but it can be switched to polar, after which the same subroutines which operate on Cartesian data can be used for Polar data: x arguments are interpreted as r values, and y arguments are interpreted as θ values.

3.1 Missing data

Missing data]

A special value can be used in a data set to indicate to SIMPLEPLOT that there is no valid data at that point; this is then represented in the picture as a hole (or gap) in the surface, contour map or waterfall chart. Figure 3.1 illustrates missing data on a shaded contour map, Figure 3.2 shows how no-data values are represented on different types of surface picture and Figure 3.3 illustrates missing data on a waterfall chart. In all these pictures, the data point corresponding to a no-data value is marked with an asterisk (*).

Figure 3.1 Missing data on contour maps

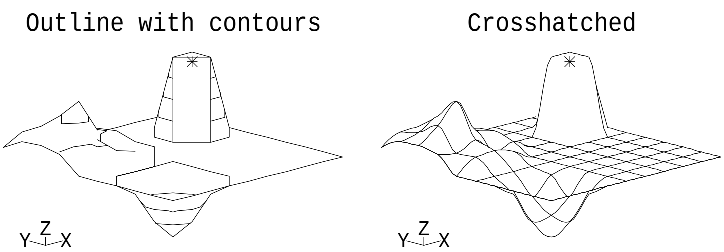


Figure 3.2 Missing data on surface pictures

Before any holes can be defined, the program must establish what value to use as a no-data value – a value is specified using `NODATA`, and `QNODAT` inquires the current no-data value:

`NODATA(RVAL)` specifies the value, `RVAL`, which SIMPLEPLOT is to interpret as a no-data value. By default, any value equal to 1.0×10^{-20} is interpreted as missing.

`QNODAT(RVAL)` inquires the current no-data value; it may be either the default or a value set by a prior call to `NODATA`. The argument of `QNODAT` must be the name of a `REAL` variable which receives the no-data value.

No-data values affect all plotting and related activities – all plotting subroutines ignore coordinates for which either ordinate corresponds to the no-data value. If `NODATA` is called to specify your own choice of the no-data value, care should be taken to choose a value outside the range of normal x-y plotting.

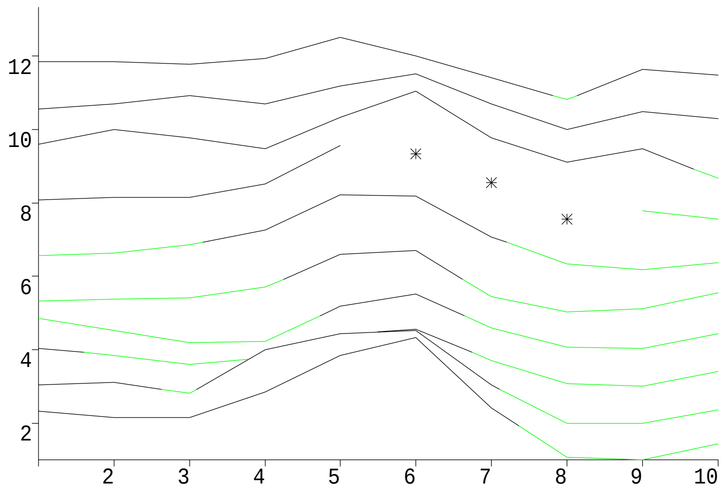


Figure 3.3 Missing data on a waterfall chart

3.2 Gridded 3-D data

Gridded 3-D data]

3-D plotting is performed from data supplied to SIMPLEPLOT as a table of z values for every combination of x and y in an x-y grid or for every combination of r and θ in an r-θ grid. The z values must be held in a 2-D `REAL` array, `Z2ARR`, dimensioned `Z2ARR(NX,NY)`.

Cartesian grids are illustrated in Figure 3.4 (below) and polar grids in Figure 3.12.

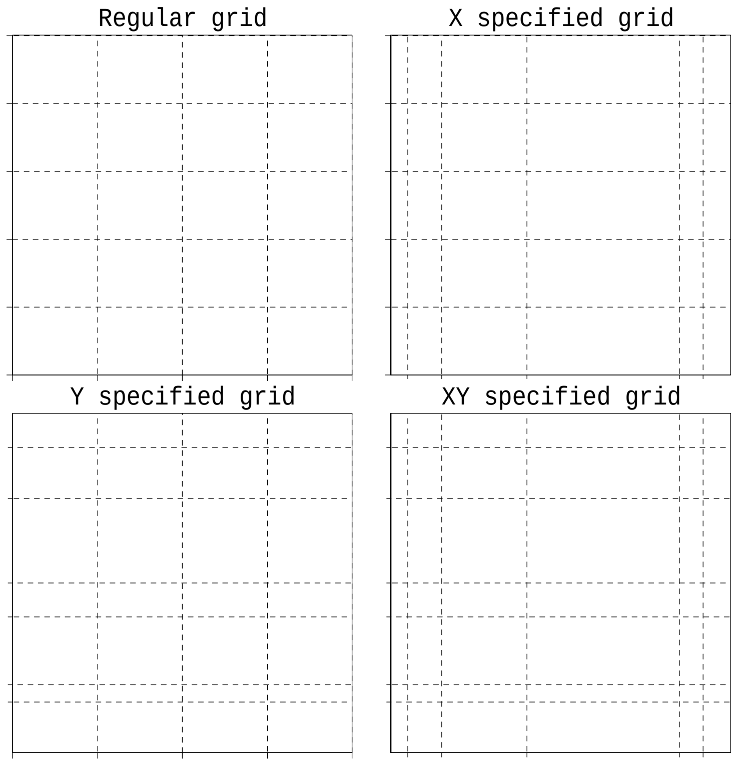


Figure 3.4 Cartesian data grids

There are two alternative methods for defining the coordinates of the grid, *equally-spaced* and *specified*; either may be used for x and either may be used for y.

3.2.1 Equally-spaced grids

When a coordinate of data grid is *equally-spaced*, its values are not included with the 3-D data. Details of the *equally-spaced* x and/or y values of the data grid may be supplied by prior calls of `SFEQX` and/or `SFEQY`.

Equally-spaced x values of a data grid are supplied by

```
CALL SFEQX(XSTART,XSTEP)
```

`XSTART` is the x value at the first grid line, and `XSTEP` is the x grid interval. `NX` data columns have x values covering the range `XSTART`, to `XSTART`+(`NX`-1)×`XSTEP`.

Equally-spaced y values of a data grid are supplied by

```
CALL SFEQY(YSTART,YSTEP)
```

YSTART is the y value at the first grid line, and YSTEP is the y grid interval. NY data rows have y values covering the range YSTART, to YSTART+(NY-1)×YSTEP.

3.2.2 Gridded data structures

Gridded data structures]

Regular-gridded data represent function values, $z = f(x, y)$, at the intersections of a grid with NX equally-spaced x values and NY equally-spaced y values. Subroutines and functions for regular-gridded data have names beginning with RG (ReGular), for example, RGSURF(Z2ARR,NX,NY).

x-specified tartan-gridded data represent function values, $z = f(x, y)$, at intersections of a grid with specified x values and equally-spaced y values; the NX x values are held in a REAL array XARR, from XARR(1) to XARR(NX), in ascending or descending order. Subroutines and functions for x-specified data have names beginning with X, for example, XSURF(Z2ARR,NX,NY,XARR).

y-specified tartan-gridded data represent function values, $z = f(x, y)$, at intersections of a grid with specified y values and equally-spaced x values; the NY y values are held in a REAL array YARR, from YARR(1) to YARR(NY), in ascending or descending order. Subroutines and functions for y-specified data have names beginning with Y, for example, YSURF(Z2ARR,NX,NY,YARR).

x-y specified tartan-gridded data represent function values, $z = f(x, y)$, at intersections of a grid with specified y values and specified x values; the NX x values of the grid are held in a REAL array XARR, from XARR(1) to XARR(NX) and the NY y values are held in a REAL array YARR, from YARR(1) to YARR(NY), each in ascending or descending order. Subroutines and functions for x-y specified data have names beginning with XY, for example, XYSURF(Z2ARR,NX,NY,XARR,YARR).

3.2.3 Finding the limits of gridded data

Equally-spaced grids Equally-spaced x values of a data grid are supplied by SFEQX(XSTART,XSTEP). The minimum and maximum values of x are XSTART and XSTART+(NX-1)×XSTEP. When XSTEP is positive, XSTART is the minimum value; when XSTEP is negative, XSTART is the maximum value.

Equally-spaced y values are specified by SFEQY(YSTART,YSTEP), which behaves in the same manner.

Array of grid coordinates – XARR and YARR: Arrays specifying grid coordinates must contain values in ascending or descending order. When x coordinates are supplied in an array XARR, XARR(1) and XARR(NX) are the extreme values; and when y coordinates are supplied in an array YARR, YARR(1) and YARR(NY) are the extreme values. Simple comparisons of the first and last value determine which is the minimum and which is the maximum.

Two-dimensional data array – Z2ARR: LIMEXC inquires the minimum and maximum values in the 2-D data array Z2ARR:

```
CALL LIMEXC(Z2ARR,NX*NY,ZMIN,ZMAX)
```

Table 3.1 Summary of gridded data structures
The arguments defining the 3-D data depend on whether one or both of the coordinates are specified.

Data	Esubroutines	*x values*	*y values*
Regular grid	RG*	*Equally-spaced*	*Equally-spaced*
x-specified tartan grid	X*	*Specified*	*Equally-spaced*
y-specified tartan grid	Y*	*Equally-spaced*	*Specified*
x & y-specified grid	XY*	*Specified*	*Specified*

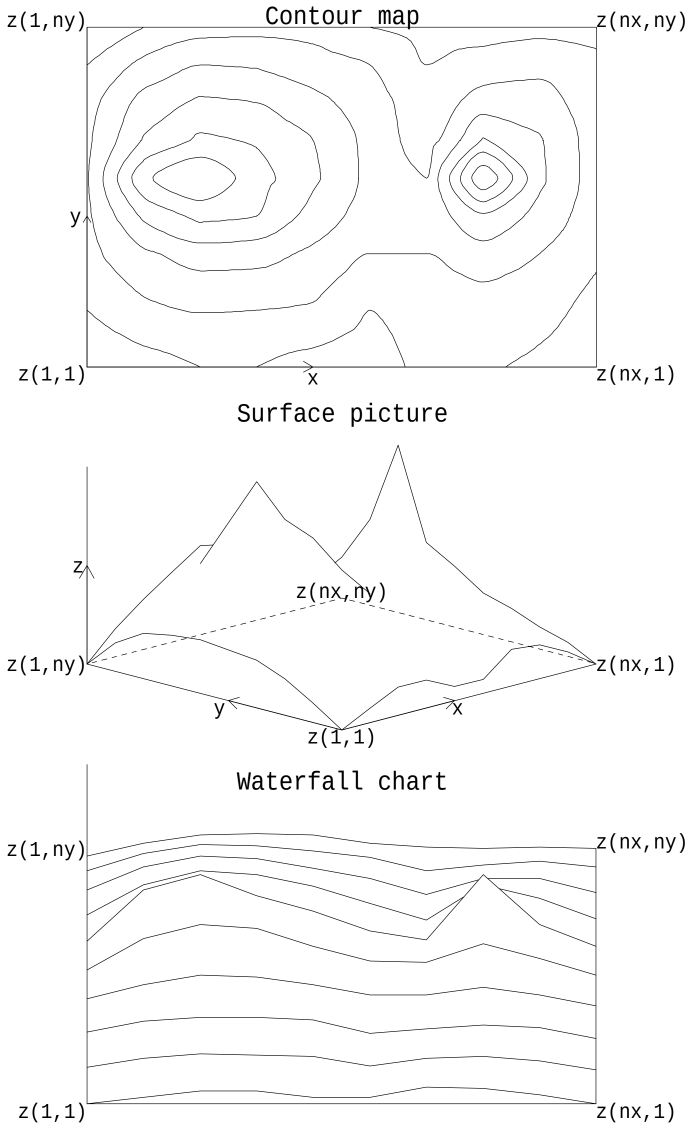


Figure 3.5 Representation of gridded data

3.2.4 Plotting gridded data

Figure 3.5 illustrates how the 2-D array of z values is represented on contour maps, surface pictures and waterfall charts.

`RGCNTS(Z2ARR,NX,NY)` draws a set of contour lines on the current picture.

`RGCONT(ZLEV,Z2ARR,NX,NY)` draws the curve(s) corresponding to the specified contour level on the current picture.

`RGCUT(X1,Y1,X2,Y2,Z2ARR,NX,NY)` draws a 2-D curve of a surface section on the current 2-D picture.

`RGSHAD(ZLEV1,ZLEV2,ISHADE,Z2ARR,NX,NY)` shades the area(s) between `ZLEV1` and `ZLEV2` on the current 2-D picture using shading pattern, `ISHADE`. The lower limit of the range is included, and the upper limit excluded.

`RGSHDS(Z2ARR,NX,NY)` draws a shaded contour map on the current 2-D picture.

`RGSURF(Z2ARR,NX,NY)` starts a new 3-D picture and draws the surface.

Similar sets of subroutines are available for plotting x-specified, y-specified and x-y specified tartan-gridded data; for example, `XSHDS(Z2ARR,NX,NY,XARR)` draws a shaded contour map of x-specified tartan-gridded data on the current 2-D picture.

`YSHDS(Z2ARR,NX,NY,YARR)` draws a shaded contour map of y-specified tartan-gridded data on the current 2-D picture.

`XYSHDS(Z2ARR,NX,NY,XARR,YARR)` draws a shaded contour map of x-y specified tartan-gridded data on the current 2-D picture.

3.3 User-defined functions of two variables

SIMPLEPLOT provides facilities for drawing surface pictures and contour maps directly from a user-defined `REAL` function with two `REAL` arguments without the need for tabulation of function values. When plotting such a function, `FUNXY`, with arguments `XVAL` and `YVAL`, the value of `FUNXY(XVAL,YVAL)` is interpreted as the surface height at the point (`XVAL`,`YVAL`).

Subroutines whose arguments include `FUNXY`, the name of a user-defined function, have names beginning with `FN` (FunctioN).

The name of the user-defined function which is to be plotted is passed to SIMPLEPLOT as an argument; this name must therefore be declared as `EXTERNAL` in each source segment in which `FUNXY` is passed as an argument.

3.3.1 Scales and function limits

By default, contours are drawn over the same ranges as the plotting scales, and surfaces are plotted over the ranges 0.0 to 10.0 in both x and y. If plotting scales have not been specified (by `SCALES`, `EQSCAL`, *etc.*), scales are linear in centimetres.

`FNAREA(XMIN,XMAX,YMIN,YMAX)` specifies `XMIN` and `XMAX`, the minimum and maximum values of x, and `YMIN` and `YMAX`, the minimum and maximum y values, to be plotted by subsequent `FN*` subroutines.

`FNAREA` does not affect the plotting scales, only the subrange of the scale over which the user-defined function is evaluated and drawn. If the area of the x-y plane described by `FNAREA` exceeds the plotting scales, SIMPLEPLOT issues a diagnostic when it tries to plot beyond the scale limits. If `FNAREA` is called with `XMIN` greater than or equal to `XMAX`, the default x values are used; similarly, if `FNAREA` is called with `YMIN` greater than or equal to `YMAX`, the default y values are used.

On contour maps where 2-D plotting scales determine the overall picture scales, `FNAREA` specifies the x and y ranges over which $z = f(x, y)$ is to be evaluated; `FNAREA` does not affect the plotting scales. On surface pictures there are no such 'plotting scales'; in this case, the ranges specified by `FNAREA` are used to define the full extent of the surface.

3.3.2 Finding the limits of user-defined functions

The limits of a user-defined function are evaluated using `LIMSFN`:

`LIMSFN(FUNXY,ZMIN,ZMAX)` looks through the function values which will be used by the `FN*` subroutines and sets `ZMIN` to the minimum, and `ZMAX` to the maximum.

The values returned by `LIMSFN` may be affected by the following specification subroutines:

`FNAREA(XMIN,XMAX,YMIN,YMAX)` specifies the ranges over which the user-defined function is to be evaluated.

`SFMESH(MX,MY)` specifies the mesh used for constructing contours and surfaces. A finer mesh may include x-y values at which more extreme z values are calculated.

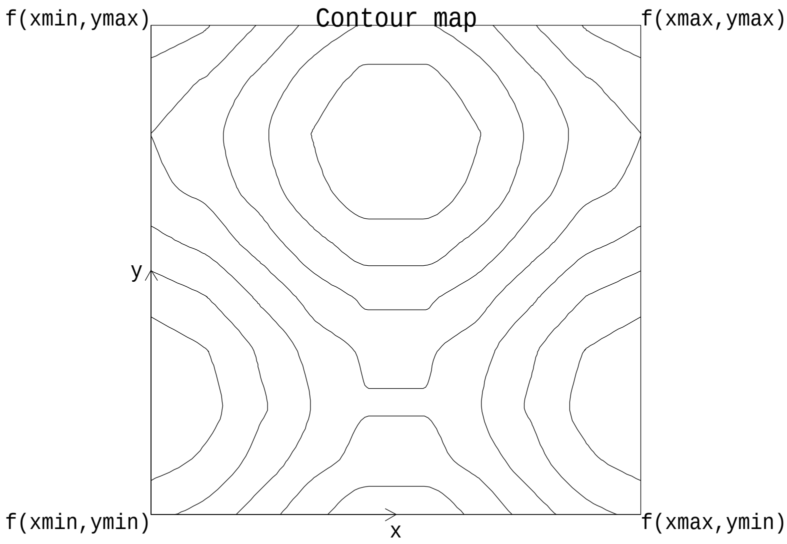
3.3.3 Plotting user-defined functions

Figure 3.6 illustrates how the z values evaluated from a user-defined function are represented on contour maps and surface pictures.

`FNCNTS(FUNXY)` draws a set of contour lines on the current picture.

`FNCONT(ZLEV,FUNXY)` draws the curve(s) corresponding to the specified contour level on the current picture.

`FNCUT(X1,Y1,X2,Y2,FUNXY)` draws a 2-D curve of a surface section on the current 2-D picture.



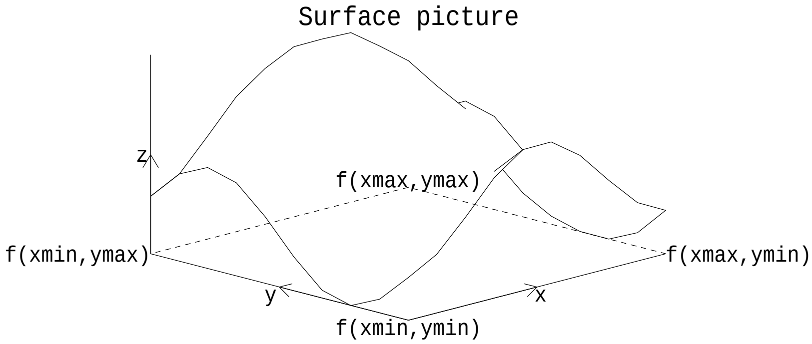


Figure 3.6 Representation of user-defined functions

`FNSHAD(ZLEV1,ZLEV2,ISHADE,FUNXY)` shades the area(s) between `ZLEV1` and `ZLEV2` on the current 2-D picture using shading pattern, `ISHADE`. The lower limit of the range is included, and the upper limit excluded.

`FNSHDS(FUNXY)` draws a shaded contour map on the current 2-D picture.

`FNSURF(FUNXY)` starts a new 3-D picture and draws the surface.

3.4 Ungridded 3-D data

Ungridded 3-D data]

An ungridded data set has x and y values specified for each individual z value. The (x, y, z) coordinates of `NPTS` data points are held in three parallel `REAL` arrays, `XARR`, `YARR` and `ZARR` such that for each `I`, `ZARR(I)` is the function value (*eg.* surface height) at position (`XARR(I)`, `YARR(I)`).

3.4.1 Ungridded data structure

In order to process three-dimensional data, Simpleplot needs the x-y plane to be divided into non-overlapping elements. Gridded data has a well-defined structure of elements and user-defined functions of two variables are evaluated by Simpleplot at points on a regular grid, but ungridded data have no such implicit structure. This section describes how:

- to specify an element structure,
- to generate an array of neighbours to reduce computation,
- to construct triangular elements using standard triangulation or normalized triangulation.

Element structure

Element structure]

The element structure required by Simpleplot is such that the area of the x-y plane covered by data is subdivided into non-overlapping area elements bounded by straight lines between data positions. Every data position must lie on the boundary of at least one area element; for gridded data, rectangular elements bounded by grid lines are used, but for ungridded data, some structuring into elements is necessary. The element structure describes which data points are combined into elements; data points are identified by their subscript values in the data arrays `XARR`, `YARR` and `ZARR`. A two-dimensional array of elements `I2ARR(NODES,NELEMS)` can be constructed such that `I2ARR(I,J)` holds the subscript of the `I`th data point of the `J`th area element, for all `I` values from 1 to `NODES`, and for all `J` values from 1 to `NELEMS`.

All elements must be enclosed by the same number of data points, but this number, `NODES`, can be any integer greater than 2; the number of elements, `NELEMS`, can be any positive integer. When `NODES` is greater than 3, care must be taken to ensure that the data points round each element are placed in `I2ARR` in the correct order to trace the element boundary.

In finite element applications, such element structuring is likely to be already known. If no elements can be supplied, Simpleplot can generate a triangular element structure (*ie.* `NODES`=3). This triangulation process is described below.

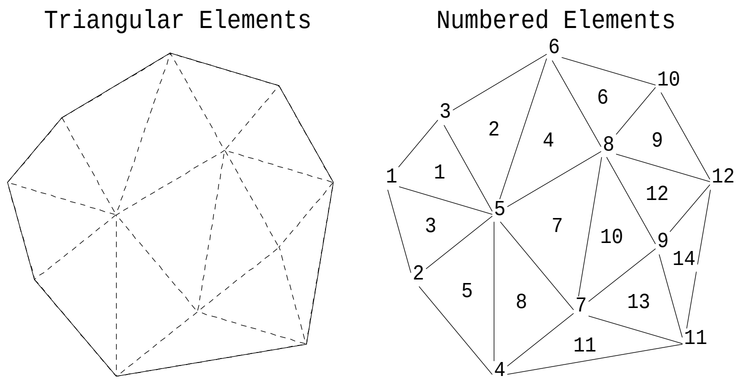


Figure 3.7 Element structure

Table 3.2 Data points and Neighbours

element	*data points*	*neighbours*
1	1 3 5	3 0 2
2	3 6 5	1 0 4
3	1 5 2	0 1 5
4	5 6 8	7 2 6
5	2 5 4	0 3 8
6	6 10 8	4 0 9
7	5 8 7	8 4 10
8	5 7 4	5 7 11
9	8 10 12	12 6 0
10	8 9 7	7 12 13
11	4 7 11	0 8 13
12	8 12 9	10 9 14
13	7 9 11	11 10 14
14	9 12 11	13 12 0

Array of neighbours

Array of neighbours]

Some computations with ungridded data are slow because of the need to search repeatedly through the elements. Considerable savings in searching time are possible if additional information is supplied about which elements are adjacent. This is supplied in the *array of neighbours*, an integer array with the same dimensions as the array of elements.

The neighbour array may be generated using `ZNEIGH` which scans the element array for details of which elements are attached to the edges of other elements, and records these details in the neighbour array. Alternatively, the neighbour array can be generated at the same time as the triangular element structure using `ZZORDR` or `ZZORDN` (see below).

It is not necessary to understand the structure of the neighbour array, but a brief explanation follows for those who are interested. Consider the `K`th element in data with `NODES` nodes per element. The element is defined as the area enclosed by joining the nodes whose subscripts are held in element array (`I2ARR`) by `NODES` straight lines. Each of the `NODES` edges either lies on the boundary of the data structure or coincides with an edge of another element. For each element, the array of neighbours contains a list of the element numbers attached to each edge, with zero when the edge lies on the boundary of the data. The order of the list of neighbours in the array of neighbours is directly related to the order of nodes in the array of elements.

Table 3.2 shows the data points and neighbours of the elements in Figure 3.7. Neighbour 0 indicates that there is no neighbour at one side of the element.

Triangulation

Triangulation]

To plot from ungridded data and to relate data points to each other, the x-y positions of the data points must be organized into an element structure. In the absence of a known element structure, Simpleplot can generate a structure consisting of triangles and, at the same time, generate the neighbour array (see above).

When the underlying x and y variables have similar units (*eg.* both the same units of distance), `ZZORDR` organizes the points into triangular elements. The number of triangular elements depends on the data configuration. The arguments of `ZZORDR` include the following:

- `XARR`, `YARR`, and `NPTS` to define the data positions;

- an array `I2ARR` dimensioned `I2ARR(3,ISIZE)` where `ISIZE` is the maximum expected number of elements;
- `ISIZE`, the maximum number of expected elements (approximately twice the number of data points);
- an `INTEGER` variable, `NELEMS`, to receive the actual number of elements.

If the resulting value of `NELEMS` equals `ISIZE`, it is possible that structuring is incomplete and the value of `ISIZE` should be increased.

Normalized triangulation

Normalized triangulation]

When ungridded data represent a function of two dissimilar variables (*eg.* time and distance), the numerical values in the x and y data sets are not comparable. `ZZORDR` triangulates data with x and y values in similar units; when the units are dissimilar, particularly when their magnitudes are significantly different, `ZZORDN` should be used, as `ZZORDR` might give unsatisfactory results or even fail.

`ZZORDR` calculates the triangles from the x-y values provided; `ZZORDN` calculates from values normalized to cover the range 0.0 to 1.0, without altering the data arrays.

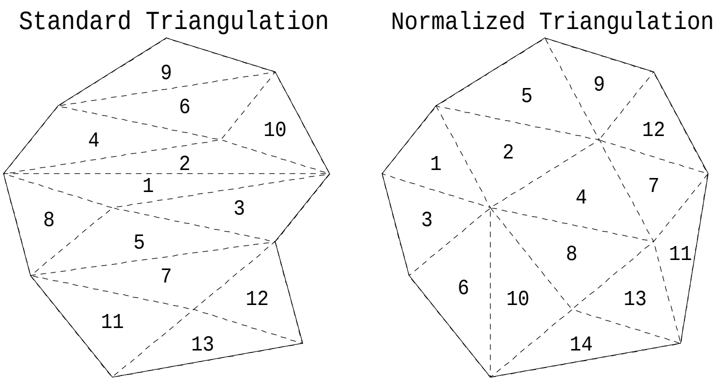


Figure 3.8 Triangulation

Triangulation can fail for a number of reasons:

- Points are all co-linear and no elements can be generated.
- The maximum size of the element array is too small and triangulation is incomplete.
- Standard triangulation fails due to the configuration of the data – normalized triangulation may be more successful.

3.4.2 Subroutine names

Subroutines and functions for manipulating or plotting ungridded data (defined by parallel arrays `XARR`, `YARR`, `ZARR`) have names beginning with `Z`. Processes which can be performed more efficiently with reference to a neighbour array have two alternative subroutines to perform them:

- Subroutines beginning with a single `Z*` which use the element structure alone,
- Subroutines beginning with `ZZ*` which include an additional argument, the neighbour array, `N2ARR(NODES,NELEMS)`

In general, since `ZZ*` subroutines are faster than their `Z*` equivalents, their use is strongly recommended. This manual describes only the `ZZ*` subroutines but details of the `Z*` subroutines are in the *SIMPLEPLOT Reference manual*.

Structuring ungridded data

The following subroutines are available for the generation of an element array and/or an array of neighbours:

`ZNEIGH(I2ARR,N2ARR,NODES,NELEMS)` generates an array of neighbours from an array of elements.

`ZZORDN(XARR,YARR,NPTS,I2ARR,N2ARR,NELEMS,ISIZE)` generates triangular elements and an array of neighbours using normalized x and y values.

`ZZORDR(XARR,YARR,NPTS,I2ARR,N2ARR,NELEMS,ISIZE)` generates triangular elements and an array of neighbours directly from x and y values.

Drawing the data structure

SIMPLEPLOT includes subroutines:

- to draw outlines of individual elements – `ZELEM`,
- to number data points and elements – `ZNUMB`,
- to draw the complete element structure – `ZZELMS`,
- to draw the data boundary – `ZZEDGE`.

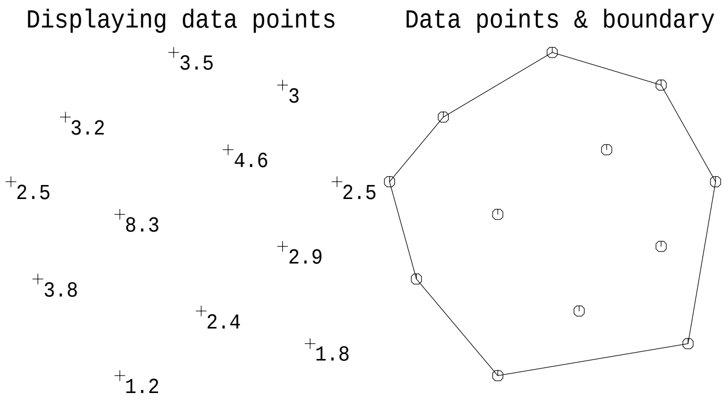


Figure 3.9 Ungridded data structure

`ZELEM(JELE,XARR,YARR,NPTS,I2ARR,NODES,NELEMS)` draws the outline of a single element on the current 2-D picture. `JELE` specifies the element number.

`ZNUMB(NTORF,ETORF)` specifies whether nodes and/or elements are to be labelled when they are drawn using `ZZELMS` or `ZELEM`. The area elements and/or the data points surrounding them can both be labelled with numbers. `NTORF` and `ETORF` are both `LOGICAL` arguments:

NTORF	*Surrounding data points*	ETORF	*Area elements*
.FALSE.	not numbered	.FALSE.	not numbered
.TRUE.	numbered	.TRUE.	numbered

By default, neither the elements nor the data points are numbered.

`ZZEDGE(XARR,YARR,NPTS,I2ARR,N2ARR,NODES,NELEMS)` draws the boundary of the data area covered by a set of elements, on the current 2-D picture.

`ZZELMS(XARR,YARR,NPTS,I2ARR,N2ARR,NODES,NELEMS)` draws the outlines of a set of elements on the current 2-D picture.

`ZZELMS` avoids redrawing any edges shared by two elements and, therefore, takes more time but generates less drawing than `ZELEM` called for every element.

Plotting ungridded 3-D data

The following subroutines plot ungridded three-dimensional data:

`ZZCNTS(XARR,YARR,ZARR,N,I2ARR,N2ARR,NODES,NELEMS)` draws a set of contour lines on the current picture.

`ZZCONT(ZLEV,XARR,YARR,ZARR,N,I2ARR,N2ARR,NODES,NELEMS)` draws the curve(s) corresponding to the specified contour level on the current picture.

`ZCUT(X1,Y1,X2,Y2,XARR,YARR,ZARR,N,I2ARR,NODES,NELEMS)` draws a 2-D curve of a surface section on the current 2-D picture.

`ZZSHAD(Z1,Z2,ISHAD,XARR,YARR,ZARR,N,I2ARR,N2ARR,NODES,NELEMS)` shades the area(s) between `Z1` and `Z2` on the current 2-D picture using shading pattern, `ISHAD`. The lower limit of the range is included, and the upper limit excluded.

`ZZSHDS(XARR,YARR,ZARR,N,I2ARR,N2ARR,NODES,NELEMS)` draws a shaded contour map on the current 2-D picture.

`ZZSURF(XARR,YARR,ZARR,N,I2ARR,N2ARR,NODES,NELEMS)` starts a new 3-D picture and draws the surface.

3.4.3 Scales and limits

`LIMEXC` returns the minimum and maximum values in the data arrays `XARR`, `YARR` and `ZARR`:

```
CALL LIMEXC(XARR,NPTS,XMIN,XMAX)
CALL LIMEXC(YARR,NPTS,YMIN,YMAX)
CALL LIMEXC(ZARR,NPTS,ZMIN,ZMAX)
```

For contour maps the 2-D plotting scales can be set using the x and y limits:

```
CALL SCALES(XMIN,XMAX,1,YMIN,YMAX,1)
```

3.4.4 Converting ungridded data to a regular grid

Plotting from ungridded data can be very slow, particularly when producing a surface picture. `KZZRG` generates a regular grid of data points, representing the same surface, from ungridded data.

`KZZRG(XARR,YARR,ZARR,N,I2ARR,N2ARR,NODES,NELEMS,Z2ARR,NX,NY)` generates a regular-gridded data set from ungridded data with neighbours.

The number of equally-spaced x values, `NX`, and the number of equally-spaced y values, `NY`, are specified as arguments. The array provided to receive the grid of values must be large enough for the chosen grid size, `Z2ARR(NX,NY)`.

Figure 3.10 illustrates a surface drawn from ungridded data using `ZZSURF`; the second picture is drawn from converted data using `RGSURF`.

No-data values are allocated to x-y grid values which lie outside the original data boundary so that the generated grid represents approximately the same area over the x-y plane as the original ungridded data.

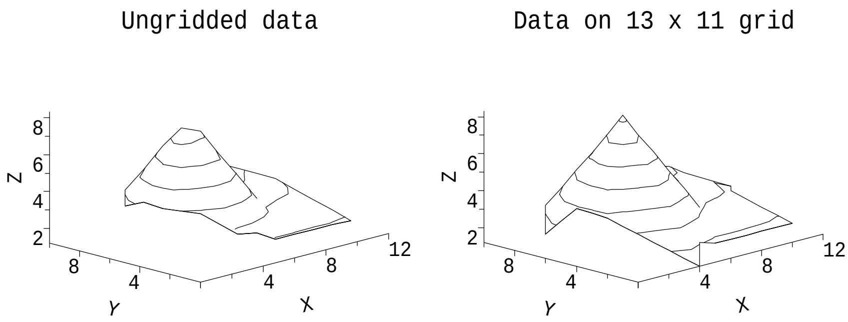


Figure 3.10 Converting ungridded data to a regular grid

3.5 Polar data

Polar data]

Contour maps can be plotted from polar data in exactly the same way as from Cartesian data (see Figure 3.11). Interpolation from polar data is also available.

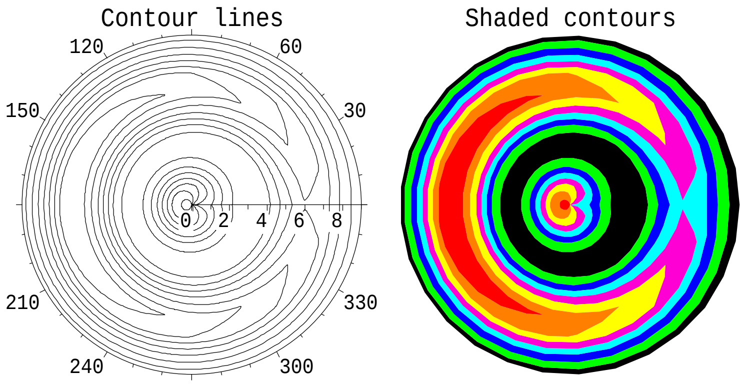


Figure 3.11 Polar contour maps

3.5.1 Coordinate interpretation

Coordinate interpretation]

When a polar picture has been started, or coordinate interpretation has been changed to polar before calling `*CNTS`, `*CONT`, `*SHDS` and `*SHAD` to plot contours, the data values are interpreted in terms of (r, θ) coordinates instead of (x, y) coordinates. Polar interpretation can be established in three ways:

`POLAR7(RADIUS,AXCAP)` is a composite plotting subroutine which starts a new picture, sets the scales, sets the coordinate interpretation to *polar* and draws a polar framework. The second argument, `AXCAP`, provides a caption for the radial axis.

If a new picture is started with `POLAR7` and `EQSCAL` has not been called, the coordinate interpretation is polar in degrees until switched by `COORDS` or the next new picture. If `EQSCAL` is active, the `XSTOP` argument of `EQSCAL` is used for the maximum radial value instead of the (`RADIUS`) argument of `POLAR7`, which is ignored.

`COORDS(IUNITS)` changes the interpretation of coordinates for subsequent plotting; `IUNITS` takes the same values as for `EQSCAL`. `COORDS` does not affect the current plotting scales but it changes the coordinate system used to refer to existing scales.

`EQSCAL(XSTART,XSTOP,YSTART,YSTOP,IUNITS)` specifies similar linear scales for 2-D Cartesian plotting or polar plotting. `EQSCAL` can specify reduced angular and radial scales for part-cycle polar charts. The scale limits can be expressed in different units indicated by the value of `IUNITS`:

`IUNITS`	*Units*	`XSTART`, `XSTOP`	`YSTART`, `YSTOP`
−1	Cartesian	Values ignored, centimetres used	
0	Cartesian	Horizontal units	Vertical units
1	Polar	Radial units	Angles in degrees
2	Polar	Radial units	Angles in radians
3	Polar	Radial units	User-defined scale

When `EQSCAL` is active, it also controls the type of any subsequent new pictures.

3.5.2 Gridded polar data

Data values on a polar grid lie at the intersections of the concentric circular arcs whose radii are equal to the set of r values, with straight lines drawn from the centre at the angles of the set of θ values. When the r values are equally spaced, the radii of the concentric circles progress in equal increments; when the θ values are equally spaced, all the angles between adjacent straight lines are equal. SIMPLEPLOT accepts gridded data on any of four different grids:

- Regular grid on which x (or r) and y (or θ) values are equally-spaced.
- Tartan grid with specified x (or r) values and equally-spaced y (or θ) values
- Tartan grid with specified y (or θ) values and equally-spaced x (or r) values
- Tartan grid with specified x (or r) and y (or θ) values

The four types of polar grid are illustrated in Figure 3.12.

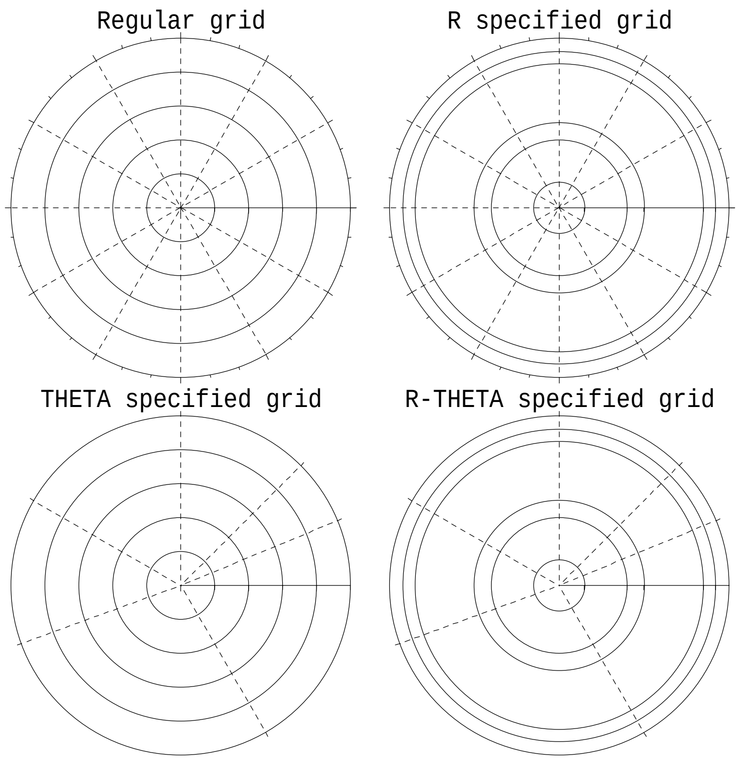


Figure 3.12 Polar data grids

The relationship between data and data grid is similar to Cartesian data. Equally-spaced r values are specified by `CALL SFEQX(XSTART,XSTEP)`. Equally-spaced θ values are specified by `CALL SFEQY(YSTART,YSTEP)`.

3.5.3 Polar functions, $z = f(r, \theta)$

Polar functions, $z = f(r, \theta)$]

By default, the area over which SIMPLEPLOT draws contours from a function $z = f(r, \theta)$ is the area of the picture; an alternative area is requested by a prior call of `FNAREA`:

```
CALL FNAREA(RMIN,RMAX,THMIN,THMAX)
```

`RMIN` and `RMAX` indicate the range of radial values to be included, and `THMIN` and `THMAX` indicate the range of angles.

3.5.4 Ungridded polar data

Ungridded data can be made up of points positioned at r and θ for each individual z value. The (r, θ, z) coordinates of `NPTS` data points are held in three parallel `REAL` arrays, `RARR`, `THARR` and `ZARR`, such that for each i, $(r_i, \theta_i, z_i) =$ (`RARR(I)`,`THARR(I)`,`ZARR(I)`).

Coordinate interpretation should be selected before generating a neighbour array and/or triangulation as well as before plotting from polar data.

4. Contour Plotting

This chapter describes how to draw and control individual contour curves and contour maps of 3-D data.

4.1 Introduction
- Contour map
- Individual contours
- Independent contour data scales and plotting scales

4.2 Controlling the contours
- Curve drawing algorithms (`CTCURV`)
- Broken line patterns (`CTBRKN` and `SQBRKN`)
- Controlling the underlying mesh (`SFMESH`)

4.3 Controlling the z scales
- Defining the z plotting range (`SFLIMS`)
- Setting the z plotting scale (`SFZSCL`)
- Controlling contour levels (`SFEQZ`, `SFEQZD` and `SQZVAL`)

4.4 Labelling contour curves
- Numbered contour levels (`CTNUMB` and `CTLABS`)
- Sequences of user-defined labels (`SQZLAB`)

4.1 Introduction

Contour lines are the lines on a map joining points of equal height or depth. There are four different forms of contour plotting with SIMPLEPLOT:

- Complete contour maps consisting of a set of contour lines (*CNTS)
- Individual contour lines (*CONT)
- Individual shaded areas between two contour levels (*SHAD)
- Complete shaded contour maps (*SHDS)

All four types of contour plotting are available with the six different data structures –RG*, X*, XY*, Y*, ZZ* and FN* (see Chapter 3).

Subroutines that control the characteristics of contour plotting have names starting CT (for ConTours), SF* (for SurFace) and SQ* (for SeQuences); SF* subroutines also affect isometric surface pictures (see Chapter 5). Contour plotting is also affected by the current 2-D plotting scales and coordinate interpretation, and the underlying mesh for 3-D data.

4.1.1 Contour map

Contour map]

For each data type, the subroutine *CNTS plots a complete map of contour curves, and a subroutine *SHDS plots a shaded contour map.

By default, SIMPLEPLOT contour maps are drawn with equally-spaced contour intervals over the full range of the data, and contour levels are not labelled.

4.1.2 Individual contours

Individual contours]

For each data type, the a subroutine *CONT plots the contour curve(s) at one data level, and *SHAD shades the region between two data levels.

4.1.3 Independent contour data scales and plotting scales

The grid coordinates are not included with equally-spaced gridded data, and when they have not been specified, they are assumed to coincide with the plotting scales: the coordinates at the first grid line are allocated the scale values at the bottom-left corner; the grid intervals are calculated internally to make the last grid line coincide with the top-right corner.

To contour equally-spaced gridded data with any other relationship to the plotting scales, the grid coordinates of the data must be specified before contouring by SFEQX/SFEQY.

When plotting scales have not been specified, natural scales are used: the bottom left-hand corner of the plot is the origin, and centimetre coordinates are assumed. 2-D plotting scales are specified using SCALES or EQSCAL:

EQSCAL(XSTART,XSTOP,YSTART,YSTOP,IUNITS) specifies proportional linear scaling in x and y. The scale limits are expressed in different units indicated by the value of IUNITS:

IUNITS	*Units*	XSTART, XSTOP	YSTART, YSTOP
−1	Cartesian	Values ignored, centimetres used	
0	Cartesian	Horizontal units	Vertical units
1	Polar	Radial units	Angles in degrees
2	Polar	Radial units	Angles in radians
3	Polar	Radial units	User-defined scale

`SCALES(XSTART,XSTOP,IXTYPE,YSTART,YSTOP,IYTYPE)` specifies linear and non-linear x and y scales for Cartesian plotting:

`IXTYPE` or `IYTYPE`	*Type of scale*
0	Natural (default)
1	Linear
2	Logarithmic
3	Normal probability (%)

When x-y plotting scales exceed the data limits the contours do not fill the whole picture. When the data limits exceed the plotting scales, part of the plotting is omitted and the following diagnostics are issued from `*CNTS`, `*CONT`, `*SHAD`, `*SHDS`:

```
(Contour curve not all in range)2
(Contour map not all in range)2
(Shaded contour not all in range)2
(Shaded contours not all in range)2
```

For more information about diagnostic messages, see Appendix E.

4.2 Controlling the contours

4.2.1 Curve drawing algorithms

Contour lines are drawn by joining interpolated points with some sort of curve:

`CTCURV(ITYPE)` specifies the curve type used for drawing contour lines according to the value of `ITYPE`:

`ITYPE`	*Curve type*
1	Tight-fitting smooth curve (default)
2	Loose-fitting smooth curve
3	Straight lines from point to point

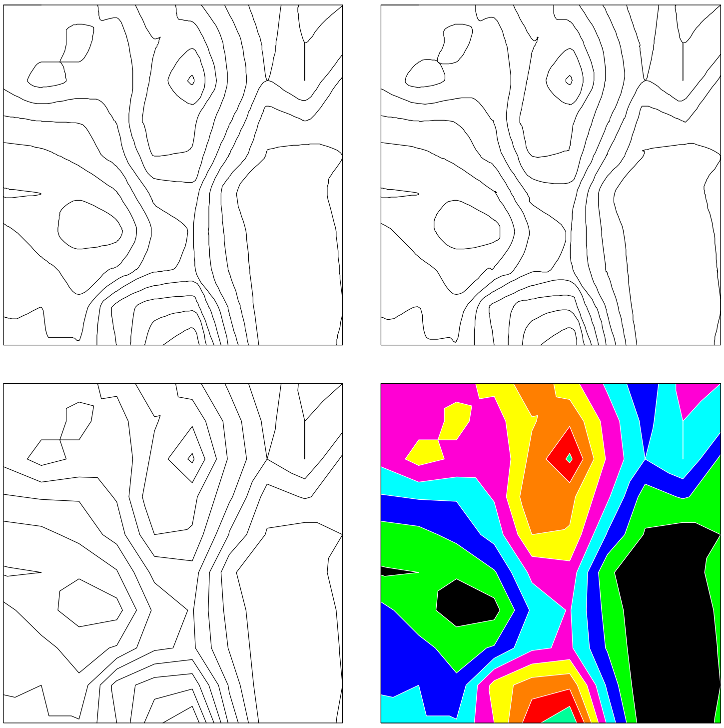


Figure 4.1 Different contour curve drawing algorithms
`CALL CTCURV(1)`, `CALL CTCURV(2)`, `CALL CTCURV(3)` and shaded contours

`CTCURV` controls the curve type used for all subroutines related to contour pictures (`*CNTS`, `*CONT` and `*CUT`) but `CTCURV` has no effect on shaded contours. Shaded contour regions are bounded by straight lines from point-to-point. A smoother appearance to the contour curves bounding such regions is achieved by increasing the underlying mesh used to calculate the points (using `SFMESH`).

4.2.2 Broken line patterns

Broken line patterns]

By default, contour curves, cross-sections and ungridded data configurations are drawn with solid lines. `CTBRKN` selects an alternative broken line pattern and `SQBRKN` specifies a sequence of line patterns:

`CTBRKN(LTYPE)` specifies the broken line pattern used for all subsequent drawing of contour lines from `*CONT` and `*CNTS`, cross sections from `*CUT`, and ungridded data configurations from `ZZEDGE`, `ZELEM` and `ZZELMS`.

`SQBRKN(IARR,NARR)` specifies a sequence of broken line patterns used for subsequent drawing of sets of lines; in particular, it affects sets of contour curves from `*CNTS`. The pattern specified by `IARR(1)` is used for the first contour level, `IARR(2)` for the second *etc.*

A maximum of 32 patterns may be specified as a sequence. When a contour map drawn by `*CNTS` shows more than 32 levels, the specified sequence is used for the first 32 levels, and the current global line style (solid default, or specified by `CTBRKN`) is used for the others.

`CALL SQBRKN(IARR,0)` restores the default.

`CTBRKN` and `SQBRKN` also affect the contour lines drawn on surface pictures (see Chapter 5).

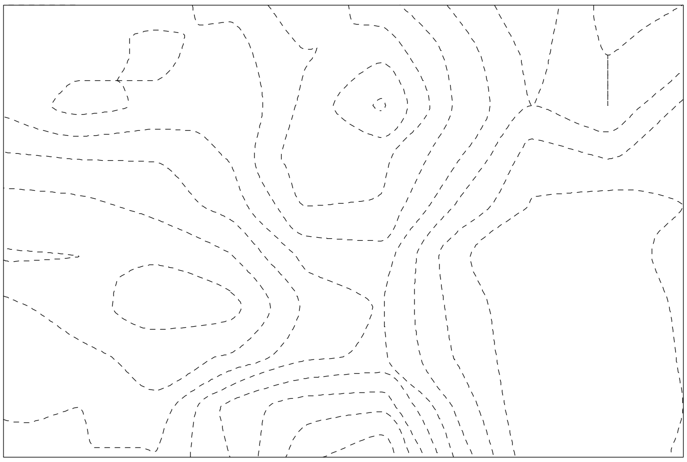


Figure 4.2 Broken line patterns
`CALL CTBRKN(-1)`

For details of the SIMPLEPLOT software broken line patterns, see Appendix D.

Broken lines can also be used as bundled pen attributes.

4.2.3 Controlling the underlying mesh

SIMPLEPLOT evaluates the points on a contour by linear interpolation along the lines of an underlying mesh; the contour curves are then constructed by joining interpolated points. `SFMESH` alters the concentration of interpolated points from which contours are constructed. Contour curves from `*CONT` and `*CNTS` can have their smoothness changed by `CTCURV`, but shaded contours from `*SHAD` and `*SHDS` are always drawn using straight lines; for these, smoother curves are produced by setting a finer mesh using `SFMESH`.

`SFMESH(MX,MY)` specifies the number of mesh lines to be used; `MX` specifies the number of lines at equally-spaced x intervals, and `MY` specifies the number of lines at equally-spaced y intervals. Larger values of `MX` and `MY` produce smoother contour curves but increase processing time.

By default, contour curves from gridded data are interpolated on the data grid. Calling `SFMESH` with `MX` and/or `MY` less than 2 restores the default.

If the mesh used differs from the data configuration, contouring may be slower.

`SFMESH` also affects surface pictures.

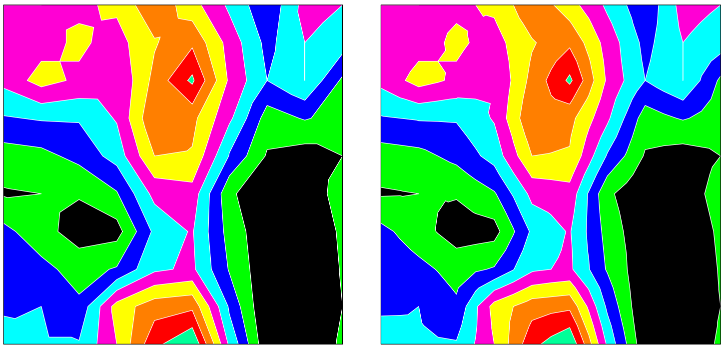


Figure 4.3 The underlying mesh
Default and after `CALL SFMESH`

4.3 Controlling the z scales

By default, contour maps are drawn with the z plotting range equal to the z data range, and the z plotting scale increasing over the z plotting range; the contour interval is chosen to give between five and ten equally-spaced levels over the z plotting scale.

4.3.1 Defining the z plotting range

By default, the z plotting range is set equal to the z data range. `SFLIMS` specifies an alternative z range for 3-D plotting.

`SFLIMS(Z1,Z2)` specifies the z plotting range; any parts of the supplied data which lie outside the range, `Z1` to `Z2`, are omitted.

When the z plotting scale has not been specified, an increasing scale over the z plotting range is used. As the default contour interval relates to the z plotting scale, it can be changed indirectly by `SFLIMS`.

The order of `Z1` and `Z2` has no significance. Calling `SFLIMS` with `Z1`=`Z2` restores the default.

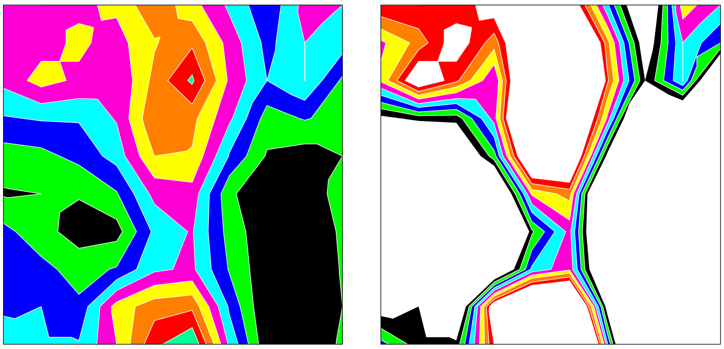


Figure 4.4 Defining the z plotting range
Data range: 0 to 90 and after `CALL SFLIMS`

Figure 4.4 shows the effect of `SFLIMS` on a shaded contour map. The first picture uses the default z plotting range; between five and ten contour intervals are allocated to cover the whole plotting scale. The second picture has a specified z plotting range lying completely within the data range; parts of the picture outside the specified range are left blank, and between five and ten contour intervals are allocated to cover the reduced range.

`SFLIMS` also affects surface pictures.

4.3.2 Setting the z plotting scale

By default, SIMPLEPLOT allocates a z plotting scale increasing over the z plotting range. `SFZSCL` specifies an alternative z plotting scale; this may be useful when comparable scales are needed for pictures of different data sets, when only part of the data range is of interest, or when a decreasing scale is required.

`SFZSCL(ZSTART,ZSTOP)` specifies the z scale ranging from `ZSTART` to `ZSTOP`.

A decreasing scale is set when `ZSTART` is greater than `ZSTOP`. Calling `SFZSCL` with `ZSTART`=`ZSTOP` restores the default. `SFZSCL` also affects surface pictures.

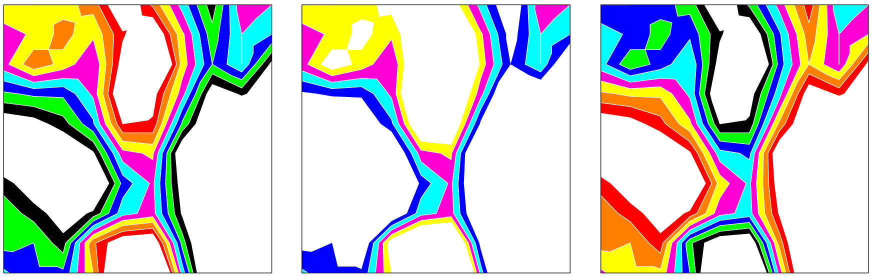


Figure 4.5 Controlling the z scale
CALL SFZSCL(20.0,60.0); SFLIMS and SFZSCL;
CALL SFZSCL(60.0, 20.0)

Figure 4.5 shows the effect of SFZSCL on a shaded contour map. The first picture shows that SFZSCL without SFLIMS behaves like SFLIMS for increasing scales. The second picture shows how SFZSCL and SFLIMS interact; the z plotting scale set by SFZSCL controls the setting of contour intervals, and the z plotting range set by SFLIMS controls what is included. The third picture shows that by reversing the z plotting scale, the contour sequence is reversed.

When the z plotting range extends beyond the z plotting scale set by SFZSCL, only part of the picture can be drawn; this contributes to the number of incomplete picture tasks and a diagnostic is issued at the end of the picture.

4.3.3 Controlling the contour levels

By default, SIMPLEPLOT chooses an offset value and an interval to produce between five and ten contour levels at suitable equally-spaced values.

SFEQZ(ZSTART,ZSTEP) sets the equally-spaced contour levels by specifying an offset value and interval. After SFEQZ has been called, sets of equally-spaced contour curves are drawn at all levels ZSTART+(n×fvarZSTEP) (where n is a positive or negative integer) lying within the z plotting range.

SFEQZD(NSTEPS,DELTA) defines the contour interval for discrete data by specifying the number of contours, NSTEPS, and the minimum interval between data values, DELTA. After SFEQZD has been called, the z plotting range is divided into NSTEPS equal intervals, allowing a margin of DELTA $\times$ 0.5 at each end of the range to accommodate discrete data values correctly. No attempt is made to ensure that the contour levels occur at 'simple' numbers.

SFEQZ overrides SFEQZD and *vice versa*.

CALL SFEQZD(0,DELTA) or CALL SFEQZ(0.0,0.0) restores the default.

SQZVAL(RARR,NARR) specifies a sequence of contour levels which may be at unequal intervals. The number of contour levels is unlimited but a maximum of 32 values may be specified as a sequence.

CALL SQZVAL(DARR,0) restores the default.

SFEQZ, SFEQZD and SQZVAL also affect the contour levels drawn on surface pictures.

Figure 4.6 Contour intervals
Default and `CALL SFEQZ(0.0, 5.0)`

4.4 Labelling contour curves

Labelling contour curves]

By default, contour levels are not labelled, but either automatically generated numbers or user-defined textual labels may be drawn. The format of the numerical labels is controlled by `FIGFMT` and `FIGSGN` .

4.4.1 Numbered contours

Numbered contours]

`CTNUMB(TORF)` specifies whether contour curves are to be labelled.

After a call of `CTNUMB(.TRUE.)`, the z value associated with each contour curve is drawn near the curve when a sequence of ten or more points is joined in one operation. `CALL CTNUMB(.FALSE.)` restores the default.

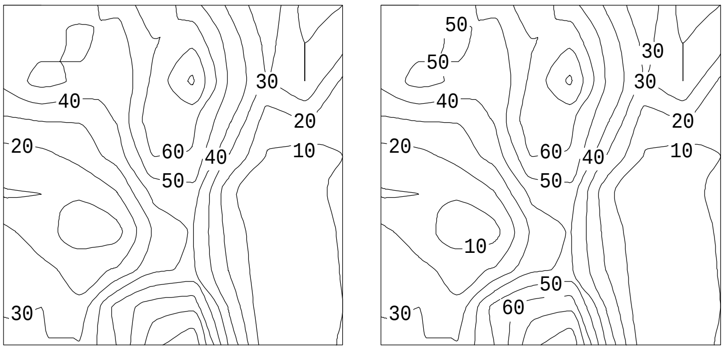


Figure 4.7 Labelling contour levels
`CALL CTNUMB(.TRUE.)` and `CALL CTLABS(5)`

`CTLABS(IFREQ)` specifies the frequency of labels included with contour curves.

After a call of `CTLABS(IFREQ)`, the z value associated with each contour curve is drawn near the curve when a sequence of `IFREQ` or more points is joined in one operation. For more frequent labelling, reduce `IFREQ` and for less frequent labelling, increase `IFREQ`.

`CTLABS(10)` is equivalent to `CTNUMB(.TRUE.)` and `CTLABS(0)` is equivalent to `CTNUMB(.FALSE.)`; `CALL CTLABS(0)` restores the default.

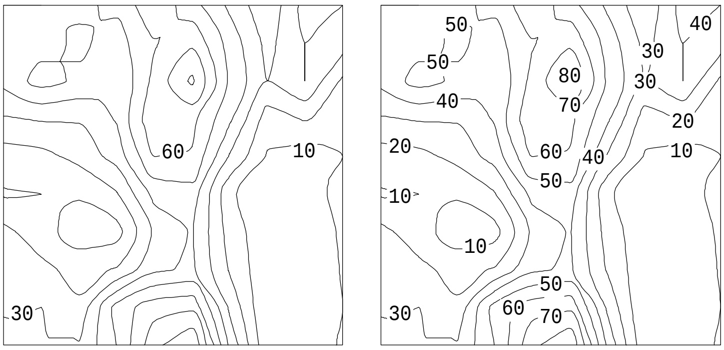


Figure 4.8 Different frequencies of contour labels
`CALL CTLABS(15)` and `CALL CTLABS(2)`

4.4.2 Sequences of contour labels

Sequences of contour labels]

An alternative set of textual labels can be supplied to be triggered by CTLABS or CTNUMB:

SQZLAB(LABARR,NARR) specifies a sequence of contour labels.

The number of contour labels is unlimited but a maximum of 32 labels may be specified as a sequence; when the sequence is exhausted, labelling ceases. The length of individual labels is restricted to a maximum of 20 characters; longer labels are truncated. CALL SQZLAB(LABARR,0) restores the default.

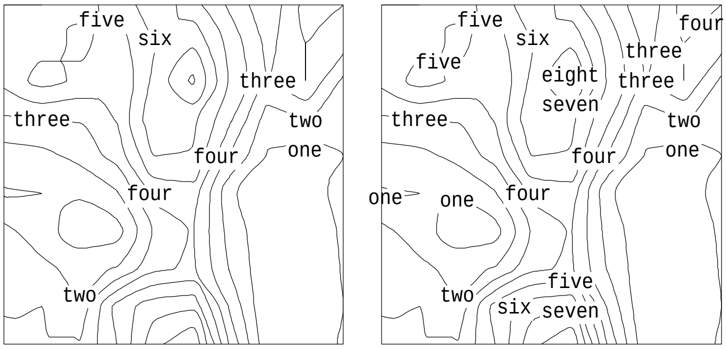


Figure 4.9 A sequence of user-defined contour labels

The CT* subroutines affect contouring subroutines and subroutines for drawing the data configuration and cross sections.

5. Surface Pictures

This chapter describes how to draw and control surface pictures of three-dimensional data.

5.1 Introduction

A single-valued continuous function of two independent variables can be represented in three-dimensional space by a surface over a fixed plane, and represented graphically by drawing the surface with hidden lines eliminated. SIMPLEPLOT can plot 3-D data as an isometric representation of a surface viewed from any corner.

Different pens can be selected to distinguish between positive and negative data, and between the top of the surface and its underside (see section 7.5.2).

Isometric plots can be annotated by an x-y-z diagram or 3-D axes.

5.1.1 Subroutine names and new pictures

Surface pictures can be drawn from any of the six data structures described in Chapter 3 using `RGSURF`, `XSURF`, `YSURF`, `XYSURF`, `FNSURF` and `ZZSURF`. All the `*SURF` subroutines start a new isometric (3-D) picture as well as draw the surface.

Subroutines that control the characteristics of ISometric plots have names beginning with `IS*`, and subroutines that control characteristics of SurFace plots (*ie.* isometrics and contours) have names beginning with `SF*` (see also Chapter 4). Sequence control subroutines (`SQ*` for SeQuence) and `CTBRKN` may also affect surface pictures when contour lines are drawn on them.

5.2 Drawing surfaces

5.2.1 Different types of surface picture

By default, the `*SURF` subroutines draw a surface outline; `ISTYPE` selects one of five types of surface picture.

`ISTYPE(ITYPE)` specifies an alternative type of surface picture:

ITYPE	*Type of surface picture*
1	Surface outline (default)
2	Surface outline and a set of contour lines
3	Crosshatched picture
4	Cascade curves $z_i = f(x)$ for a set of equally- spaced y_i values
5	Cascade curves $z_i = f(y)$ for a set of equally- spaced x_i values

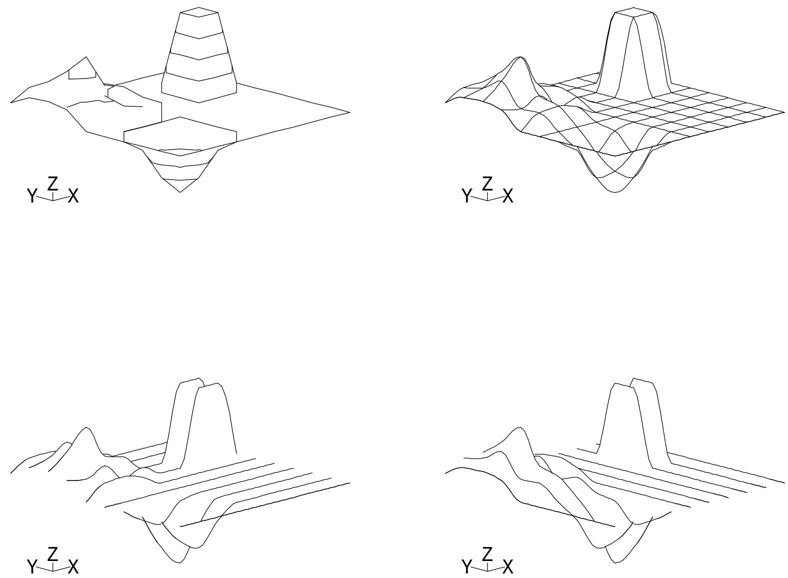


Figure 5.1 Different types of surface picture
`CALL ISTYPE(2)`, `CALL ISTYPE(3)`, `CALL ISTYPE(4)` and `CALL ISTYPE(2)`

These are described in more detail below:

Surface outlines: The surface outlines drawn with `ITYPE`=1 or 2, are drawn as fast as possible using straight lines from point-to-point; the smoothness of the outline depends on the fineness of the underlying mesh. The mesh size is altered by `ISMESH` and `SFMESH`.

Contour lines: The contour lines drawn with `ITYPE`=2 are drawn in the same way as if `*CNTS` had been called to add a set of contour lines to a simple surface outline; the appearance of the contour lines is affected by the contour control subroutines described in Chapter 4:

- `CTBRKN` and `SQBRKN` specify the broken line patterns used for contour lines.

- SFEQZ, SFEQZD and SQZVAL control the contour interval(s).

Contour labels cannot be added to surface pictures.

Crosshatched surfaces and cascade curves: The number of curves drawn in crosshatched and cascaded pictures (ITYPE=3, 4 or 5) corresponds to the number of grid lines in the underlying mesh, controlled by ISMESH and SFMESH; curves are interpolated between the points of the underlying mesh.

SQPEN may be called to define a sequence of pens to distinguish a sequence of z ranges on an isometric plot. Colour sequences can significantly reduce the speed of drawing because each change of colour is executed by a separate pass over the data set. Pen usage on surface pictures is further described in section 7.5.2.

5.2.2 Axes on surface pictures

The range of values represented on a surface picture is bounded by a cuboid. The surface can be represented from any corner (ISVIEW), rotated about a horizontal line (ISANG), or represented in mirror image (ISYUP).

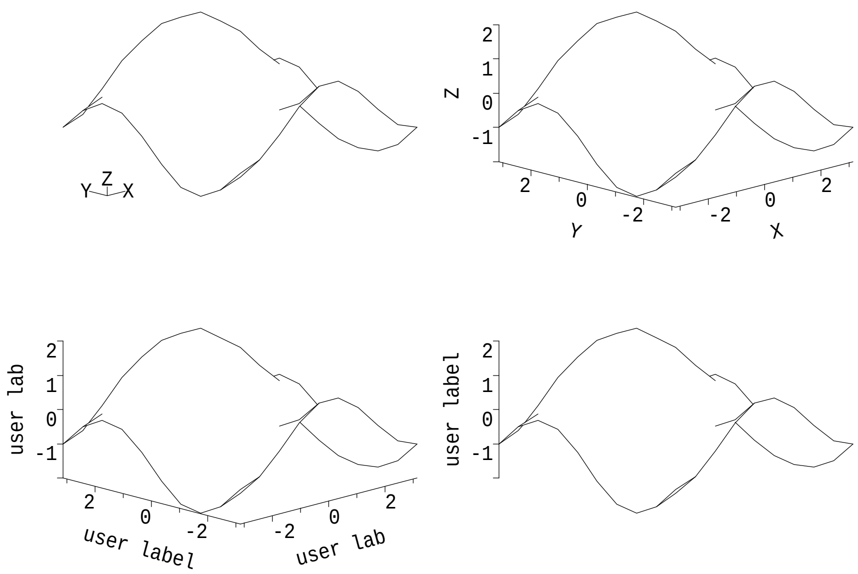


Figure 5.2 Different methods of drawing isometric axes
default; ISAXES; ISAXD7 and ISDIAG; AXIS7 and ISDIAG

By default, surface pictures are drawn with a diagram of x-y-z arrows indicating the direction of change of x, y and z. After CALL ISAXES(.TRUE.) this diagram is omitted, but x-y-z axes are drawn with subsequent surface pictures; the x-axis is labelled 'X', the y-axis is labelled 'Y', and the z-axis is labelled 'Z'. Alternatively, AXIS7 adds a single 3-D axis and ISAXD7 adds a complete sets of three axes. Automatic axes requested by ISAXES always have labels 'X', 'Y' and 'Z'; added axes can have specified labels.

These axes are annotated with the 3-D plotting scales.

Table 5.1 Methods of drawing isometric axes

Subroutine	*Effect*	*Axis labels*
ISAXES	Axes drawn with each subsequent 3-D picture	'X', 'Y' and 'Z'.
ISAXD7	Add set of axes to current 3-D picture	Specified by the arguments.
AXIS7	Add one axis to current 3-D picture	Specified by the argument.

Table 5.2 Facilities for isometric axes

Subroutine	'XI'	'YI'	'ZI'
AXCLR	√	√	√
AXGRID	×	×	×
AXIS7	√	√	√
AXLAB7	√	√	√
AXLBGP	√	√	√
AXLBJS	×	×	√
AXLOCN	√	√	√
AXRNGE	√	√	√
AXSBDV	√	√	√

AXIS7(CHAXIS,CAP) draws an individual axis on the current surface picture where CHAXIS equals 'XI', 'YI' or 'ZI' for the x-axis, y-axis or z-axis respectively; CAP is the caption for the specified axis.

ISAXD7(CAPX,CAPY,CAPZ) draws a set of x-y-z axes on the current surface picture where CAPX, CAPY and CAPZ are the captions for the x, y and z axes respectively.

ISAXES(TORF) specifies whether or not x-y-z axes are drawn by the *SURF subroutines. When axes are requested this way, the x-y-z diagram is omitted. CALL ISAXES(.TRUE.) has effect until the next CALL ISAXES(.FALSE.) restores the default.

ISDIAG(TORF) specifies whether or not an x-y-z diagram is drawn by the *SURF subroutines. When axes are to be drawn using ISAXD7 or AXIS7, CALL ISDIAG(.FALSE.) is needed. CALL ISDIAG(.FALSE.) has effect until the next CALL ISDIAG(.TRUE.).

The three methods for drawing axes on surface pictures are summarized in Table 5.1 and illustrated in Figure 5.2.

Axis limitations

In order to avoid interference between axes and surface pictures, the only positions allowed for each isometric axis are the two outer extremes of the surface, controllable by AXLOCN. The z-axis may be at either side of the surface (default left). By default, the x-y axes are at the minimum value of z – bottom when when z is increases upwards, and top when z increases downwards (see Figure 5.10, page 50).

Grids are not available on surface pictures therefore AXGRID is not available for isometric axes. Table 5.2 summarizes the facilities which are available with isometric axes. Further details can be found in the *SIMPLEPLOT Reference manual*.

5.2.3 Filling the space available

Filling the space available]

By default, surface pictures are scaled so that all rotations (using ISANG) are drawn to the same scale within the picture size. This allows comparisons of rotated surfaces but may result in a very small picture with a large empty area around it. `ISFULL` specifies whether or not surfaces are to be drawn as big as possible within the picture limits, but without changing the aspect ratio.

`ISFULL(TORF)` specifies whether surface pictures fill the space available.

After a call of `CALL ISFULL(.TRUE.)` each surface picture is drawn on a scale to make the surface as big as possible within the picture limits. The surfaces in Figure 5.3 are drawn with a box around the picture area.

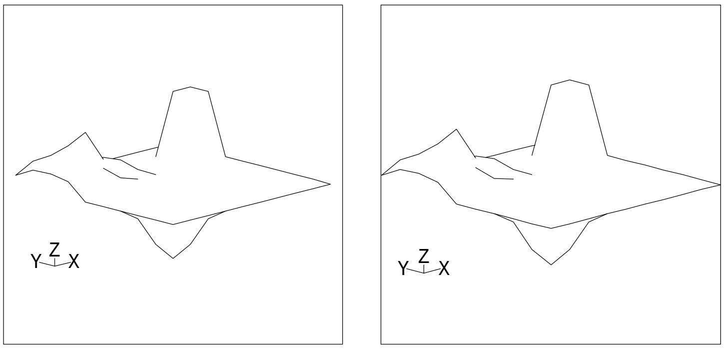


Figure 5.3 Filling the space available
`ISFULL(.FALSE.)` and `ISFULL(.TRUE.)`

5.3 Controlling the projection

SIMPLEPLOT can plot 3-D data as an isometric representation of a surface viewed from any corner. The surface may be rotated vertically by changing the angle between the horizontal plane and its base plane through 360°. The z scale of the plot can be adjusted by specifying the surface height relative to its width.

5.3.1 Vertical rotation

Vertical rotation]

By default, surfaces are drawn such that the angle between the horizontal plane and the base plane is 15°. `ISANG` specifies an alternative angle (see Figure 5.4).

`ISANG(ANGLE)` specifies the angle in degrees at which surface pictures are drawn.

> The viewing corner of the data (see `ISVIEW`) is positioned at the front of the picture before rotation is performed, and therefore will not be at the front of the picture for angles between 90.0° and 270.0°.
>
> `CALL ISANG(15.0)` restores the default.

By default, the x-y axes are drawn at the minimum value of z. Figure 5.4 shows how the position of the x-y axes switches from the bottom of the picture to the top as the surface is rotated by `ISANG`.

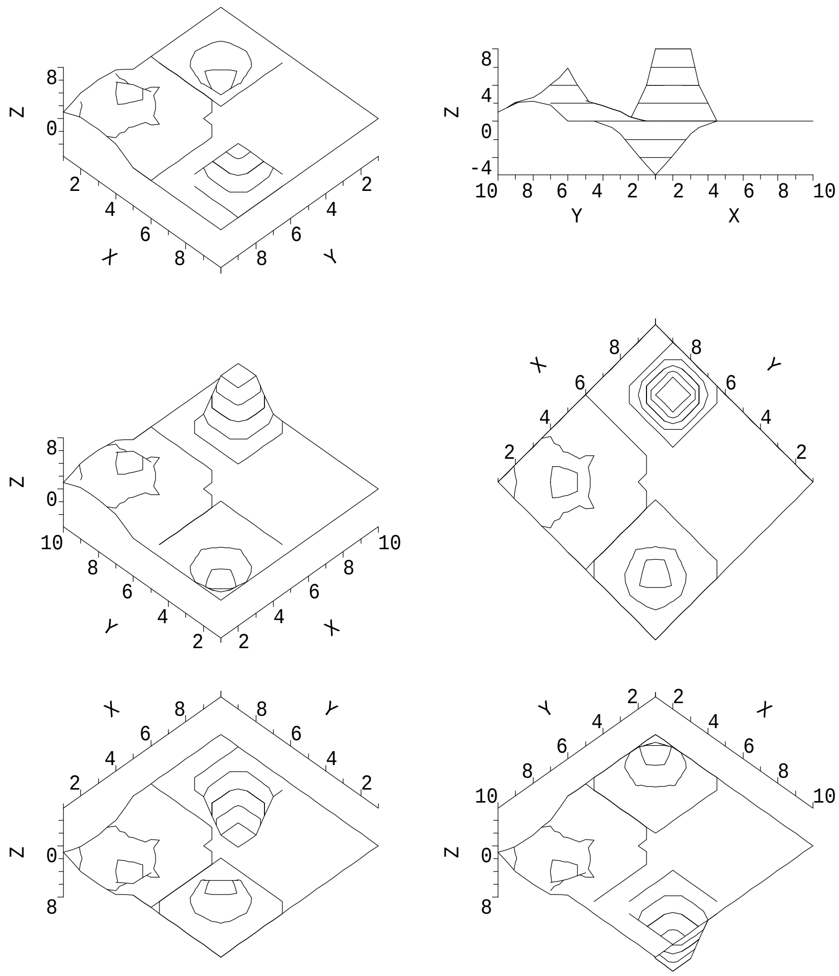


Figure 5.4 Changing the angle of view
`CALL ISANG(-45.0)`, (0.0), (45.0), (90.0), (135.0) and (225.0)

5.3.2 Horizontal rotation

Horizontal rotation]

Surface pictures cover a rectangular region of the x-y plane (even when the data structure plotted by *SURF is non-rectangular). By default, the surface is viewed from the corner of minimum x and minimum y. Four horizontal rotations are available, allowing the surface to be viewed from any of its corners. `ISVIEW` specifies which corner to be at the front.

`ISVIEW(ICORNR)` specifies the viewing corner; corners are numbered 0, 1, 2, 3 in a clockwise direction, starting from 0 at the first data point (*ie.* minimum x and minimum y).

The viewing corner is positioned at the front of the picture before any rotation is performed.

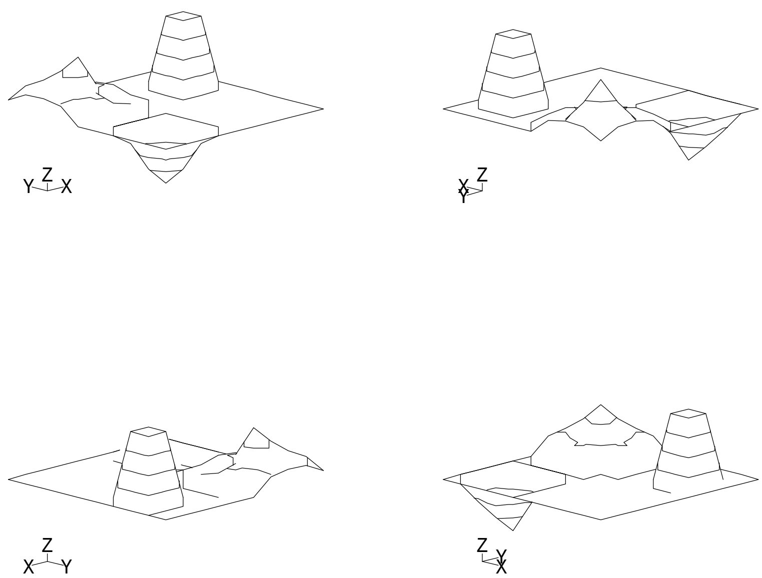


Figure 5.5 Views from different corners
`CALL ISVIEW(0)` (default), `CALL ISVIEW(1)`, `CALL ISVIEW(2)` and `CALL ISVIEW(3)`

5.3.3 Height control

Height control]

By default, the z scale of isometric plots is set to make the surface height 0.4 of its width. `ISRISE` specifies the heights of surfaces.

`ISRISE(FACTOR)` specifies the surface heights on surface pictures as `FACTOR`×*picture width*. `CALL ISRISE(0.4)` restores the default.

`ISRISE` modifies the length of the vertical scale of a surface, without changing the range of z values covered.

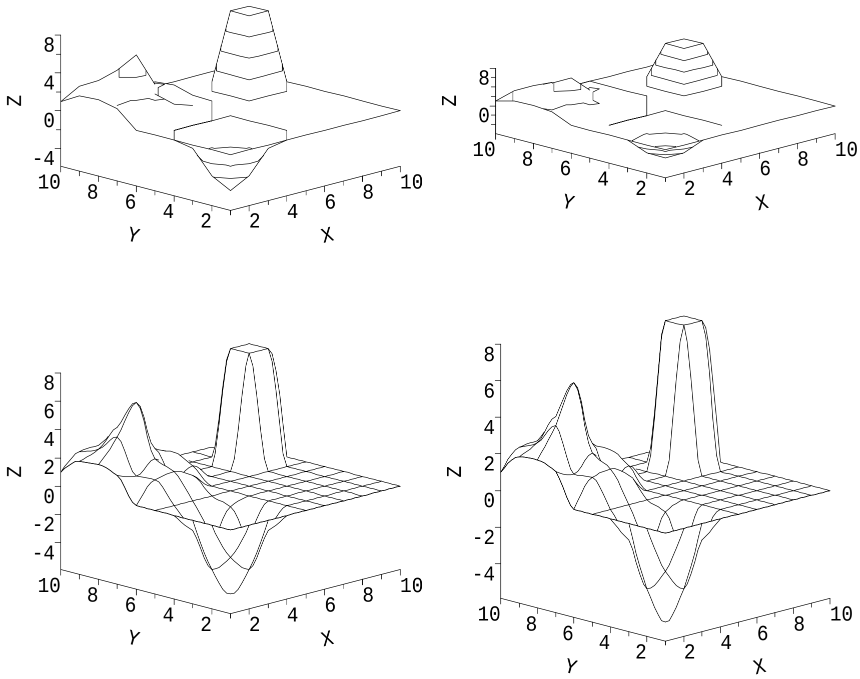


Figure 5.6 Adjusting the height
`CALL ISRISE(0.4)` (default), `CALL ISRISE(0.2)`, `CALL ISRISE(0.6)` and `CALL ISRISE(0.8)`

5.3.4 Mirror images

Mirror images]

By default, Simpleplot surface pictures show data with a *right-handed* coordinate system, showing x increasing from left to right and y increasing upwards. This is the convention for Cartesian plotting. The convention for tabulated data is a *left-handed* coordinate system with y increasing downwards; surface pictures using the left-handed coordinate system are mirror images of right-handed ones. `ISYUP` specifies the direction of the y scale on surface pictures.

`ISYUP(TORF)` specifies the direction of change of y on surface pictures.
  If `TORF=.FALSE.`, then y values are represented as increasing downwards, as is conventional for tables.

`ISYUP` specifies that a left-handed coordinate system is to be used. The left-handed coordinate system is a mirror image of the right-handed system.

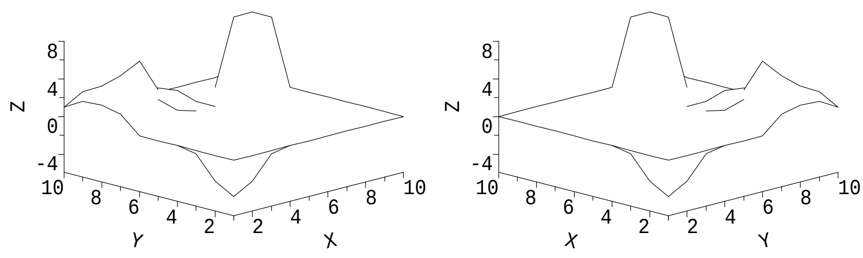


Figure 5.7 Mirror images
`ISYUP(.TRUE.)` and `ISYUP(.FALSE.)`

5.4 Controlling the surface

5.4.1 Smoothness of curves

Smoothness of curves]

Outline surfaces are drawn as rapidly as possible using straight lines between points on an underlying mesh; on crosshatched and cascaded pictures (see `ISTYPE`), all the visible parts of mesh lines are drawn, and curves are interpolated between mesh intersections. By default, the number of steps used for this interpolation depends on the picture size; fewer steps are used on smaller pictures.

`ISCURV(NSTEPS)` specifies the number of interpolated steps between mesh intersections on crosshatched and cascaded surface pictures. After `ISCURV` has been called, each piece of curve between intersections is drawn in `NSTEPS` steps, interpolated on a smooth curve through the intersections. Small values of `NSTEPS` increase speed, large values increase smoothness.

Calling `ISCURV` with `NSTEPS` less than 2 (the minimum value possible) restores the default.

The fineness of the underlying mesh is controlled by `ISMESH` or `SFMESH`.

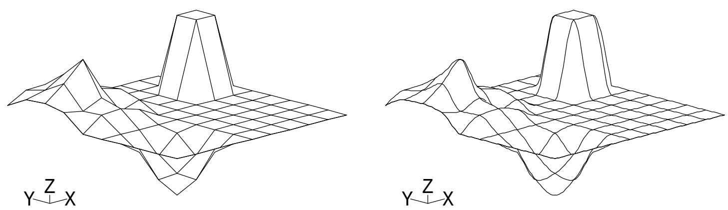


Figure 5.8 Controlling the interpolation
`CALL ISCURV(2)` and `CALL ISCURV(10)`

5.4.2 Controlling the underlying mesh

The processing of all surface pictures is based on an underlying mesh of lines at equally-spaced x values and equally-spaced y values; the x and y mesh intervals are always equal in size on the picture. This mesh corresponds to the lines drawn on crosshatched (and cascaded) surfaces. By default, the relative lengths of the x and y scales depend on the relative x and y data ranges; when the data ranges are unspecified (only possible with equally-spaced gridded data) the relative lengths depend on the numbers of points in the data grid.

`ISMESH` specifies the concentration of mesh to be used, retaining similar scale length proportions to the default. `SFMESH` specifies the mesh to be used, possibly changing the scale length proportions.

`ISMESH(MXY)` specifies the total number of x lines and y lines (`MXY`=`MX`+`MY`).

The relative values of `MX` and `MY` determine the relative lengths of the x scale and y scale in the picture. The distribution of `MXY` between `MX` and `MY` depends on the data type and the x-y ranges of the data. For equally-spaced gridded data, if either of the x-y scales has not been specified, `MX` and `MY` are given values proportional to the numbers of data values in the x and y directions; when both of the x-y scales are known, `MX` and `MY` are set in a similar ratio to x-range : y-range, unless that ratio would produce either `MX` or `MY` less than 5, in which case `MX`=`MY`=`MXY`/2.

By default, for data on a regular grid the data grid is used, and for all other types of data `MXY`=20 is used. Calling `ISMESH` with `MXY` less than 4 restores the default.

`SFMESH(MX,MY)` specifies the numbers of mesh lines to be used; `MX` specifies the number of lines at equally-spaced x intervals, and `MY` specifies the number of lines at equally-spaced y intervals. Calling `SFMESH` with `MX` and/or `MY` less than 2 restores the default.

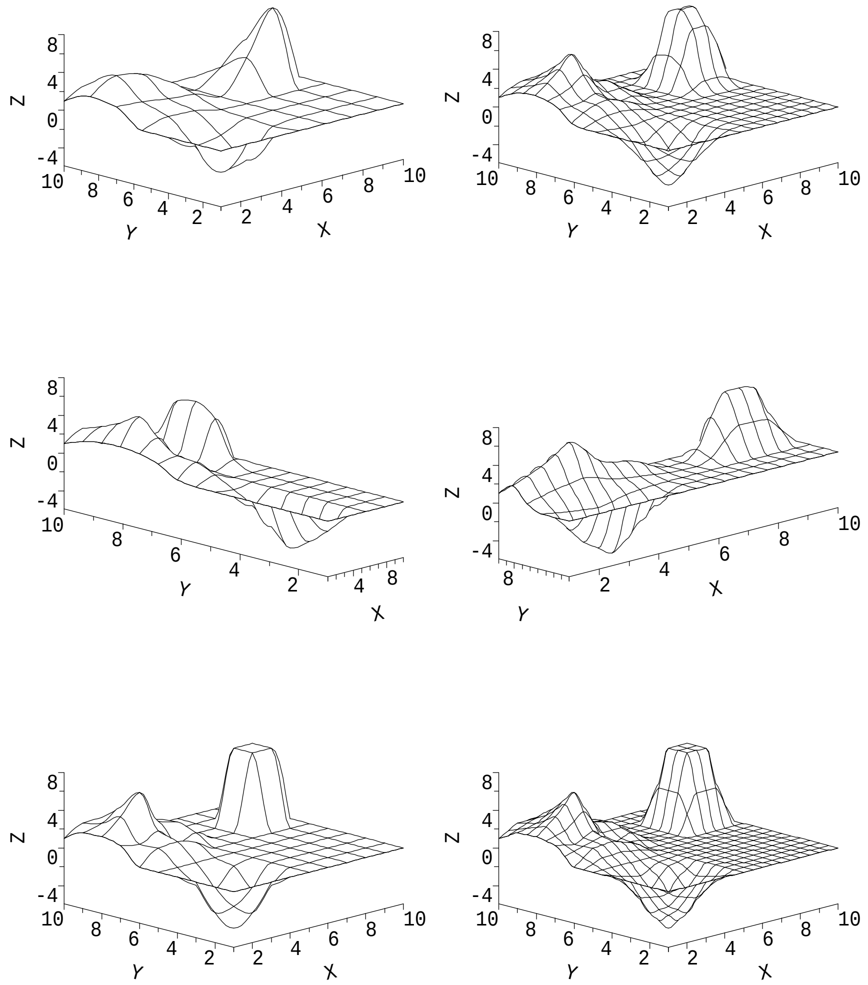


Figure 5.9 Underlying mesh
`CALL ISMESH(15)`, `CALL ISMESH(30)`, `CALL SFMESH(5,15)`, `CALL SFMESH(20,6)`, `CALL SFMESH(NX,NY)` and `CALL SFMESH(2*NX-1,2*NY-1)`

`MX` and `MY` define the mesh used for calculations, but also determine the number of curves drawn in crosshatched and cascaded pictures (`ISTYPE(ITYPE)` with `ITYPE`=3, 4 or 5).

When the mesh used differs from the data configuration, plotting may be slow.

Larger values of `MX` and `MY` produce surface pictures with finer detail but the processing is necessarily slower.

`ISMESH` only affects surface pictures; `SFMESH` affects surface pictures, contour curves and also determines the underlying mesh used for evaluating user-defined functions.

`ISMESH` has no effect while `SFMESH` is active.

5.5 Controlling the z scales

5.5.1 Setting the z plotting scale

By default, surface pictures are drawn with an increasing z scale covering the z plotting range. SFZSCL specifies the z scale of surface pictures and contour maps independently of the data set; this may be useful when comparable scales are needed for different pictures, when only part of the data range is of interest, or when a decreasing scale is required.

SFZSCL(ZSTART,ZSTOP) specifies a z scale ranging from ZSTART at the bottom of the picture to ZSTOP at the top. If ZSTART is greater than ZSTOP, the scale is decreasing.

Calling SFZSCL with ZSTART=ZSTOP restores the default. SFZSCL also affects the z scale on contour maps and map keys.

If the user-specified scale cannot accommodate the entire surface, part of the picture cannot be drawn; this contributes to the number of incomplete picture tasks and a diagnostic is issued at the end of the picture.

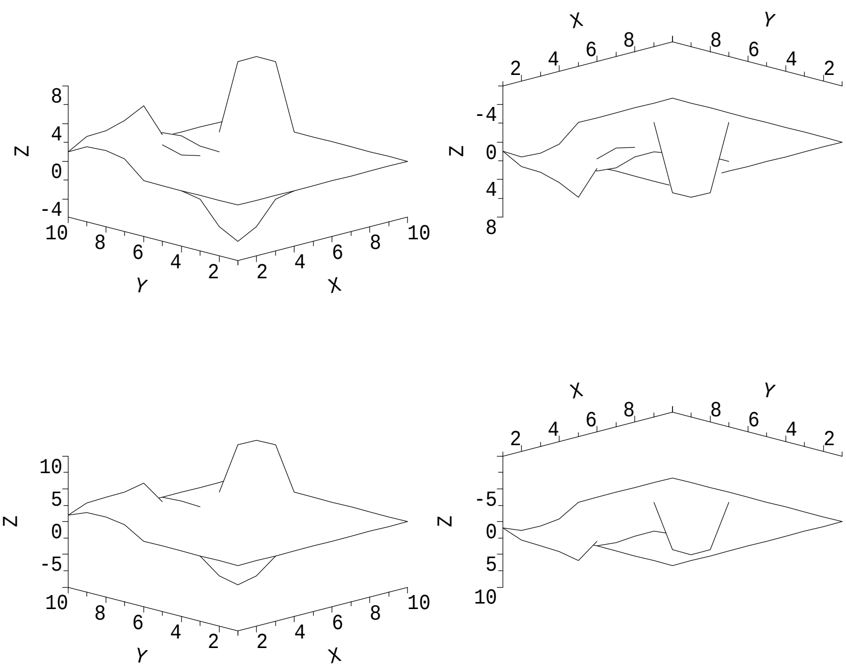


Figure 5.10 Setting the z scale
default, CALL SFZSCL(8.0,-6.0), CALL SFZSCL(-10.0,10.0), CALL SFZSCL(10.0,-10.0)

5.5.2 Defining the z plotting range

By default, the z plotting range is set equal to the z data range. SFLIMS specifies an alternative z range for 3-D plotting.

SFLIMS(Z1,Z2) specifies the z plotting range; any parts of the supplied data which lie outside the range Z1 to Z2 are then omitted.

The order of Z1 and Z2 has no significance. Calling SFLIMS with Z1=Z2 restores the default. SFLIMS also affects the range of data represented on contour maps and map keys.

On surfaces, holes are left in the surface where data values lie above or below the specified limits. `SFLIMS` alone acts like `SFZSCL` for increasing scales, but when both have been called, the plotting scale is controlled by `SFZSCL` and the plotting range is the intersection of the data range, the range specified by `SFLIMS`, and the scale specified by `SFZSCL`.

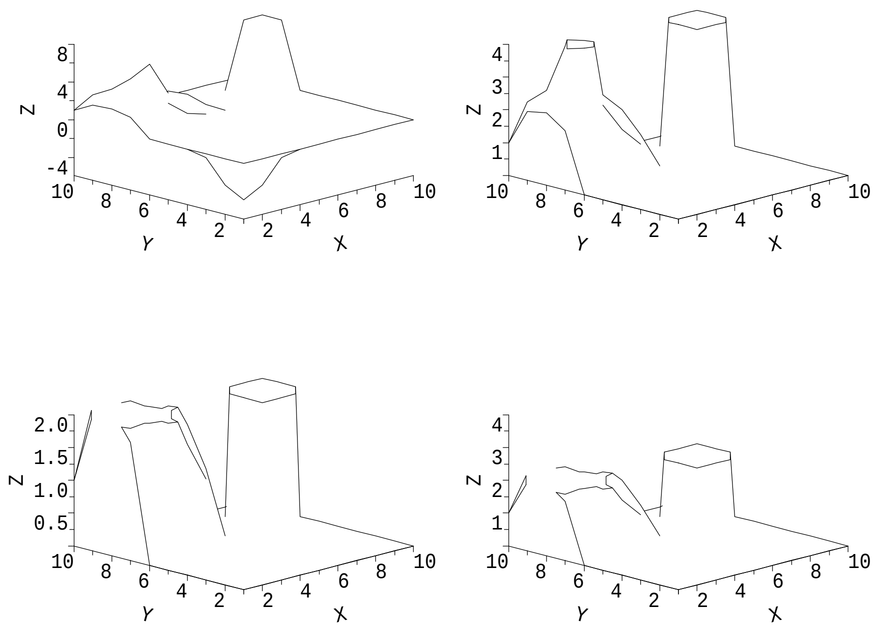


Figure 5.11 Changing the z range
default; `CALL SFZSCL(0.0,4.0)`; `CALL SFLIMS(0.0,2.0)`; `SFLIMS` and `SFZSCL`

5.6 Contour lines on surface pictures

A set of contour lines can be drawn with the surface (using `ISTYPE(2)`) or added to the current surface picture by `*CNTS`; individual contour curves are added using `*CONT`.

The intervals between contours on surfaces are controlled in exactly the same way as they are on contour maps.

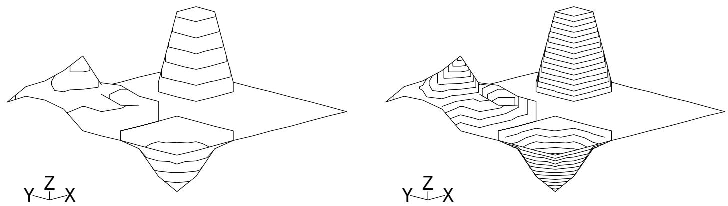


Figure 5.12 Contour intervals on surfaces
`CALL SFEQZ(0.0,1.5)` and `CALL SFEQZ(0.0,0.5)`

5.7 Isometric drawing

Isometric drawing]

On surface pictures, isometric coordinates are described by three variables, (x, y, z), but the underlying coordinate system remains Cartesian and can be addressed as such. Any of the general plotting subroutines can be used by first converting a 3-D isometric coordinate into a 2-D user coordinate using `KISXY`.

6. Waterfall Charts

This chapter describes how to draw and control waterfall charts:

6.1 Introduction

A waterfall chart consists of a family of curves plotted against a single independent variable. Each curve is displaced from the previous curve by a constant offset and is masked by the previous curve. A waterfall chart is comparable with the cascade version of a surface picture, where the surface is represented by a series of curves of z against x for a set of equally-spaced y values.

`WFCHT` plots a waterfall chart on the current 2-D picture, complete with annotated axes. Alternatively, `WFDRAW` adds waterfall curves individually.

Waterfall characteristics which can be specified include

- Number of curves on a waterfall chart – `WFNCVS`
- Displacement between curves – `WFSTEP`
- Alternative label scale – `WFEQL`
- Alternative numeric scale for data – `WFEQN`
- Alternative numeric scale limits – `WFNSCL`
- Alternative limits for individual curves – `WFZSCL`
- Data value at which curves change pens – `WFZLEV`
- Pens used by `WFZLEV`, and to draw masked curves – `WFPNS`

6.1.1 A simple waterfall chart

`WFCHT(Z2ARR,NPTS,NCURVS,CAPN,CAPL)` draws a waterfall chart of data held in the 2-dimensional array, `Z2ARR(NPTS,NCURVS)`, where `NCURVS` is the number of curves to be plotted. Each curve corresponding to the points `Z2ARR(1,n)` to `Z2ARR(NPTS,n)` is offset from the last by an equal step and masked by curves already drawn on the waterfall chart.

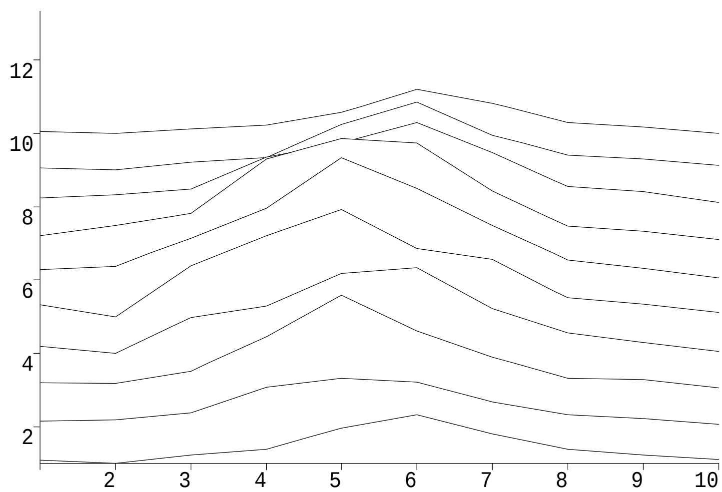


Figure 6.1 Waterfall chart

`NPTS` and `NCURVS` must both be positive integers; if they are not, a diagnostic is generated and nothing is drawn. The maximum number of points that can be plotted as a waterfall curve is 1024, the minimum number of points is 2.

`WFCHT` draws numeric and label axes which can be configured using appropriate `AX*` subroutines with `CHAXIS='NW'` and `'LW'` respectively.

`WFCHT` must always be preceded by the *new picture* subroutine `NEWPIC`.

6.1.2 Drawing individual waterfall curves

`WFDRAW(ZARR,NPTS)` draws a single curve of data contained in `ZARR(NPTS)`. Each curve corresponding to the points `ZARR(1)` to `ZARR(NPTS)` is offset from the last by an equal step, and masked by curves already drawn on the waterfall chart.

The value of `NPTS` in the first call of `WFDRAW` on a chart sets the value for the number points per curve on that chart; subsequent curves on the same chart are either short or incomplete if they have different values of `NPTS`. `NPTS` must be a positive integer; if it is not, a diagnostic is generated and nothing is drawn. The maximum number of points per waterfall curve is 1024.

`WFDRAW` draws a waterfall curve on the current 2-D picture therefore the first call to `WFDRAW` must be preceded by the *new picture* subroutine `NEWPIC`.

`WFNCVS(NCURVS)` specifies that the label scale is to accommodate `NCURVS` curves. If more than `NCURVS` curves are requested, an error message is generated and no further drawing is done. If the number of curves specified by `WFCHT` is less than `NCURVS`, there is a space at the top of the label scale.

`NCURVS` must be a positive integer; if not, the default is restored.

By default, a waterfall chart drawn by `WFCHT` is scaled to accommodate the number of curves in the supplied data; following a call to `WFNCVS`, the label scale is calculated to accommodate `NCURVS` curves for each waterfall chart.

6.2 Controlling the scales

A waterfall chart has a numeric scale for the independent variable, and a label scale over which a series of staggered curves is plotted. Waterfall scales are evaluated and set when the first waterfall curve is drawn on the current picture either as part of a complete chart drawn by a call of `WFCHT` or an individual curve drawn by `WFDRAW`.

`WFEQL`, `WFEQN`, `WFNSCL` and `WFZSCL` all affect the underlying scales, and should therefore be called before any waterfall curves are drawn on the current picture.

If an individual axis is to be drawn to display these scales, it should be drawn after the first curve has been drawn.

6.2.1 Controlling the numeric scale

The numeric scale represents the independent variable against which all the curves are plotted. In the current version, the numeric scale is always represented horizontally. `WFNSCL` specifies the range of the numeric scale, and `WFEQN` specifies the relationship between the data points and the numeric scale. (see Figure 6.2).

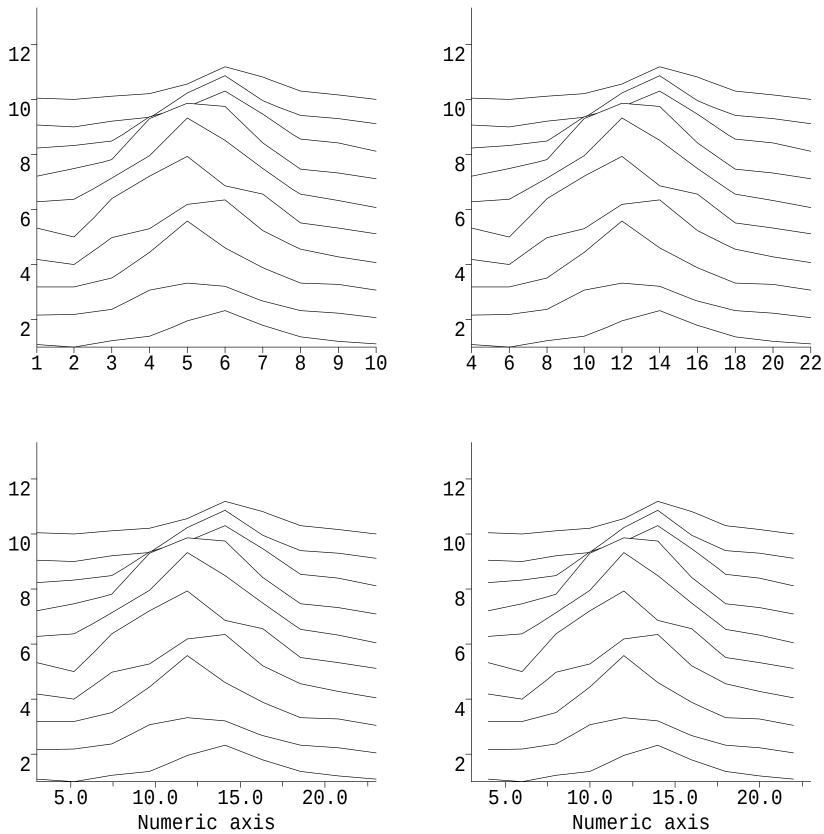


Figure 6.2 Controlling the numeric scale
Default; `CALL WFEQN(4.0,2.0))`; `CALL WFNSCL(2.0,23.0)`; `CALL WFEQN(4.0,2.0)` and `CALL WFNSCL(3.0,23.0)`

`WFNSCL(START,STOP)` specifies the plotted range of the numeric scale from `START` to `STOP`.

By default, the numeric scale covers the full range of the data points. Calling `WFNSCL` with `START`=`STOP` restores the default.

`WFEQN(ESTART,ESTEP)` specifies the relationship between the data points and the numeric scale; `ESTART` is the scale value at the first point of each curve, and `ESTEP` is the scale interval between the equally-spaced points in the curves. Each curve covers the range `ESTART` to `ESTART`+(`ESTEP`×(`NPTS`-1)), where `NPTS` is the number of points in a curve.

By default, when `WFNSCL` has specified a range for the scale, the first point of each curve is positioned at the start of the scale, and a scale interval is allocated to make the curves fill the scale; when there is no specified range, 1.0 is used for the first point and the interval. Calling `WFEQN` with `ESTEP`=0.0 restores the default.

6.2.2 Controlling the label scale

The label scale is a composite scale which represents both y and z – the curves are placed at equally-spaced y values, and each curve spans a range of z values. By default, the label scale is equivalent to a count of the equally-spaced curves, extended sufficiently to accommodate the z scale. `WFEQL` specifies an alternative label scale.

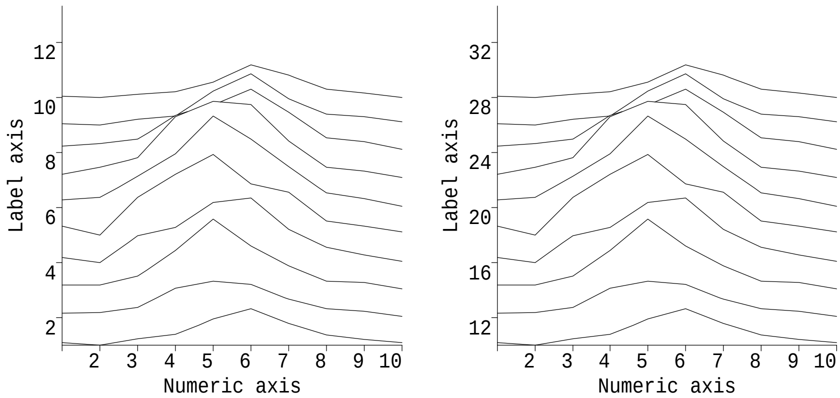


Figure 6.3 Controlling the label scale
Default and `CALL WFEQL(10.0,2.0)`

`WFEQL(CSTART,CSTEP)` specifies the relationship between the curves and the label scale; `CSTART` is the scale value at the first curve, and `CSTEP` is the scale interval between the equally-spaced curves.

Calling `WFEQL` with `CSTEP`=0.0 restores the default.

The label scale defined by `WFEQL` affects axis annotations, but has no effect on the curves (see Figure 6.3).

6.2.3 Controlling the z scale

Each waterfall curve spans a range of z values; this range is incorporated into the label scale (see section 6.2.2). By default, the limits of the z scale are set to the minimum and maximum of the data set supplied when the first curve is drawn. When a complete waterfall chart is drawn using `WFCHT`, the default z scale is set from the range of z values in all curves; but when individual curves are drawn using `WFDRAW`, the default z scale can only be set from the range of z values in the first curve alone, and this may not cover the full range of z values to be plotted. `WFZSCL` specifies a z scale to be used for all the curves on a waterfall chart.

`WFZSCL(ZSTART,ZSTOP)` specifies the range of z to be accommodated on each curve. If `ZSTART` > `ZSTOP`, the curves are inverted (see Figure 6.4).

Calling `WFZSCL` with `ZSTART`=`ZSTOP` restores the default.

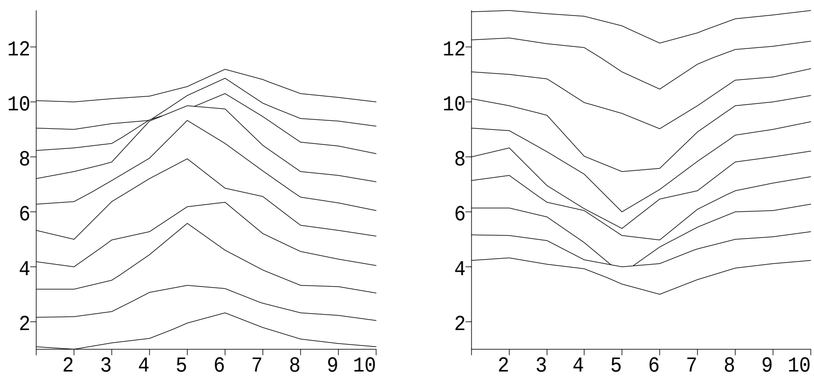


Figure 6.4 Controlling the z scale
Default and `CALL WFZSCL(ZMAX,ZMIN)`

6.2.4 Controlling the displacement between curves

Each waterfall curve is scaled to occupy an equal portion of the label scale, and the curves are displaced from each other by an equal step. By default, each curve is displaced from the next by 0.3 of the span of a curve. `WFSTEP` specifies an alternative displacement (see Figure 6.5).

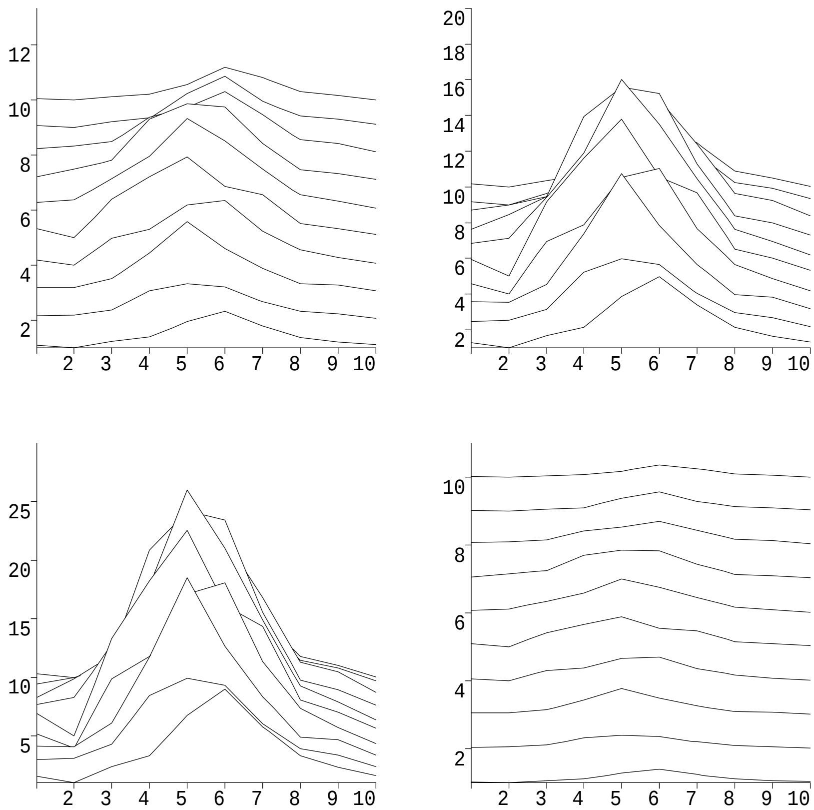


Figure 6.5 Controlling the displacement between curves
`CALL WFSTEP(0.3)` (default), `CALL WFSTEP(0.1)`, `CALL WFSTEP(0.05)`, `CALL WFSTEP(1.0)`

`WFSTEP(ZDISP)` specifies that curves are drawn at a displacement `ZDISP`$\times$ R from each other, where R is the range of a single curve. The label scale is then calculated to cover a range dependent on a displacement factor, `ZDISP`, and the number of curves being accommodated.

`CALL WFSTEP(-1.0)` restores the default, which is equivalent to `CALL WFSTEP(0.3)`.

6.3 Controlling pen usage

6.3.1 Waterfall pen pointers

Different parts of the curves are drawn using different pen pointers so that independent changes to the visibility or appearance of the parts can be achieved through pointer adjustments:

Pointer	*Pen usage*
[1]	Unmasked part of curve with $z \geq$ `ZLEVEL`
[2]	Masked part of curve with $z \geq$ `ZLEVEL`
[3]	Unmasked part of curve with $z <$ `ZLEVEL`
[4]	Masked part of curve with $z <$ `ZLEVEL`

`WFPNS(IPEN1,IPEN2,IPEN3,IPEN4)` specifies the pens to be associated with the four pen pointers. Parts of the plotting are omitted if the relevant pen pointer is set to −1. Rubbing out may be performed by setting the relevant pen pointers to zero, the background colour.

`CALL WFPNS(1,-1,1,-1)` restores the default.

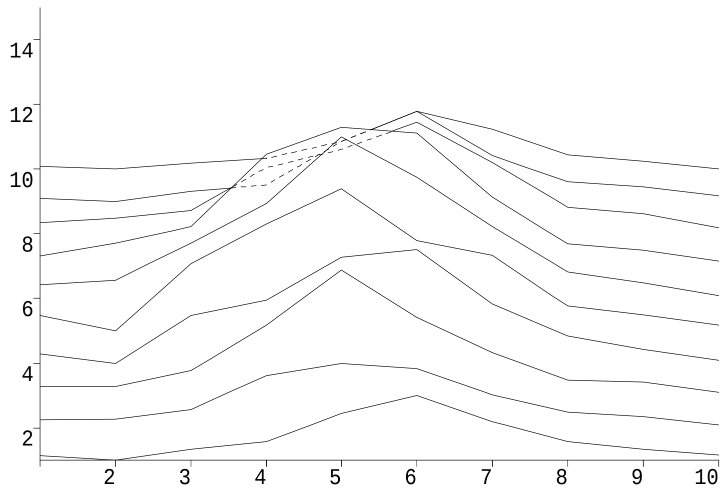


Figure 6.6 Unmasking waterfall curves
`CALL WFPNS(1,2,1,2)`

Pointers set by `WFPNS` are used for waterfall charts independently of pointers set by `SETPNS`. This allows different defaults, and simultaneous setting of one set of pointers for the waterfall charts using `WFPNS`, and another set for axes using `SETPNS`.

6.3.2 Changing the pen threshold

Pen pointers can set pens for different parts of the curves. By default, a switch of pens is made at z=0; `WFZLEV` specifies an alternative value, `ZLEVEL`, for switching pens.

`WFZLEV(ZLEVEL)` sets to `ZLEVEL` the z threshold at which pens should be changed.

`CALL WFZLEV(0.0)` restores the default.

For further details about controlling pen usage, see section 7.5.2.

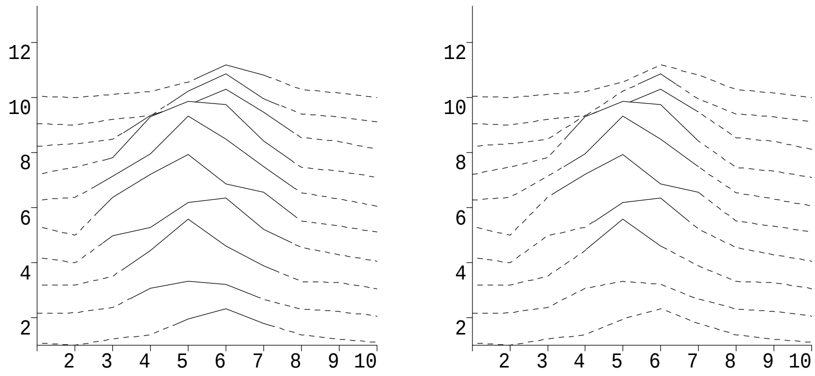


Figure 6.7 Changing the pen threshold
`CALL WFZLEV(0.0)` (default) and `CALL WFZLEV(10.0)` (both with `CALL WFPNS(1,-1,2,-1)`)

6.4 Additional facilities

6.4.1 Inquiry and conversion

`KWZVAL(ICURVE,ZVAL,VALUE)` delivers a value for z = `ZVAL` on curve `ICURVE` in terms of the label scale.

`QWZSCL(ZSTART,ZSTOP)` delivers the z limits for each curve; by default, this corresponds to the highest and lowest values in `Z2ARR`.

`QWZSCL` and `KWZVAL` must be called after scales have been set by `WFCHT`, `WFDRAW` or `WFZSCL`.

6.4.2 Resetting all defaults

`WFINIT` resets all waterfall characteristics to their default values; this does not include the characteristics of waterfall axes.

7. Additional 3-D facilities

This chapter describes additional 3-D facilities:

7.1 Interpolation – `*CALC`

7.2 Cross section – to draw a section through 3-D data along the straight path between two specified x-y points

7.3 Sequences of contours

7.4 Labelling and keys on surface pictures and contour maps

- Data range and contour interval
- Keys for shaded contour maps

7.5 Storing and retrieving curve coordinates

- Labelling contour curves
- Controlling the label position

7.6 Colour and pen control

- Bundles
- Controlling pen usage
- Pens for shaded areas

7.1 Interpolation

Interpolation]

SIMPLEPLOT can interpolate the surface height at a specified point on a 3-D surface. There is a separate `REAL` function for each type of data (see Chapter 3).

Name	*Data Type*
`RGCALC`	Regular grid
`XCALC`	x-specified tartan
`YCALC`	y-specified tartan
`XYCALC`	xy-specified tartan
`ZZCALC`	Ungridded

The choice of which function to use depends entirely on the type of data available.

The grid coordinates should be specified using `SFEQX` and/or `SFEQY` before interpolation is attempted with any equally-spaced coordinates.

Interpolation is performed with polar data in the same way as with Cartesian data. After `COORDS` or `EQSCAL` has been called to switch coordinate interpretation to one of the polar conventions, the arguments, (x, y) are interpreted as (r, θ).

7.2 Cross section

Cross section]

Simpleplot can draw the curve showing a cross section through a 3-D surface, along the straight path between two specified points. There is a separate subroutine for each type of data (see Chapter 3).

Name	*Data Type*
`RGCUT`	Regular grid
`XCUT`	x-specified tartan
`YCUT`	y-specified tartan
`XYCUT`	xy-specified tartan
`ZCUT`	Ungridded
`FNCUT`	User-defined

The choice of which subroutine to use depends entirely on the type of data available.

Although it is derived from 3-D data, a cross section is a 2-dimensional curve drawn on a standard framework. When drawing cross sections from equally-spaced data, `SFEQX` and/or `SFEQY` must be called before `*CUT` to ensure the correct interpretation of the data grid.

Cross sections pick up the attributes of contour plotting – the line style specified by `CTBRKN` and the curve drawing algorithm specified by `CTCURV`.

7.3 Sequences of contours

Sequences of contours]

By default, plotting which consists of a sequence of objects resulting from a single subroutine call, uses attributes in a defined order. A family of SeQuence subroutines `SQ*`, allows user control of the order. In 3-D plotting, sequences are particularly applicable to sets of contour lines, or ranges on contour maps and surface pictures. The maximum number of items which may be specified as a sequence is 32.

`SQBRKN(IARR,NARR)` specifies a sequence of `NARR` broken line patterns in `INTEGER` array `IARR`. Patterns are specified in the range $-6 \ldots 6$ (see Figure D.1). The sequence of broken lines set by `SQBRKN` is used by `*CNTS` for contour curves. By default, the same line pattern is used for all contour curves, and is solid unless specified by `CTBRKN`.

When the sequence is exhausted, the default line pattern is used.

`CALL SQBRKN(IARR,0)` restores the default.

`SQPEN(IARR,NARR)` specifies a sequence of `NARR` pens to be used in `INTEGER` array `IARR`. Pens values can be -1, 0, 1, 2, 3, ... *etc.* The sequence of pens set by `SQPEN` is used by `*CNTS` for contour curves.

When the sequence is exhausted, the pen pointed to by pen pointer 1 is used.

`CALL SQPEN(IARR,0)` restores the default.

`SQSHAD(IARR,NARR)` specifies a sequence of `NARR` shading patterns in `INTEGER` array `IARR`. Shading patterns can be -1, 0, 1, 2, 3, ... *etc.* (see Figure D.2). The sequence of shading patterns set by `SQSHAD` is used by `*SHDS` for contour maps. By default, shading patterns are used in numerical order, 1, 2, 3, *etc.*

When the sequence is exhausted, the default shading sequence is resumed.

`CALL SQSHAD(IARR,0)` restores the default.

Details of the SIMPLEPLOT software shading patterns are found in Appendix D.

`SQZLAB(LABARR,NARR)` specifies a sequence of `NARR` contour labels to be used in array `LABARR`. Labels may be up to 20 characters long. The sequence of labels set by `SQZLAB` is used by `*CNTS` and `*SHDS` when labelling has been enabled.

When the sequence is exhausted, labelling stops.

`CALL SQZLAB(LABARR,0)` restores the default.

`SQZVAL(DARR,NARR)` specifies a sequence of `NARR` contour levels to be used in `REAL` array `DARR`. Contour levels must be monotonic, *ie.* all different and in ascending or descending order. The sequence of levels set by `SQZVAL` is used by `*CNTS` and `*SHDS` .

`CALL SQZVAL(DARR,0)` restores the default.

7.4 Labelling and keys

Labelling and keys]

In addition to the standard facilities for adding titles, keys and caption areas to a picture or to the SIMPLEPLOT page, there are some labelling facilities which are specific to 3-D plotting:

7.4.1 Data range and contour interval

`SFLAB` adds a label to the current picture after a surface picture or contour map has been plotted. The label refers explicitly to the data already plotted, and therefore can only be written after some drawing has been done. `SFLAB` only has effect when called after `*SURF`, `*CNTS` or `*SHDS`.

`SFLAB` adds a label under the picture; the label indicates the range of data values, and when equally-spaced contours are shown, the contour interval.

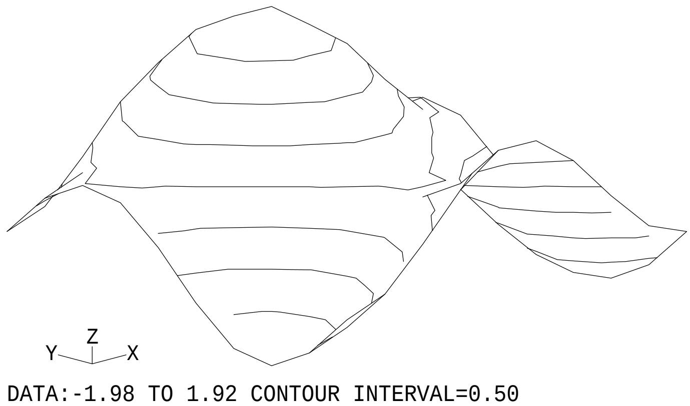


Figure 7.1 Labelling with `SFLAB`

Figure 7.1 illustrates the effect of calling `SFLAB` after a `*SURF` subroutine with contours.

The values written by `SFLAB` are inquired by `QSFLAB`. The values returned by `QSFLAB` are the minimum and maximum z values, and the contour interval. Figure 7.2 shows the values returned by `QSFLAB` written in titles on a surface picture and a contour map; when equally-spaced contours are not included in the picture, a zero contour interval is returned.

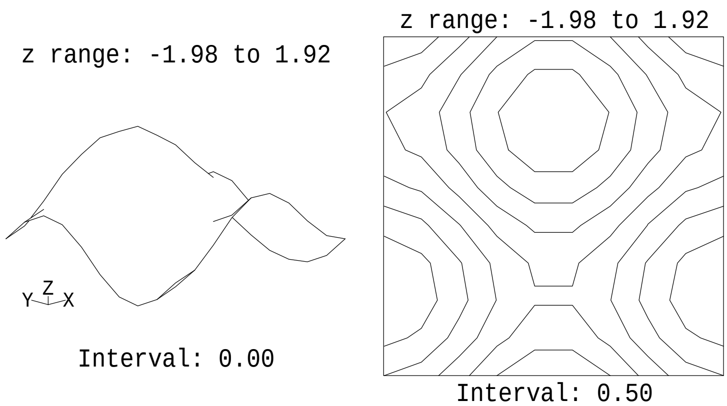


Figure 7.2 Alternative labelling with `QSFLAB`

Table 7.1 Descriptive positions of titles, keys and captions

VCHAR	*Vertical position*	HCHAR	*Horizontal position*
`'N'`	North (highest possible)		
`'H'`	Higher than group	`'W'`	West (far left)
`'O'`	Over picture/group	`'P'`	Preceding
`'T'`	Top of picture/group	`'L'`	to the Left
`'C'`	Centre	`'C'`	Centre
`'B'`	Bottom of picture/group	`'R'`	to the Right
`'U'`	Under picture/group	`'F'`	Following
`'L'`	Lower than group	`'E'`	East (far right)
`'S'`	South (lowest possible)		

7.4.2 Keys for shaded contour maps

`MPK7H` and `MPK7V` draw complete keys to shaded contour maps, automatically relating the shading patterns to the z scales in the same way as the `*SHDS` subroutines. `MPK7H` draws a horizontal key, and `MPK7V` draws a vertical key. (see Figure 7.3).

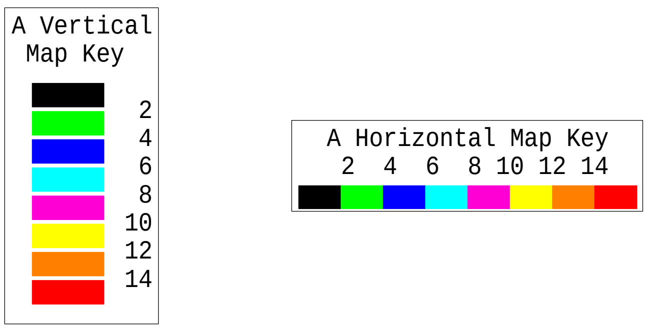


Figure 7.3 Vertical and horizontal map keys for the range 0.0–15.0

`MPK7H(VCHAR,HCHAR,ZLEFT,ZRIGHT,CAP)` draws a complete horizontal key to a shaded contour map of data ranging from `ZLEFT` to `ZRIGHT`. Samples of the patterns are drawn side-by-side, with the contour labels above.

`MPK7V(VCHAR,HCHAR,ZTOP,ZBOTTM,CAP)` draws a complete vertical key to a shaded contour map of data ranging from `ZTOP` to `ZBOTTM`. Samples of the patterns are drawn in a vertical list, with the contour labels alongside.

A map key is positioned on the SIMPLEPLOT page in the same way as other keys; `VCHAR` and `HCHAR` are single characters representing the initial letters of vertical and horizontal positions (see Table 7.1). `CAP` provides a heading which is centred at the top of the key; a box is drawn around the key unless cancelled by a prior call of `BOXKY`. The key is associated only with the current picture and the area is masked against overdrawing; this is equivalent to `ITYPE`=2 with `DEFKEY`. To prevent the picture obscuring any of the key, `MPK7H` and `MPK7V` should be called after the new picture has been started but before the contour map is drawn.

In order to relate keys to the shaded contour maps drawn by SIMPLEPLOT both `MPK7H` and `MPK7V` interact with the specification subroutines which control the `*SHDS` subroutines:

- `MPTYPE` specifies whether ordinary shading (default) or fast area-fill is to be used.
- `SFEQZ` specifies the equally-spaced z values where patterns change.
- `SFEQZD` specifies the equally-spaced z values centred on pattern spans.

- SFLIMS imposes limits on the range plotted.
- SFZSCL defines the z scale on the picture.
- SQSHAD replaces the default sequence of shading patterns by a user-defined sequence.
- SQZLAB replaces the default numerical contour labels by a set of user-defined text labels.
- SQZVAL overrides the normal strategy of contouring at equal intervals, by specifying a sequence of z values for contours.

These are described in greater detail below:

Scales and data ranges: Arguments ZLEFT and ZRIGHT of MPK7H, or ZTOP and ZBOTTM of MPK7V, specify the data range for the key. If ZLEFT=ZRIGHT or ZTOP=ZBOTTM, the scale specified by SFZSCL is picked up; but if none has been set, nothing is drawn and MPK7H/MPK7V issue the following diagnostic:

```
(MAP KEY OMITTED: CONSTANT DATA)
```

The order of key entries is reversed by interchanging ZLEFT with ZRIGHT, or ZTOP with ZBOTTM.

Following calls of SFLIMS and/or SFZSCL, the data range represented in the key is the intersection of the specified data range with those specified by SFLIMS and SFZSCL. If the intersection is null, nothing is drawn, and MPK7H/MPK7V issue the following diagnostic:

```
(MAP KEY OMITTED: NULL RANGE)
```

Each shaded contour covers a data range which includes the lower limit, and excludes the upper limit. When the maximum data value coincides with a contour level, the shading pattern for the range rising from that level can only occur in the map to represent a plateau at the top of the data range. Because of this possibility, the shading pattern is included in keys. The extra level is avoided by specifying a slightly reduced data range for the key.

Relating data values to shading patterns: When increasing contour levels are used (either specified by a prior call of SFEQZ, SFEQZD, SQZVAL or by default), the lowest value in the z scale is represented by the first shading pattern (either pattern 1 or the first pattern in the sequence specified by SQSHAD). When decreasing contour levels have been specified by SFEQZ, SFEQZD or SQZVAL, the highest value in the z scale is represented by the first shading pattern. By default, the z scale coincides with the data range, but SFZSCL defines it independently.

Map key samples and labels: If SFEQZD has been called to specify the contour levels, key labels are centred on samples; all other methods of controlling contour levels result in key labels between samples. Map keys are not affected by DEFKYW.

Map key captions: By default, the caption for a map key, CAP, is centred over the key; on vertical map keys, numerical key labels are right-justified and textual key labels are left-justified. If the caption is ‘null’ (*ie.* contains only space characters), the space for caption is omitted.

7.5 Storing and retrieving curve coordinates

The (x, y) coordinates of some internally generated curves can be retained by SIMPLEPLOT. Curves whose coordinates can be remembered include:

- Contour curves drawn with `*CONT` on 2-D pictures (*ie.* contours drawn on surface pictures cannot be retrieved),
- Cross-sectional curves drawn by `*CUT`,
- Element boundaries drawn by `ZZELMS`,
- Ungridded data boundaries drawn by `ZZEDGE`.

Coordinates are retained following a request by `CTHOLD` , and can subsequently be retrieved by `QCURVE`.

`CTHOLD(ICODE)` specifies whether coordinates of internally generated curves are to be saved and/or plotted according to the value of `ICODE`:

`ICODE`	*Graphics output*	*Coordinates of contour*
1	Yes	Forgotten (default)
2	No	Remembered
3	Yes	Remembered

`QCURVE(MAXPTS,XARR,YARR,NPTS)` inquires the coordinates of a previously remembered curve. Each call of `QCURVE` can receive coordinates of between 0 and `MAXPTS` points in parallel arrays, `XARR` and `YARR`. The remembered coordinates can constitute any number of separate curve segments which can be extracted by repeated calls of `QCURVE`. Curves containing more than `MAXPTS` points also require multiple calls of `QCURVE`. For each call of `QCURVE`, `NPTS` is set to the number of points received, and should be interpreted as follows:

`NPTS`	*Interpretation*
`NPTS`=`MAXPTS`	The curve may be incomplete either because there are more points in this curve segment or because there is another curve segment; continuation of the current segment is signified by the first point received by next call of `QCURVE` duplicating the value of last point of the previous segment, `XARR(MAXPTS)`, `YARR(MAXPTS)`.
0 <`NPTS`<`MAXPTS`	(`XARR(NPTS)`,`YARR(NPTS)`) is the final point of a curve.
`NPTS`=0	No more coordinates to be retrieved – the end of the process.

If there are no stored coordinates to retrieve, the following diagnostic is issued:

`(Curve storage empty)`[1]

7.5.1 Labelling contour curves

Labelling contour curves]

The contour curves returned by `QCURVE` can be drawn and labelled individually using `LABCV7`.

`LABCV7(XARR,YARR,NARR,LTYPE,CAP)` draws a labelled curve using broken line pattern, `LTYPE`, corresponding to the following points held in parallel arrays, `XARR` and `YARR`:

(`XARR(1)`, `YARR(1)`), (`XARR(2)`, `YARR(2)`), ...
(`XARR(NARR)`, `YARR(NARR)`)

The curve is constructed from straight lines from point-to-point. A closed curve is obtained by using data in which `XARR(NARR)`=`XARR(1)` and `YARR(NARR)`=`YARR(1)`.

The label, `CAP`, is drawn along the curve, with the base of each letter parallel to the curve; if the label is too long, it is truncated.

The label is drawn in the direction of the curve, that is, from (`XARR(1)`, `YARR(1)`) towards (`XARR(NARR)`,`YARR(NARR)`). This may result in an upside-down label: to reverse the direction, the coordinates of the curve must be reversed.

The following diagnostics may be issued:

```
(Data curve exceeds scales)2
(Insufficient number of valid points)1
(Requested point unavailable)1
(Text truncated on curve label)2
```

7.5.2 Controlling the label position

The position of the label on labelled curves is controlled by specifying a reference point along the curve, and the justification of the label relative to the curve and the reference point. By default, labels drawn on curves by `LABCV7` are vertically centred on the curve and left-justified to the midpoint of the curve, (`XARR(NARR/2)`,`YARR(NARR/2)`). `CVLBJS` specifies an alternative justification of the label and `CVLBPS` specifies an alternative position of the label. Labels can be drawn relative to the highest or lowest point on the curve by setting `IPOS` to a value evaluated using `LIMIDX` which finds the array index of the maximum and minimum of data in an array.

`CVLBJS(VCHAR,HCHAR)` specifies the justification of the label drawn on a labelled curve according to the values of `VCHAR` and `HCHAR`:

`VCHAR`	*Vertical justification*
`'D'`	Default
`'B'`	Bottom of the letters level with the curve
`'C'`	Centre of the letters level with the curve (default)
`'T'`	Top of the letters level with the curve

`HCHAR`	*Horizontal justification*
`'D'`	Default
`'C'`	Label is centred on (`XARR(IPOS)`,`YARR(IPOS)`)
`'L'`	Label starts at (`XARR(IPOS)`,`YARR(IPOS)`) (default)
`'R'`	Label ends at (`XARR(IPOS)`,`YARR(IPOS)`)

`CVLBPS(IPOS)` specifies the position of the label drawn on a labelled curve. Labels are drawn relative to the point (`XARR(IPOS)`,`YARR(IPOS)`), where `XARR` and `YARR` are the parallel arrays of coordinates which are passed as arguments to `LABCV7`.

If there are no valid points between `IPOS` and the end of the curve, a diagnostic is issued and the label is omitted.

If `IPOS` is greater than `NARR` (the number of points in `XARR` and `YARR`), the default is used for the curve currently being drawn. If `IPOS` is less than 1, the default is restored for all subsequent curves.

`LIMIDX(DARR,NARR,IMIN,IMAX)` returns the positions of the minimum and maximum values of a `REAL` array. That is to say, `DARR(IMIN)` is the lowest value in the array, and `DARR(IMAX)` is the highest value.

If `DARR` contains more than one value equal to the minimum or maximum, `IMIN` and `IMAX` refer to the first occurrence.

`LIMIDX` ignores any values which are equal to the current no-data value.

When `XARR(IPOS)` or `YARR(IPOS)` is equal to the current no-data value, the label is justified against the next valid point; with `'L'`eft or `'C'`entre horizontal justification, the next point is `IPOS`+1; with `'R'`ight justification, the next point is `IPOS`-1.

7.6 Colour and pen control

Colour and pen control]

SIMPLEPLOT controls the use of colour with pens which may correspond to an actual physical plotting pen or to one of a series of available colours. Pens are referred to by values −1, 0, 1, 2, 3, . . . *etc.* as described in Table 7.2.

Table 7.2 Pen identification

Pen value	*Interpretation*
−1	Plotting is omitted on all devices
0	Plotting is done in background colour; on some devices this produces the effect of rubbing out but on others (*eg.* pen plotters) plotting is omitted
1, 2, 3. . .	Pens/bundles are selected as determined by the device driver

7.6.1 Bundles

Bundles]

By default, colour is the only line-drawing attribute associated with pens and, therefore, on monochrome devices different pens are not distinguished at all. `BUNLPR` specifies the attributes which are to be combined to distinguish different pens when they are used for line drawing even on monochrome devices. This facility is useful when demarcation is required, and different details are acceptable on devices with different capabilities.

`BUNLPR(CHATTR)` specifies the attributes which are to be used and their order of significance. `CHATTR` can consist of the initials of one, two or three of the following attributes in the required order:

`CHATTR`	*Attribute*	*Cyclic order*
`'C'`	Colour	1, 2, 3, . . . `NPENS`
`'S'`	Style	0, −1, −2. . . −6, 1, 2 . . . 6
`'T'`	Thickness	1, 2, 3, 4

If a selected pen has a number which exceeds the number of attribute combinations in the current bundle, the effect is undefined.

If linestyle is a bundled attribute when it is directly selected by the `LTYPE` argument of `BRKN*`, or `*BRKN`, `LTYPE` overrides the linestyle of the bundle, but the other bundled attributes remain as they would be for the current pen. To select the bundled linestyle, use `LTYPE` = −999.

7.6.2 Controlling pen usage

Pens can be selected individually or *via* four pen pointers. These pen pointers are identified with different types of plotting operations, and different parts of composite operations. The pointers identified with specific plotting operations are indicated by numbers in square brackets within the subroutine specifications in the *SIMPLEPLOT Reference manual* and as described below. The following subroutines control the use of pens:

`BUNLPR(CHATTR)` specifies the priority of bundled line-drawing attributes.

`PEN(IPEN)` specifies the pen to be used for all subsequent drawing.

`SETPNS(IPEN1,IPEN2,IPEN3,IPEN4)` specifies the pens associated with the four pen pointers [1], [2], [3] and [4] for all plotting except waterfall curves.

`WFPNS(IPEN1,IPEN2,IPEN3,IPEN4)` specifies the pens associated with the four pen pointers [1], [2], [3] and [4] on waterfall curves.

By default, the four pen pointers are each associated with pen number 1. `SETPNS` affects all subsequent plotting (except that of waterfall curves) until `SETPNS` or `PEN` is called. `CALL PEN(1)` restores the default.

Pen usage on surfaces

Surfaces are drawn using the four pen pointers as described in Table 7.3. Different pens may be selected to draw the top of the surface and its underside, so that the front edge can be recognized by the change of pen across the picture. When only one side of the surface is visible, the pen reserved for the other side is used along its front edge alone. Figure 7.4 illustrates how the four pen pointers affect a crosshatched surface.

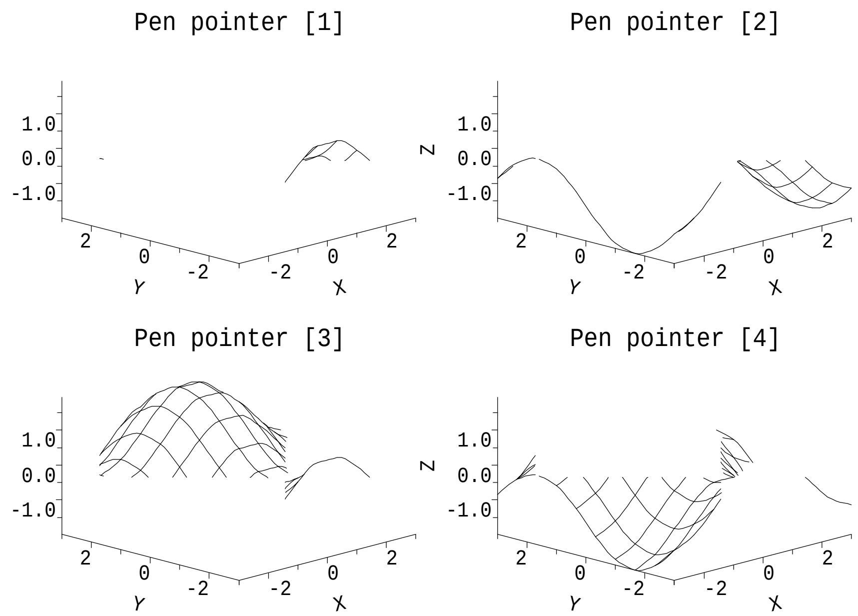


Figure 7.4 Pen usage on crosshatched surfaces

Zero is included with both 'negative' and 'positive'; a contour line for $z = 0.0$ on a surface is drawn

Table 7.3 Pen pointers controlled by `SETPNS`

Pen pointer	*General plotting*	*Surface pictures*	*Sets of contour curves*
[1]	All general line plotting, axes	Positive data on the underside	Positive levels (except extremes)
[2]	All text	Negative data on the underside	Highest level (if more than one level)
[3]	Symbols, major grid lines	Positive data on the top side	Lowest level
[4]	Minor grid lines	Negative data on the top side	Zero and −ve levels (except extremes)

twice, first with the pen for positive values and then in the pen for negative values.

Pen usage for sets of contour curves

When a single contour curve is drawn using a *CONT subroutine, the pen is selected by the first pen pointer, [1]. Subroutines which draw a sequence of contour lines (*CNTS) use the four pen pointers as described in Table 7.3 (page 73); Figure 7.5 illustrates how SETPNS affects line contour maps.

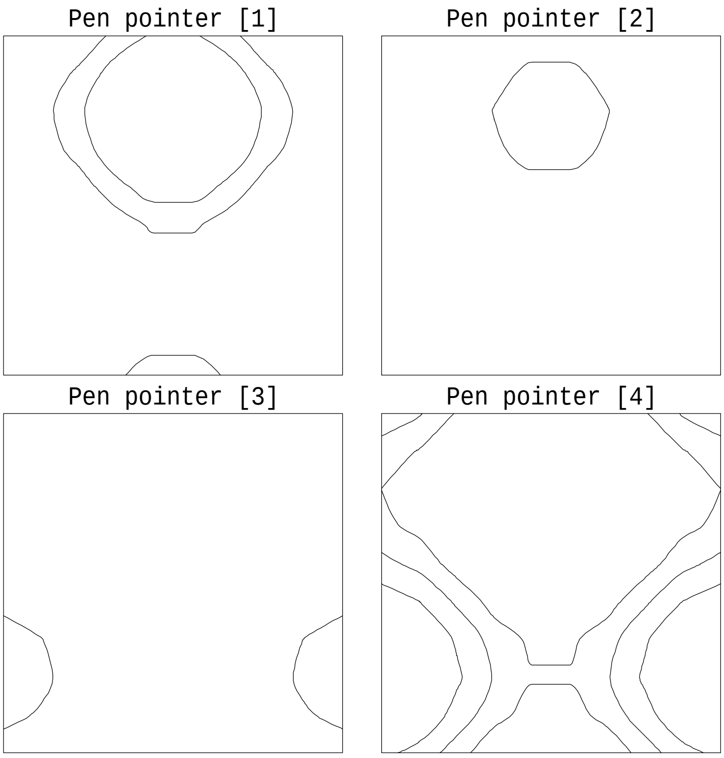


Figure 7.5 Pen usage on line contour maps

Although Figure 7.5 illustrates the use of pen pointers on a 2-D contour map, similar rules apply when a set of contours is added to a surface picture except the contour for $z = 0.0$ is drawn twice (see above).

Pen usage on waterfall curves

Waterfall curves are drawn using the four pen pointers as defined by WFPNS and WFZLEV:

Pointer	*Pen usage*
[1]	Unmasked part of curve with $z \geq$ ZLEVEL
[2]	Masked part of curve with $z \geq$ ZLEVEL
[3]	Unmasked part of curve with $z <$ ZLEVEL
[4]	Masked part of curve with $z <$ ZLEVEL

where ZLEVEL is the threshold specified by WFZLEV. Pointers set by WFPNS are used for waterfall curves independently of pointers set by SETPNS, which may be different. Figure 7.6 illustrates how WFPNS affects line contour maps.

7.6.3 Pens for shaded areas

The details of shading patterns used for 3-D plotting can be controlled using general facilities for shading control. The following subroutines are particularly relevant to contour maps:

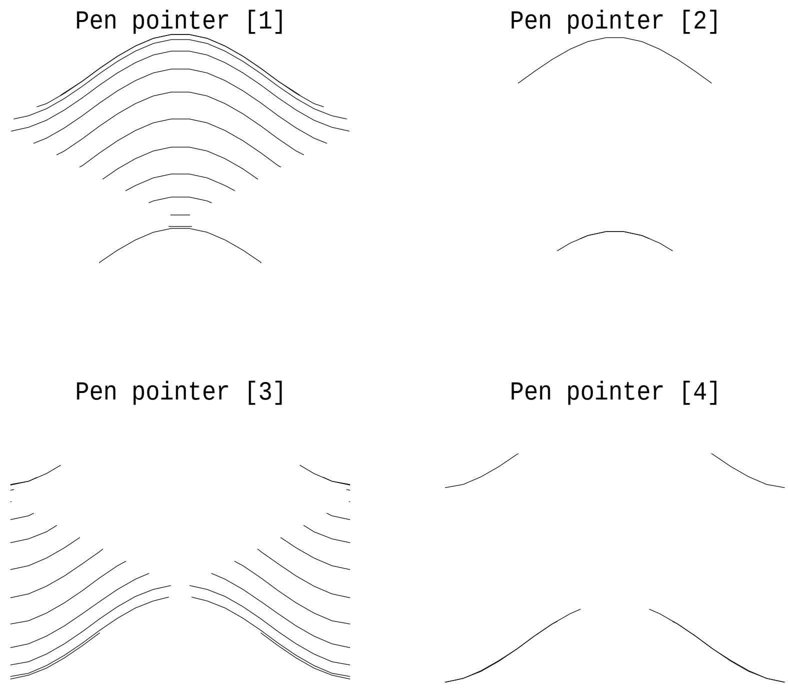


Figure 7.6 Pen usage on waterfall curves

`SHEDGE(ITYPE)` specifies whether to draw a boundary around a shaded area.

By default, SIMPLEPLOT draws shaded areas on a contour map without a boundary. `SHEDGE` can override the default (see Figure 7.7).

`ITYPE`	*Shading edge*
−1	Default mode, determined by SIMPLEPLOT
0	No edge drawn
1	Edge drawn

Drawn edges are not always noticeable. When a drawn boundary of shaded areas is requested by `SHEDGE`, the same pen is used for boundary and shading pattern; when an empty area is plotted, (pattern −1), it is outlined using the pen selected by `SHPEN` or (if none has been selected) the pen currently selected for drawing lines, [1].

`SHPEN(IPEN)` specifies single colour shading patterns with the specified pen.

`IPEN`	*Interpretation*
−1	Plotting is omitted on all devices
0	Plotting is done in background colour; on some devices this produces the effect of rubbing out but on others (*eg.* pen plotters) plotting is omitted
1, 2, 3...	Pens/colours are selected as determined by the device driver

`SHPEN` can be called before drawing a shaded contour map, to get shading patterns of one colour, with the intensity gradually decreasing from solid colour for the first level.

Further information about shading can be found in the *SIMPLEPLOT Primer*.

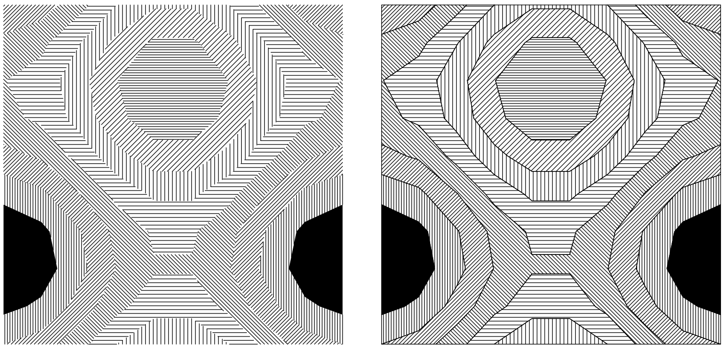


Figure 7.7 Boundaries for shaded contour regions
Default: without edges; `CALL SHEDGE(1)`: with edges

8. Cookbook

This chapter consists entirely of example programs and explanatory notes. The notes accompanying each example explain those subroutines which have not occurred in any earlier examples, or which are being used in a new way.

8.1 Contour Plotting
- Shaded contour map
- Line contours
- Individual contours
- Contour maps of tartan-gridded data
- Contour maps of a user-defined function

8.2 Surface Picture
- A simple surface picture
- Isometric drawing
- Surface pictures of tartan-gridded data
- Surface pictures of a user-defined function
- Surface pictures of ungridded data

8.3 Waterfall Chart
- A simple waterfall chart
- Drawing individual waterfall curves
- Distinguishing between data levels on a waterfall chart
- Conversion on waterfall charts

8.4 Advanced Examples
- Scales on contour maps and map keys
- Contour maps of ungridded data
- Extra labelling on contour maps
- Sequences on contour maps
- Labelling contour curves
- Polar contour map
- Waterfall charts and cascade surface pictures
- Shaded contour maps and surfaces
- Drawing a cross section

8.1 Contour plotting

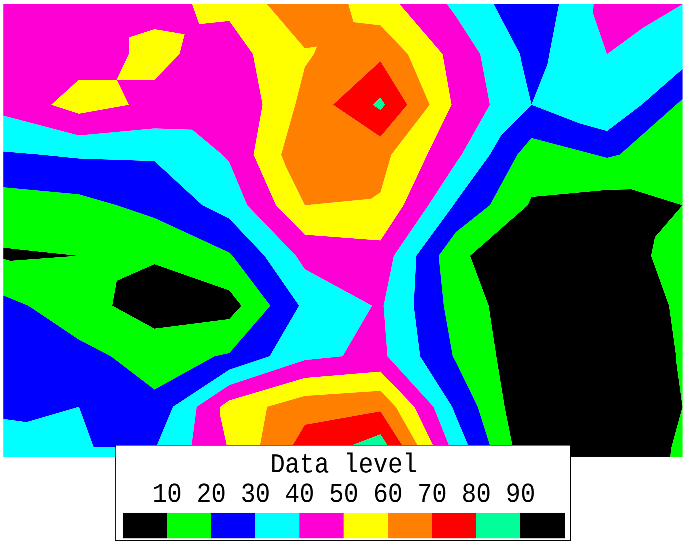
Contour plotting]

8.1.1 Shaded contour map

Shaded contour map]

This example illustrates how to:

- draw a key for a shaded contour map,
- plot a 2-D array of regular-gridded data as a shaded contour map.




```
      PROGRAM CONT1
      PARAMETER(NX=10,NY=10)
      REAL Z2ARR(NX,NY)
C Read data
      OPEN(UNIT=10,FILE='cont1.dat',STATUS='OLD')
      READ(10,*)Z2ARR
      CLOSE(10)
C Start picture, find data limits and draw key of data within limits
      CALL NEWPIC
      CALL LIMEXC(Z2ARR,NX*NY,ZMIN,ZMAX)
      CALL MPK7H('U','C',ZMIN,ZMAX,'Data level')
C Draw shaded contours
      CALL RGSHDS(Z2ARR,NX,NY)
C terminate plotting
      CALL ENDPLT
      END
```

Example 1. Shaded contours

Explanation of subroutines

When the first SIMPLEPLOT subroutine is called, the first diagnostic message is produced; for example:

`(SIMPLEPLOT Mark 2-14(001)F)`

`NEWPIC` starts a new 2-D picture without drawing any axis framework.

`LIMEXC(Z2ARR,NX*NY,ZMIN,ZMAX)` returns the minimum and maximum values in a `REAL` array, ignoring any no-data values.

`MPK7H(VCHAR,HCHAR,ZLEFT,ZRIGHT,CAP)` draws a complete key to the shading patterns used to cover the range `ZLEFT` to `ZRIGHT` with the current contour interval. Subsequent drawing is masked against the key.

`RGSHDS(Z2ARR,NX,NY)` draws a shaded contour map on the current 2-D picture.

`ENDPLT` must be called to terminate plotting. It empties any plotting buffers, closes the plotting device, and triggers the output of any outstanding diagnostic messages; for example:

```
(END OF PICTURE)
```

and then outputs the following messages:

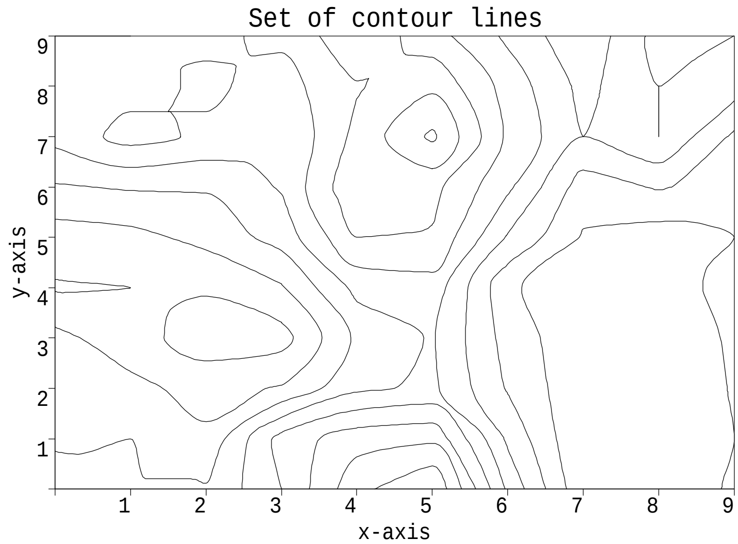
```
(DEVICE CLOSED)
(SIMPLEPLOT CLOSED)
```

8.1.2 Line contours

Line contours]

This example illustrates how to:

- plot a 2-D array of regular-gridded data as a set of line contours.




```
      PROGRAM CONT2
      PARAMETER(NX=10,NY=10)
      REAL Z2ARR(NX,NY)
C read data
      OPEN(UNIT=10,FILE='cont1.dat',STATUS='OLD')
      READ(10,*)Z2ARR
      CLOSE(UNIT=10)
C specify scales to relate to data
      CALL SCALES(0.0,REAL(NX-1),1,0.0,REAL(NY-1),1)
C start new 2-D picture and draw axes
      CALL AXES7('x-axis', 'y-axis')
C draw contours with perimeter and title
      CALL RGCNTS(Z2ARR,NX,NY)
      CALL PERIM
      CALL TITLE7('O','C','Set of contour lines')
C terminate plotting
      CALL ENDPLT
      END
```

Example 2. Line contours

Explanation of subroutines

`SCALES(XSTART,XSTOP,IXTYPE,YSTART,YSTOP,IYTYPE)` defines scales for subsequent Cartesian plotting. Both the x scale and the y scale are described by three arguments – the value at the start of the scale, the value at the end of the scale and a code number indicating the type of scale (1=linear).

`AXES7(CAPX,CAPY)` starts a new 2-D picture and draws an axis framework.

`RGCNTS(Z2ARR,NX,NY)` draws a complete set of contour lines representing the z heights supplied in the `NX`×`NY` array, `Z2ARR`.

`PERIM` draws a box around the current picture.

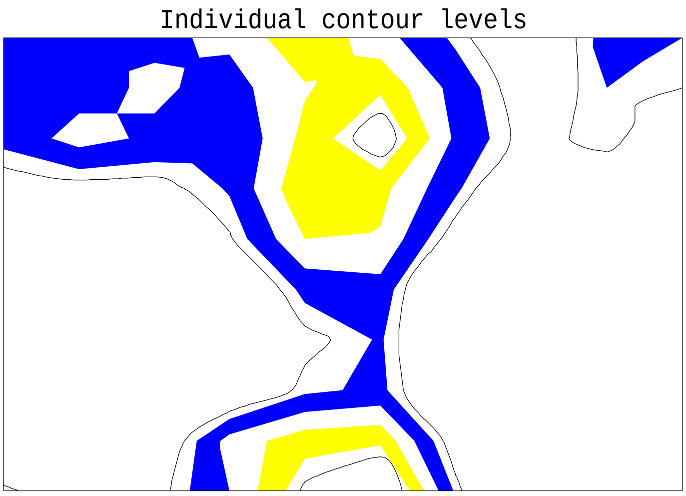
`TITLE7(VCHAR,HCHAR,CAP)` draws the text string, `CAP`, `'O'`ver the picture, and horizontally `'C'`entred, as specified by `VCHAR` and `HCHAR`.

8.1.3 Individual contours

Individual contours]

Similar subroutines to *CNTS and *SHDS are available for plotting individual contour levels. This example illustrates how to:

- draw individual contour lines,
- shade an individual contour region.




```
      PROGRAM CONT3
      PARAMETER(NX=10,NY=10)
      REAL Z2ARR(NX,NY)
C read data
      OPEN(UNIT=10,FILE='cont1.dat',STATUS='OLD')
      READ(10,*)Z2ARR
      CLOSE(UNIT=10)
C start 2-D picture and draw single contours
      CALL NEWPIC
      CALL RGCONT(35.0,Z2ARR,NX,NY)
      CALL RGCONT(75.0,Z2ARR,NX,NY)
C shadev contour bands
      CALL RGSHAD(40.0,50.0,4,Z2ARR,NX,NY)
      CALL RGSHAD(60.0,70.0,7,Z2ARR,NX,NY)
C add perimeter and title and terminate plotting
      CALL PERIM
      CALL TITLE7('O','C','Individual contour levels')
C terminate plotting
      CALL ENDPLT
      END
```

Example 3. Individual contour levels

Explanation of subroutines

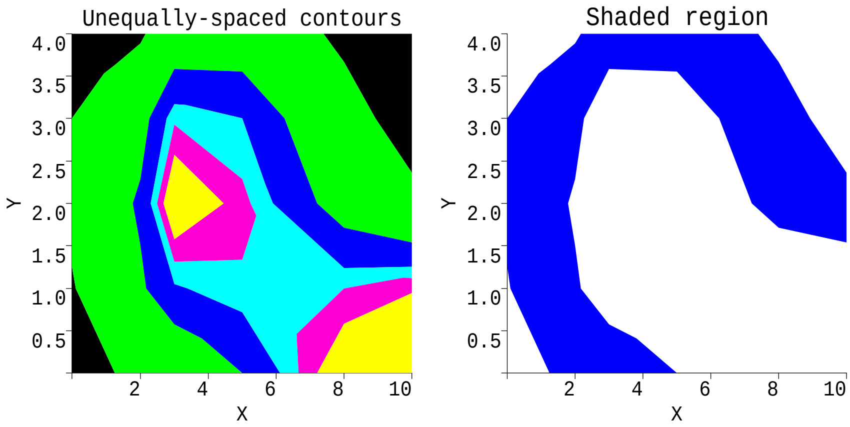
RGCONT(ZLEV,Z2ARR,NX,NY) draws all curves corresponding to the contour level, z=ZLEV, on the current 2-D or isometric picture.

RGSHAD(ZLEV1,ZLEV2,ISHADE,Z2ARR,NX,NY) draws all shaded contour regions between the two contour levels, z=ZLEV1 and z=ZLEV2, on the current 2-D picture, using shading pattern ISHADE. min(ZLEV1,ZLEV2) is included in the shaded region and max(ZLEV1,ZLEV2) is excluded.

8.1.4 Contour maps of tartan-gridded data

This example illustrates how to:

- draw a shaded contour map of x-specified tartan-gridded data,
- draw an individual shaded region,
- draw contour lines at user-defined levels.




```
      PROGRAM CONT4
      PARAMETER (NX=6,NY=5,NZ=5,Y1=0.0,DY=1.0)
      REAL Z2ARR(NX,NY),XARR(NX),HEIGHT(NZ)
      DATA Z2ARR /0.5, 1.3, 1.8, 3.0, 5.7, 6.8,
     +            0.9, 2.8, 3.9, 4.4, 4.5, 4.9,
     +            1.3, 3.2, 5.8, 4.7, 2.4, 1.4,
     +            1.0, 2.5, 4.4, 4.0, 1.6, 0.3,
     +            0.0, 0.8, 2.0, 2.2, 0.7, 0.0/
      DATA XARR  /0.0,2.0,3.0,5.0,8.0,10.0/
      DATA HEIGHT/1.0,3.0,4.0,4.5,5.0/
C specify grouping and set scales
      CALL GROUP(2,1)
      CALL SCALES(XARR(1),XARR(NX),1,Y1,Y1+(NY-1)*DY,1)
C draw a set of unequally-spaced contour ranges
      CALL AXES7('X','Y')
      CALL SQZVAL(HEIGHT,NZ)
      CALL XSHDS(Z2ARR,NX,NY,XARR)
      CALL TITLE7('O','C','Unequally-spaced contours')
C draw a single contour range
      CALL AXES7('X','Y')
      CALL XSHAD(HEIGHT(1),HEIGHT(2),4,Z2ARR,NX,NY,XARR)
      CALL TITLE7('O','C','Shaded region')
C terminate plotting
      CALL ENDPLT
      END
```

Example 4. Contour maps of tartan-gridded data

Explanation of subroutines

`GROUP(NHORIZ,NVERT)` specifies that subsequent pictures are to be grouped together, with `NHORIZ` pictures across the page and `NVERT` pictures down the page. The order of pictures is from left to right and from top to bottom.

`SQZVAL(RARR,NARR)` specifies a sequence of contour levels.

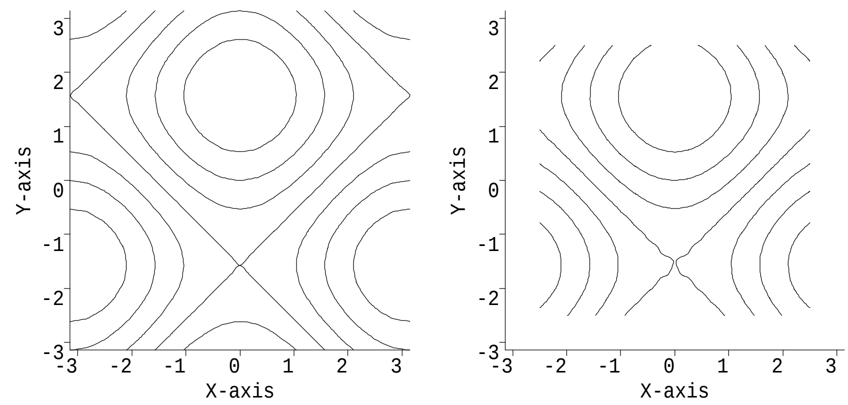
`XSHDS(Z2ARR,NX,NY,XARR)` draws a shaded contour map on the current 2-D picture.

`XSHAD(ZLEV1,ZLEV2,ISHADE,Z2ARR,NX,NY,XARR)` draws all shaded contour regions between the two contour levels, z=`ZLEV1` and z=`ZLEV2`, on the current 2-D picture, using shading pattern `ISHADE`. z_{min}=min(`ZLEV1`,`ZLEV2`) is included in the shaded region and z_{max}=max(`ZLEV1`,`ZLEV2`) is excluded.

8.1.5 Contour maps of a user-defined function

This example illustrates how to:

- draw a contour map of a user-defined function,
- increase the mesh size to produce smoother curves,
- specify function limits and/or scale limits.




```
      PROGRAM CONT5
      PARAMETER (PI=3.14159)
      EXTERNAL COSPSN
C specify layout and axis characteristics
      CALL GROUP(2,1)
      CALL AXLOCN('*C','P')
      CALL SFMESH(21,21)
C Evaluate function and draw contour over scale limits
      CALL SCALES(-PI,PI,1,-PI,PI,1)
      CALL AXES7('X-axis','Y-axis')
      CALL FNCNTS(COSPSN)
C Draw 2nd picture with axis & function scales defined separately
      CALL AXES7('X-axis','Y-axis')
      CALL FNAREA(-2.5,2.5,-2.5,2.5)
      CALL FNCNTS(COSPSN)
C terminate plotting
      CALL ENDPLT
      END
C user-defined function of two REAL variables
      REAL FUNCTION COSPSN(X,Y)
      COSPSN=COS(X)+SIN(Y)
      END
```

Example 5. Contour maps of a user-defined function

Explanation of subroutines

`AXLOCN('*C','P')` specifies that all Cartesian axes are to be positioned `'P'`receding the picture; they would normally be at zero, and obscure part of these pictures.

`SFMESH(MX,MY)` specifies the underlying mesh used for evaluating a user-defined function and for constructing contour lines.

`FNCNTS(FUNXY)` draws a contour map representing the specified function of two `REAL` variables over the current limits.

`FNAREA(XMIN,XMAX,YMIN,YMAX)` specifies the limits of the independent variables (x and y) used for evaluating a function, $z = f(x, y)$.

8.2 Surface picture

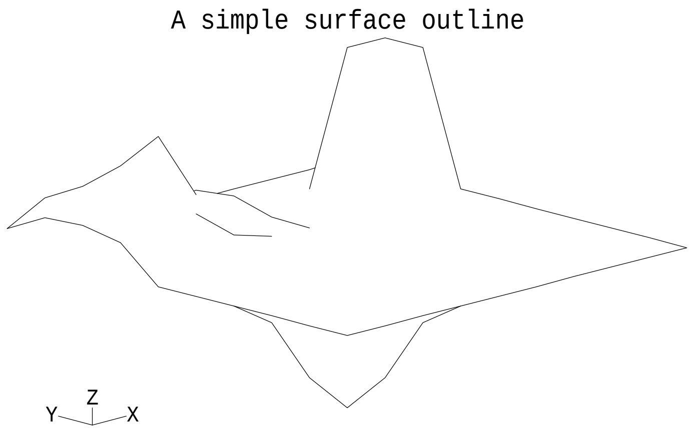
Surface picture]

8.2.1 A simple surface picture

This example illustrates how to draw an outline surface picture from a 2-D array of regular-gridded data.

Explanation of subroutines

`RGSURF(Z2ARR,NX,NY)` starts a new 3-D picture and draws a surface picture of the regular-gridded data in the `NX`×`NY` array, `Z2ARR`.



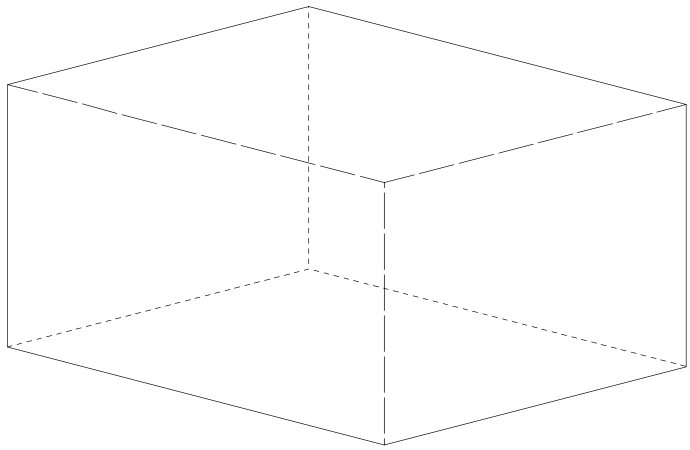

```
      PROGRAM SURF1
      PARAMETER(NX=10,NY=10)
      REAL Z2ARR(NX,NY)
      DATA Z2ARR/0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     +           0.0,-2.6,-3.8,-1.4, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     +           0.0,-3.8,-5.9,-2.3, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     +           0.0,-1.4,-2.3,-0.4, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     +           0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     +           0.0, 1.5, 1.7, 1.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
     +           1.8, 1.4, 2.3, 0.4, 1.1, 0.0, 0.0, 8.0, 8.0, 0.0,
     +           2.2, 3.8, 5.9, 2.3, 1.7, 0.0, 0.0, 8.0, 8.0, 0.0,
     +           2.1, 2.6, 3.8, 1.4, 1.5, 0.0, 0.0, 0.0, 0.0, 0.0,
     +           1.0, 2.1, 2.2, 1.8, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0/
C draw surface picture
      CALL RGSURF(Z2ARR,NX,NY)
C draw title and terminate plotting
      CALL TITLE7('T','C','A simple surface outline')
      CALL ENDPLT
      END
```

Example 6. A simple surface picture

8.2.2 Isometric drawing

Isometric drawing]

This example illustrates general isometric plotting using the underlying coordinate system.



Explanation of subroutines

SFZSCL(ZSTART,ZSTOP) specifies the z scale.

ISNEW starts a new isometric (3-D) picture. x-y scales cover the ranges specified by FNAREA, and the z scale is as specified by a previous call of SFZSCL.

KISXY(X3,Y3,Z3,X2,Y2) converts a point, (X3,Y3,Z3), on a surface picture to its 2-D equivalent, (X2,Y2).

BREAK is called to move to the next point specified by BRKNPT.

BRKNPT(X,Y,LTYPE) draws a straight line from the current pen position to (X,Y). The final argument, LTYPE, specifies the broken line pattern to be used (see Figure D.1, page 161).

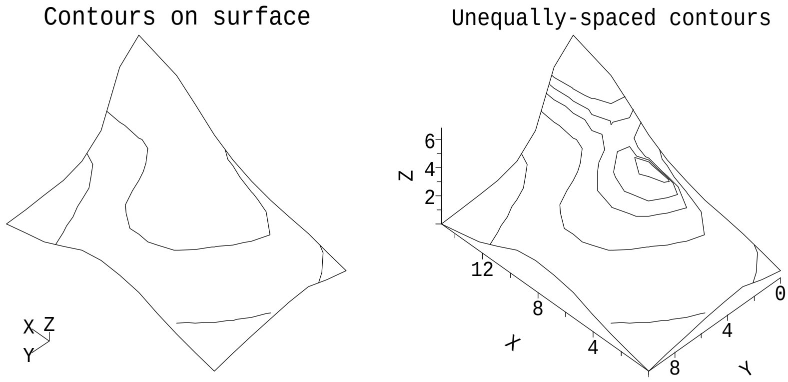
```
      PROGRAM SURF2
      PARAMETER(XMIN=0.0,XMAX=1.5,YMIN=0.0,YMAX=2.0,ZMIN=0.0,ZMAX=1.0)
C specify scales & start null isometric picture
      CALL FNAREA(XMIN,XMAX,YMIN,YMAX)
      CALL SFZSCL(ZMIN,ZMAX)
      CALL ISNEW
C draw frame
      CALL ISLINE(XMIN,YMIN,ZMIN,XMAX,YMIN,ZMIN,0)
      CALL ISLINE(XMAX,YMIN,ZMIN,XMAX,YMIN,ZMAX,0)
      CALL ISLINE(XMAX,YMIN,ZMAX,XMAX,YMAX,ZMAX,0)
      CALL ISLINE(XMAX,YMAX,ZMAX,XMIN,YMAX,ZMAX,0)
      CALL ISLINE(XMIN,YMAX,ZMAX,XMIN,YMAX,ZMIN,0)
      CALL ISLINE(XMIN,YMAX,ZMIN,XMIN,YMIN,ZMIN,0)
C draw front lines
      CALL ISLINE(XMIN,YMIN,ZMAX,XMAX,YMIN,ZMAX,1)
      CALL ISLINE(XMIN,YMIN,ZMAX,XMIN,YMIN,ZMIN,1)
      CALL ISLINE(XMIN,YMIN,ZMAX,XMIN,YMAX,ZMAX,1)
C draw back lines
      CALL ISLINE(XMAX,YMAX,ZMIN,XMAX,YMAX,ZMAX,-1)
      CALL ISLINE(XMAX,YMAX,ZMIN,XMAX,YMIN,ZMIN,-1)
      CALL ISLINE(XMAX,YMAX,ZMIN,XMIN,YMAX,ZMIN,-1)
C terminate plotting
      CALL ENDPLT
      END
C subroutine to draw a line using isometric coordinates
      SUBROUTINE ISLINE(X31,Y31,Z31,X32,Y32,Z32,LTYPE)
      CALL KISXY(X31,Y31,Z31,XC1,YC1)
      CALL KISXY(X32,Y32,Z32,XC2,YC2)
      CALL BREAK
      CALL BRKNPT(XC1,YC1,LTYPE)
      CALL BRKNPT(XC2,YC2,LTYPE)
      CALL BREAK
      END
```

Example 7. Null isometric picture

8.2.3 Surface pictures of tartan-gridded data

This example illustrates how to:

- draw surface pictures of y-specified tartan-gridded data,
- add individual contour lines to the current surface picture,
- draw surface pictures with a set of contour lines at user-defined intervals.




```
      PROGRAM SURF3
      PARAMETER (NX=6,NY=5,NLEVEL=5)
      REAL Z2ARR(NX,NY),YARR(NY),HEIGHT(NLEVEL)
      DATA Z2ARR/ 0.5, 1.3, 1.8, 3.0, 5.7, 6.8,
     +            0.9, 2.8, 3.9, 4.4, 4.5, 4.9,
     +            1.3, 3.2, 5.8, 4.7, 2.4, 1.4,
     +            1.0, 2.5, 4.4, 4.0, 1.6, 0.3,
     +            0.0, 0.8, 2.0, 2.2, 0.7, 0.0/
      DATA YARR  /0.0,2.0,3.0,5.0,10.0/
      DATA HEIGHT/1.0,3.0,4.0,4.5,5.0/
C specify layout and surface characteristics
      CALL GROUP(2,1)
      CALL ISANG(45.0)
      CALL ISVIEW(1)
C x-scale starts at 0.0 with intervals of 3.0
      CALL SFEQX(0.0,3.0)
C draw surface with added contours
      CALL YSURF(Z2ARR,NX,NY,YARR)
      CALL YCONT(1.0,Z2ARR,NX,NY,YARR)
      CALL YCONT(3.0,Z2ARR,NX,NY,YARR)
      CALL TITLE7('O','C','Contours on surface')
C specify that surfaces are drawn with contours
      CALL ISTYPE(2)
C draw surface with axes and specified contour levale
      CALL ISAXES(.TRUE.)
      CALL SQZVAL(HEIGHT,NLEVEL)
      CALL YSURF(Z2ARR,NX,NY,YARR)
      CALL TITLE7('O','C','Unequally-spaced contours')
C terminate plotting
      CALL ENDPLT
      END
```

Example 8. Surface pictures of tartan-gridded data

Explanation of subroutines

`ISANG(ANGLE)` specifies that the surface is to be projected at an angle in degrees above the horizontal.

`ISVIEW(1)` specifies that the surface is to be viewed from corner number 1, corresponding to (`XMIN`,`YMAX`).

`SFEQX(XSTART,XSTEP)` specifies the data values associated with the equally-spaced x values of the data grid.

`YSURF(Z2ARR,NX,NY,YARR)` starts a new isometric picture and draws a surface representing the y-specified tartan-gridded z values in `Z2ARR`.

`YCONT(ZLEV,Z2ARR,NX,NY,YARR)` draws all curves corresponding to the contour level, z=`ZLEV`, on the current 2-D or isometric picture.

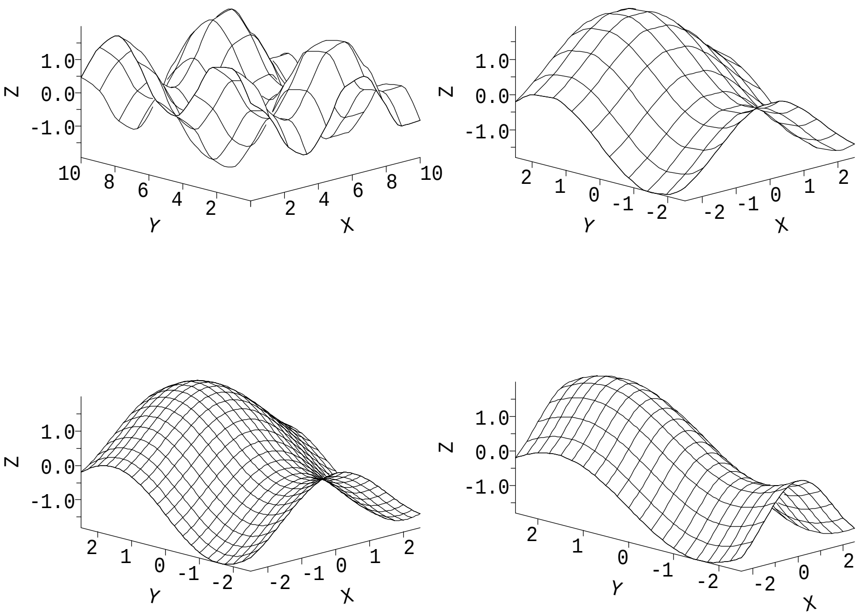
`ISTYPE(2)` specifies that surfaces are to consist of a simple outline with a set of contour lines added.

`ISAXES(.TRUE.)` specifies that axes are to be drawn on all subsequent surface pictures.

8.2.4 Surface pictures of a user-defined function

This example illustrates how to:

- draw surface pictures of a user-defined function,
- restrict the function to a specified area of the x-y plane,
- increase the concentration of points in the mesh,
- control x-y proportions by specifying the mesh.



Explanation of subroutines

`ISTYPE(3)` specifies crosshatched surfaces.

`FNSURF(FUNXY)` starts a new isometric picture and draws a surface representing a user-defined function of two variables.

`ISMESH(MXY)` specifies the concentration of mesh lines drawn on a crosshatched surface and used for evaluating the function.

The default mesh size is 20 for all surface type except a regular grid. On a regular grid, the default mesh is `NX`+`NY`, where `NX` and `NY` are the dimensions of the grid.

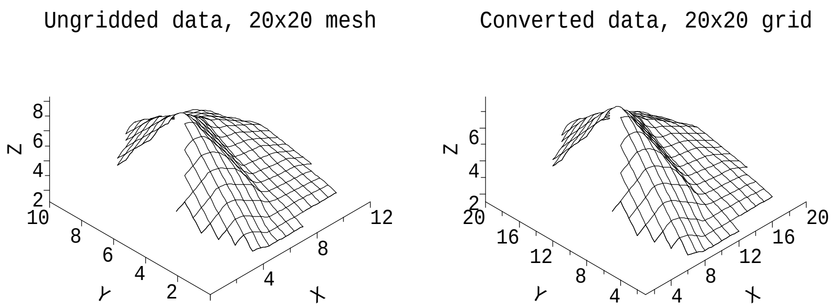
```
      PROGRAM SURF4
      EXTERNAL COSPSN
C specify layout and surface characteristics
      CALL GROUP(2,2)
      CALL ISAXES(.TRUE.)
      CALL ISTYPE(3)
C draw surface of user-defined function with default limits
      CALL FNSURF(COSPSN)
C specify limits and draw surface of the same function
      CALL FNAREA(-2.5,2.5,-2.5,2.5)
      CALL FNSURF(COSPSN)
C draw surface with finer mesh
      CALL ISMESH(42)
      CALL FNSURF(COSPSN)
C draw surface with different x-y proportions
      CALL SFMESH(11,21)
      CALL FNSURF(COSPSN)
      CALL ENDPLT
      END
C user-defined function of two REAL variables
      REAL FUNCTION COSPSN(X,Y)
      COSPSN=COS(X)+SIN(Y)
      END
```

Example 9. Surface pictures of a user-defined function

8.2.5 Surface pictures of ungridded data

This example illustrates how to:

- inquire the current no-data value,
- draw surface pictures of ungridded data,
- convert the ungridded data to a regular grid,
- draw an equivalent surface from the converted data.




```
      PROGRAM SURF5
      PARAMETER (NPTS=12,NODES=3,ISIZE=25,MX=20,MY=20)
      REAL XA(NPTS),YA(NPTS),ZA(NPTS),Z2ARR(MX,MY)
      INTEGER I2ARR(NODES,ISIZE),N2ARR(NODES,ISIZE)
      DATA XA/0.0,1.0,2.0,4.0,4.0, 6.0,7.0,8.0,10.0,10.0,11.0,12.0/
      DATA YA/6.0,3.0,8.0,0.0,5.0,10.0,2.0,7.0, 4.0, 9.0, 1.0, 6.0/
      DATA ZA/2.5,3.8,3.2,1.2,8.3, 3.5,2.4,4.6, 2.9, 3.0, 1.8, 2.5/
C specify layout
      CALL GROUP(2,1)
      CALL ISTYPE(3)
      CALL ISANG(35.26)
      CALL ISAXES(.TRUE.)
      CALL QNODAT(RNODAT)
      ZA(1)=RNODAT
      CALL SFMESH(MX,MY)
C triangulate data and draw surface of ungridded data
      CALL ZZORDR(XA,YA,NPTS,I2ARR,N2ARR,NELEMS,ISIZE)
      CALL ZZSURF(XA,YA,ZA,NPTS,I2ARR,N2ARR,NODES,NELEMS)
      CALL TITLE7('T','C','Ungridded data, 20x20 mesh')
C convert data and draw surface of gridded data
      CALL KZZRG(XA,YA,ZA,NPTS,I2ARR,N2ARR,NODES,NELEMS,Z2ARR,MX,MY)
      CALL RGSURF(Z2ARR,MX,MY)
      CALL TITLE7('T','C','Converted data, 20x20 grid')
      CALL ENDPLT
      END
```

Example 10. Surface pictures of ungridded data

Explanation of subroutines

QNODAT(RVAL) inquires the current no-data value. This example sets Z(1) to the current no-data value, to give a data set containing a 'hole'.

ZZORDR(XARR,YARR,NPTS,I2ARR,N2ARR,NELEMS,ISIZE) organizes ungridded data into triangular elements and neighbours.

ZZSURF(XARR,YARR,ZARR,NPTS,I2ARR,N2ARR,NODES,NELEMS) starts a new isometric picture and draws a surface representing the ungridded z values with neighbours.

`KZZRG(XARR,YARR,ZARR,N,I2ARR,N2ARR,NODES,NELEMS,Z2ARR,NX,NY)` generates a regular-gridded data set from ungridded data with neighbours.

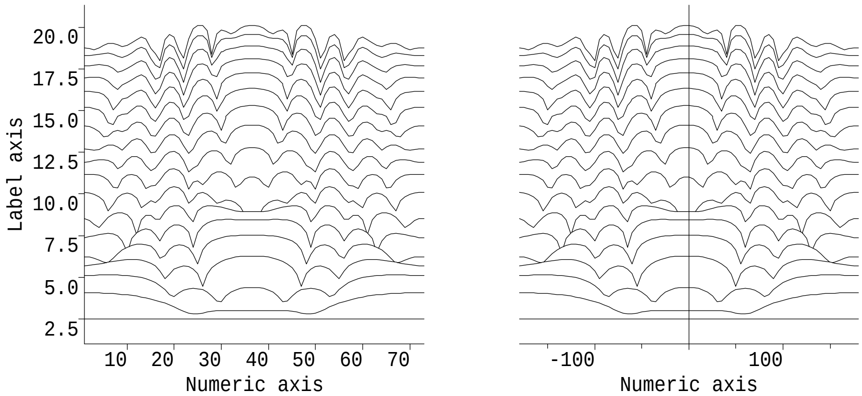
8.3 Waterfall chart

Waterfall chart]

8.3.1 A simple waterfall chart

This example illustrates how to:

- draw a waterfall chart,
- change the numerical scale of a waterfall chart,
- alter waterfall axes.




```
      PROGRAM WATER1
      PARAMETER(NPTS=73,NCURVE=18)
      REAL Z2ARR(NPTS,NCURVE)
      OPEN(UNIT=10, FILE='water1.dat', status='OLD')
      READ(10,*) Z2ARR
      CLOSE(10)
      CALL GROUP(2,1)
C start picture and draw waterfall chart
      CALL NEWPIC
      CALL WFCHT(Z2ARR,NPTS,NCURVE,'Numeric axis','Label axis')
C set numeric scale -180 .. 180 and clear label scale
      CALL WFEQN(-180.0, 5.0)
      CALL AXCLR('LW', 1)
C start new picture and draw same chart
      CALL NEWPIC
      CALL WFCHT(Z2ARR,NPTS,NCURVE,'Numeric axis',' ')
C terminate plotting
      CALL ENDPLT
      END
```

Example 11. A simple waterfall chart

Explanation of subroutines

`WFCHT(Z2ARR,NPTS,NCURVS,CAPN,CAPL)` draws a waterfall chart of data held in the 2-dimensional array, `Z2ARR(NPTS,NCURVS)`, where `NCURVS` is the number of curves to be plotted. Each curve n, corresponding to the points `Z2ARR(1,`n`)` to `Z2ARR(NPTS,`n`)`, is offset from the last by an equal step and masked by curves already drawn on the waterfall chart.

`NPTS` and `NCURVS` must both be positive integers; if they are not, a diagnostic is generated and nothing is drawn. A maximum of 1024 and a minimum of 2 points may be plotted as a waterfall curve.

WFCHT draws numeric and label axes which are configured using appropriate AX* subroutines with CHAXIS='NW' and 'LW' respectively.

WFCHT must be preceded by the *new picture* subroutine NEWPIC.

WFEQN(ESTART,ESTEP) relates the equally-spaced points to the numeric scale. The first point corresponds to ESTART, and subsequent points are drawn at intervals of ESTEP.

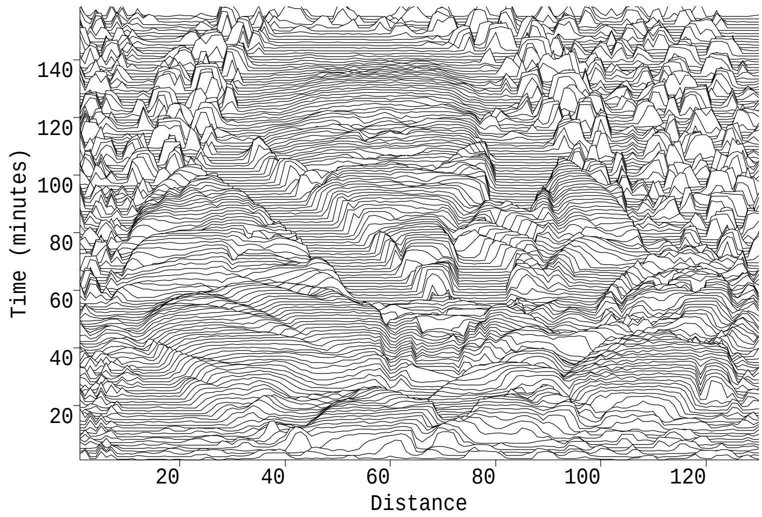
A curve with NPTS points will cover the range ESTART to ESTART + (ESTEP × (NCURVS-1)).

AXCLR('LW',1) specifies that the annotation is to be omitted when the label axis is drawn.

8.3.2 Drawing individual waterfall curves

This example illustrates how to:

- specify the number of curves on a chart,
- add curves one at a time.




```
      PROGRAM WATER2
      PARAMETER(NX=130,NY=155)
      REAL ZARR(NX)
C specify NY curves, start picture and open data file
      CALL WFNCVS(NY)
      CALL NEWPIC
      OPEN(UNIT=10, FILE='water2.dat', STATUS='OLD')
C read data for NY curves and draw them
      DO 10 J=1,NY
        READ(10,*) ZARR
        CALL WFDRAW(ZARR, NX)
10    CONTINUE
      CLOSE(10)
C add axes and terminate plotting
      CALL AXIS7('NW','Distance')
      CALL AXIS7('LW','Time (minutes)')
      CALL ENDPLT
      END
```

Example 12. Drawing individual waterfall curves

Explanation of subroutines

WFNCVS(NCURVS) specifies that the label scale is to accommodate NCURVS curves. NCURVS must be a positive integer; if not, the default is restored.

WFDRAW(ZARR,NPTS) draws a single curve of data contained in ZARR(NPTS). Each curve corresponding to the points ZARR(1) to ZARR(NPTS) is offset from the last by an equal step, and masked by curves already drawn on the waterfall chart.

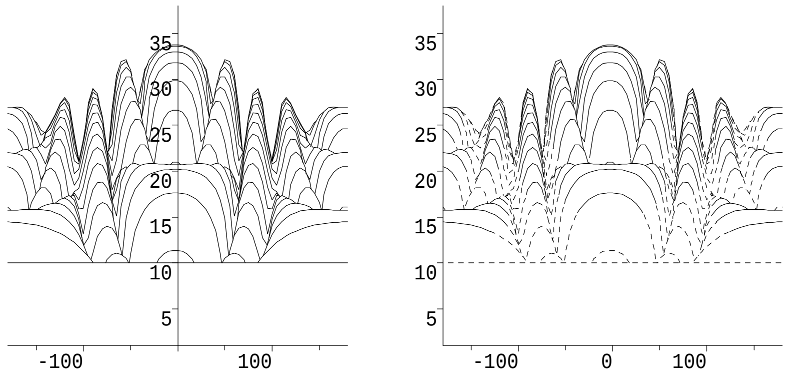
WFDRAW draws a waterfall curve on the current 2-D picture therefore the first call to WFDRAW must be preceded by the *new picture* subroutine NEWPIC.

AXIS7(CHAXIS,CAP) draws an axis for axis types 'NW' (numeric water) and 'LW' (label water).

8.3.3 Distinguishing between data levels on a waterfall chart

This example illustrates how to:

- specify the range of the numeric scale,
- specify the displacement between waterfall curves,
- use bundles to select different line styles,
- mark a data level on waterfall curves.




```
      PROGRAM WATER3
      PARAMETER(NPTS=73,NCURVS=18)
      REAL Z2ARR(NPTS,NCURVS)
C read data
      OPEN(UNIT=10, FILE='water1.dat', status='OLD')
      READ(10,*) Z2ARR
      CLOSE(10)
C set grouping, numeric scale and curve displacement
      CALL GROUP(2,1)
      CALL WFNSCL(-180.0, 180.0)
      CALL WFSTEP(0.05)
C start a new picture and draw first chart
      CALL NEWPIC
      CALL WFCHT(Z2ARR, NPTS, NCURVS, ' ', ' ')
C set bundles and mark points below -10.0 with pen 2
      CALL BUNLPR('SCT')
      CALL WFPNS(1,-1,2,-1)
      CALL WFZLEV(-10.0)
C draw second waterfall chart with Label axis Preceding
      CALL AXLOCN('LW', 'P')
      CALL NEWPIC
      CALL WFCHT(Z2ARR, NPTS, NCURVS, ' ', ' ')
C terminate plotting
      CALL ENDPLT
      END
```

Example 13. Distinguishing between data levels on a waterfall chart

Explanation of subroutines

`WFNSCL(START,STOP)` specifies the limits of the numeric scale.

`WFSTEP(ZDISP)` specifies the displacement between curves. `ZDISP` must be between 0.0 and 1.0. As `ZDISP` decreases, curves will be closer together; if `ZDISP` is 1.0, curves will be drawn clear of each other.

`BUNLPR(CHATTR)` specifies the order of priority of bundled attributes. `'SCT'` gives a sequence ordered by Style, Colour and Thickness.

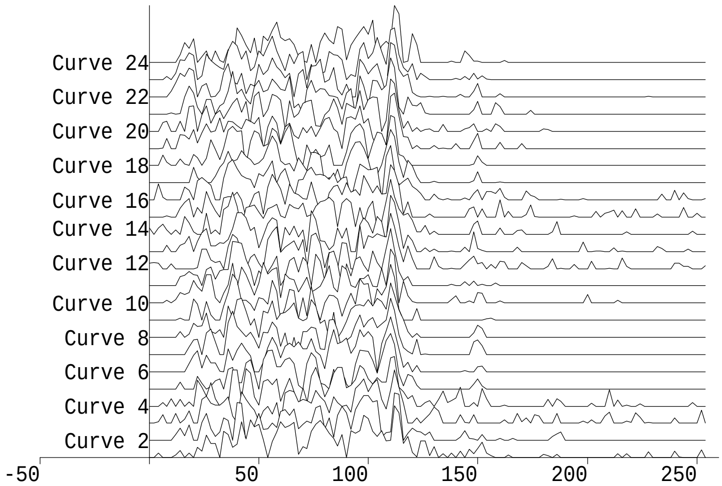
`WFPNS(IPEN1,IPEN2,IPEN3,IPEN4)` specifies the pens to be associated with the four pen pointers. `IPEN1` and `IPEN3` refer to the unmasked part of the curve. In this case pen -1 is used for the masked parts of the curve which means they are omitted.

`WFZLEV(ZLEVEL)` specifies the threshold z value for changing pens; the parts of the curve with a z value higher than `ZLEVEL` are drawn using the pens associated with the first pen pointer (unmasked) and the second (masked).

8.3.4 Conversion on waterfall charts

This example demonstrates how to:

- set waterfall data independently of the numeric scale,
- find the position of a point on a given waterfall curve.




```
      PROGRAM WATER4
      PARAMETER(NPTS=128,NCURVE=24)
      REAL Z2ARR(NPTS,NCURVE)
      CHARACTER*2 ISTR
      OPEN(UNIT=10, FILE='water4.dat', status='OLD')
      READ(10,*) Z2ARR
      CLOSE(10)
C set numeric scales from -50.0 to 260.0
      CALL WFNSCL(-50.0, 260.0)
C set numeric data from 0.0 to 256.0
      CALL WFEQN(0.0, 2.0)
C start new picture and draw chart with clear label axis
      CALL NEWPIC
      CALL AXCLR('LW', 1)
      CALL WFCHT(Z2ARR, NPTS, NCURVE, ' ', ' ')
C specify label justification Centre Right
      CALL LABJST('C', 'R')
C label relative to first point of every second curve
      DO 10 ICURVE=2,NCURVE,2
        CALL KWZVAL(ICURVE,Z2ARR(1,ICURVE),ZVAL)
        CALL KNUMB(ICURVE,ISTR)
        CALL CP7LB(0.0, ZVAL, 'Curve '//ISTR)
10    CONTINUE
C terminate plotting
      CALL ENDPLT
      END
```

Example 14. Waterfall chart of ungridded data

Explanation of subroutines

`LABJST(VCHAR,HCHAR)` specifies the vertical (`'C'`entred) and horizontal (`'R'`ight) justification of subsequent labels.

`KWZVAL(ICURVE,ZVAL,VALUE)` delivers a value for z=`ZVAL` on curve `ICURVE` in terms of the label scale.

`KNUMB(IVAL,STR)` converts an `INTEGER`, `IVAL`, to a string.

`CP7LB(X,Y,CAP)` draws the caption, `CAP`, at point (`X`,`Y`).

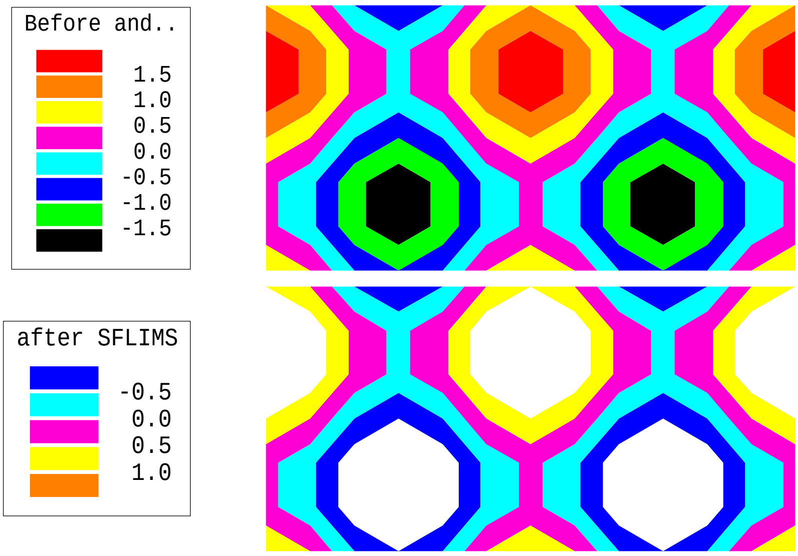
This example also show the interaction between `WFNSCL`, which sets the scales for the numeric axis, and `WFEQN`, which sets the numeric scales for the data to be plotted.

8.4 Advanced examples

8.4.1 Scales on contour maps and map keys

This example illustrates how to:

- specify the z scales for a contour map,
- specify a range of z values to be plotted within this scale.




```
      PROGRAM EXTRA1
      PARAMETER(PI=3.141593,ZMIN=-2.0,ZMAX=2.0,ZINT=0.5)
      EXTERNAL COSPSN
C specify layout
      CALL GROUP(1,2)
C specify scales and contour intervals
      CALL EQSCAL(-2.0*PI,2.0*PI,-PI,PI,0)
      CALL SFEQZ(ZMIN,ZINT)
      CALL SFZSCL(ZMIN,ZMAX)
C start new picture, draw key and shaded map
      CALL NEWPIC
      CALL MPK7V('C', 'P', ZMAX-ZINT, ZMIN, 'Before and..')
      CALL FNSHDS(COSPSN)
C specify limits, draw second picture then terminate plotting
      CALL SFLIMS(-1.0,1.0)
      CALL NEWPIC
      CALL MPK7V('C', 'P', ZMIN, ZMAX, 'after SFLIMS')
      CALL FNSHDS(COSPSN)
      CALL ENDPLT
      END
C
      REAL FUNCTION COSPSN(X,Y)
      COSPSN=COS(X)+SIN(Y)
      END
```

Example 15. Controlling the z scale

Explanation of subroutines

`EQSCAL(XSTART,XSTOP,YSTART,YSTOP,0)` specifies proportional scales, from `XSTART` to `XSTOP` in x, and from `YSTART` to `YSTOP` in y.

`SFEQZ(ZSTART,ZSTEP)` specifies that contours are to be drawn at `ZSTART` (if plottable), and at intervals of `ZSTEP` from `ZSTART`.

`MPK7V(VCHAR,HCHAR,ZTOP,ZBOTTM,CAP)` draws a vertical key to the shading patterns to cover the range `ZTOP` to `ZBOTTM` with the current contour interval. The data range represented on the key is the intersection of the ranges specified by `SFLIMS` and `SFZSCL`.

Each key entry represents the shaded region between two contour levels, such that $z_{min} \leq z < z_{max}$; this explains why the second map key includes five contour levels when only four appear on the map.

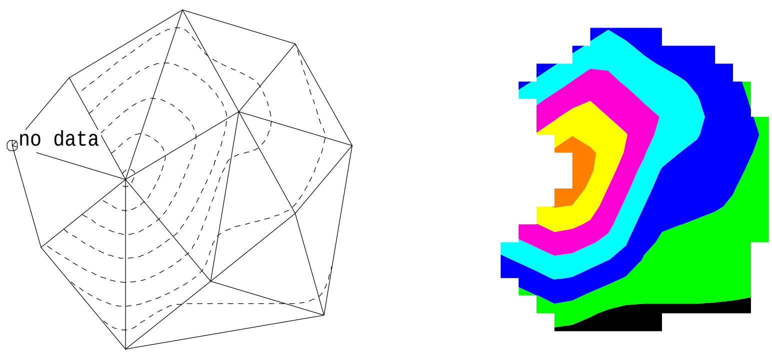
`FNSHDS(FUNXY)` draws a shaded contour map from the user-defined function `FUNXY` on the current picture.

`SFLIMS(Z1,Z2)` specifies the sub-range of the z scale to be plotted on contour maps and map keys.

8.4.2 Contour maps of ungridded data

This example illustrates how to:

- use the current no-data value to modify a data set,
- draw the element structure of ungridded data,
- mask subsequent text.




```
      PROGRAM EXTRA2
      PARAMETER (NPTS=12,NODES=3,ISIZE=25,MX=20,MY=20)
      REAL XA(NPTS),YA(NPTS),ZA(NPTS),Z2ARR(MX,MY)
      INTEGER I2ARR(NODES,ISIZE),N2ARR(NODES,ISIZE)
      DATA XA/0.0,1.0,2.0,4.0,4.0, 6.0,7.0,8.0,10.0,10.0,11.0,12.0/
      DATA YA/6.0,3.0,8.0,0.0,5.0,10.0,2.0,7.0, 4.0, 9.0, 1.0, 6.0/
      DATA ZA/2.5,3.8,3.2,1.2,8.3, 3.5,2.4,4.6, 2.9, 3.0, 1.8, 2.5/
C inquire no-data value and modify data
      CALL QNODAT(RNODAT)
      ZA(1)=RNODAT
C specify layout
      CALL GROUP(2,1)
      CALL LIMEXC(XA,NPTS,XMIN,XMAX)
      CALL LIMEXC(YA,NPTS,YMIN,YMAX)
      CALL SCALES(XMIN,XMAX,1,YMIN,YMAX,1)
C triangulate data and draw elements
      CALL NEWPIC
      CALL ZZORDR(XA,YA,NPTS,I2ARR,N2ARR,NELEMS,ISIZE)
      CALL CHMASK(.TRUE.)
      CALL CP7PT(XA(1),YA(1),1,'no data')
      CALL ZZELMS(XA,YA,NPTS,I2ARR,N2ARR,NODES,NELEMS)
C draw contours with line pattern -1
      CALL CTBRKN(-1)
      CALL ZZCNTS(XA,YA,ZA,NPTS,I2ARR,N2ARR,NODES,NELEMS)
C draw shaded contours
      CALL NEWPIC
      CALL KZZRG(XA,YA,ZA,NPTS,I2ARR,N2ARR,NODES,NELEMS,Z2ARR,MX,MY)
      CALL RGSHDS(Z2ARR,MX,MY)
      CALL ENDPLT
      END
```

Example 16. Contour maps of ungridded data

Explanation of subroutines

`CHMASK(.TRUE.)` specifies that text is not overdrawn by subsequent plotting.

`CP7PT(X,Y,MKTYPE,CAP)` draws the symbol, `MKTYPE`, centred at point (`X`,`Y`) and draws the caption, `CAP`, above and to the right of the symbol.

`ZZELMS(XARR,YARR,NPTS,I2ARR,N2ARR,NODES,NELEMS)` draws the outlines of a set of elements on the current 2-D picture.

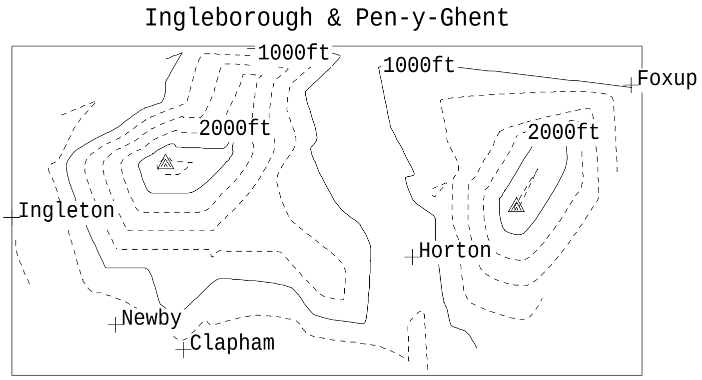
`CTBRKN(LTYPE)` specifies the broken line style to be used on subsequent lines, cross sections and representations of ungridded data configuration.

`ZZCNTS(XARR,YARR,ZARR,NPTS,I2ARR,N2ARR,NODES,NELEMS)` draws a set of contours from ungridded data on the current 2-D picture.

8.4.3 Extra labelling on contour maps

This example illustrates how to:

- add labels to a contour map,
- specify a sequence of line patterns and labels,
- control the contour interval.



Explanation of subroutines

`CTLABS(IFREQ)` specifies the minimum number of points which make up a contour curve for it to be labelled.

`SQZLAB(LABARR,NARR)` specifies a sequence of contour labels.

`SQBRKN(IARR,NARR)` specifies a sequence of broken line patterns to be used when plotting a sequence of contour lines.

`MARKPT(X,Y,MKTYPE)` moves to the point (`X`,`Y`), without drawing and draws the marker symbol, `MKTYPE` (see Figure D.5).

`BRKNBX(X1,Y1,X2,Y2,LTYPE)` draws a box with corners at (`X1`,`Y1`) and (`X2`,`Y2`)) using line style `LTYPE`. Line style 0 is a solid line.

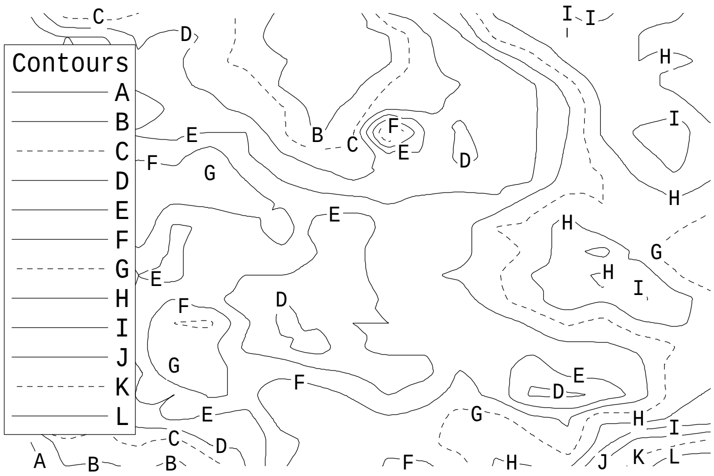
```
      PROGRAM EXTRA3
      PARAMETER(NN=3,NPTS=67,MAXE=NPTS*2+1,NVILL=5,NPK=2,NLEV=8)
      REAL X(NPTS),Y(NPTS),Z(NPTS),XVIL(NVILL),YVIL(NVILL)
      INTEGER IARR(NN,MAXE),NBARR(NN,MAXE),BKNSQ(NLEV),IPEAK(NPK)
      CHARACTER CHVIL(NVILL)*8,LABSQ(NLEV)*6
      DATA XVIL /69.8,87.1,74.6,81.0,72.7/
      DATA YVIL /73.1,76.8,69.4,72.0,70.1/
      DATA IPEAK/11,58/
      DATA BKNSQ/ -1,-1,0,-1,-1,-1,0,-1/
      DATA LABSQ/' ',' ','1000ft',' ',' ',' ','2000ft',' '/
      DATA CHVIL/'Ingleton','Foxup','Clapham','Horton','Newby'/
C read data and perform triangulation
      OPEN(UNIT=10,FILE='extra3.dat',STATUS='OLD')
      READ(10,*)X,Y,Z
      CLOSE(UNIT=10)
      CALL ZZORDR(X,Y,NPTS,IARR,NBARR,NELEM,MAXE)
C evaluate limits and set equal scaling
      CALL LIMEXC(X,NPTS,XMIN,XMAX)
      CALL LIMEXC(Y,NPTS,YMIN,YMAX)
      CALL EQSCAL(XMIN,XMAX,YMIN,YMAX,0)
C define labels, line styles and contour levels
      CALL CTLABS(5)
      CALL SQZLAB(LABSQ,NLEV)
      CALL SQBRKN(BKNSQ,NLEV)
      CALL SFEQZ(500.0,250.0)
C start new picture and label the CHVILages (masked)
      CALL CHMASK(.TRUE.)
      CALL NEWPIC
      DO 10,I=1,NVILL
        CALL CP7PT(XVIL(I),YVIL(I),32,CHVIL(I))
10    CONTINUE
C draw contours and mark the two peaks and add titles
      CALL ZZCNTS(X,Y,Z,NPTS,IARR,NBARR,NN,NELEM)
      DO 20,I=1,NPK
         N=IPEAK(I)
         CALL MARKPT(X(N),Y(N),37)
20    CONTINUE
      CALL BRKNBX(XMIN,YMIN,XMAX,YMAX,0)
      CALL TITLE7('T','C','Ingleborough & Pen-y-Ghent')
      CALL ENDPLT
      END
```

Example 17. Labelling on contour maps

8.4.4 Sequences on contour maps

This example illustrates how to:

- specify sequences of pens and labels,
- reduce the width of a vertical key.



Explanation of subroutines

`KSCALE(START,STOP,DIV,VSTART,VSTOP,VDIV)` returns `VSTART`, `VSTOP` and `VDIV` to give a 'nice' range of values and interval which include `START` and `STOP`.

`DEFKYW(NCHARS)` specifies the width of key samples.

`QKYCAP(LABARR,NARR,ITYPE,NROWS,NCOLMS)` inquires the key size.

`DEFKEY(2,VCHAR,HCHAR,NROWS,NCOLMS)` defines a key area on the current picture which may not be overdrawn.

`ADDCP7(CAP)` adds the caption, `CAP`, to the current key.

`SETPNS(IPEN1,IPEN2,IPEN3,IPEN4)` specifies the pens to be associated with the four pen pointers used for general plotting but not waterfalls.

`LINEK7(LTYPE,CAP)` draws an annotated line in the current key area. `LTYPE` −999 draws a line with the current bundled attributes.

`SQPEN(IARR,NARR)` specifies a sequence of pens/bundles to be used when plotting a sequence of contour lines.

```
      PROGRAM EXTRA4
      PARAMETER(NX=20,NY=20,NPATT=12,NLABS=12)
      REAL Z2ARR(NX,NY)
      INTEGER IBUNDL(NPATT)
      CHARACTER LABELS(NLABS)
      DATA IBUNDL/1,1,2,1,1,1,2,1,1,1,2,1/
      DATA LABELS/'A','B','C','D','E','F','G','H','I','J','K','L'/
C read data
      OPEN(UNIT=10,FILE='extra4.dat',STATUS='OLD')
      READ(10,*)Z2ARR
      CLOSE(UNIT=10)
C specify plotting characteristics
      CALL BUNLPR('SCT')
C find limits and specify scales
      CALL LIMEXC(Z2ARR,NX*NY,ZMIN,ZMAX)
      CALL KSCALE(ZMIN,ZMAX,6.0,ZLO,ZHI,ZINT)
      CALL SFEQZ(0.0,ZINT)
      CALL SFZSCL(ZLO,ZHI)
C start new picture and draw key
      CALL NEWPIC
      CALL DEFKYW(7)
      CALL QKYCAP(LABELS,NLABS,1,NROWS,NCOLMS)
      CALL DEFKEY(2,'C','W',NROWS+1,NCOLMS)
      CALL ADDCP7('Contours')
      DO 10 I=1,NLABS
        CALL SETPNS(IBUNDL(I),1,1,1)
        CALL LINEK7(-999,LABELS(I))
 10   CONTINUE
C specify sequences and labelling characteristics
      CALL SQPEN(IBUNDL,NPATT)
      CALL SQZLAB(LABELS,NLABS)
      CALL CTLABS(2)
C draw contour map and perimeter
      CALL RGCNTS(Z2ARR,NX,NY)
      CALL ENDPLT
      END
```

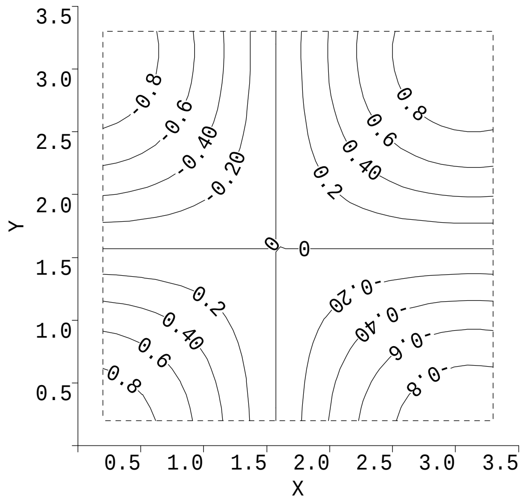
Example 18. Sequences on contour maps

8.4.5 Labelling contour curves

Labelling contour curves]

This example illustrates how to:

- save and retrieve the coordinates used to plot contour curves,
- convert a real number into a text string.



Explanation of subroutines

`LIMSFN(FUNXY,ZMIN,ZMAX)` finds the range of values used by the `FN*` subroutines, and sets `ZMIN` to the minimum and `ZMAX` to the maximum.

`CTHOLD(2)` specifies that coordinates of internally generated curves are to be saved but not plotted. `CTHOLD(1)` restores the default.

`FNCONT(ZLEV,FUNXY)` draws a contour curve z=`ZLEV` for the specified function of two `REAL` variables over the current function limits.

`QCURVE(MAXPTS,XARR,YARR,NPTS)` returns the coordinates of a curve.

`KREAL(RVAL,STR)` converts a `REAL` value, `RVAL`, to a string.

`CVLBJS(VCHAR,HCHAR)` specifies the justification of the label on a curve.

`LABCV7(XARR,YARR,NARR,LTYPE,CAP)` draws a curve with broken line pattern `LTYPE`, labelled `CAP`.

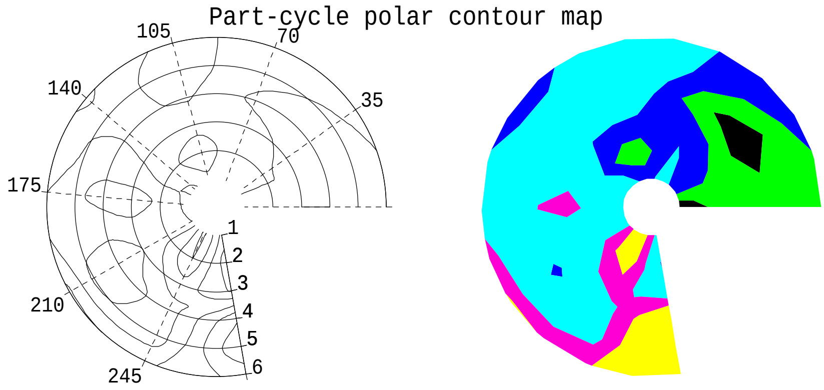
```
      PROGRAM EXTRA5
      PARAMETER(MAXARR=31, FNMIN=0.2, FNMAX=3.3)
      REAL XARR(MAXARR),YARR(MAXARR)
      CHARACTER ZSTR*5
      EXTERNAL COSXMY
C increase mesh size and set scales
      CALL SFMESH(31,31)
      CALL EQSCAL(0.0,3.5,0.0,3.5,0)
C specify area in which function is drawn
      CALL FNAREA(FNMIN,FNMAX,FNMIN,FNMAX)
C inquire function limits and calculate z limits/step
      CALL LIMSFN(COSXMY,FMIN,FMAX)
      CALL KSCALE(FMIN,FMAX,0.0,ZMIN,ZMAX,ZSTEP)
      NSTEPS=INT((ZMAX-ZMIN)/ZSTEP+0.5)
C start new picture and draw X-Y axes
      CALL AXES7('X','Y')
C Remember contour lines from a function; Centre labels
      CALL CTHOLD(2)
      CALL CVLBJS('C','C')
      DO 20 I=1,NSTEPS
C Define contour level, calculate, & convert to string
        ZLEV=ZMIN+REAL(I)*ZSTEP
        CALL FNCONT(ZLEV,COSXMY)
        CALL KREAL(ZLEV,ZSTR)
C inquire coordinates of curve to label
 10     CALL QCURVE(MAXARR,XARR,YARR,NPTS)
        IF (NPTS.NE.0) THEN
          CALL LABCV7(XARR,YARR,NPTS,0,ZSTR)
          GOTO 10
        ENDIF
 20   CONTINUE
C draw broken box around drawing region
      CALL BRKNBX(FNMIN,FNMIN,FNMAX,FNMAX,-1)
      CALL ENDPLT
      END
C user-defined function
      REAL FUNCTION COSXMY(X,Y)
      COSXMY=COS(X)*COS(Y)
      END
```

Example 19. Labelling contour curves

8.4.6 Polar contour map

This example illustrates how to:

- plot a contour map from polar data,
- specify restricted polar scales for a part cycle chart.




```
      PROGRAM EXTRA6
      PARAMETER (NR=6,NTH=9,NH=5)
      PARAMETER (RSTART=1.0,THSTRT=0.0,RSTEP=1.0,THSTEP=35.0)
      REAL Z2ARR(NR,NTH),ZHEIGH(NH)
      DATA ZHEIGH/0.5,1.0,1.5,3.0,4.0/
C read data
      OPEN(UNIT=10,FILE='extra6.dat',STATUS='OLD')
      READ(10,*)Z2ARR
      CLOSE(UNIT=10)
C specify layout
      CALL GROUP(2,1)
C specify data intervals and polar scales
      CALL SFEQX(RSTART,RSTEP)
      CALL SFEQY(THSTRT,THSTEP)
      RSTOP=RSTART+(NR-1)*RSTEP
      THSTOP=THSTRT+(NTH-1)*THSTEP
      CALL EQSCAL(RSTART,RSTOP,THSTRT,THSTOP,1)
C start new 2-d picture and draw data grid
      CALL AXSBDV('RP',0.0,RSTEP)
      CALL AXSBDV('AP',0.0,THSTEP)
      CALL AXGRID('*P',1,-1)
      CALL AXES7(' ',' ')
      CALL RGCNTS(Z2ARR,NR,NTH)
C draw unequally spaced shaded contour map
      CALL NEWPIC
      CALL SQZVAL(ZHEIGH,NH)
      CALL SFMESH(NR,NTH*2)
      CALL RGSHDS(Z2ARR,NR,NTH)
      CALL TITLE7('H','C','Part-cycle polar contour map')
      CALL ENDPLT
      END
```

Example 20. Polar contour map

Explanation of subroutines

`SFEQY(YSTART,YSTEP)` specifies the data values associated with the equally-spaced y values of the data grid.

`EQSCAL(RSTART,RSTOP,THSTRT,THSTOP,IUNITS)` specifies equal linear scales for Cartesian or polar plotting. In this example, `EQSCAL` specifies reduced angular and radial scales. `IUNITS` gives the angular units:

`IUNITS`	*Units*	`RSTART`, `RSTOP`	`THSTRT`, `THSTOP`
1	Polar	Radial units	Angles in degrees
2	Polar	Radial units	Angles in radians
3	Polar	Radial units	User-defined scale

`AXSBDV(CHAXIS,OFFSET,DELTA)` specifies subdivisions on axis type `CHAXIS`.

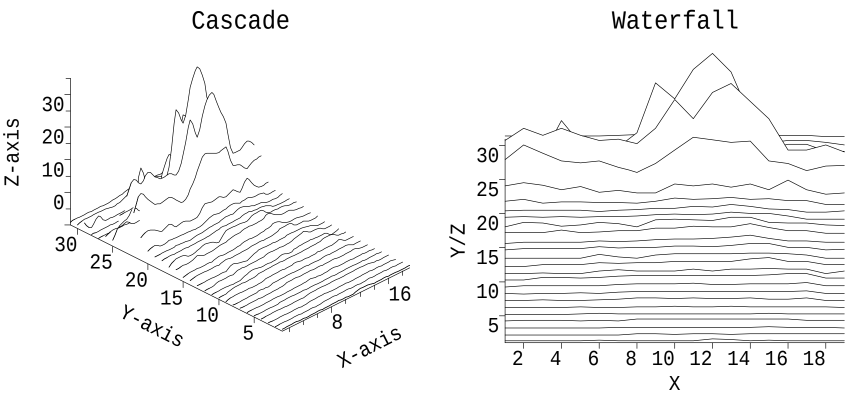
`AXGRID(CHAXIS,ILEVEL,LTYPE)` requests grids along the `CHAXIS` scale.

`AXES7(CAP1,CAP2)` starts a new polar picture (following the call to `EQSCAL`), draws a radial axis labelled `CAP1` at `THETA`=`THSTRT`, and (when the angular range is less than a cycle) another radial axis at `THETA`=`THSTOP` labelled with `CAP2`; the angular axis is drawn at `R`=`RSTOP` covering the angular range `THSTRT` to `THSTOP`.

8.4.7 Waterfall charts and cascade surface pictures

This example illustrates how to:

- draw a set of x-y-z axes on an isometric picture,
- draw cascade curves on the surface which correspond to the waterfall curves.




```
      PROGRAM EXTRA7
      PARAMETER (NX=19,NY=31)
      REAL Z2ARR(NX,NY)
C read data
      OPEN(UNIT=10,FILE='extra7.dat',STATUS='OLD')
      READ(10,*) Z2ARR
      CLOSE(UNIT=10)
C specify layout and scales
      CALL GROUP(2,1)
      CALL LIMEXC(Z2ARR,NX*NY,ZMIN,ZMAX)
      CALL KSCALE(ZMIN,ZMAX,0.0,ZSTART,ZSTOP,ZDIV)
C specify surface characteristics and draw surface
      CALL SFZSCL(ZSTART,ZSTOP)
      CALL ISANG(30.0)
      CALL ISRISE(0.5)
      CALL ISDIAG(.FALSE.)
      CALL ISTYPE(4)
      CALL RGSURF(Z2ARR,NX,NY)
      CALL ISAXD7('X-axis','Y-axis','Z-axis')
      CALL TITLE7('T','C','Cascade')
C draw waterfall chart
      CALL WFZSCL(ZSTART,ZSTOP)
      CALL WFSTEP(0.05)
      CALL AXRNGE('LW',1.0,REAL(NY))
      CALL NEWPIC
      CALL WFCHT(Z2ARR,NX,NY,'X','Y/Z')
      CALL TITLE7('T','C','Waterfall')
      CALL ENDPLT
      END
```

Example 21. Waterfalls and cascades

Explanation of subroutines

`ISRISE(FACTOR)` specifies the height of a surface picture. The z scale becomes more elongated as `FACTOR` increases. Default is `ISRISE(0.4)`.

`ISDIAG(.FALSE.)` specifies that the x-y-z diagram is to be omitted when the surface is drawn.

`ISTYPE(4)` specifies that surfaces are to be drawn as a series of cascade curves, $z_i = f(x, y_i)$.

`ISAXD7(CAPX,CAPY,CAPZ)` draws a set of x-y-z axes on a surface picture with axis captions `CAPX`, `CAPY` and `CAPZ`.

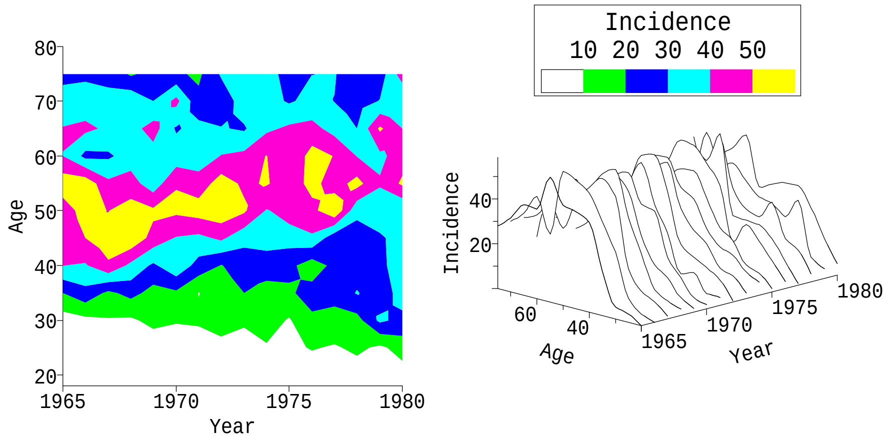
`WFZSCL(ZSTART,ZSTOP)` specifies the limits of z values for each curve.

`AXRNGE('LW',START,STOP)` specifies a sub-range of the label scale over which the waterfall label axis is to be drawn.

8.4.8 Shaded contour maps and surfaces

This example illustrates how to:

- draw both contour maps and a surface of the same data,
- draw similar 2-D and 3-D axes.




```
      PROGRAM EXTRA8
      PARAMETER(NYEARS=16,NAGE=12)
      REAL Z2ARR(NYEARS,NAGE)
C read data
      OPEN(UNIT=10,FILE='extra8.dat',STATUS='OLD')
      READ(10,*) Z2ARR
      CLOSE(UNIT=10)
C specify layout and scales
      CALL GROUP(2,1)
      CALL SFEQX(1965.0,1.0)
      CALL SFEQY(20.0,5.0)
      CALL SCALES(1965.0,1980.0,1,18.0,80.0,1)
C specify axis characteristics
      CALL AXSBDV('X*',1965.0,5.0)
      CALL AXLBGP('X*', 0)
      CALL AXLBJS('**','C')
C start new 2-D picture, draw axes and draw shaded contours
      CALL AXES7('Year','Age')
      CALL SHPATT(-1,1)
      CALL RGSHDS(Z2ARR,NYEARS,NAGE)
C specify surface characteristics
      CALL ISDIAG(.FALSE.)
      CALL ISTYPE(5)
C draw surface picture and add axes
      CALL RGSURF(Z2ARR,NYEARS,NAGE)
      CALL ISAXD7('Year','Age','Incidence')
C draw key to shaded contours
      CALL LIMEXC(Z2ARR,NYEARS*NAGE,ZMIN,ZMAX)
      CALL MPK7H('O','C',ZMIN,ZMAX,'Incidence')
      CALL ENDPLT
      END
```

Example 22. Comparison of data in different forms

Explanation of subroutines

`AXLBGP(CHAXIS,0)` specifies no labelling for axis type `CHAXIS` where the axes intersect.

`AXLBJS('**','C')` specifies that axis annotation labels on all axes are to be centred relative to their tick mark; even the wildcard, `'**'`, has no effect on the x and y isometric axes for which the justification of axis labels is fixed.

`SHPATT(ISHADE,IPOS)` specifies that the `IPOS`th shading pattern in the current sequence is to be `ISHADE`.

`ISTYPE(5)` specifies that surfaces are to be drawn as a series of cascade curves, $z_i = f(x_i, y)$.

8.4.9 Drawing a cross section

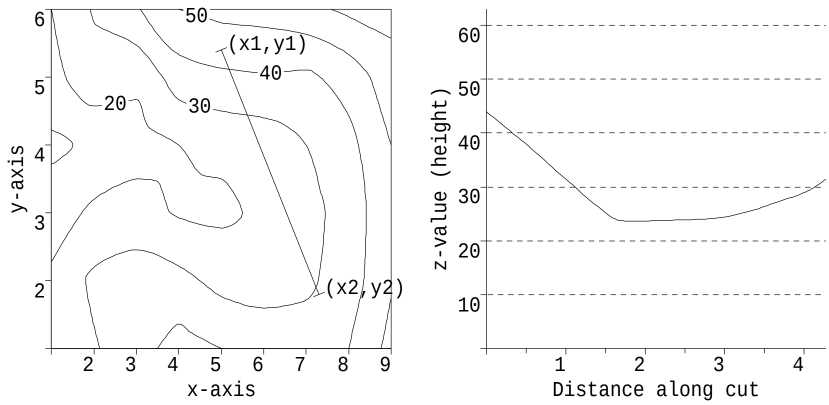
This example illustrates how to:

- plot the line of cross-sectional cut on a contour map,
- draw a cross-sectional curve representing the heights between (x_1, y_1) and (x_2, y_2).

The correspondence between the curve and the cut is seen by comparing the crossing points of the curve and the grid lines in the second picture with the contours and the line drawn by `RANGE` in the first picture. The variable, `DIST`, is set equal to the distance between the two points:

$$\texttt{DIST} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

This value is used to calculate a suitable x scale on the second picture.



Explanation of subroutines

`RANGE(X1,Y1,X2,Y2)` draws a line, on the current 2-D picture, between (`X1`,`Y1`) and (`X2`,`Y2`) with short perpendicular lines across the ends.

`CP7PT(X,Y,16,CAP)` draws the caption `CAP` next to a blank symbol (see Figure D.5, page 164).

`RGCUT(X1,Y1,X2,Y2,Z2ARR,NX,NY)` draws a cross section of the surface (representing the regular-gridded data in `Z2ARR`) corresponding to the 'cut' from (`X1`,`Y1`) to (`X2`,`Y2`).

```
      PROGRAM EXTRA9
      PARAMETER (NX=9,NY=6,X1=5.0,Y1=5.4,X2=7.3,Y2=1.8
     +,X0=1.0,DX=1.0,Y0=1.0,DY=1.0)
      REAL Z2ARR(NX,NY)
C read data
      OPEN(UNIT=10,FILE='extra9.dat',STATUS='OLD')
      READ(10,*)Z2ARR
      CLOSE(UNIT=10)
C specify layout and plotting characteristics
      CALL GROUP(2,1)
      CALL CTLABS(5)
C specify scales and relationship between data and the data grid
      CALL SFEQX(X0,DX)
      CALL SFEQY(Y0,DY)
      CALL SCALES(X0,X0+(NX-1)*DX,1,Y0,Y0+(NY-1)*DY,1)
C start new 2-D picture, draw axes and draw line contour map
      CALL AXES7('x-axis','y-axis')
      CALL RGCNTS(Z2ARR,NX,NY)
      CALL PERIM
C draw range and mark end points
      CALL RANGE(X1,Y1,X2,Y2)
      CALL CP7PT(X1,Y1,16,'(x1,y1)')
      CALL CP7PT(X2,Y2,16,'(x2,y2)')
C find limits and specify scales for cross section
      DIST=SQRT((X2-X1)**2+(Y2-Y1)**2)
      CALL LIMEXC(Z2ARR,NX*NY,ZMIN,ZMAX)
      CALL SCALES(0.0,DIST,1,0.0,ZMAX,1)
      CALL AXGRID('YC',1,-1)
C start new 2-d picture, draw axes and draw cross section
      CALL AXES7('Distance along cut','z-value (height)')
      CALL RGCUT(X1,Y1,X2,Y2,Z2ARR,NX,NY)
C terminate plotting
      CALL ENDPLT
      END
```

Example 23. Line of cut and cross section

A. Subroutine Specifications

This appendix gives brief formal specifications for the SIMPLEPLOT subroutines used in this manual. Full specifications are given in the *SIMPLEPLOT Reference manual*. All specifications are given in a similar format whether they are classified as graphics, specification or auxiliary subroutines. For example:

SUBROUTINE NAME (ARG1, ARG2)

Name

`NAME` – brief summary line

Availability Section 1, 2, 4, 5, 6, 7 or +, released version 2-*n*.

Arguments Throughout the specifications, arguments of type `INTEGER` have been given names starting with `I`–`N`, and arguments of type `REAL` have been given names starting with the letters `A`–`H` and `O`–`Z`.

IN only arguments are identified as *expression*; a variable name, a constant value or an expression may be used for such arguments.

INOUT or OUT only arguments are identified as *variable*.

`RVAL`	`REAL` expression	Expression with floating point value
`IVAL`	`INTEGER` expression	Expression with integer value
`TORF`	`LOGICAL` expression	Expression with logical value (`.TRUE.` or `.FALSE.`)
`CHAR`	`CHARACTER*1`	Expression with single character value
`CHTYPE`	`CHARACTER*2`	Expression with two character value
`CAP`	`STRING` expression	Expression with any length character value
`VARX`	`REAL` variable	Variable of type `REAL`
`STR`	`STRING` variable	Variable of type `CHARACTER*`*n*
`LABARR`	`STRING` array	Array of string values
`IARR`	`INTEGER` array	1-dimensional array of `INTEGER` values
`I2ARR`	`INTEGER` 2-D array	2-dimensional array of `INTEGER` values
`DARR`	`REAL` array	1-dimensional array of `REAL` values
`D2ARR`	`REAL` 2-D array	2-dimensional array of `REAL` values
`D3ARR`	`REAL` 3-D array	3-dimensional array of `REAL` values

SUBROUTINE ADDCP7 (CAP)
SUBROUTINE ADDCAP (CAP, NCAP)

Name

`ADDCP7` – to draw a caption in a previously defined area.

Availability Section 1, released before version 2-5.

Arguments

`CAP`	`STRING` expression	Caption
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

SUBROUTINE AXCLR (CHAXIS, ILEVEL)

Name

`AXCLR` – to specify the level of annotation drawn with the axis.

Availability Section 1, released version 2-11.

Arguments

`CHAXIS`	`CHARACTER*2`	Axis type
`ILEVEL`	`INTEGER` expression	Level of axis annotation

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
`CHAXIS`	`XC`	`YC`	`RP`	`AP`	`XI`	`YI`	`ZI`	`NB`	`LB`	`NW`	`LW`	`X3`	`Y3`	`Z3`	`U3`
	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

`ILEVEL`	*Effect*
0	Tick marks and annotation labels are drawn
1	Only axis line drawn

SUBROUTINE AXES7 (CAPX, CAPY)
SUBROUTINE AXES (CAPX, NCAPX, CAPY, NCAPY)

Name

`AXES7` – to start a new picture and draw axes.

Availability Section 1, released before version 2-5.

Arguments

`CAPX`	`STRING` expression	Caption for horizontal axis
`CAPY`	`STRING` expression	Caption for vertical axis
`NCAPX`	`INTEGER` expression	Number of characters in `CAPX`
`NCAPY`	`INTEGER` expression	Number of characters in `CAPY`

SUBROUTINE AXGRID (CHAXIS, ILEVEL, LTYPE)

Name

`AXGRID` – to specify the style of grids drawn in association with axis subdivisions.

Availability Section 1, released version 2-11.

Arguments

`CHAXIS`	`CHARACTER*2`	Axis type
`ILEVEL`	`INTEGER` expression	Level of grid
`LTYPE`	`INTEGER` expression	Type of broken line pattern

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
CHAXIS	XC	YC	RP	AP	XI	YI	ZI	NB	LB	NW	LW	X3	Y3	Z3	U3
	√	√	√	√	×	×	×	√	√	√	√	×	×	×	×

ILEVEL	*Level of grid*
0	No grid
1	Grid lines drawn [3] at annotated and major unannotated tick marks
2	Grid lines drawn [4] at all tick marks including minor

SUBROUTINE AXIS7 (CHAXIS, CAP)
SUBROUTINE AXIS (CHAXIS, CAP, NCAP)

Name

AXIS7 – to draw an axis on the current picture.

Availability Section 1, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
CAP	STRING expression	Axis caption
NCAP	INTEGER expression	Number of characters in CAP

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
CHAXIS	XC	YC	RP	AP	XI	YI	ZI	NB	LB	NW	LW	X3	Y3	Z3	U3
	√	√	√	√	√	√	√	√	√	√	√	√	√	√	×

SUBROUTINE AXLBGP (CHAXIS, ILEVEL)

Name

AXLBGP – to specify the level of axis annotation near an intersection.

Availability Section 4, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
ILEVEL	INTEGER expression	Level of annotation at intersection

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
CHAXIS	XC	YC	RP	AP	XI	YI	ZI	NB	LB	NW	LW	X3	Y3	Z3	U3
	√	√	√	×	√	√	√	√	×	√	√	√	√	√	√

ILEVEL	*Effect*
0	All annotation drawn
1	Annotation omitted near intersection

SUBROUTINE AXLBJS (CHAXIS, JCHAR)

Name

AXLBJS – to specify the justification of axis annotation.

Availability Section 4, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
JCHAR	CHARACTER*1	Justification of annotation

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
CHAXIS	XC	YC	RP	AP	XI	YI	ZI	NB	LB	NW	LW	X3	Y3	Z3	U3
	√	√	√	×	×	×	√	√	×	√	√	×	×	×	×

JCHAR	*Justification of annotation*
'B'	Between tick marks (centred)
'C'	Centered on the tick mark
'F'	Following the tick mark
'P'	Preceding the tick mark
'D'	Default position

SUBROUTINE AXLOCN (CHAXIS, LCHAR)

Name

AXLOCN – to specify the location of an axis relative to the picture.

Availability Section 4, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
LCHAR	CHARACTER*1	Location of axis

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
CHAXIS	XC	YC	RP	AP	XI	YI	ZI	NB	LB	NW	LW	X3	Y3	Z3	U3
	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

LCHAR	*Location of axis*
'P'	Preceding the other axis
'F'	Following the other axis
'D'	Default

SUBROUTINE AXRNGE (CHAXIS, START, STOP)

Name

AXRNGE – to specify the subrange over which an axis is to be drawn.

Availability Section 1, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
START	REAL expression	Start of axis in units of scale
STOP	REAL expression	End of axis in units of scale

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
CHAXIS	XC	YC	RP	AP	XI	YI	ZI	NB	LB	NW	LW	X3	Y3	Z3	U3
	√	√	√	√	√	√	√	√	×	√	√	√	√	√	√

SUBROUTINE AXSBDV (CHAXIS, OFFSET, DELTA)

Name

AXSBDV – to specify the interval at which a linear axis is to be subdivided and annotated.

Availability Section 1, released version 2-11.

Arguments

`CHAXIS`	`CHARACTER*2`	Axis type
`OFFSET`	`REAL` expression	Offset value for subdivisions
`DELTA`	`REAL` expression	Interval between subdivisions of axis in units of the scale

axis type	Cartesian		Polar		Isometric			Bars		Water		ViSualization			
`CHAXIS`	`XC`	`YC`	`RP`	`AP`	`XI`	`YI`	`ZI`	`NB`	`LB`	`NW`	`LW`	`X3`	`Y3`	`Z3`	`U3`
	$\surd^L$	$\surd^L$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

L: Linear scales only

SUBROUTINE BREAK

Name

`BREAK` – to force a break between separate sequences of point-by-point plotting.

Availability Section 1, released before version 2-5.

Arguments

None.

SUBROUTINE BRKNBX (X1, Y1, X2, Y2, LTYPE)

Name

`BRKNBX` – to draw a box, using a specified broken line pattern.

Availability Section 1, released version 2-8.

Arguments

`X1`, `Y1`, `X2`, `Y2`	`REAL` expressions	Coordinates of opposite corners of the box, specified in units of the plotting scales
`LTYPE`	`INTEGER` expression	Type of broken line pattern

SUBROUTINE BRKNPT (X, Y, LTYPE)

Name

`BRKNPT` – to draw a broken line to the point (x, y).

Availability Section 1, released version 2-9.

Arguments

`X`, `Y`	`REAL` expressions	Coordinates of a point specified in units of the plotting scale
`LTYPE`	`INTEGER` expression	Type of broken line pattern

SUBROUTINE BUNLPR (CHATTR)

Name

`BUNLPR` – to specify the order of priority of bundled line-drawing attributes.

Availability Section 1, released version 2-13.

Argument

`CHATTR`	`CHARACTER*3`	Attributes to be bundled

CHATTR	*Attribute*	*Cyclic order*
'C'	Colour	1, 2, 3, ... NPENS
'S'	Style	0, −1, −2 ... , 1, 2 ...
'T'	Thickness	1, 2, 3, 4

SUBROUTINE CHMASK (TORF)

Name

CHMASK – to specify whether text is to be masked.

Availability Section 4, released version 2-9.

Argument

TORF	LOGICAL expression	Whether text is to be masked

SUBROUTINE COORDS (IUNITS)

Name

COORDS – to change the interpretation of coordinates.

Availability Section 1, released version 2-12.

Argument

IUNITS	INTEGER expression	Type of units

IUNITS	*Coordinate interpretation*
0	Cartesian (x, y)
1	polar, $z = f(r, \theta)$, θ in degrees
2	polar, $z = f(r, \theta)$, θ in radians
3	polar – user-defined angular scale (see POLRNG)

SUBROUTINE CP7LB (X, Y, CAP)
SUBROUTINE CAPLB (X, Y, CAP, NCAP)

Name

CP7LB – to draw a caption at (x, y).

Availability Section 4, released version 2-6.

Arguments

X, Y	REAL expressions	Coordinates of a point, specified in units of the plotting scales
CAP	STRING expression	Caption
NCAP	INTEGER expression	Number of characters in CAP

SUBROUTINE CP7PT (X, Y, MKTYPE, CAP)
SUBROUTINE CAPPT (X, Y, MKTYPE, CAP, NCAP)

Name

CP7PT – to draw a marker symbol at (x, y) with a caption.

Availability Section 1, released before version 2-5.

Arguments

X, Y	REAL expressions	Coordinates of a point specified in units of the plotting scales
MKTYPE	INTEGER expression	Type of marker symbol
CAP	STRING expression	Caption
NCAP	INTEGER expression	Number of characters in CAP

SUBROUTINE CTBRKN (LTYPE)

Name

`CTBRKN` – to specify the broken line pattern used for contours and surface sections.

Availability Section 2, released before version 2-5.

Argument

`LTYPE`	`INTEGER` expression	Type of broken line pattern

SUBROUTINE CTCURV (ITYPE)

Name

`CTCURV` – to specify the curve type used for drawing contours.

Availability Section 2, released before version 2-5.

Argument

`ITYPE`	`INTEGER` expression	Method of curve drawing

`ITYPE`	*Curve type*
1	Tight fitting smooth curve
2	Looser fitting smooth curve
3	Straight lines from point to point

SUBROUTINE CTHOLD (ICODE)

Name

`CTHOLD` – to specify whether coordinates of internally generated curves are to be saved.

Availability Section plus, released version 2-13.

Argument

`ICODE`	`INTEGER` expression	Behaviour of subsequent curve drawing

`ICODE`	*Graphics output*	*Coordinates of contour*
1	Yes	Forgotten (default)
2	No	Stored
3	Yes	Stored

SUBROUTINE CTLABS (IFREQ)

Name

`CTLABS` – to specify the frequency of labels included with contour curves.

Availability Section 2, released version 2-6.

Argument

`IFREQ`	`INTEGER` expression	Frequency of labels

SUBROUTINE CTNUMB (TORF)

Name

`CTNUMB` – to specify whether contour curves are to be labelled.

Availability Section 2, released before version 2-5.

Argument

`TORF`	`LOGICAL` expression	Whether labelling is required

SUBROUTINE CVLBJS (VJUST, HJUST)

Name

`CVLBJS` – to specify the justification of the label drawn on a labelled curve.

Availability Section 4, released version 2-13.

Arguments

`VJUST`	`CHARACTER*1`	Vertical justification of label
`HJUST`	`CHARACTER*1`	Horizontal justification of label

`VJUST`	*Vertical justification*
`'D'`	Default
`'B'`	Bottom of the letters level with the curve
`'C'`	Centre of the letters level with the curve (default)
`'T'`	Top of the letters level with the curve

`HJUST`	*Horizontal justification*
`'D'`	Default
`'C'`	Label is centred on the reference point
`'L'`	Label starts at the reference point (default)
`'R'`	Label ends at the reference point

SUBROUTINE CVLBPS (IPOS)

Name

`CVLBPS` – to specify the reference point for the label drawn on a labelled curve.

Availability Section 4, released version 2-13.

Argument

`IPOS`	`INTEGER` expression	Position of label, given as an index of data arrays

SUBROUTINE DEFKEY (ITYPE, VCHAR, HCHAR, NROWS, NCOLMS)

Name

`DEFKEY` – to define an area for a key box.

Availability Section 1, released version 2-11.

Arguments

`ITYPE`	`INTEGER` expression	Type of reservation
`VCHAR`	`CHARACTER*1`	Vertical position of area
`HCHAR`	`CHARACTER*1`	Horizontal position of area
`NROWS`	`INTEGER` expression	Number of lines in area
`NCOLMS`	`INTEGER` expression	Maximum number of characters in any key caption

SUBROUTINE DEFKYW (NCHARS)

Name

`DEFKYW` – to specify the width of samples in a key box.

Availability Section 4, released version 2-12.

Argument

`NCHARS`	`INTEGER` expression	Width of sample in equivalent number of characters (or columns)

SUBROUTINE DIAGLV (ILEVEL)

Name

`DIAGLV` – to specify the level of diagnostics.

Availability Section plus, released version 2-9.

Argument

`ILEVEL`	`INTEGER` expression	Level of diagnostics

`ILEVEL`	*Description*	*Messages*
0	No messages output at all	None
1	Brief messages (default)	Type 1
2	Level 1 messages plus fuller details	Type 1+2
3	Level 1 messages plus subroutine trace	Type 1+3
4	Level 2 messages plus subroutine trace	Type 1+2+3

SUBROUTINE ENDPLT

Name

`ENDPLT` – to close the plotting device and SIMPLEPLOT at the end of plotting.

Availability Section 1, released before version 2-5.

Arguments

None.

SUBROUTINE EQSCAL (XSTART, XSTOP, YSTART, YSTOP, IUNITS)

Name

`EQSCAL` – to specify similar linear scales for Cartesian or polar plotting.

Availability Section 1, released version 2-10.

Arguments

`XSTART`	`REAL` expression	Value at start of x (or r) scale
`XSTOP`	`REAL` expression	Value at end x (or r) scale
`YSTART`	`REAL` expression	Value at start of y (or θ) scale
`YSTOP`	`REAL` expression	Value at end of y (or θ) scale
`IUNITS`	`INTEGER` expression	Type of units

IUNITS	*Units*	XSTART, XSTOP	YSTART, YSTOP
−1	Cartesian	*Values ignored, centimetres used*	
0	Cartesian	Horizontal units	Vertical units
		RSTART, RSTOP	THSTRT, THSTOP
1	Polar	Radial units	Angles in degrees
2	Polar	Radial units	Angles in radians
3	Polar	Radial units	User-defined scale

SUBROUTINE FIGFMT (CHFORM, NUMINT, NDEC)

Name

FIGFMT – to specify the format of REAL numbers drawn on pictures.

Availability Section 1, released version 2-11.

Arguments

CHFORM	CHARACTER*1	Type of format
NUMINT	INTEGER expression	Number of digits before point
NDEC	INTEGER expression	Number of decimal places

CHFORM	*Effect*
'F'	for fixed-point representation of all values
'E'	for floating-point representation of all values
'G'	'E' for very small or large values, 'F' otherwise

SUBROUTINE FIGSGN (SIGN, ESIGN)

Name

FIGSGN – to specify the sign conventions for positive numbers and exponents.

Availability Section 1, released version 2-11.

Arguments

SIGN	CHARACTER*1	Representation of sign of positive number
ESIGN	CHARACTER*1	Representation of sign of positive exponent

SIGN or ESIGN	*Description*	*Examples*	
'+'	Always include sign	+12.4	+0.123E+4
' '	Space before +ve values	12.4	0.123E 4
'D'	Default (no + sign)	12.4	0.124E4

SUBROUTINE FNAREA (XMIN, XMAX, YMIN, YMAX)

Name

FNAREA – to specify the x-y plotting ranges for 3-D functions, $z = f(x, y)$.

Availability Section 2, released before version 2-5.

Arguments

XMIN	REAL expression	Minimum x value included
XMAX	REAL expression	Maximum x value included
YMIN	REAL expression	Minimum y value included
YMAX	REAL expression	Maximum y value included

SUBROUTINE FNCNTS (FUNXY)

Name

FNCNTS – to draw a contour map from a 3-D function $z = f(x, y)$.

Availability Section 2, released before version 2-5.

Argument

FUNXY	function name	REAL function with two REAL arguments, also declared in an EXTERNAL statement

SUBROUTINE FNCONT (ZLEV, FUNXY)

Name

FNCONT – to draw a contour curve from a 3-D function $z = f(x, y)$.

Availability Section 2, released before version 2-5.

Arguments

ZLEV	REAL expression	Contour level
FUNXY	function name	REAL function with two REAL arguments, also declared in an EXTERNAL statement

SUBROUTINE FNCUT (X1, Y1, X2, Y2, FUNXY)

Name

FNCUT – to draw a 2-D curve of a surface section from a function $z = f(x, y)$.

Availability Section 2, released before version 2-5.

Arguments

X1, Y1, X2, Y2	REAL expressions	Coordinates of two points
FUNXY	function name	REAL function with two REAL arguments, also declared in an EXTERNAL statement

SUBROUTINE FNSHAD (ZLEV1, ZLEV2, ISHADE, FUNXY)

Name

FNSHAD – to draw the shaded area between two contour levels from a 3-D function $z = f(x, y)$.

Availability Section plus, released version 2-5.

Arguments

ZLEV1, ZLEV2	REAL expressions	Contour levels
ISHADE	INTEGER expression	Shading pattern number
FUNXY	function name	REAL function with two REAL arguments, also declared in an EXTERNAL statement

SUBROUTINE FNSHDS (FUNXY)

Name

`FNSHDS` – to draw a shaded contour map from a 3-D function $z = f(x, y)$.

Availability Section plus, released version 2-5.

Argument

`FUNXY`	function name	`REAL` function with two `REAL` arguments, also declared in an `EXTERNAL` statement

SUBROUTINE FNSURF (FUNXY)

Name

`FNSURF` – to start a new picture and draw a surface from a 3-D function $z = f(x, y)$.

Availability Section 2, released before version 2-5.

Argument

`FUNXY`	function name	`REAL` function with two `REAL` arguments, also declared in an `EXTERNAL` statement

SUBROUTINE GROUP (NHORIZ, NVERT)

Name

`GROUP` – to specify how pictures are to be grouped on the SIMPLEPLOT page.

Availability Section 1, released before version 2-5.

Arguments

`NHORIZ`	`INTEGER` expression	Number of pictures to be placed horizontally
`NVERT`	`INTEGER` expression	Number of pictures to be placed vertically

SUBROUTINE ISANG (ANGLE)

Name

`ISANG` – to specify the angle at which surface pictures are drawn.

Availability Section 2, released before version 2-5.

Argument

`ANGLE`	`REAL` expression	Angle in degrees

SUBROUTINE ISAXD7 (CAPX, CAPY, CAPZ)
SUBROUTINE ISAXDR (CAPX, NCAPX, CAPY, NCAPY, CAPZ, NCAPZ)

Name

`ISAXD7` – to draw 3-D axes on an isometric picture.

Availability Section plus, released version 2-9.

Arguments

`CAPX`	`STRING` expression	x-axis caption
`CAPY`	`STRING` expression	y-axis caption
`CAPZ`	`STRING` expression	z-axis caption
`NCAPX`	`INTEGER` expression	Number of characters in `CAPX`
`NCAPY`	`INTEGER` expression	Number of characters in `CAPY`
`NCAPZ`	`INTEGER` expression	Number of characters in `CAPZ`

SUBROUTINE ISAXES (TORF)

Name

`ISAXES` – to specify whether surface pictures are to include 3-D axes.

Availability Section plus, released version 2-9.

Argument

`TORF`	`LOGICAL` expression	Whether axes required

SUBROUTINE ISCURV (NSTEPS)

Name

`ISCURV` – to specify the smoothness of curves on crosshatched and cascade surface pictures.

Availability Section 2, released before version 2-5.

Argument

`NSTEPS`	`INTEGER` expression	Number of steps

SUBROUTINE ISDIAG (TORF)

Name

`ISDIAG` – to specify whether surface pictures are to include an x-y-z diagram.

Availability Section 2, released version 2-6.

Argument

`TORF`	`LOGICAL` expression	Whether a diagram is required

SUBROUTINE ISFULL (TORF)

Name

`ISFULL` – to specify whether surface pictures fill the space available.

Availability Section 2, released version 2-7.

Argument

`TORF`	`LOGICAL` expression	Whether to fill the space

SUBROUTINE ISMESH (MXY)

Name

`ISMESH` – to specify the fineness of detail in surface pictures.

Availability Section 2, released before version 2-5.

Argument

`MXY`	`INTEGER` expression	Total number of mesh lines used in x and y

SUBROUTINE ISNEW

Name

ISNEW – to start a null isometric picture.

Availability Section 2, released before version 2-5.

Arguments

None.

SUBROUTINE ISRISE (FACTOR)

Name

ISRISE – to specify the proportions of surface pictures.

Availability Section 2, released before version 2-5.

Argument

FACTOR	REAL expression	Specifies the maximum height of the surface above the base plane

SUBROUTINE ISTYPE (ITYPE)

Name

ISTYPE – to specify the type of surface picture.

Availability Section 2, released before version 2-5.

Argument

ITYPE	INTEGER expression	Type of picture

ITYPE	*Type of surface picture*
0	No lines drawn
1	Surface outline (default)
2	Surface outline and a set of contours
3	Crosshatched picture
4	Set of curves of z against x for a set of equally-spaced y values
5	Set of curves of z against y for a set of equally-spaced x values

SUBROUTINE ISVIEW (ICORNR)

Name

ISVIEW – to specify the view point for surface pictures.

Availability Section 2, released before version 2-5.

Argument

ICORNR	INTEGER expression	Viewing corner

SUBROUTINE ISYUP (TORF)

Name

ISYUP – to specify the direction of change of y on surface pictures.

Availability Section 2, released before version 2-5.

Argument

TORF	LOGICAL expression	Whether y increases upwards

SUBROUTINE KISXY (X3, Y3, Z3, X2, Y2)

Name

`KISXY` – to convert a point (x, y, z) on a surface picture to its 2-D equivalent.

Availability Section 2, released version 2-11.

Arguments

`X3`, `Y3`, `Z3`	`REAL` expressions	Coordinates of a point, specified in 3-D picture units
`X2`, `Y2`	`REAL` variables	To receive the coordinates of a point specified in internal 2-D picture units

SUBROUTINE KNUMB (IVAL, STR)

Name

`KNUMB` – to convert an `INTEGER` value to the equivalent text string.

Availability Section 1, released version 2-11.

Arguments

`IVAL`	`INTEGER` expression	Value to be converted
`STR`	`STRING` variable	To receive text string

SUBROUTINE KREAL (RVAL, STR)

Name

`KREAL` – to convert a `REAL` value to the equivalent text string.

Availability Section 1, released version 2-11.

Arguments

`RVAL`	`REAL` expression	Value to be converted
`STR`	`STRING` variable	To receive text string

SUBROUTINE KSCALE (START, STOP, DIV, VSTART, VSTOP, VDIV)

Name

`KSCALE` – to convert scale limits such that they span whole subdivisions.

Availability Section 1, released version 2-12.

Arguments

`START`	`REAL` expression	Value required near beginning of scale
`STOP`	`REAL` expression	Value required near end of scale
`DIV`	`REAL` expression	Subdivision required
`VSTART`	`REAL` variable	To receive value to use at beginning of scale
`VSTOP`	`REAL` variable	To receive value to use at end of scale
`VDIV`	`REAL` variable	To receive subdivision value to use

SUBROUTINE KWZVAL (ICURVE, ZVAL, RVAL)

Name

`KWZVAL` – to convert a z value on a waterfall chart to its equivalent in terms of the label scale.

Availability Section plus, released version 2-13.

Arguments

`ICURVE`	`INTEGER` expression	Number of relevant curve
`ZVAL`	`REAL` expression	Coordinate of a point specified in data units
`RVAL`	`REAL` variable	Equivalent value in units of the label scale

SUBROUTINE KZZRG (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS, Z2ARR, NX, NY)

Name

`KZZRG` – to generate a regular grid of values from ungridded (x, y, z) coordinates.

Availability Section 2, released version 2-11.

Arguments

`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values in parallel arrays
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements
`Z2ARR`	`REAL` 2-D array	To receive values over a regular grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`

SUBROUTINE LABCV7 (XARR, YARR, NARR, LTYPE, CAP)
SUBROUTINE LABCV (XARR, YARR, NARR, LTYPE, CAP, NCAP)

Name

`LABCV7` – to draw a labelled curve.

Availability Section 4, released version 2-13.

Arguments

`XARR`, `YARR`	`REAL` arrays	Coordinates of points, in units of the plotting scales
`NARR`	`INTEGER` expression	Number of points
`LTYPE`	`INTEGER` expression	Type of broken line pattern
`CAP`	`STRING` expression	Label for curve
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

SUBROUTINE LABJST (VJUST, HJUST)

Name

`LABJST` – to specify the justification of labels drawn by `CP7LB`, `CP7LBM`, `CP7XC` or `CP7YC`.

Availability Section 4, released version 2-6.

Arguments

`VJUST`	`CHARACTER*1`	Vertical justification of label
`HJUST`	`CHARACTER*1`	Horizontal justification of label

`VJUST`	*Vertical justification*
`'D'`	Default
`'T'`	Top of letters
`'C'`	Halfway between `'T'` and `'B'`
`'B'`	Bottom of letters (not including descenders)

`HJUST`	*Horizontal justification*
`'D'`	Default
`'L'`	At the left (beginning) of the label
`'C'`	Halfway between `'L'` and `'R'`
`'R'`	At the right (end) of the label
`'P'`	Preceding the label
`'F'`	Following the label

SUBROUTINE LIMEXC (DARR, NARR, VARMIN, VARMAX)

Name

`LIMEXC` – to find the minimum and maximum values in a `REAL` array.

Availability Section 1, released before version 2-5.

Arguments

`DARR`	`REAL` array	Data values
`NARR`	`INTEGER` expression	Number of elements of `DARR` to be examined
`VARMIN`	`REAL` variable	To receive minimum value
`VARMAX`	`REAL` variable	To receive maximum value

SUBROUTINE LIMIDX (DARR, NARR, IMIN, IMAX)

Name

`LIMIDX` – to find the position of each of the minimum and maximum values of a `REAL` array.

Availability Section 1, released version 2-13.

Arguments

`DARR`	`REAL` array	Data values
`NARR`	`INTEGER` expression	Number of elements in `DARR` to be examined
`IMIN`	`INTEGER` variable	Position of minimum value
`IMAX`	`INTEGER` variable	Position of maximum value

SUBROUTINE LIMSFN (FUNXY, VARMIN, VARMAX)

Name

`LIMSFN` – to find the range of function values to be used for contours of a 3-D function $z = f(x, y)$.

Availability Section 2, released version 2-11.

Arguments

`FUNXY`	function name	`REAL` function with two `REAL` arguments, also declared in an `EXTERNAL` statement
`VARMIN`, `VARMAX`	`REAL` variables	To receive values of limits

SUBROUTINE LINEK7 (LTYPE, CAP)
SUBROUTINE LINEKY (LTYPE, CAP, NCAP)

Name

`LINEK7` – to draw an annotated sample of a broken line pattern in a key.

Availability Section 1, released before version 2-5.

Arguments

`LTYPE`	`INTEGER` expression	Type of broken line pattern
`CAP`	`STRING` expression	Caption
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

SUBROUTINE MARKPT (X, Y, MKTYPE)

Name

`MARKPT` – to draw a marker symbol at a specified point.

Availability Section 1, released before version 2-5.

Arguments

`X`, `Y`	`REAL` expressions	Coordinates of a point specified in units of the plotting scales
`MKTYPE`	`INTEGER` expression	Type of marker symbol

SUBROUTINE MPK7H (VCHAR, HCHAR, ZLEFT, ZRIGHT, CAP)
SUBROUTINE MPKYH (VCHAR, HCHAR, ZLEFT, ZRIGHT, CAP, NCAP)

Name

`MPK7H` – to draw a complete horizontal key to a shaded contour map.

Availability Section plus, released version 2-10.

Arguments

`VCHAR`	`CHARACTER*1`	Vertical position of key
`HCHAR`	`CHARACTER*1`	Horizontal position of key
`ZLEFT`	`REAL` expression	z value at left of key
`ZRIGHT`	`REAL` expression	z value at right of key
`CAP`	`STRING` expression	Single caption for key
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

SUBROUTINE MPK7V (VCHAR, HCHAR, ZTOP, ZBOTTM, CAP)
SUBROUTINE MPKYV (VCHAR, HCHAR, ZTOP, ZBOTTM, CAP, NCAP)

Name

MPK7V – to draw a complete vertical key to a shaded contour map.

Availability Section plus, released version 2-10.

Arguments

VCHAR	CHARACTER*1	Vertical position of key
HCHAR	CHARACTER*1	Horizontal position of key
ZTOP	REAL expression	z value at top of key
ZBOTTM	REAL expression	z value at bottom of key
CAP	STRING expression	Single caption for key
NCAP	INTEGER expression	Number of characters in CAP

SUBROUTINE NEWPIC

Name

NEWPIC – to start a new 2-D picture without drawing an axis framework.

Availability Section 1, released before version 2-5.

Arguments

None.

SUBROUTINE NODATA (RVAL)

Name

NODATA – to specify the REAL value to be used to represent no-data values.

Availability Section 1, released version 2-10.

Argument

RVAL	REAL expression	No-data value

SUBROUTINE PEN (IPEN)

Name

PEN – to select the pen to be used for all plotting.

Availability Section 1, released before version 2-5.

Argument

IPEN	INTEGER expression	Pen number

IPEN	*Pen usage*
-1	plotting is omitted on *all* devices
0	plotting is done in background colour; on some devices this produces the effect of rubbing out but on others (*eg.* pen plotters) plotting is omitted
1, 2, 3 ...	pens/bundles are selected as determined by the current device, palette (see PENHLS and PENRGB) and the current bundled attributes (see BUNLPR)

SUBROUTINE PERIM

Name

`PERIM` – to draw a rectangular box around the current picture.

Availability Section 1, released version 2-5.

Arguments

None.

SUBROUTINE POLAR7 (RADIUS, CAP)
SUBROUTINE POLAR (RADIUS, CAP, NCAP)

Name

`POLAR7` – to start a new polar picture and draw axis framework.

Availability Section 1, released before version 2-5.

Arguments

`RADIUS`	`REAL` expression	Maximum value of radial scale
`CAP`	`STRING` expression	Caption for radial axis
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

SUBROUTINE QCURVE (MAXPTS, XARR, YARR, NPTS)

Name

`QCURVE` – to return the coordinates of a stored curve.

Availability Section plus, released version 2-13.

Arguments

`MAXPTS`	`INTEGER` expression	Maximum points to receive per call
`XARR`, `YARR`	`REAL` arrays	Dimensioned at least `MAXPTS` to receive coordinates of points
`NPTS`	`INTEGER` variable	To receive number of points

SUBROUTINE QKYCAP (LABARR, NARR, ITYPE, NROWS, NCOLMS)

Name

`QKYCAP` – to inquire the size of a key or caption area to hold a set of captions.

Availability Section plus, released version 2-11.

Arguments

`LABARR`	`STRING` array	Set of labels
`NARR`	`INTEGER` expression	Number of labels in `LABARR`
`ITYPE`	`INTEGER` expression	Intended use of text strings
`NROWS`	`INTEGER` variable	To receive number of rows
`NCOLMS`	`INTEGER` variable	To receive nominal number of columns

`ITYPE`	*Text usage*
1	Keys or captions relating to pictures
2	Keys or captions relating to groups/pages

SUBROUTINE QNODAT (VARW)

Name

`QNODAT` – to inquire the current value representing no-data.

Availability Section 1, released version 2-10.

Argument

`VARW`	`REAL` variable	To receive no-data value

SUBROUTINE QSFLAB (VARMIN, VARMAX, VARSTP)

Name

`QSFLAB` – to inquire of the range of data and contour interval on latest 3-D plot.

Availability Section 2, released version 2-11.

Arguments

`VARMIN`	`REAL` variable	To receive first z value
`VARMAX`	`REAL` variable	To receive last z value
`VARSTP`	`REAL` variable	To receive contour interval

SUBROUTINE QWZSCL (ZSTART, ZSTOP)

Name

`QWZSCL` – to inquire the current z scales for curves on waterfall charts.

Availability Section plus, released version 2-13.

Arguments

`ZSTART`, `ZSTOP`	`REAL` variables	To receive range of z covered

SUBROUTINE RANGE (X1, Y1, X2, Y2)

Name

`RANGE` – to draw a line indicating a range of values.

Availability Section 1, released before version 2-5.

Arguments

`X1`, `Y1`	`REAL` expressions	Coordinates of the first point
`X2`, `Y2`	`REAL` expressions	Coordinates of the second point

SUBROUTINE RGCNTS (Z2ARR, NX, NY)

Name

`RGCNTS` – to draw a contour map from 3-D data on a regular grid.

Availability Section 2, released before version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`

SUBROUTINE RGCONT (ZLEV, Z2ARR, NX, NY)

Name

`RGCONT` – to draw a contour curve from 3-D data on a regular grid.

Availability Section 2, released before version 2-5.

Arguments

`ZLEV`	`REAL` expression	Contour level
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`

SUBROUTINE RGCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY)

Name

`RGCUT` – to draw a 2-D curve of a surface section from 3-D data on a regular grid.

Availability Section 2, released before version 2-5.

Arguments

`X1`, `Y1`, `X2`, `Y2`	`REAL` expressions	Coordinates of two points
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`

SUBROUTINE RGSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY)

Name

`RGSHAD` – to shade the area between two contour levels from 3-D data on a regular grid.

Availability Section plus, released version 2-5.

Arguments

`ZLEV1`, `ZLEV2`	`REAL` expressions	Contour levels
`ISHADE`	`INTEGER` expression	Shading pattern number
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`

SUBROUTINE RGSHDS (Z2ARR, NX, NY)

Name

`RGSHDS` – to draw a shaded contour map from 3-D data on a regular grid.

Availability Section plus, released version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`

SUBROUTINE RGSURF (Z2ARR, NX, NY)

Name

`RGSURF` – to start a new picture and draw a surface from 3-D data on a regular grid.

Availability Section 2, released before version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`

SUBROUTINE SCALES (XSTART, XSTOP, IXTYPE, YSTART, YSTOP, IYTYPE)

Name

`SCALES` – to specify Cartesian scales for all 2-D plotting.

Availability Section 1, released before version 2-5.

Arguments

`XSTART`	`REAL` expression	x scale value at left edge
`XSTOP`	`REAL` expression	x scale value at right edge
`IXTYPE`	`INTEGER` expression	Type of horizontal scale
`YSTART`	`REAL` expression	y scale value at bottom edge
`YSTOP`	`REAL` expression	y scale value at top edge
`IYTYPE`	`INTEGER` expression	Type of vertical scale

`IXTYPE` or `IYTYPE`	*Type of scale*
0	Centimetre
1	Linear
2	Logarithmic
3	Normal probability (%)

SUBROUTINE SETPNS (IPEN1, IPEN2, IPEN3, IPEN4)

Name

`SETPNS` – to specify the pens associated with the four pen pointers.

Availability Section 1, released before version 2-5.

Arguments

`IPEN1`	`INTEGER` expression	Pen associated with pen pointer [1]
`IPEN2`	`INTEGER` expression	Pen associated with [2]
`IPEN3`	`INTEGER` expression	Pen associated with [3]
`IPEN4`	`INTEGER` expression	Pen associated with [4]

`IPEN`i	*Pen usage*
-1	plotting is omitted on all devices
0	plotting is done in background colour; this has no effect on pen plotters
1, 2, 3 ...	pens/bundles are selected as determined by the current device, palette and bundled attributes

SUBROUTINE SFEQX (XSTART, XSTEP)

Name

SFEQX – to specify the equally-spaced x values to be associated with gridded 3-D data.

Availability Section 2, released before version 2-5.

Arguments

XSTART	REAL expression	x value of first data column
XSTEP	REAL expression	x interval between data columns

SUBROUTINE SFEQY (YSTART, YSTEP)

Name

SFEQY – to specify the equally-spaced y values to be associated with gridded 3-D data.

Availability Section 2, released before version 2-5.

Arguments

YSTART	REAL expression	y value of first data row
YSTEP	REAL expression	y interval between data rows

SUBROUTINE SFEQZ (ZSTART, ZSTEP)

Name

SFEQZ – to specify the equal spacing of contours.

Availability Section 2, released before version 2-5.

Arguments

ZSTART	REAL expression	Offset value for contour levels
ZSTEP	REAL expression	Interval between contour levels

SUBROUTINE SFEQZD (NSTEPS, DELTA)

Name

SFEQZD – to specify the equal spacing of contours for discrete data.

Availability Section plus, released version 2-11.

Arguments

NSTEPS	INTEGER expression	Number of contour intervals
DELTA	REAL expression	Minimum interval between discrete data values

SUBROUTINE SFLAB

Name

SFLAB – to annotate the current 3-D picture with the range of values displayed.

Availability Section 2, released before version 2-5.

Arguments

None.

SUBROUTINE SFLIMS (Z1, Z2)

Name

`SFLIMS` – to specify the z plotting range.

Availability Section 2, released before version 2-5.

Arguments

`Z1`	`REAL` expression	z value at start of z range
`Z2`	`REAL` expression	z value at end of z range

SUBROUTINE SFMESH (MX, MY)

Name

`SFMESH` – to specify a mesh to be used for constructing contours and surfaces.

Availability Section plus, released version 2-11.

Arguments

`MX`	`INTEGER` expression	Number of x mesh lines used
`MY`	`INTEGER` expression	Number of y mesh lines used

SUBROUTINE SFZSCL (ZSTART, ZSTOP)

Name

`SFZSCL` – to specify the z scale of surface pictures and contour maps independently of the data set.

Availability Section 2, released version 2-12.

Arguments

`ZSTART`	`REAL` expression	Start of z scale
`ZSTOP`	`REAL` expression	End of z scale

SUBROUTINE SHPATT (ISHADE, IPOS)

Name

`SHPATT` – to specify one of a sequence of shading patterns.

Availability Section 4, released version 2-5.

Arguments

`ISHADE`	`INTEGER` expression	Pattern number
`IPOS`	`INTEGER` expression	Indicating which of the sequence is to be set (1–32)

`ISHADE`	*Shading pattern*
−1	an empty area
0	solid fill with background colour
1, 2, 3...	hardware/software patterns

SUBROUTINE SQBRKN (IARR, NARR)

Name

`SQBRKN` – to specify a sequence of broken line patterns.

Availability Section plus, released version 2-12.

Arguments

`IARR`	`INTEGER` array	Pattern numbers for each contour line
`NARR`	`INTEGER` expression	Number of elements in `IARR` (1–32)

SUBROUTINE SQPEN (IARR, NARR)

Name

`SQPEN` – to specify a sequence of pens/bundles.

Availability Section plus, released version 2-12.

Arguments

`IARR`	`INTEGER` array	Pen numbers for each contour
`NARR`	`INTEGER` expression	Number of elements in `IARR` (1–32)

`IARR(`*i*`)`	*Pen usage*
−1	plotting is omitted on all devices
0	plotting is done in background colour; on some devices this produces the effect of rubbing out but on others (*eg.* pen plotters) plotting is omitted
1, 2, 3 ...	pens/bundles are selected as determined by the current device, palette (see `PENHLS` and `PENRGB`) and the current bundled attributes (see `BUNLPR`)

SUBROUTINE SQSHAD (IARR, NARR)

Name

`SQSHAD` – to specify a sequence of shading patterns.

Availability Section 4, released version 2-12.

Arguments

`IARR`	`INTEGER` array	Pattern numbers for each shaded area
`NARR`	`INTEGER` expression	Number of elements in `IARR` (1–32)

`IARR(`*i*`)`	*Shading pattern*
−1	an empty area
0	solid fill with background colour
1, 2, 3 ...	hardware/software patterns

SUBROUTINE SQZLAB (LABARR, NARR)

Name

`SQZLAB` – to specify a sequence of contour labels.

Availability Section plus, released version 2-12.

Arguments

`LABARR`	`STRING` array	Contour labels
`NARR`	`INTEGER` expression	Number of elements in `LABARR` (1–32)

SUBROUTINE SQZVAL (DARR, NARR)

Name

`SQZVAL` – to specify a sequence of contour levels.

Availability Section plus, released version 2-12.

Arguments

`DARR`	`REAL` array	Contour levels
`NARR`	`INTEGER` expression	Number of elements in `DARR` (1–32)

SUBROUTINE TITLE7 (VCHAR, HCHAR, CAP)
SUBROUTINE TITLE (VCHAR, HCHAR, CAP, NCAP)

Name

`TITLE7` – to draw a text string as a title to the picture, group or page.

Availability Section 1, released before version 2-5.

Arguments

`VCHAR`	`CHARACTER*1`	Vertical position of title
`HCHAR`	`CHARACTER*1`	Horizontal position of title
`CAP`	`STRING` expression	Caption
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

SUBROUTINE WFCHT (Z2ARR, NPTS, NCURVS, CAPN, CAPL)

Name

`WFCHT` – to draw a waterfall chart on the current picture.

Availability Section plus, released version 2-13.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NPTS`	`INTEGER` expression	Number of columns in `Z2ARR`
`NCURVS`	`INTEGER` expression	Number of rows in `Z2ARR`
`CAPN`	`STRING` expression	Numeric axis caption
`CAPL`	`STRING` expression	Label axis caption

SUBROUTINE WFDRAW (ZARR, NPTS)

Name

`WFDRAW` – to draw a single curve on a waterfall chart.

Availability Section plus, released version 2-13.

Arguments

`ZARR`	`REAL` array	Data values of curve
`NPTS`	`INTEGER` expression	Number of points in `ZARR`

SUBROUTINE WFEQL (CSTART, CSTEP)

Name

`WFEQL` – to specify the equally-spaced values on the label scale of a waterfall chart.

Availability Section plus, released version 2-13.

Arguments

`CSTART`	`REAL` expression	Minimum scale value
`CSTEP`	`REAL` expression	Interval between curves

SUBROUTINE WFEQN (ESTART, ESTEP)

Name

`WFEQN` – to specify the equally-spaced values on the numeric scale of a waterfall chart.

Availability Section plus, released 2-13.

Arguments

`ESTART`	`REAL` expression	Value of first data column
`ESTEP`	`REAL` expression	Interval between data columns

SUBROUTINE WFINIT

Name

`WFINIT` – to reset all defaults for waterfall charts.

Availability Section plus, released version 2-13.

Arguments

None.

SUBROUTINE WFNCVS (NCURVS)

Name

`WFNCVS` – to specify the number of curves to be accommodated on waterfall charts.

Availability Section plus, released version 2-13.

Argument

`NCURVS`	`INTEGER` expression	Total number of waterfall curves

SUBROUTINE WFNSCL (START, STOP)

Name

`WFNSCL` – to specify the numeric scale for waterfall charts.

Availability Section plus, released version 2-13.

Arguments

`START`	`REAL` expression	Value at start of numeric scale
`STOP`	`REAL` expression	Value at end of numeric scale

SUBROUTINE WFPNS (IPEN1, IPEN2, IPEN3, IPEN4)

Name

`WFPNS` – to specify the pens used in waterfall charts.

Availability Section plus, released version 2-13.

Arguments

`IPEN1`	`INTEGER` expression	Pen associated with [1]
`IPEN2`	`INTEGER` expression	Pen associated with [2]
`IPEN3`	`INTEGER` expression	Pen associated with [3]
`IPEN4`	`INTEGER` expression	Pen associated with [4]

Pointer	*Pen usage*
[1]	Unmasked part of curve with $z \geq$ `ZLEVEL`
[2]	Masked part of curve with $z \geq$ `ZLEVEL`
[3]	Unmasked part of curve with $z <$ `ZLEVEL`
[4]	Masked part of curve with $z <$ `ZLEVEL`

`IPEN`i	*Pen usage*
-1	plotting is omitted on all devices
0	drawn in background colour
1, 2, 3 ...	pens/bundles as specified by `PENHLS`/`PENRGB` and `BUNLPR`

SUBROUTINE WFSTEP (ZDISP)

Name

`WFSTEP` – to specify the displacement between waterfall curves.

Availability Section plus, released version 2-13.

Argument

`ZDISP`	`REAL` expression	Displacement between curves relative to the span of a curve

SUBROUTINE WFZLEV (ZLEVEL)

Name

`WFZLEV` – to specify the data value at which waterfall curves can change pens.

Availability Section plus, released version 2-13.

Argument

`ZLEVEL`	`REAL` expression	Data value for pen change

SUBROUTINE WFZSCL (ZSTART, ZSTOP)

Name

`WFZSCL` – to specify the z scales for curves on subsequent waterfall charts.

Availability Section plus, released version 2-13.

Arguments

`ZSTART`, `ZSTOP`	`REAL` expressions	Range of z covered

REAL FUNCTION XCALC (X, Y, Z2ARR, NX, NY, XARR)

Name

XCALC – to interpolate $z = f(x, y)$ from 3-D data on an x-specified grid.

Availability Section 2, released before version 2-5.

Arguments

X, Y	REAL expressions	Coordinates of a point
Z2ARR	REAL 2-D array	Data values over a grid
NX, NY	INTEGER expressions	Number of columns and rows in Z2ARR
XARR	REAL array	NX x values of the grid

SUBROUTINE XCNTS (Z2ARR, NX, NY, XARR)

Name

XCNTS – to draw a contour map from 3-D data on an x-specified grid.

Availability Section 2, released before version 2-5.

Arguments

Z2ARR	REAL 2-D array	Data values over a grid
NX, NY	INTEGER expressions	Number of columns and rows in Z2ARR
XARR	REAL array	NX x values of the grid

SUBROUTINE XCONT (ZLEV, Z2ARR, NX, NY, XARR)

Name

XCONT – to draw a contour curve from 3-D data on an x-specified grid.

Availability Section 2, released before version 2-5.

Arguments

ZLEV	REAL expression	Contour level
Z2ARR	REAL 2-D array	Data values over a grid
NX, NY	INTEGER expressions	Number of columns and rows in Z2ARR
XARR	REAL array	NX x values of the grid

SUBROUTINE XCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY, XARR)

Name

XCUT – to draw a 2-D curve of a surface section from 3-D data on an x-specified grid.

Availability Section 2, released before version 2-5.

Arguments

X1, Y1, X2, Y2	REAL expressions	Coordinates of two points
Z2ARR	REAL 2-D array	Data values over a grid
NX, NY	INTEGER expressions	Number of columns and rows in Z2ARR
XARR	REAL array	NX x values of the grid

SUBROUTINE XSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY, XARR)

Name

`XSHAD` – to shade the area between two contour levels from 3-D data on an x-specified grid.

Availability Section plus, released version 2-5.

Arguments

`ZLEV1`, `ZLEV2`	`REAL` expressions	Contour levels
`ISHADE`	`INTEGER` expression	Shading pattern number
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid

SUBROUTINE XSHDS (Z2ARR, NX, NY, XARR)

Name

`XSHDS` – to draw a shaded contour map from 3-D data on an x-specified grid.

Availability Section plus, released version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid

SUBROUTINE XSURF (Z2ARR, NX, NY, XARR)

Name

`XSURF` – to start a new picture and draw a surface from 3-D data on an x-specified grid.

Availability Section 2, released before version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid

REAL FUNCTION XYCALC (X, Y, Z2ARR, NX, NY, XARR, YARR)

Name

`XYCALC` – to interpolate $z = f(x, y)$ from 3-D data on an x-y specified grid.

Availability Section 2, released before version 2-5.

Arguments

`X`, `Y`	`REAL` expressions	Coordinates of a point
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE XYCNTS (Z2ARR, NX, NY, XARR, YARR)

Name

`XYCNTS` – to draw a contour map from 3-D data on an x-y specified grid.

Availability Section 2, released before version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE XYCONT (ZLEV, Z2ARR, NX, NY, XARR, YARR)

Name

`XYCONT` – to draw a contour curve from 3-D data on an x-y specified grid.

Availability Section 2, released before version 2-5.

Arguments

`ZLEV`	`REAL` expression	Contour level
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE XYCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY, XARR, YARR)

Name

`XYCUT` – to draw a 2-D curve of a surface section from 3-D data on an x-y specified grid.

Availability Section 2, released before version 2-5.

Arguments

`X1`, `Y1`, `X2`, `Y2`	`REAL` expressions	Coordinates of two points
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE XYSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY, XARR, YARR)

Name

`XYSHAD` – to shade the area between two contour levels from 3-D data on an x-y specified grid.

Availability Section plus, released version 2-5.

Arguments

`ZLEV1`, `ZLEV2`	`REAL` expressions	Contour levels
`ISHADE`	`INTEGER` expression	Shading pattern number
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE XYSHDS (Z2ARR, NX, NY, XARR, YARR)

Name

`XYSHDS` – to draw a shaded contour map from 3-D data on an x-y specified grid.

Availability Section plus, released version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE XYSURF (Z2ARR, NX, NY, XARR, YARR)

Name

`XYSURF` – to start a new picture and draw a surface from 3-D data on an x-y specified grid.

Availability Section 2, released before version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`XARR`	`REAL` array	`NX` x values of the grid
`YARR`	`REAL` array	`NY` y values of the grid

REAL FUNCTION YCALC (X, Y, Z2ARR, NX, NY, YARR)

Name

`YCALC` – to interpolate $z = f(x, y)$ from 3-D data on a y-specified grid.

Availability Section 2, released before version 2-5.

Arguments

`X`, `Y`	`REAL` expressions	Coordinates of a point
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE YCNTS (Z2ARR, NX, NY, YARR)

Name

`YCNTS` – to draw a contour map from 3-D data on a y-specified grid.

Availability Section 2, released before version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE YCONT (ZLEV, Z2ARR, NX, NY, YARR)

Name

`YCONT` – to draw a contour curve from 3-D data on a y-specified grid.

Availability Section 2, released before version 2-5.

Arguments

`ZLEV`	`REAL` expression	Contour level
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE YCUT (X1, Y1, X2, Y2, Z2ARR, NX, NY, YARR)

Name

`YCUT` – to draw a 2-D curve of a surface section from 3-D data on a y-specified grid.

Availability Section 2, released before version 2-5.

Arguments

`X1`, `Y1`, `X2`, `Y2`	`REAL` expressions	Coordinates of two points
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE YSHAD (ZLEV1, ZLEV2, ISHADE, Z2ARR, NX, NY, YARR)

Name

`YSHAD` – to shade the area between two contour levels from 3-D data on a y-specified grid.

Availability Section plus, released version 2-5.

Arguments

`ZLEV1`, `ZLEV2`	`REAL` expressions	Contour levels
`ISHADE`	`INTEGER` expression	Shading pattern number
`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE YSHDS (Z2ARR, NX, NY, YARR)

Name

`YSHDS` – to draw a shaded contour map from 3-D data on a y-specified grid.

Availability Section plus, released version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE YSURF (Z2ARR, NX, NY, YARR)

Name

`YSURF` – to start a new picture and draw a surface from 3-D data on a y-specified grid.

Availability Section 2, released before version 2-5.

Arguments

`Z2ARR`	`REAL` 2-D array	Data values over a grid
`NX`, `NY`	`INTEGER` expressions	Number of columns and rows in `Z2ARR`
`YARR`	`REAL` array	`NY` y values of the grid

SUBROUTINE ZCUT (X1, Y1, X2, Y2, XARR, YARR, ZARR, NPTS, I2ARR, NNODES, NELEMS)

Name

`ZCUT` – to draw a 2-D curve of a surface section from ungridded 3-D data (without neighbours).

Availability Section 2, released before version 2-5.

Arguments

`X1`, `Y1`, `X2`, `Y2`	`REAL` expressions	Coordinates of two points
`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values in parallel arrays
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZELEM (JELE, XARR, YARR, NPTS, I2ARR, NNODES, NELEMS)

Name

`ZELEM` – to draw a single area element.

Availability Section 2, released before version 2-5.

Arguments

`JELE`	`INTEGER` expression	Element number to be drawn
`XARR`, `YARR`	`REAL` arrays	(x, y) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZNEIGH (I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZNEIGH` – to set up an array of neighbours from an array of elements.

Availability Section 2, released version 2-5.

Arguments

`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	To receive element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZNUMB (NTORF, ETORF)

Name

`ZNUMB` – to specify whether nodes and/or elements are to be numbered.

Availability Section 2, released before version 2-5.

Arguments

`NTORF`	`LOGICAL` expression	Whether to number nodes
`ETORF`	`LOGICAL` expression	Whether to number elements

REAL FUNCTION ZZCALC (X, Y, XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZCALC` – to interpolate $z = f(x, y)$ from ungridded 3-D data (with neighbours).

Availability Section 2, released version 2-5.

Arguments

`X`, `Y`	`REAL` expressions	Coordinates of a point
`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZZCNTS (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZCNTS` – to draw contours from ungridded 3-D data (with neighbours).

Availability Section 2, released version 2-5.

Arguments

`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZZCONT (ZLEV, XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZCONT` – to draw a contour curve from ungridded 3-D data (with neighbours).

Availability Section 2, released version 2-5.

Arguments

`ZLEV`	`REAL` expression	Contour level
`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZZEDGE (XARR, YARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZEDGE` – to draw around the boundary of the data area covered by a set of elements (with neighbours).

Availability Section 2, released version 2-5.

Arguments

`XARR`, `YARR`	`REAL` arrays	(x, y) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZZELMS (XARR, YARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZELMS` – to draw the configuration of a set of area elements (with neighbours).

Availability Section 2, released version 2-5.

Arguments

`XARR`, `YARR`	`REAL` arrays	(x, y) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZZORDN (XARR, YARR, NPTS, I2ARR, N2ARR, NVAR, ISIZE)

Name

`ZZORDN` – to reconfigure (x, y) coordinates into triangular elements and an array of neighbours (using normalized values).

Availability Section 2, released version 2-9.

Arguments

`XARR`, `YARR`	`REAL` arrays	(x, y) coordinates of data values in parallel arrays
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	To receive data element structure, `I2ARR(3,ISIZE)`
`N2ARR`	`INTEGER` 2-D array	To receive neighbour array, `N2ARR(3,ISIZE)`
`NVAR`	`INTEGER` variable	To receive number of elements
`ISIZE`	`INTEGER` expression	second dimension of `I2ARR` and `N2ARR`

SUBROUTINE ZZORDR (XARR, YARR, NPTS, I2ARR, N2ARR, NVAR, ISIZE)

Name

`ZZORDR` – to reconfigure (x, y) coordinates into triangular elements and an array of neighbours.

Availability Section 2, released version 2-5.

Arguments

`XARR`, `YARR`	`REAL` arrays	(x, y) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	To receive data element structure, `I2ARR(3,ISIZE)`
`N2ARR`	`INTEGER` 2-D array	To receive neighbour array, `N2ARR(3,ISIZE)`
`NVAR`	`INTEGER` variable	To receive number of elements
`ISIZE`	`INTEGER` expression	second dimension of `I2ARR` and `N2ARR`

SUBROUTINE ZZSHAD (ZLEV1, ZLEV2, ISHADE, XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZSHAD` – to shade the area between two contour levels from ungridded 3-D data (with neighbours).

Availability Section plus, released version 2-5.

Arguments

`ZLEV1`, `ZLEV2`	`REAL` expressions	Contour levels
`ISHADE`	`INTEGER` expression	Shading pattern number
`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZZSHDS (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZSHDS` – to draw a shaded contour map from ungridded 3-D data (with neighbours).

Availability Section plus, released version 2-5.

Arguments

`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

SUBROUTINE ZZSURF (XARR, YARR, ZARR, NPTS, I2ARR, N2ARR, NNODES, NELEMS)

Name

`ZZSURF` – to start a new picture and draw a surface from ungridded 3-D data (with neighbours).

Availability Section 2, released version 2-5.

Arguments

`XARR`, `YARR`, `ZARR`	`REAL` arrays	(x, y, z) coordinates of data values
`NPTS`	`INTEGER` expression	Number of data points
`I2ARR`	`INTEGER` 2-D array	Data element structure
`N2ARR`	`INTEGER` 2-D array	Element numbers of neighbours
`NNODES`	`INTEGER` expression	Number of nodes per element
`NELEMS`	`INTEGER` expression	Number of elements

D. Graphic Details

This appendix illustrates the graphical details of SIMPLEPLOT.

D.1 Broken line patterns

Broken line patterns]

CTBRKN and SQBRKN specify broken line patterns for contour curves and BRKNBX, BRKNPT, LABCV7 and LINEK7 draw lines with a specified broken line pattern. These subroutines identify the line pattern by values in the range -6 to $+6$. The number of software line patterns is unlimited but patterns beyond the usual range, $-6\ldots6$, have longer patterns and may not be easily distinguishable from one another. The number of hardware broken line patterns is also unlimited in theory but, in practice, there are fewer than are available in software. Figure D.1 illustrates the thirteen SIMPLEPLOT software broken line patterns.

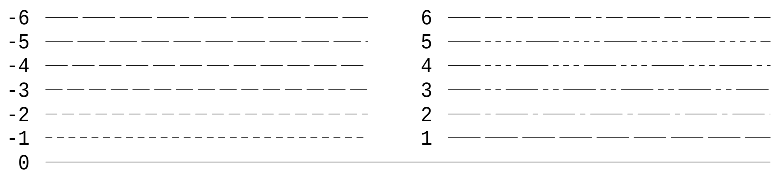


Figure D.1 SIMPLEPLOT software broken line patterns

D.2 Shading patterns

Shading patterns]

Shading patterns are a function of colour (where available) and pattern. The precise details of the patterns depend on the output device but are always chosen to give distinct appearances.

SIMPLEPLOT uses hardware shading by default when possible. The availability and number of hardware shading patterns depend on the device you are using. On all devices, SIMPLEPLOT resorts to software shading patterns when the hardware patterns have been exhausted. Figure D.2 illustrates hardware shading on a monochrome device with ten hardware shading patterns.

The patterns used for software shading consist of parallel hatching lines with adjustable angles and line separation. Figure D.3 illustrates software shading on a monochrome device. Software shading patterns consist of sets of equally-spaced parallel lines of one colour. The default sequence for a monochrome device is as follows:

- Pattern -1 is empty but is outlined using the pen selected by SHPEN, or the pen currently selected for drawing lines (pen pointer 1).

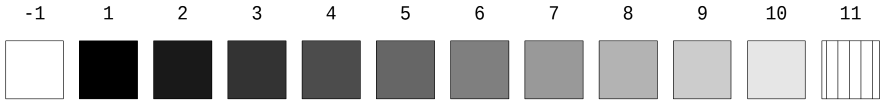


Figure D.2 Typical hardware shading patterns on a monochrome device

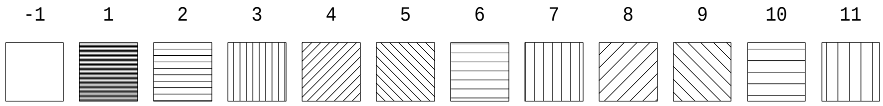


Figure D.3 Software shading patterns on a monochrome device

- Pattern 0 (not illustrated) is as near to solid as the device permits, using background colour; if the device cannot draw in background colour, pattern 0 is equivalent to pattern −1.
- Pattern 1 is as near to solid as the device permits and is not affected by settings for the angle or separation of hatching lines.
- Patterns 2–5 use the four shading angles with a small line separation.
- Patterns 6–9 use the four shading angles with a larger line separation.
- *etc.*

If a very large pattern number is chosen, very large line separation is used.

By default, SIMPLEPLOT uses four shading angles, 0°, 90°, 45° and 135°; the number of colours used is set to the maximum number of colours available on the device, and the minimum separation between shading lines is set to the thickness of the lines drawn on the device. `SHDESC` can be called before a shading operation, to specify alternatives for all these shading characteristics – the number of shading angles used is the number of different angles specified; the number of shading colours can be any positive value and the minimum separation between shading lines can be set to any value greater than or equal to the standard thickness of lines on the device.

D.3 Font

Font]

By default, SIMPLEPLOT uses the most appropriate hardware characters available on a graphics device to write text. In addition to hardware text, a set of simple software characters, proportionally spaced fonts (Hershey characters) and an adjustable fixed width font are available. Please note:

- *Hardware* fonts differ between graphics devices, therefore lettering which fits comfortably on one device may be smaller or larger on another.
- The *simple software* font is designed always to be clearly readable and may appear relatively larger on some low resolution graphics devices.
- Other software fonts are drawn independently of the resolution of the graphics device and may be illegible on some devices.

Figure D.4 illustrates the character sets available.

CHSET(0)	Hardware
CHSET(1)	Software
CHSET(2)	CARTOGRAPHIC
CHSET(3)	Simplex Roman
CHSET(4)	Duplex Roman
CHSET(5)	Complex Roman
CHSET(6)	Small Complex Roman
CHSET(7)	Triplex Roman
CHSET(8)	*Complex Italic*
CHSET(9)	*Small Complex Italic*
CHSET(10)	*Triplex Italic*
CHSET(11)	*Simplex Script*
CHSET(12)	*Complex Script*
CHSET(13)	Τινπμεω Ελληνικα
CHSET(14)	Γονπμεω Ελληνικα
CHSET(15)	Τναμμ Γονπμεω Ελληνικα
CHSET(16)	Вомплдч Вшсиллив
CHSET(17)	English Gothic
CHSET(18)	German Gothic
CHSET(19)	Italian Gothic
CHSET(20)	**Solid**
CHSET(21)	Outline
CHSET(22)	COMPLEX MATHS
CHSET(23)	Big Complex Maths
CHSET(24)	**Solid Roman**
CHSET(25)	Outline Roman
CHSET(51)	Adjustable ANSI (#)
CHSET(52)	Adjustable UK (£)
CHSET(-9)	Alternative Hardware Font

Figure D.4 SIMPLEPLOT character sets

D.4 Marker symbols

Marker symbols]

The standard SIMPLEPLOT marker symbols are identified in the range 0–16 and individual Hershey marker symbols can be selected from the range 17–84. A full range of software symbols is given in Figure D.5.

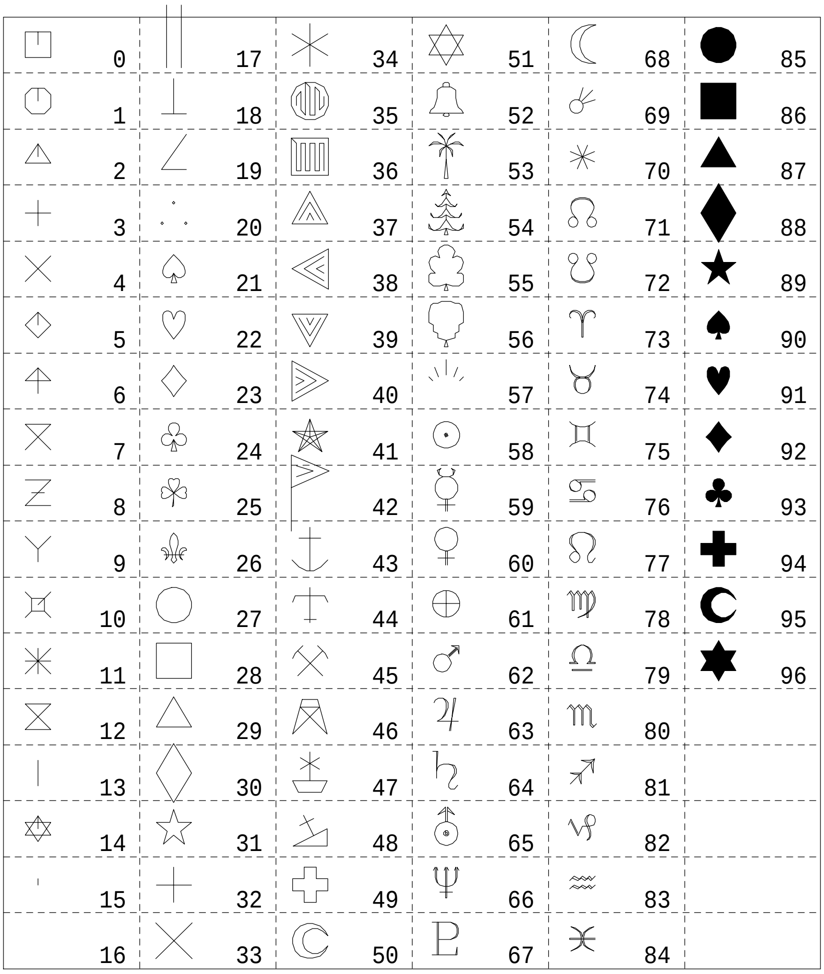


Figure D.5 SIMPLEPLOT marker symbols

E. Diagnostics

This appendix describes the diagnostics messages which SIMPLEPLOT issues both as a result of normal operation and of error conditions.

E.1 Normal operation

In a simple program diagnostic messages are produced to monitor progress of the program, as follows:

- When the first SIMPLEPLOT subroutine is called, the first diagnostics are produced:

  `(SIMPLEPLOT Mark 2-14(000)F)`

- When the first picture is started, the first graphics instructions are sent to the device; this produces the second diagnostic message:

  `(DEVICE OPENED: `*`device_name`*`)`

- When `ENDPLT` is called, SIMPLEPLOT closes the plotting device, triggers the output of diagnostic messages still outstanding for the last picture:

  `(END OF PICTURE)`

- Finally, SIMPLEPLOT outputs the following messages:

  ```
  (DEVICE CLOSED)
  (SIMPLEPLOT CLOSED)
  ```

Diagnostic messages are issued through the *diagnostic channel* which is opened unless diagnostics have been explicitly suppressed by `CALL DIAGLV(0)` (see below). The diagnostic channel is closed by `ENDPLT` along with other I/O channels used by SIMPLEPLOT. Details about directing the destination of diagnostic messages are found in your *Host Specific Information.*

`DIAGLV` changes the level of diagnostic reporting.

`DIAGLV(ILEVEL)` specifies one of five diagnostic levels, according to the value of `ILEVEL`, to output combinations of the three types of message:

`ILEVEL`	*Description*	*Messages*
0	No messages output at all	None
1	Brief messages (default)	Type 1
2	Level 1 messages plus fuller details	Type 1+2
3	Level 1 messages plus subroutine trace	Type 1+3
4	Level 2 messages plus subroutine trace	Type 1+2+3

The default is restored by `CALL DIAGLV(1)`.

E.2 Diagnostic messages

Diagnostic messages]

The different diagnostic messages fall into the following categories:

- Progress reports
- Errors in data or arguments
- Exceeding the picture or page limits
- Plotting with no active picture or page
- Trace of subroutine calls

These are described in detail below.

E.2.1 Progress reports

Progress reports]

SIMPLEPLOT reports on occurrences of definite events within your program which are a direct effect of what you have requested. The most important diagnostic messages in this category are listed below:

`(SIMPLEPLOT Mark 2-14(`*nnn*`)`*X*`)`[1]
`(SIMPLEPLOT CLOSED)`[1]
`(DEVICE CLOSED)`[1]
`(DEVICE OPENED: `*device name*`)`[1]
`(END OF GROUP)`[1]
`(END OF PICTURE)`[1]
`(START OF GROUP)`[1]

These messages are all of type 1 which are issued at diagnostic levels 1, 2 and 4, and are given in predominantly upper-case letters.

E.2.2 Errors in data or arguments

SIMPLEPLOT will always attempt to make sense of conflicting requests and, whenever possible, something is drawn. However, there are circumstances when the value of arguments or data do not make sense and an error messages is issued; for example,

`(DATA GRID NOT MONOTONIC)`[1]
`(INVALID ARGUMENT: ARRAY SIZE)`[1]
`(SURFACE OMITTED: CONSTANT DATA)`[1]
`(Triangulation data duplicated)`[1]

All these messages are of type 1 and are issued at diagnostic levels 1, 2 and 4.

E.2.3 Exceeding the picture or page limits

In a similar way to the errors described above, SIMPLEPLOT records any attempt to plot beyond the picture or page limits. At diagnostic level 1, these errors are all counted towards *incomplete tasks* messages and, at the end of every picture/page, a summary of incomplete tasks is given:

`(No. of incomplete page tasks= `n`)`[1]
`(>9999 incomplete page tasks)`[1]
`(No. of incomplete picture tasks= `n`)`[1]
`(>9999 incomplete picture tasks)`[1]

For example,

`(No. of incomplete page tasks=22)`[1]

At diagnostic level 2 (or 4) more information about these incomplete tasks is given; for example,

`(Key/caption area full)`[2]
`(Maximum no. of masked areas reached)`[2]
`(Symbol spills over boundary)`[2]
`(Title truncated)`[2]

Similarly, the following messages are output no more than once per user call:

`(`*`plotting`*` not all in range)`[2]
`(Data curve exceeds scales)`[2]
`(Inappropriate axis type: `*`XX`*`)`[2]
`(Invalid axis type: `*`XX`*`)`[1]
`(Isometric axes inappropriate)`[2]
`(Isometric coordinates inappropriate)`[1]
`(Surface label inappropriate)`[2]
`(Surface not all within picture)`[2]

In addition, any attempt to plot with coordinates out of range (of the plotting scales) results in the output of the offending coordinates:

(**x,y**)[2] – for a single point.
(**x_2,y_2,x_2,y_2**)[2] – for two points.

E.2.4 Plotting with no active picture or page

If plotting is attempted while there is no active picture or page, SIMPLEPLOT accumulates similar errors (which may be the result of a single programming error) and issues a single line summary. These errors are classed according to whether the attempted plotting is page- or picture-related:

(n `omissions, no active page)`[1]
`(>999 omissions, no active page)`[1]
(n `omissions, no active picture)`[1]
`(>999 omissions, no active picture)`[1]

Diagnostic level 2 or 4 must be specified to get more information about these incomplete tasks; for example,

`(Key/caption attempted with no page)`[2]
`(Title attempted with no picture)`[2]

E.2.5 Trace of subroutine calls

At diagnostic levels 3 and 4, each call of a SIMPLEPLOT subroutine produces a diagnostic of the form:

`(** `*`subroutine name`*` **)`

For example,

```
(** DIAGLV **)
(** SFEQX **)
(** SFEQY **)
(** RGSURF **)
(** ENDPLT **)
```

E.3 List of Messages

The following list includes all diagnostic messages which are issued by the subroutines described in this manual, and an explanation of why they occur. The number given after each message indicates the type of message, and therefore at which diagnostic level it is output.

`(**`x, y`**)`^{2} – the (x, y) coordinates of a point to which plotting has been omitted.

`(**`x_1, y_1, x_2, y_2`**)`^{2} – the coordinates of two points, (x_1, y_1) and (x_2, y_2), between which plotting has been omitted.

`(>999 omissions, no active page)`^{1} – more than 999 omissions have accumulated before a page has been started.

`(>999 omissions, no active picture)`^{1} – more than 999 omissions have accumulated before a picture has been started.

`(>9999 incomplete page tasks)`^{1} – more than 9999 incomplete page tasks have accumulated by the end of the page.

`(>9999 incomplete picture tasks)`^{1} – more than 9999 incomplete picture tasks have accumulated by the end of the picture.

`(`n` omissions, no active page)`^{1} – n omissions have accumulated before a page has been started.

`(`n` omissions, no active picture)`^{1} – n omissions have accumulated before a picture has been started.

`(Caption truncated)`^{2} – the caption added to a key or caption area (*eg.* using `ADDCP7`, `LINEK7` or `SHDEK7`) is too long to fit within the predefined area and has been truncated.

`(Constant data: default Z scale used)`^{1} – `WFCHT` or `WFDRAW` has been called with a set of equal z values.

`(Contour curve not all in range)`^{2} – the extent of a contour curve (drawn using `*CONT`) exceeds the current picture scales.

`(Contour map not all in range)`^{2} – the extent of a set of contour curves (drawn using `*CNTS`) exceeds the current picture scales.

`(Curve storage empty)`^{1} – `QCURVE` has been called when there are no stored curve coordinates to be retrieved.

`(Data curve exceeds scales)`^{2} – the extent of a curve (*eg.* drawn using `LABCV7`) exceeds the current picture scales.

`(DATA GRID NOT MONOTONIC)`^{1} – a tartan grid (x, x-y or y) has been specified, or `SQZVAL` has been called, with non-monotonic values.

`(DEVICE CLOSED)`^{1} – the current device has been closed.

`(DEVICE OPENED: `*`device name`*`)`^{1} – communication with the device has begun.

`(Element boundary not all in range)`^{2} – the extent of an element boundary (*eg.* drawn using `ZZEDGE`) exceeds the current picture scales.

`(Element not all in range)`^{2} – the extent of a single element (drawn using `ZELEM`) exceeds the current picture scales.

`(Elements not all in range)`[2] – the extent of a set of elements (*eg.* drawn using `ZZELMS`) exceeds the current picture scales.

`(END OF GROUP)`[1] – `GROUP` has been called and the specified configuration has been completed next picture will be on a new SIMPLEPLOT page.

`(END OF PICTURE)`[1] – the current picture has been completed either by an explicit call (*eg.* `ENDPLT`) or because a new picture has been started.

`(Grouping discontinued)`[1] – `GROUP` has been called to restore the default picture configuration (one per page).

`(Inappropriate axis type: XX)`[2] – one of the axis drawing subroutines (*eg.* `AXIS7`, `AXLAB7`) has been called with an inappropriate type of axis, `CHAXIS`=*XX*.

`(Insufficient number of valid points)`[1] – `LABCV7` has been called with an array which contains less than two valid points (*ie.* not equal to the current no-data value).

`(INVALID ARGUMENT: ARRAY SIZE)`[1] – an array has not been given valid dimensions.

`(Invalid axis type: XX)`[1] – one of the axis subroutines has been called with an unrecognized type of axis, `CHAXIS`=*XX*.

`(Isometric axes inappropriate)`[2] – isometric axes have been requested (using `ISAXD7`) when the current picture is not a surface picture.

`(Isometric coordinates inappropriate)`[1] – `KISXY` has been called to convert isometric coordinates when the current picture is not a surface picture.

`(Key/caption area full)`[2] – there is no room for the caption/key entry (*eg.* requested by `ADDCP7`, `LINEK7` or `SHDEK7`).

`(Key/caption attempted with no page)`[2] – a key or caption area has been defined (*eg.* using `DEFKEY`, `MPK7H` or `MPK7V`) before a page has been started.

`(Key/caption attempted with no picture)`[2] – a key or caption area has been defined using picture-related positional descriptors before a picture has been started.

`(Key/caption height reduced)`[2] – the size of a key or caption area (defined using `DEFKEY`, `MPK7H` or `MPK7V`) has been reduced due to restrictions of the current page and/or picture size and the size of text.

`(Key/caption width reduced)`[2] – the width of a key or caption area (defined using `DEFKEY`, `MPK7H` or `MPK7V`) has been reduced due to restrictions of the current page and/or picture size and the size of text.

`(MAP KEY OMITTED: CONSTANT DATA)`[1] – a map key has been requested (using `MPK7H` or `MPK7V`) with the limits of the data range equal, and no z scale specified elsewhere (by `SFZSCL`).

`(MAP KEY OMITTED: NULL RANGE)`[1] – a map key has been requested (using `MPK7H` or `MPK7V`) which would cover a null z range.

`(Maximum no. of keys/captions reached)`[2] – the number of blanked or reserved key and caption areas (defined using `DEFKEY`, `MPK7H` or `MPK7V`) has exceeded the maximum of 9.

`(Maximum no. of masked areas reached)`[1] – the number of masked areas used for individual marker symbols and text strings has exceeded the maximum of 50; subsequent marked areas take the place of the oldest areas and overdrawing of these areas may occur.

`(No active key/caption area)`[2] – there is no defined area for the caption/key entry (requested by `ADDCP7`, `LINEK7` or `SHDEK7`).

`(No current waterfall Z scale)`[1] – `KWZVAL` or `QWZSCL` has been called when there are no existing waterfall scales.

`(No room for key/caption area)`[2] – The space available for a key or caption area (*eg.* defined using `DEFKEY`, `MPK7H` or `MPK7V`) is not even sufficient for one line of text containing only one character.

`(No. of incomplete page tasks=` *n*`)`[1] – *n* incomplete page tasks have accumulated by the end of the page.

`(No. of incomplete picture tasks=` *n*`)`[1] – *n* incomplete picture tasks have accumulated by the end of the picture.

`(None of title will fit in)`[2] – The space available for a title is not wide enough for a single character.

`(Null box, nothing drawn)`[2] – the extent of a box (*eg.* drawn using `BRKNBX`) exceeds the current picture scales.

`(POLAR OMITTED: ZERO RADIUS)`[1] – `POLAR7` has been called with zero value for maximum radial scale value.

`(Range >=100; linear scale used)`[1] – `SCALES` has been called for a normal probability scale which exceeds 100.

`(Range through 0; linear scale used)`[1] – `SCALES` has been called for a non-linear scale which includes zero.

`(Requested point unavailable)`[1] – `CVLBPS` has been specified a reference point for `LABCV7` which cannot be used.

`(Shaded contour not all in range)`[2] – the extent of a shaded contour region (drawn using `*SHAD`) exceeds the current picture scales.

`(Shaded contours not all in range)`[2] – the extent of a shaded contour map (drawn using `*SHDS`) exceeds the current picture scales.

`(SIMPLEPLOT CLOSED)`[1] – is issued by `ENDPLT` and indicates that all activity by Simpleplot has finished and all associated files have been closed.

`(SIMPLEPLOT Mark 2-14(`*nnn*`)`*X*`)`[1] – indicates that Simpleplot is in use; it is issued by the first call to any Simpleplot subroutine except `DIAGLV` with `ILEVEL`=0.

`(START OF GROUP)`[1] – `GROUP` has been called and the next picture will be the first in a group formation.

`(String too short for INTEGER)`[1] – `KNUMB` has been called with a string variable which is not large enough to hold the converted `INTEGER`.

`(String too short for REAL)`[1] – `KREAL` has been called with a string variable which is not large enough to hold the converted `REAL` number.

`(Surface label inappropriate)`[2] – `SFLAB` has been called when there is no current z scale (for a contour map or a surface picture) to be labelled.

`(Surface not all within picture)`[2] – the extent of a surface (drawn using `*SURF` exceeds the current picture scales.

`(SURFACE OMITTED: CONSTANT DATA)`[1] – surface drawing has been attempted for an array containing values which are all the same.

`(Surface section not all in range)`^{2} – the extent of a cross-sectional curve (drawn using `*CUT`) exceeds the current picture scales.

`(Symbol clipped)`^{2} – a marker symbol (*eg.* drawn by `CP7PT` or `MARKPT`) has exceeded the current clipping window (the picture or the page).

`(Symbol spills over boundary)`^{2} – a marker symbol (*eg.* drawn by `CP7PT` or `MARKPT`) is centred on the edge of the current picture.

`(Text clipped)`^{2} – a caption (*eg.* drawn by `CP7PT`) or title has exceeded the current clipping window (the picture or the page).

`(Text omitted: too many lines)`^{2} – the key/caption area is not full but the caption/key entry contains more lines than can be accommodated (*ie.* the caption includes active escape sequences to insert new lines).

`(Text truncated on curve label)`^{2} – the label of a labelled curve would have extended beyond the end of the curve and has therefore been truncated.

`(Title attempted with no page)`^{2} – a title has been requested before a page has been started.

`(Title attempted with no picture)`^{2} – a title has been requested using picture-related positional descriptors before a picture has been started.

`(Title omitted below bottom)`^{2} – an additional title has been requested towards the bottom of the SIMPLEPLOT page but the is not enough room beneath the existing line(s) of title.

`(Title too tall)`^{2} – The space available for a title is not even sufficient for one line of text.

`(Title truncated)`^{2} – a title is too long to fit within the limiting area (which depends on the position) and has been truncated.

`(Too many axis intervals: default used)`^{1} – the axis interval (specified using `AXSBDV`) is less than $1.0^{-4} \times$ axis range.

`(Too many contours: default used)`^{1} – `SFEQZ` or `SFEQZD` has been called to specify a contour interval less than $1.0^{-4} \times$ contour range.

`(Too many points in waterfall curve)`^{1} – `WFCHT` or `WFDRAW` has been called with more than 1024 points.

`(Too many waterfall curves on picture)`^{1} – `WFDRAW` has been called when the picture already contains number of curves specified by `WFNCVS`.

`(Triangulation data duplicated)`^{1} – data cannot be triangulated because of duplicated data (*eg.* (x, y, z_1) and (x, y, z_2)).

`(Triangulation failed)`^{1} – triangulation process has failed; try again with `ZZORDN` to normalize data first.

`(Triangulation impossible)`^{1} – data cannot be triangulated, *eg.* points are co-linear.

`(Triangulation incomplete)`^{1} – maximum size of element array is too small to store all elements.

`(Waterfall chart not all in range)`^{2} – the extent of a waterfall chart (drawn using `WFCHT`) exceeds the current picture scales.

`(Waterfall curve not all in range)`^{2} – the extent of a waterfall curve (drawn using `WFDRAW`) exceeds the current picture scales.

`(WATERFALL OMITTED: INVALID DIMENSIONS)`^{1} – 2-D array has not been given valid dimensions.

G. Glossary

Angular axis the circular axis on a polar plot which represents the angular (θ) scale.

Annotation see *Axis annotation*.

Area fill see *Shading*.

Aspect ratio The ratio of the width to the height of a rectangular area; for example, an aspect ratio of 2.0 indicates an area twice as wide as it is high.

Axis the framework on an x-y plot, polar plot, histogram or bar chart.

Axis annotation label(s) at axis subdivision(s).

Axis caption text used to identify an axis.

Bundles set of attributes associated with one of the following graphical element types: line, marker symbol or text.

Caption a text entry added to a predefined caption area (or key area).

Caption area an area defined for subsequent captions.

Cartesian coordinate system in which points are described by horizontal and vertical distances from a fixed origin.

Cascade picture a type of surface picture in which curves of z against y or z against x are plotted at equally-spaced intervals on the underlying mesh.

Composite plotting plotting where a single subroutine performs a number of different operations.

Contour interval the difference in altitude represented by the space between two contour lines on a map.

Contour level the height (or z value) at which a contour line is drawn, or which bounds one side of a shaded area.

Contour line a line on a map or chart joining points of equal height or depth.

Contour map a diagrammatic representation of a surface made up of contour lines.

Coordinate pair a numerical description of the position of a point, (x, y) or (r, θ).

Crosshatching drawing two or more sets of parallel lines which cross each other.

Data grid a grid of intersecting straight lines in x and y (or r and θ) whose points of intersection are where the data points $z = f(x, y)$ (or $z = f(r, \theta)$) occur.

Default a pre-set value or condition which you can either change or allow to stand.

Default page according to the device, the area within which a group of one or more pictures is positioned.

Device see *Graphics device*.

Device driver a set of subroutines which implements graphic functions used for graphics devices.

Element array a 2-D array describing the non-overlapping elements of the x-y plane which describe the configuration of ungridded data.

Font a set of graphic representations of the standard character set (*eg.* italic).

Framework the axis structure of a graph.

General graph plotting plotting using coordinate pairs, (x, y) or (r, θ), which are interpreted according to the current plotting scales and coordinate system.

Graph a picture showing the relationship of one variable to another.

Graphics device the physical display medium (*eg.* plotter or Windows).

Group a set of pictures within a page surrounded by a periphery.

Host Specific Information the information which tells you how to execute a program which contains calls to SIMPLEPLOT on your computer system.

Isometric axis one of the x, y or z axes which can be drawn on a surface picture.

Isometric coordinates the (x, y, z) coordinates represented on a surface picture.

Isometric projection an axonometric projection for which all three mutually perpendicular axes have equal scaling.

Justification the position of text relative to a reference point.

Key area or **key box** an area defined for subsequent key entries.

Key entry a description, within the key box, which identifies a key sample *eg.* line, marker symbol *etc.*

Label a text string positioned at specified coordinates.

Labelled curve a curve made up of straight lines from point-to-point along which a label is written.

Landscape the orientation of a page set on its long axis; see also *Portrait.*

Layout the design of the SIMPLEPLOT page – number of pictures, size of margins *etc.*

Margin an area of space around a picture, used for axis annotation.

Marker symbol a special symbol, used for identifying data points.

Masked area an area around symbols, characters, keys or caption areas which will not be overdrawn.

Monotonic consistently increasing or decreasing in value.

Neighbour array a 2-D array, with the same dimensions as the element array, describing the which elements are adjacent.

New picture the start of a new 2-D picture, 3-D surface picture, *etc.*

Origin the fixed point, (0,0), from which coordinates are positioned.

Page a SIMPLEPLOT page is the area in which all drawing will be performed; it may be made up of more than one picture.

Parallel arrays two 1-D arrays of the same dimension whose elements contain related items, *eg.* the x and y coordinates of a point.

Periphery the margin left around a group of pictures at the edge of the SIMPLEPLOT page.

Pen a physical pen or colour which is selected by number for drawing.

Pen pointers SIMPLEPLOT's four internal indicators of pen usage; each pointer is assigned to a *pen.*

Picture the pictorial representation of one or more data sets; it may be a surface picture, contour map, waterfall chart, cross section, *etc.* A picture is also the current plotting area.

Plotting scales see *Scales.*

Polar a radial form of 2-D plotting.

Polar data data representing a function of two variables such that $z = f(r, \theta)$.

Portrait the orientation of a page set on its short axis; see also *Landscape.*

Radial axis the axis on a polar plot which represents the radial scale.

Regular-gridded data data representing a continuous function of two variables where each z value occurs at the intersections of an equally-spaced grid over the x-y plane.

Scales or **Plotting scales** the range over which data is represented – it may be marked on an axis.

Shaded contour map a diagrammatic representation of a surface made up of shaded contour intervals.

Shading the filling of an area using a distinguishable combination of pattern and colour.

Tartan-gridded data data representing a continuous function of two variables where each z value occurs at the intersections of an x-y grid where the x values and/or y values are specified separately.

Title a text string used as the title of a picture or group of pictures.

Ungridded data data representing a continuous function of two variables where each z value occurs at specified (x, y) coordinates on the x-y plane.

User coordinates coordinate pairs – (x, y) or (r, θ) – interpreted in terms of the current 2-D plotting scales.

Waterfall chart consists of a family of curves plotted against a single independent variable. Each curve is displaced from the previous curve by a constant offset and each curve is masked by the previous curve.

S. Subroutine Summary

The numbers marked against the subroutine name indicate in which section of the library the subroutine is included.

Starting a new picture

`AXES7`[1]	start a new 2-D picture and draw axes
`ISNEW`[2]	start a new surface picture
`FNSURF`[2]	start picture and draw surface of a user-defined 3-D function
`NEWPIC`[1]	start a new picture
`POLAR7`[1]	specify scales, start a new 2-d polar picture and draw axes
`RGSURF`[2]	start picture and draw surface (data on regular grid)
`XSURF`[2]	start picture and draw surface (data on x-specified grid)
`XYSURF`[2]	start picture and draw surface (data on x-y specified grid)
`YSURF`[2]	start picture and draw surface (data on y-specified grid)
`ZZSURF`[2]	start picture and draw surface (ungridded data with neighbours)

Data manipulation

`KZZRG`[2]	convert ungridded data with neighbours to regular grid
`LIMEXC`[1]	find the exclusive range of values in an array
`LIMINC`[1]	find the inclusive range of values in an array
`LIMIDX`[1]	find positions of maximum and minimum values in an array
`LIMSFN`[2]	find the range of a 3-D function
`NODATA`[1]	specify `REAL` value to represent missing data
`QNODAT`[1]	inquire current missing data value
`ZELEM`[2]	draw outline of a single element
`ZNEIGH`[2]	generate neighbour array
`ZNUMB`[2]	specify node and element numbering
`ZZEDGE`[2]	draw periphery of data (with neighbours)
`ZZELMS`[2]	draw element outlines (with neighbours)
`ZZORDN`[2]	structure data into elements and neighbours (normalized)
`ZZORDR`[2]	structure data into elements and neighbours

Plotting three-dimensional data

Surface pictures, Contour maps and Cross sections

`CTBRKN`[2]	specify broken line pattern for contours and cross sections
`CTCURV`[2]	specify curve type for contours and cross sections
`CTHOLD`[+]	specify whether coordinates of 3-D curves are to be stored
`CTLABS`[2]	specify frequency of contour labels
`CTNUMB`[2]	specify whether contours are to be labelled
`ISANG`[2]	specify angle for representing surfaces
`ISCURV`[2]	specify smoothness for crosshatch/cascade lines on surfaces
`ISDIAG`[2]	specify whether to include x-y-z arrows on surface pictures
`ISFULL`[2]	specify whether surface pictures are to fill space available
`ISMESH`[2]	specify fineness of detail for surface pictures

X *Y* *Z*	*regular* *regular* *matrix*	*array* *regular* *matrix*	*regular* *array* *matrix*	*array* *array* *matrix*	*array* *array* *array*	*regular* *regular* *function*
Surface	*RGSURF*[2]	*XSURF*[2]	*YSURF*[2]	*XYSURF*[2]	*ZZSURF*[2]	*FNSURF*[2]
Contour map	*RGCNTS*[2]	*XCNTS*[2]	*YCNTS*[2]	*XYCNTS*[2]	*ZZCNTS*[2]	*FNCNTS*[2]
Contour line	*RGCONT*[2]	*XCONT*[2]	*XCONT*[2]	*XYCONT*[2]	*ZZCONT*[2]	*FNCONT*[2]
Cross-section	*RGCUT*[2]	*XCUT*[2]	*YCUT*[2]	*XYCUT*[2]	*ZCUT*[2]	*FNCUT*[2]
Interpolation	*RGCALC*[2]	*XCALC*[2]	*YCALC*[2]	*XYCALC*[2]	*ZZCALC*[2]	*N/A*
Shaded map	*RGSHDS*[+]	*XSHDS*[+]	*YSHDS*[+]	*XYSHDS*[+]	*ZZSHDS*[+]	*FNSHDS*[+]
Shaded region	*RGSHAD*[+]	*XSHAD*[+]	*YSHAD*[+]	*XYSHAD*[+]	*ZZSHAD*[+]	*FNSHAD*[+]
Waterfall *Chart*	*WFCHT*[+] *WFDRAW*[+]			*Convert data* *to regular grid*		

`ISRISE`[2] specify width:height proportions for surface pictures
`ISTYPE`[2] specify type of surface picture
`ISVIEW`[2] specify view point for surface pictures
`QCURVE`[+] inquire coordinates of a stored curve or curve segments
`QSFLAB`[2] inquire surface picture and contour map values
`SFEQZ`[2] specify spacing of contours
`SFEQZD`[2] specify contour intervals for discrete data
`SFLAB`[2] draw caption of range of values displayed
`SFLIMS`[2] specify z plotting range without changing scale
`SFMESH`[+] specify mesh for contour curves and surface pictures
`SFZSCL`[2] specify z scale for surface pictures and contour maps
`SQZVAL`[+] specify sequence of contour levels
`SQZLAB`[+] specify sequence of contour labels

Waterfall charts

`WFCHT`[+] draw waterfall chart on the current 2-D picture
`WFDRAW`[+] draw an individual waterfall curve
`WFEQL`[+] specify equally-spaced waterfall label scale
`WFEQN`[+] specify equally-spaced waterfall numeric scale
`WFINIT`[+] reset default waterfall characteristics
`WFNCVS`[+] specify number of waterfall curves
`WFNSCL`[+] specify limits of numeric waterfall scale
`WFPNS`[+] specify pens for 4 pen pointers on waterfall charts
`WFSTEP`[+] specify displacement between waterfall curves
`WFZLEV`[+] specify data level that pen changes on waterfall curve
`WFZSCL`[+] specify z scale of waterfall charts
`KWZVAL`[+] convert z value on curve to waterfall label scale
`QWZSCL`[+] inquire limits of waterfall z scale

Controlling 2-D and 3-D data scales

`COORDS`[1] change interpretation of coordinates
`EQSCAL`[1] specify similar linear scales for Cartesian/polar pictures
`FNAREA`[2] specify plotting ranges for 3-D functions
`ISYUP`[2] specify direction of change of y for surface pictures
`KSCALE`[1] convert scale limits such that they span whole subdivisions
`SCALES`[1] specify both horizontal and vertical scales
`SFEQX`[2] specify equally-spaced x values associated with 3-D data
`SFEQY`[2] specify equally-spaced y values associated with 3-D data
`SFLIMS`[2] specify z plotting range without changing scale
`SFZSCL`[2] specify z scale for surface pictures and contour maps

Point-by-point plotting and annotation

`BREAK`[1]	force a break between joined points
`BRKNBX`[1]	draw a box using specified line style
`BRKNPT`[1]	draw a straight line to a specified point using specified line style
`CP7PT`[1]	draw a symbol annotated with a caption at point
`CVLBJS`[4]	specify justification of lettering on a labelled curve
`CVLBPS`[4]	specify reference point for label on a labelled curve
`KISXY`[2]	convert coordinate on a surface picture to (x, y)
`KNUMB`[1]	convert `INTEGER` to text string
`KREAL`[1]	convert `REAL` to text string
`LABCV7`[4]	draw a labelled curve
`MARKPT`[1]	draw a marker symbol at a specified point
`RANGE`[1]	draw a line indicating a range of values
`SQBRKN`[+]	specify sequence of broken line patterns

Shading control

`SHEDGE`[+]	specify whether to draw a shading boundary
`SHPATT`[4]	specify one of a sequence of shading patterns
`SHPEN`[4]	specify pen number for monochrome shading
`SQSHAD`[4]	specify sequence of shading patterns

Axes

`AXCLR`[1]	specify level of automatically generated axis labels
`AXES7`[1]	start a new 2-D picture and draw axes
`AXGRID`[1]	specify style and level of grids at axis subdivisions
`AXIS7`[1]	draw an axis
`AXLAB7`[1]	draw an individual axis annotation label
`AXLBGP`[4]	specify level of annotation at axis intersections
`AXLBJS`[4]	specify justification of axis annotation labels
`AXLOCN`[4]	specify location of an axis w.r.t. the picture
`AXRNGE`[1]	specify the sub-range of axis
`AXSBDV`[1]	specify the axis subdivision interval
`ISAXD7`[+]	draw x-y-z axes to a surface picture
`ISAXES`[+]	specify whether x-y-z axes are drawn on surface pictures
`POLAR7`[1]	specify scales, start a 2-D polar picture, draw axes

Titles, keys and captions

`ADDCP7`[1]	draw a caption in defined area
`DEFKEY`[1]	define an area for keys
`DEFKYW`[4]	specify width of samples in a key box
`LINEK7`[1]	draw key to broken line pattern
`MPK7H`[+]	draw a complete horizontal key to a shaded contour map
`MPK7V`[+]	draw a complete vertical key to a shaded contour map
`SFLAB`[2]	draw caption of range of values displayed
`SHDEK7`[4]	draw key to a shading pattern
`TITLE7`[1]	draw a caption as a title to a picture, group or page

Layout

`GROUP`[1]	specify group layout of pictures

`PERIM`[1]	draw rectangular perimeter around current picture

Pen/bundle control

`BUNLPR`[+]	specify order of precedence of bundled line-drawing attributes
`PEN`[1]	specify single pen/bundle for all drawing
`SETPNS`[1]	specify pens/bundles for 4 pen pointers
`SHPEN`[4]	specify pen/bundle number for monochrome shading
`SQPEN`[+]	specify sequence of pens/bundles
`WFPNS`[+]	specify pens for 4 pen pointers on waterfall charts
`WFZLEV`[+]	specify data level that pen changes on waterfall curve

Device and job control

`DIAGLV`[1]	specify level of diagnostics
`ENDPLT`[1]	terminate plotting, close graphics device and SIMPLEPLOT
`INITSP`[1]	reset all defaults
`WFINIT`[+]	reset default waterfall characteristics

Index