SIMPLEPLOT 2-15

SIMPLEPLOT
Primer

BUSS Ltd.

SIMPLEPLOT Primer

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 Example Programs

List of Figures

List of Tables

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. It still achieves this goal but SIMPLEPLOT has been developed into a much more powerful tool which can also be used for professional software applications.

Six separate sections constitute the complete SIMPLEPLOT Mark 2:

1. The basic package for conventional graph plotting – x-y plots and polar plots.

2. Additional subroutines for 3-dimensional plotting – contour maps and isometric surface pictures of 3-D data

4. Additional subroutines for presentation graphics – bar charts, histograms and pie charts.

5. Additional subroutines for plotting functions of three variables – perspective pictures of 4-dimensional data.

6. The SIMPLEPLOT mapping module – for representing data based on geographical coordinate systems.

7. The SIMPLEPLOT ViSualization module – for full colour modelling of functions of 2- 3- and 4-variables.

There is also Section + which provides extra facilities with a SIMPLEPLOT library that has at least Sections *1*, *2* and *4*. SIMPLEPLOT-PLUS refers to a SIMPLEPLOT library which is made up from Sections *1*, *2*, *4* and +.

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

SIMPLEPLOT is written in standard FORTRAN, and can be used with any language which can call FORTRAN subroutines.

Graphics device interface

The SIMPLEPLOT library is independent of any single graphics system but, rather than describe SIMPLEPLOT as either device *independent* or device *dependent*, it is more appropriate to describe it as 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.

The SIMPLEPLOT library is usually supplied with the BUSS *Single Entry Point* (S.E.P.) device driver system; this includes interfaces to a large number of graphics devices, and the range of validated device drivers is constantly being extended. Moreover, SIMPLEPLOT can address graphics devices directly, or through a separate low-level graphics systems (*eg.* GKS, CGM) or graphics languages (*eg.* PostScript).

What sort of pictures can you draw?

This manual describes facilities for plotting 2-dimensional data:

- x-y plots with linear, logarithmic and normal probability scales
- Polar charts and more general circular charts
- Curve drawing, and linear or quadratic regression curves
- Plotting of polynomial functions and user-defined functions
- General 2-D point-by-point plotting with Cartesian or polar coordinate systems
- Pie charts
- Bar charts
- Histograms

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

1. Introduction

The *SIMPLEPLOT Primer* is an introductory manual for the SIMPLEPLOT library of graph drawing subroutines. It can be used as a manual in its own right but, for detailed explanations of all SIMPLEPLOT subroutines, please refer to the *SIMPLEPLOT Reference manual*.

1.1 Overview

The *SIMPLEPLOT Primer* is organized as a tutorial, followed by a cookbook, technical appendices and reference material:

- Introduction to the SIMPLEPLOT Primer
- Tutorial – Chapters 2–5
  2. Getting started with SIMPLEPLOT
  3. General graph plotting – x-y plots and polar plots
  4. Special charts – histograms, bar charts and pie charts
  5. Additional facilities – layout control, titles, keys, captions, text control, axes, shading and colour.
- Cookbook – Chapter 6
- Technical Appendices
  A. Subroutine specifications – brief specification of all subroutines used in this manual.
  D. Device driver information – the Single Entry Point device driver system.
  G. Graphic details – shading patterns, broken line styles, marker symbols and character sets.
  M. Messages – diagnostics messages and explanations for some common problems.
  S. Summary of subroutines – single line summaries of subroutines used in this manual.
- Reference Material – Glossary and Index

How you choose to use the *SIMPLEPLOT Primer* will depend on your experience of SIMPLEPLOT.

1.2 Target audience

The *SIMPLEPLOT Primer* has been written with the following readers in mind:

- Newcomers to SIMPLEPLOT who want an introduction to using SIMPLEPLOT to produce simple pictorial representations of data.
- Long time (but not extensive) users of the Fortran version of SIMPLEPLOT who are now choosing to program in C.
- Experienced users who probably need not read it at all for themselves but who may be using it to teach others.

All readers should be familiar with programming in FORTRAN on their host computer system.

1.3 Related documents

Related documents include:

- The *SIMPLEPLOT Reference manual* which contains full specifications for all SIMPLEPLOT-PLUS subroutines and in which *Host Specific Information* should be available.

1.4 How to report problems

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

1.5 How to use this manual

If you are a newcomer to SIMPLEPLOT you may find it easier to get started if you use this manual in the following way:

1. Look through the chapter introductions and decide whether Chapter 3 or Chapter 4 is required. Then, either,
   - Convert an existing program into a plotting program by adding calls of subroutines from one of these chapters alone, or,
   - Find an example program which is the closest to what you want to produce and adapt it for your data.
2. Execute your program according to your host computer's requirements for a SIMPLEPLOT program.
3. When the program works and produces basic graphs, it can be enhanced by using some of the subroutines specified in Chapter 5.

Having gained confidence in using the subroutines, you (as a new user) should then be able to go on and use any combination of subroutines for which you have an application.

1.6 Conventions

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

Example programs: The example programs are written in standard FORTRAN and have been designed to be as brief as possible. In Chapters 3 and 4, the example programs concentrate on how to draw simple representations of data without refinements such as titles, keys or additional labelling. The examples in Chapter 6 combine the additional features of Chapter 5 with the data plotting facilities described in Chapters 3 and 4; although as short as possible, these cookbook programs are necessarily more complex (but therefore more realistic).

In all example programs, small data sets are included in the program using in-line static array initializers; longer data sets are read from files which are included as part of the software distribution kit.

Diagnostics from the example programs are only included for Example 1 but a full explanation of all diagnostic messages issued by subroutines described in this manual is given in Appendix M, *Messages*, and some common problems are discussed in section M.4.

Function specifications: For full details of these and other SIMPLEPLOT subroutines please refer to the *SIMPLEPLOT Reference manual*.

Illustrations: All figures and output from example programs are produced using SIMPLEPLOT version 2-13 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 such that lines are made up from dotted lines.

2. Getting Started

This chapter describes how to start using SIMPLEPLOT to draw pictures of your data. The basic elements of SIMPLEPLOT are introduced and should give you a general overview of what to expect from SIMPLEPLOT.

2.1 SIMPLEPLOT output

2.2 Simple plotting – an example of some basic plotting concepts

2.3 An overview of SIMPLEPLOT subroutines

2.4 General plotting procedure

2.5 The next step

Specific details, such as what devices are available and how to execute a SIMPLEPLOT program, cannot be included as they are dependent on how SIMPLEPLOT has been implemented at your site.

2.1 SIMPLEPLOT output

Once a program includes calls to SIMPLEPLOT subroutines it will produce two additional types of output, diagnostic output and graphical output.

2.1.1 Diagnostic output

SIMPLEPLOT issues diagnostic messages to provide information about progress, to warn of omissions and to report fatal errors. For example, a typical execution of a SIMPLEPLOT program which draws a single picture may produce the following diagnostics:

```
(SIMPLEPLOT Mark 2-13(000)F)
(DEVICE OPENED: device_name)
(END OF PICTURE)
(DEVICE CLOSED)
(SIMPLEPLOT CLOSED)
```

The diagnostics messages which SIMPLEPLOT issues both as a result of normal operation and of error conditions are described in Appendix M.

2.1.2 Graphical output

The graphical output from a SIMPLEPLOT program depends to a great extent on the graphics device but, whatever device is used, SIMPLEPLOT automatically organizes your plotting within the available space. Data are plotted within a *picture*, pictures are laid out in groups, groups are positioned within pages and, where necessary, SIMPLEPLOT positions pages within the available plotting space. The process of starting a new picture can be initiated without any preliminary planning. The following terms are used to refer various components of your graphical output and illustrated in Figure 2.1:

- A *graphics device* is the physical display medium.
- A SIMPLEPLOT *page* corresponds to a region on the device within which drawing may be performed. A page has width and height defined in centimetres. All drawing is restricted to the current page.
- A *group* is constructed from a set of pictures thus forming a region within a page with a *periphery* between the edge of the group and the page. By default, only one picture is drawn per page.
- A *picture* is the region on a page within which data is plotted. No plotting can be performed until a picture is started. A picture is surrounded by a *margin* which is typically used for axis annotation.

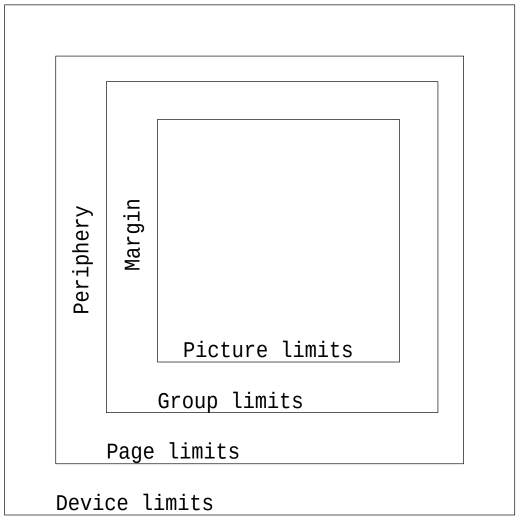
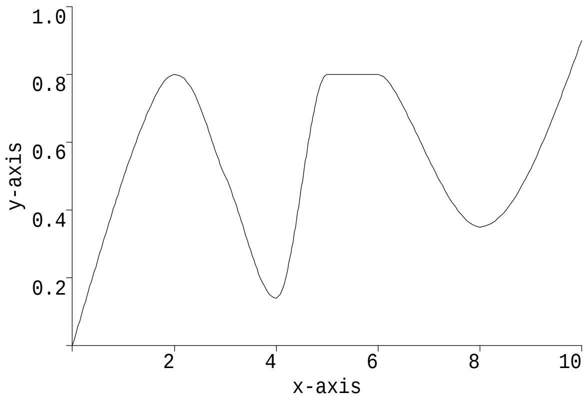


Figure 2.1 SIMPLEPLOT page layout – one picture per page

2.2 Simple plotting

Example 1 is a simple FORTRAN program which draws a curve through a set of coordinates. These coordinates are stored in parallel arrays, `XARR` and `YARR`, such that each point is represented by the coordinate pair (`XARR(I)`,`YARR(I)`), `I`=1 ... `NARR`.




```
      PROGRAM PRIM01
      PARAMETER(NARR=8)
      REAL XARR(NARR),YARR(NARR)
      DATA XARR/0.0,2.0,3.0,4.0,5.0,6.0,8.0,10.0/
      DATA YARR/0.0,0.8,0.5,0.14,0.8,0.8,0.35,0.9/
C specify plotting scales
      CALL SCALES(0.0,10.0,1, 0.0,1.0,1)
C start picture and draw pair of axes
      CALL AXES7('x-axis','y-axis')
C draw curve of data
      CALL BRKNCV(XARR,YARR,NARR,0)
C terminate plotting
      CALL ENDPLT
      END
```

Example 1. A simple picture

Explanation of subroutines

`SCALES(XSTART,XSTOP,IXTYPE,YSTART,YSTOP,IYTYPE)` defines the scales for subsequent Cartesian plotting. Each scale is 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 (0=natural, 1=linear, 2=logarithmic or 3=normal probability (%)).

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

`(SIMPLEPLOT Mark 2-13(000)F)`[1]

Refer to Appendix M for more information about diagnostic messages.

`AXES7(CAPX,CAPY)` starts a new picture and draws a pair of axes annotated with the current plotting scales. The two arguments provide the captions for the horizontal axis and the vertical axis.

When the first picture is started, the graphics device is opened, and the first graphics instructions are generated; the second diagnostic message is also produced:

`(DEVICE OPENED:` *`device_name`*`)`[1]

Refer to Appendix D for more information about device selection.

`BRKNCV(XARR,YARR,NARR,LTYPE)` draws the curve which connects the points represented by the following coordinate pairs:

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

The data points are interpreted in terms of the plotting scales which are operating when `BRKNCV` is called.

The last argument, `LTYPE`, specifies which of the broken line patterns shown in Figure G.1 (see page 156) is to be used. `LTYPE`=0 represents a solid line.

`ENDPLT` must be called when plotting is complete. It empties any plotting buffers which are in use, closes the plotting device, triggers the output of diagnostic messages still outstanding for the last picture,

`(END OF PICTURE)`[1]

and then outputs the following messages:

`(DEVICE CLOSED)`[1]
`(SIMPLEPLOT CLOSED)`[1]

The number given after each diagnostic message indicates the diagnostic level at which it is output (see Appendix M for more information).

2.3 An overview of SIMPLEPLOT subroutines

All SIMPLEPLOT subroutines fall into three distinct categories:

- *Graphics* subroutines generate graphics (*ie.* instructions which are sent to the graphics device).
- *Specification* subroutines specify characteristics which affect the next and subsequent calls of relevant *graphics* subroutines.
- *Auxiliary* subroutines perform auxiliary tasks.

The use of *graphics*, *specification* and *auxiliary* subroutines is described below. The order for calling SIMPLEPLOT subroutines is not strict and is summarized in section 2.4.2.

2.3.1 Graphics subroutines

Graphics subroutines fall into four distinct categories – new picture subroutines, data plotting subroutines, labelling subroutines and axis drawing subroutines. Most graphics subroutines perform a single task and fall into just one of these categories, but a few *composite plotting* subroutines perform a number of tasks.

New picture subroutines

Each picture is started by a call of a new picture subroutine which prepares for a new picture in a way suitable to the device; for example, a new empty area of paper is found, or the screen is cleared. The first call of a new picture subroutine in a program automatically causes any procedures needed for opening the plotting device to be executed before the picture is started. All plotting must be preceded by a call of a new picture subroutine. The new picture subroutines described in this manual are `AXES7`, `NEWPIC` and `POLAR7`.

Data plotting subroutines

A wide range of subroutines are available for plotting coordinate data, (x, y) or (r, θ), which all have arguments which are interpreted as coordinates according to the scales and coordinate system in operation when they are called:

- *Plotting sets of data* with lines (`BRKNAR`, `BRKNCV` or `BRKNFN`), with marker symbols (`MARKAR` or `MARKCV`) and with shaded areas (`SHDEAR`, `SHDECV` or `SHDEFN`).
- *Point-by-point plotting and annotation* (`BRKNPT`, `CP7LB`, `CP7PT`, `DRAWLN`, `MARKPT` or `RANGE`).
- *Extra plotting subroutines* (`ARROW`, `BRKNBX`, `BRKNCL`, `SHDEBX` or `SHDECL`).

These subroutines are described in Chapter 3.

In addition to these general plotting subroutines, there are subroutines which draw 'special' charts – bar charts (`BARCHT`), histograms (`HSTGRM`) and pie charts (`PIECHT`). These subroutines are described in Chapter 4.

Labelling subroutines

Textual labels can be added to a picture at any stage after it has been started using a *labelling* subroutine. The subroutines which draw titles (or define key and caption areas) place them according to arguments which describe their positions – `TITLE7`, `DEFCAP`, `DEFKEY`, `SHKEYS`. Individual keys and captions can be added to pre-defined areas using `ADDCP7`, `PUTCP7`, `BLNKKY`, `BOTHK7`, `LINEK7`, `MARKK7` and `SHDEK7`. The labelling subroutines are described in Chapter 5.

Axis drawing subroutines

In addition to the *new picture* subroutines which also draw axes (`AXES7` and `POLAR7`) an individual axis can be drawn on an existing picture or labels can be added to an existing axis (`AXIS7`, `AXLAB7`, `AXTXT7`). These axis drawing subroutines are described in section 5.6.

2.3.2 Specification subroutines

Most *graphics* subroutines behave in a default manner which can be changed by calling the appropriate *specification* subroutine.

- Specify scales and coordinate interpretation
- Specify drawing characteristics
- Specify layout

Changes continue to apply to all plotting done until a further change is caused by another call. None of the *specification* subroutines draws anything.

Specify scales and coordinate interpretation

Special charts are usually scaled according to the data represented but scales for general plotting on x-y plots and polar plots usually need to be specified (by `POLAR7`, `SCALES`, `XSCALE`, `YSCALE` or `EQSCAL`) to accommodate the data to be plotted. If general graph plotting is started before any *specify scales* subroutine has been called, coordinates are assumed to be in centimetres relative to an origin in the bottom, left-hand corner of the picture. Scales can be changed after a picture has been started, if required.

For general graph plotting, the coordinate system used by SIMPLEPLOT is changed using `COORDS`. `EQSCAL` and `POLAR7` also affect the coordinate system.

Specify drawing characteristics

The *specify drawing* subroutines modify plotting characteristics. Some of these subroutines relate to specific tasks, others have a more general effect on drawing characteristics. For example, `CVTYPE` specifies the type of curve to be drawn by the curve drawing subroutines such as `BRKNAR`, but `SETPNS` specifies colours for all plotting.

The subroutines which specify axis characteristics (*eg.* `AXCRSS`, `AXGRID`, `AXRNGE`) affect `AXIS7`, `AXLAB7` and `AXTXT7` and the subroutines which draw axes as part of their operation (*eg.* `AXES7`, `BARCHT`, `HSTGRM` and `POLAR7`). The axis subroutines are described in section 5.6.

Specify layout

The default layout of pictures is tailored to the type of device but can be modified by prior calls of *specify layout* subroutines (*eg.* `PICSIZ`, `GROUP`). The layout subroutines are described in section 5.1.

2.3.3 Auxiliary subroutines

A few subroutines have no direct effect on other SIMPLEPLOT operations, but perform auxiliary tasks which may be needed in conjunction with plotting.

- *Data Manipulation* – `LIMEXC`, `LIMINC`, `POLOUT`, `POLY`.
- *Conversion subroutines* – `KNUMB`, `KREAL`, `KSCALE`.
- *Device control* – `DEVNAM`, `DEVNO`, `DIAGLV`, `ENDPLT`.

2.4 General plotting procedure

A simple SIMPLEPLOT program consists of four main steps:

1. Specify picture characteristics (including scales)
2. Start a new picture (with or without axis framework)
3. Plot data (which can be set up at any suitable stage prior to the actual plotting)
4. Terminate plotting

This procedure can become more complex if a program is to plot more than one picture, especially if each picture has different characteristics.

2.4.1 Plotting more than one picture

If more than one picture is to be drawn, each picture is drawn on a separate SIMPLEPLOT page unless a group of $n \times m$ pictures is specified. In this case a new SIMPLEPLOT page is started only when the group on the current page is completed. For multi-picture programs the procedure can be expressed as follows:

1. Specify global page and picture characteristics
2. For each page:
   - Specify individual page characteristics
   - For each picture:
     - Specify individual picture characteristics
     - Start new picture
     - Plot data

   All the plotting on one picture must be completed before another is started because SIMPLEPLOT only allows plotting on the current picture.
3. Terminate plotting

No formal termination of pictures and pages is required – a picture is considered finished when the next picture is started or plotting is terminated altogether. Similarly a page is finished when a new picture is started for which there is no more room in the group of pictures on the current page.

2.4.2 Order of subroutines

The order for calling SIMPLEPLOT subroutines is not strict and can be summarized in three simple rules:

1. A page must have been started before anything is drawn. When no page has been started, the start of the first (or only) picture starts a page.
2. *Specification* subroutines must be called before the affected *graphics* subroutines – when a *graphics* subroutine is called, the only user-specified details which affect its output are those which have already been specified in prior calls of relevant *specification* subroutines.

   Specification subroutines are called for three different types of use:

   - *Global* – to affect all plotting throughout the program;
   - *Picture* – to affect all of a picture;
   - *Local* – to affect a particular task within a picture.

Although specification subroutines can be called anywhere in a program, their effects remain until specifically cancelled; mistakes can be avoided by observing the following guidelines:

- insert *global* calls at the beginning of the program;
- insert *picture* calls before the call to the new picture subroutine;
- insert *local* calls before the task to which they relate.

If in doubt about the longer term effect of these subroutines, cancel them immediately after the task or picture to which they relate has been completed. Descriptions of specification subroutines include details of how to restore the default behaviour. Alternatively, `INITSP` resets all SIMPLEPLOT defaults.

3. `ENDPLT` must be called at the end to finish off properly – it empties any plotting buffers which are in use, closes the plotting device, triggers the output of diagnostic messages still outstanding for the last picture, and then outputs the final diagnostic messages.

Some care is needed to ensure that specifications which affect the size and position of a *new picture* are fixed before the picture is started. In particular, specifications which change the size of text can indirectly affect the layout.

2.5 The next step

How you use SIMPLEPLOT depends largely on the form your data takes and the sort of picture you need to produce. This manual describes how to plot six different types of picture (see Figure 2.2):

- x-y plots
- Polar charts
- Histograms
- Bar charts
- Pie charts
- Text pictures or diagrams

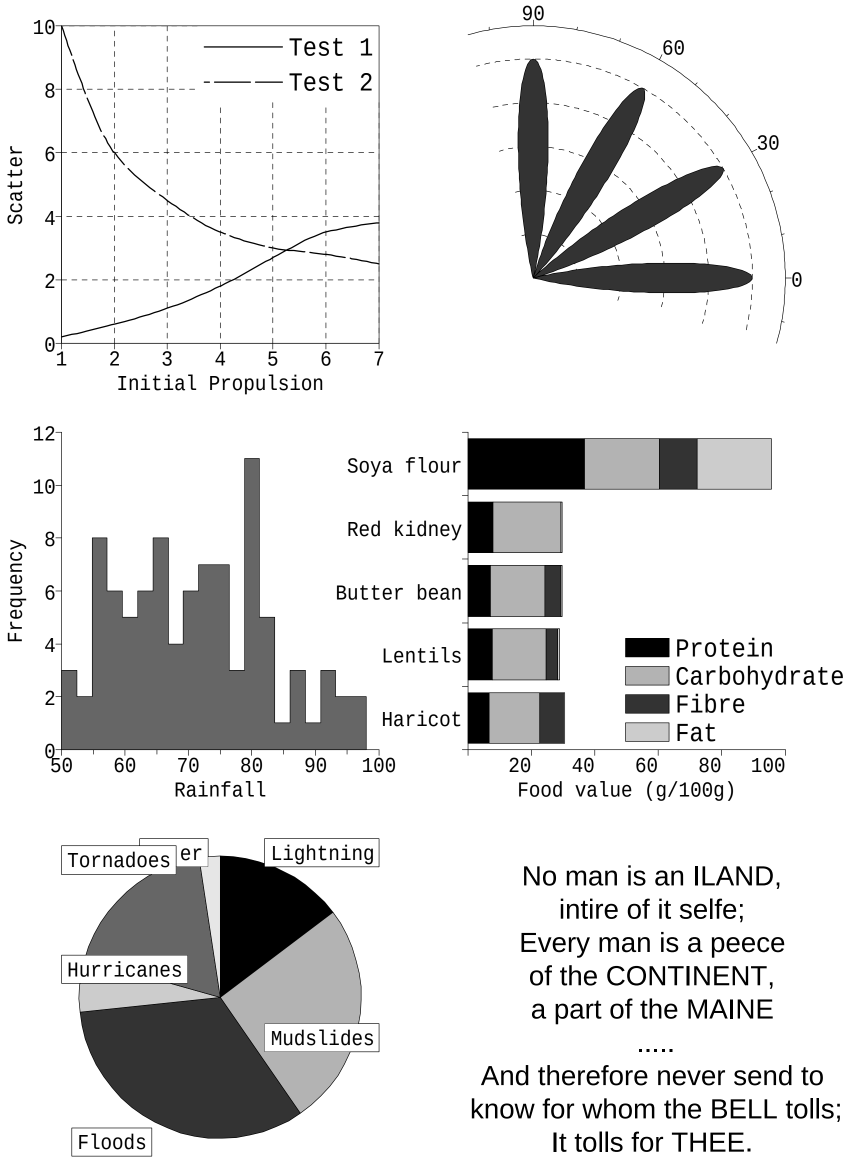


Figure 2.2 SIMPLEPLOT pictures

The six different types of picture are treated in one of two ways by SIMPLEPLOT. x-y plots, polar plots and diagrams fall into the category of *general graph plotting* described in Chapter 3. Histograms, bar charts and pie charts are *special charts* which are described in Chapter 4.

For each picture type, the following questions must be addressed:

- How do you set scales?
- How do you start a new picture?
- How do you draw the axis framework?
- How do you plot the data?
- What form must the data be in?

Some care may be needed when more than one type of picture is plotted from a single program.

The example programs in Chapters 3 and 4 illustrate the variety of plotting facilities for the different types of picture. Chapter 5 describes additional features which can be added to your picture and also how to control the basic features such as axes, shading and colour. The examples in Chapter 6 combine the additional features of Chapter 5 with the data plotting facilities described in Chapters 3 and 4.

3. General Graph Plotting

This chapter describes SIMPLEPLOT's graph drawing capabilities for plotting two-dimensional data:

3.1 Introduction to general graph plotting

General graph plotting embodies the more conventional use of SIMPLEPLOT for plotting graphs of data points which are described as coordinate pairs, (x, y) or (r, θ), depending on whether a Cartesian or polar coordinate system is required.

There are a number of alternative methods for specifying scales, plotting data and starting a new picture which can be chosen according to the needs of your application. These methods are described in this section but, more importantly, they are used throughout the examples in this chapter and in most examples in Chapter 6.

3.1.1 Data structures

SIMPLEPLOT is primarily designed to draw graphical representations of data; although it can perform many other graphic operations, it is especially designed to handle coordinate data in a variety of different ways to suit your needs. The following data configurations can be represented on an x-y plot (or polar plot):

- An array holding the $y = f(x)$ (or $r = f(\theta)$) function values corresponding to equally-spaced x (or θ) values:

  `BRKNAR` – draw a curve from function values in array with a broken line

  `MARKAR` – draw a set of symbols from function values in an array

  `SHDEAR` – shade a curve from function values in an array

  A selection of curve-drawing techniques is available.

- Parallel arrays which hold the x and y (or r and θ) coordinates of each point to be plotted:

  `BRKNCV` – draw a curve from arrays with a broken line

  `MARKCV` – draw a set of symbols from arrays of coordinates

  `SHDECV` – shade curve from arrays of coordinates

  A selection of curve-drawing techniques is available.

- Individual coordinates, (x, y) (or (r, θ)), to be plotted point-by-point:

  `BREAK` – force break between joined points

  `BRKNPT` – draw a straight line to a specified point with a broken line

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

  Curve-drawing techniques are not provided with these facilities.

- User-defined function, $y = f(x)$ (or $r = f(\theta)$), to be plotted over the limits of the existing plotting scale or over a pre-defined range:

  `BRKNFN` – draw a user defined function with a broken line

  `SHDEFN` – shade user defined function

- Similar sets of subroutines draw circles (`BRKNCL` or `SHDECL`) and rectangles (`BRKNBX` or `SHDEBX`).

All data are interpreted with respect to the current plotting scales and coordinate system. All these subroutines plot on the current picture and all plotting, except caption drawing (using `CP7LB` or `CP7PT`), is clipped to the picture boundary.

3.1.2 New pictures and framework

New pictures can be started with no framework or as part of a composite task which also draws the axis framework for a Cartesian x-y plot or a polar plot:

`AXES7` – start a new picture and draw axes

`NEWPIC` – start a new picture

`POLAR7` – specify scales, start a picture, draw polar axes

The type of picture started by `AXES7` and `NEWPIC` depends on the system of coordinate interpretation which is active when the new picture is started. If `EQSCAL` or `COORDS` has been called to specify polar units, then a new polar picture is started and `AXES7` draws a polar framework over full quadrants; otherwise, a Cartesian picture is started (for an x-y plot) and `AXES7` draws a pair of Cartesian axes.

3.1.3 Plotting scales and coordinate interpretation

For all general graph plotting on x-y plots and polar plots, the *specify scales* subroutines can be used to define both scale limits and coordinate interpretation. When scales have not been specified, coordinates are interpreted as centimetres relative to an origin in the bottom left-hand corner of the picture; these default scales are also referred to as *natural* scales. However, it is usually necessary to define scales which relate to the data to be plotted or to provide device independence. Linear and non-linear scales for Cartesian plotting can be specified using `SCALES`, `XSCALE` or `YSCALE`:

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

These different types of scales are illustrated in Figure 3.1.

It may be necessary to ensure that both the x and the y scales are proportional in order to represent similar quantities (*eg.* distance); also, identical x and y scaling are needed for polar plotting to avoid distortion. `EQSCAL` can be used to request proportional linear scales for either Cartesian or polar plotting. If the $x : y$ proportions of specified scale values differ from the picture proportions, identical x and y scaling is achieved by making one of the scales extend beyond the requested values at both ends. When scaling is controlled by `EQSCAL`, the default extent of Cartesian axes drawn is the requested ranges rather than the extended ranges which may be used internally to fit the picture size.

The relationship between the coordinate system and scales is defined by a scales subroutine or can be modified by a subsequent call of `COORDS`. Scales subroutines change both the coordinate system and the plotting scales and override any existing call to `COORDS`. `SCALES`, `XSCALE` and `YSCALE` set the coordinate interpretation to Cartesian whereas `EQSCAL` sets it to the specified units.

The specified scales and system of coordinate interpretation are used for every subsequent new picture until cancelled by a call of another scales subroutine.

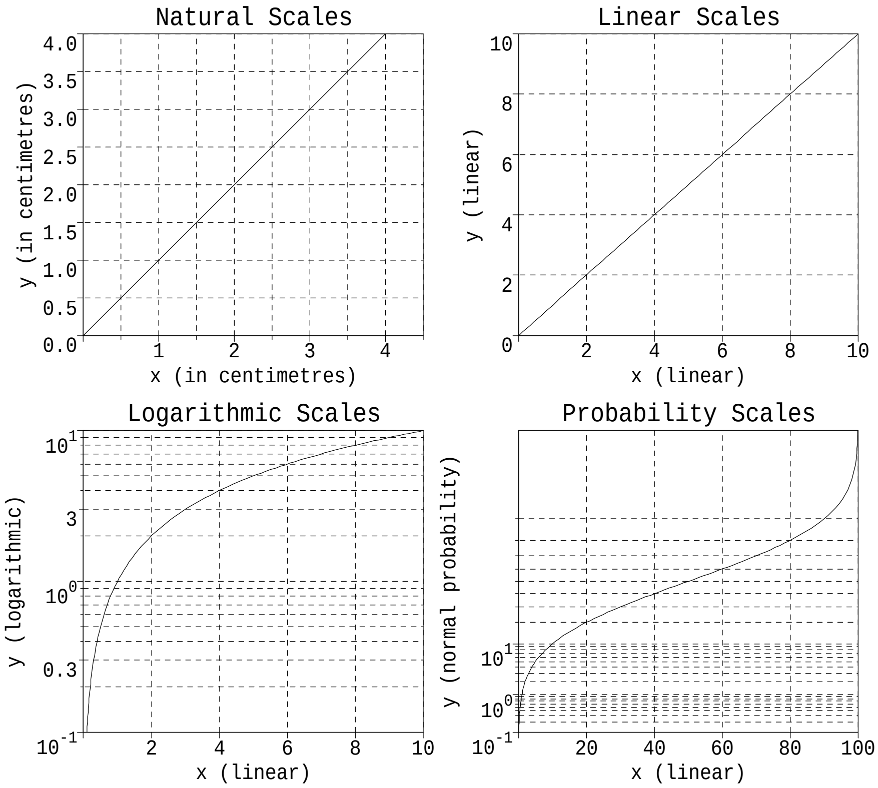


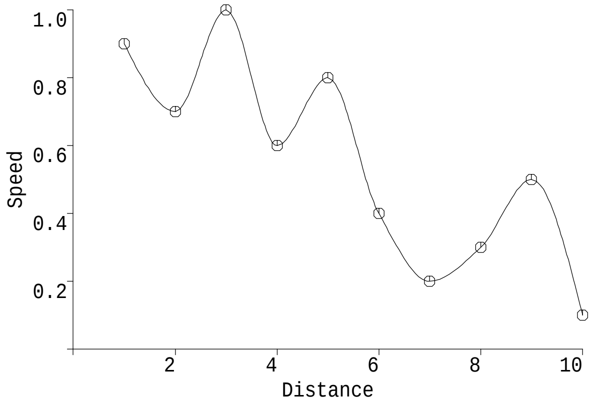
Figure 3.1 The function, $f(x) = x$, plotted on different types of scales

3.2 x-y plots

3.2.1 A simple x-y plot

This example illustrates how:

- to specify linear plotting scales,
- to draw a simple x-y plot from a set of function values, $y_i = f(x_i)$, stored in an array, these y values are plotted against equally-spaced x values.




```
      PROGRAM PRIMO2
      PARAMETER(X1=1.0,DX=1.0,NARR=10,NSTEP=1)
      REAL DARR(NARR)
      DATA DARR/0.9,0.7,1.0,0.6,0.8,0.4,0.2,0.3,0.5,0.1/
C specify plotting scales
      CALL SCALES(0.0,10.0,1, 0.0,1.0,1)
C start picture and draw pair of axes
      CALL AXES7('Distance','Speed')
C draw curve of data
      CALL BRKNAR(X1,DX,DARR,NARR,0)
C draw marker symbols at each data value
      CALL MARKAR(X1,DX,DARR,NARR,1,NSTEP)
      CALL ENDPLT
      END
```

Example 2. A simple x-y plot

Explanation of subroutines

`AXES7(CAPX,CAPY)` is a composite plotting subroutine which starts a new picture and draws a pair of axes annotated with the existing plotting scales. The first argument, `CAPX`, provides the caption for the horizontal axis, and `CAPY` provides the caption for the vertical axis. In this example, the horizontal axis is `"Distance"`, and the vertical axis is `"Speed"`.

`BRKNAR(X1,DX,DARR,NARR,LTYPE)` draws the curve which connects the following points:

(`X1`,`DARR[0]`), (`X1+DX`,`DARR[1]`),...
(`X1+(NARR-1)*DX`,`DARR[NARR-1]`)

These points represent a set of values from a single array whose elements hold function values $y_i = f(x_i)$, corresponding to equally-spaced x values, $x_i = x_1 + (i-1)\Delta x$.

The last argument, `LTYPE`, specifies which of the broken line patterns shown in Figure G.1 (see page 156) is to be used; `LTYPE`=0 represents a solid line.

`MARKAR(X1,DX,DARR,NARR,MKTYPE,NSTEP)` draws the marker symbol, `MKTYPE`, centred on each of the following points:

(X1, DARR[0]), (X1+DX, DARR[NSTEP]),
(X1+2*DX, DARR[2*NSTEP]),...

These points represent a set of values from a single array where the array elements hold function values $y_i = f(x_i)$ corresponding to equally-spaced x values, $x_i = x_1 + (i-1)\Delta x$.

The fifth argument, MKTYPE, specifies which of the marker symbols shown in Figure G.5 is to be used; standard SIMPLEPLOT marker symbols can be selected by MKTYPE=0–16, and Hershey marker symbols can be selected by MKTYPE=17–96 (see Figure G.5, page 159).

The final argument, NSTEP, specifies the interval between the points to be marked; this enables you to specify that only every nth point is to be plotted. In this example *every* point is plotted (NSTEP=1).

SCALES(XSTART,XSTOP,IXTYPE,YSTART,YSTOP,IYTYPE) specifies linear scales in x and y (IXTYPE=1, IYTYPE=1) for subsequent Cartesian plotting.

3.2.2 Different types of curves

SIMPLEPLOT can draw curves using a number of different curve-drawing methods which can be specified using `CVTYPE`:

`CVTYPE(ITYPE)` specifies the type of curve drawn by `BRKNAR` and `BRKNCV` (or `SHDEAR` and `SHDECV`). Five different types of curve drawing are available:

`ITYPE`	*Curve type*
1	A tight-fitting smooth curve through every data point, which is confined between the same limits as the data (default)
2	A loose-fitting smooth curve through every data point
3	Straight lines from point to point through the data; these lines are used to join points of a curve are physically straight even on a polar plot or non-linear scales
4	Linear regression
5	Quadratic regression

When `CVTYPE` has not been called, `BRKNAR`/`BRKNCV` (and `SHDEAR`/`SHDECV`) draw a *tight-fitting curve* (type 1).

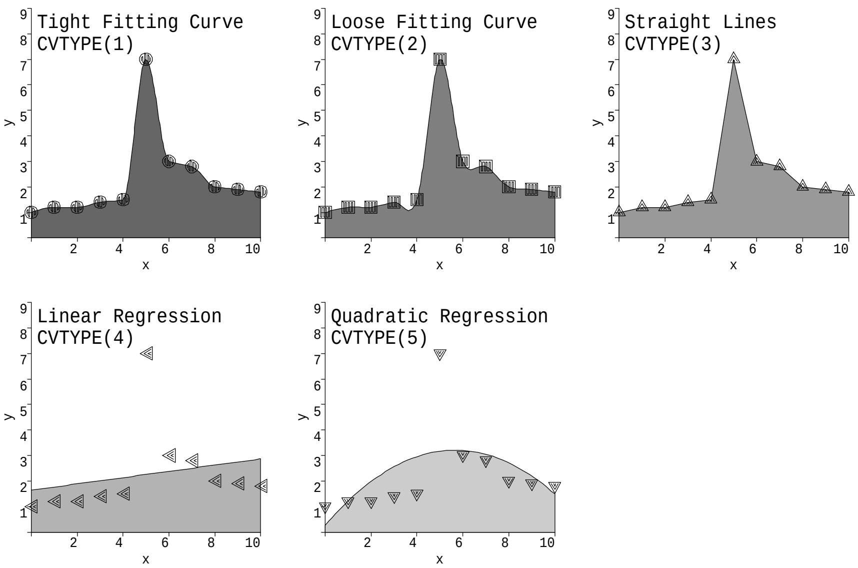


Figure 3.2 Different types of curves

Curve-fitting techniques

Tight-fitting curves are drawn using a cubic spline method developed by Judy Butland[1] to meet the needs of the graphical representation of data. The resultant curve passes through every data point, increases where data increases, decreases where data decreases, and is confined between the same limits as the data.

Because of these constraints, sudden changes in data will result in sharp corners in the curve, and the curve is not invariant under rotation.

Loose-fitting curves are drawn using a similar cubic spline technique developed by McConologue.[2]

[1] Butland, J. (1980) *A method of interpolating reasonable-shaped curves through any data*, in Proc. Computer Graphics 80, pp. 409–422, (Online Publications, Middlesex, U.K.)

[2] McConologue, D. J. (1970) *A quasi-intrinsic scheme for passing a smooth curve through a discrete set of points*, Comp. J., vol. 13, pp. 392–396. and McConologue, D. J. (1971) *Algorithm 66 – An automatic French-curve procedure for use with an incremental plotter*, Comp. J., vol. 14, pp. 207–209.

The resultant curve should not have sharp corners, passes through every data point, and is invariant under rotation.

But, the curve may extend higher (or lower) than data at turning points, and a sudden sharp change of direction in data may be shown by an extraneous loop in the curve.

This type of curve is suitable for drawing pictures, or plotting data where interpolation of curve behaviour can be justified.

For a general review of curve-drawing methods, please refer to Brodlie.[3]

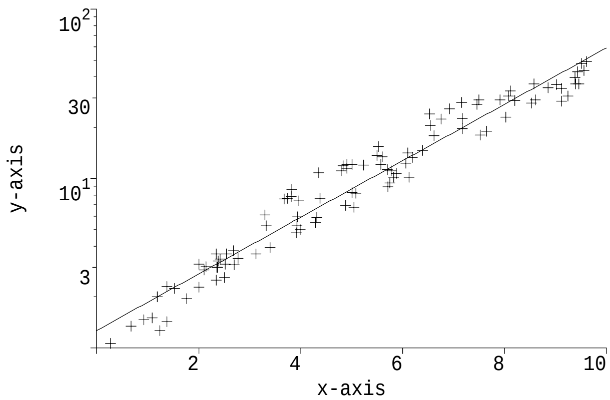
Linear and quadratic regression curves are first and second order functions fitted to the data points. They enable you to illustrate the association between the dependent variable, x, and the independent variable, $y = f(x)$.

[3] Brodlie, K.W. (1985) *Methods of Drawing Curves* in *Fundamental Algorithms for Computer Graphics*, ed. Earnshaw, (Springer-Verlag), pp. 303–323.

3.2.3 A scattergram

This example illustrates how:

- to specify logarithmic plotting scales,
- to draw a set of marker symbols at coordinates stored in parallel arrays,
- to plot a correlation curve through the data points using linear regression.




```
      PROGRAM PRIM03
      PARAMETER(NARR=100,NSTEP=1)
      REAL XARR(NARR),YARR(NARR)
C read (x,y) data from file
      OPEN(UNIT=10,FILE='prim03.dat',STATUS='OLD')
      DO 10 I=1,NARR
        READ(10,*)XARR(I),YARR(I)
 10   CONTINUE
      CLOSE(10)
C specify plotting scales
      CALL SCALES(0.0,10.0,1,1.0,100.0,2)
C start picture and draw framework
      CALL AXES7('x-axis','y-axis')
C draw symbol at every point
      CALL MARKCV(XARR,YARR,NARR,3,NSTEP)
C plot linear regression curve through data
      CALL CVTYPE(4)
      CALL BRKNCV(XARR,YARR,NARR,0)
      CALL ENDPLT
      END
```

Example 3. Marking points with symbols

Explanation of subroutines

`BRKNCV(XARR,YARR,NARR,LTYPE)` draws the curve which connects the following points:

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

The data points are interpreted in terms of the plotting scales which are operating when `BRKNCV` is called.

The last argument, `LTYPE`, specifies which of the broken line patterns shown in Figure G.1 (see page 156) is to be used. `LTYPE`=0 represents a solid line.

`CVTYPE(4)` specifies that a linear regression curve is to be plotted.

`MARKCV(XARR,YARR,NARR,MKTYPE,NSTEP)` draws the marker symbol, `MKTYPE`, centred on each of the following points:

(XARR(1),YARR(1)), (XARR(1+NSTEP), YARR(1+NSTEP)), ...
(XARR(1+2*NSTEP), YARR(1+2*NSTEP))

MKTYPE specifies which marker symbol is to be used; standard SIMPLEPLOT markers are selected by MKTYPE=0–16, and Hershey markers are selected by MKTYPE=17–96 (see Figure G.5, page 159). NSTEP specifies the interval between the points to be marked; this enables you to specify that only every nth point is to be plotted. In this example every point is plotted (NSTEP=1).

SCALES(XSTART,XSTOP,IXTYPE,YSTART,YSTOP,IYTYPE) with IXTYPE=1 and IYTYPE=2, specifies a linear x-scale and a logarithmic y-scale for subsequent Cartesian plotting. Care must be taken to ensure that a logarithmic (or normal probability) scale does not include zero. If zero is included, SIMPLEPLOT uses a linear scale and issues the following diagnostic:

`(Range through 0; linear scale used)`[1]

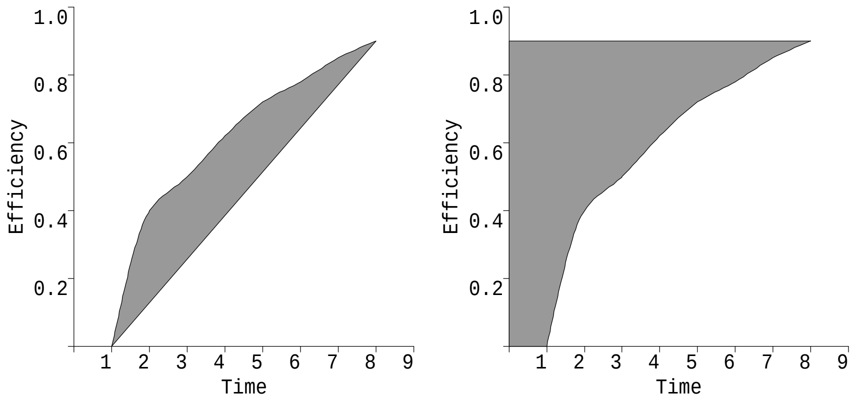
Similarly a normal probability scale must lie within the range 0.0–100.0.

prim03.dat is included in the SIMPLEPLOT software distribution kit.

3.2.4 Different shaded areas for open curves

This example illustrates how:

- to use alternative strategies for shading the area 'under' an open curve,
- to shade under a curve defined by function values held in a single array,
- to draw a group of pictures.




```
      PROGRAM PRIM04
      PARAMETER(NARR=8,X1=1.0,DX=1.0)
      REAL DARR(NARR)
      DATA DARR/0.0,0.4,0.5,0.62,0.72,0.78,0.85,0.9/
C specify layout with 2 pictures side-by-side
      CALL GROUP(2,1)
C specify plotting scales
      CALL SCALES(0.0,REAL(NARR+1),1, 0.0,1.0,1)
C start first picture and draw axes
      CALL AXES7('Time','Efficiency')
      CALL SHTYPE(0)
      CALL SHDEAR(X1,DX,DARR,NARR,7)
C start second picture and draw axes
      CALL AXES7('Time','Efficiency')
      CALL SHTYPE(2)
      CALL SHDEAR(X1,DX,DARR,NARR,7)
      CALL ENDPLT
      END
```

Example 4. Shaded areas under unclosed curves

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 in which the pictures are plotted is from left to right and from top to bottom of the page.

`SHDEAR(X1,DX,DARR,NARR,ISHADE)` shades the area enclosed by the curve that connects the following points:

(`X1`, `DARR(1)`), (`X1+DX`, `DARR(2)`),...
(`X1+NARR*DX`, `DARR(NARR)`)

These points represent a set of values from a single array where the array elements hold function values $y_i = f(x_i)$ corresponding to equally-spaced x values, $x_i = x_1 + (i-1)\Delta x$.

The type of curve depends on `CVTYPE` and, if the curve is not closed, the area shaded depends on `SHTYPE`. `ISHADE` specifies which shading pattern is to be used (see Figures G.2 and G.3, page 157).

`SHTYPE(ITYPE)` specifies the type of closure for shading unclosed curves:

ITYPE	*Shading area for unclosed curves*
0	Area contained by joining last point of curve to first with a straight line
1	Area between the curve and the horizontal axis or bottom of picture (default)
2	Area between the curve and the vertical axis or left-hand edge of the picture

`SHTYPE` has no effect on regression curves (`CVTYPE(4)` and (5)), polar plots, or curves of user-defined functions (see section 3.2.5).

Similar subroutines

`SHDECV(XARR,YARR,NARR,ISHADE)` shades the area enclosed by the curve corresponding to the following points:

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

A closed curve can be obtained by using data in which `XARR(NARR)` = `XARR(1)` and `YARR(NARR)` = `YARR(1)` (see Example 34).

3.2.5 Curves from user-defined functions

This example illustrates how:

- to draw curves of user-defined functions,
- to avoid discontinuities in user-defined functions.

Explanation of subroutines

`BRKNFN(FUNX,LTYPE)` evaluates the function whose name is specified as the argument, and draws the curve of the function over the range of the picture. The intervals at which the function is evaluated are determined by SIMPLEPLOT and vary with the curvature of the curve to ensure that a smooth drawing is produced.

The function must be a `float` function with one `float` argument; the function argument is interpreted in terms of the x-scale of the picture, and the computed function value in terms of the y-scale. As the argument of `BRKNFN` is the name of a function, FORTRAN requires that it be declared in advance of it's usage.

The final argument, `LTYPE`, specifies the type of broken line pattern from Figure G.1 (see page 156) to be used. `LTYPE`$=-1$ represents a broken line with short dashes.

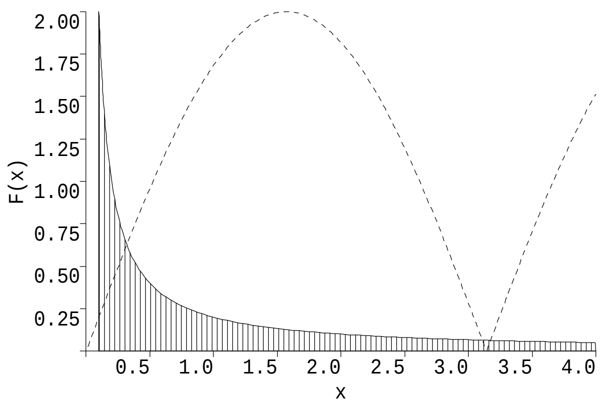
`FNRNGE(START,STOP)` specifies a restricted range of the independent variable over which subsequent calls of `BRKNFN` and `SHDEFN` are to be plotted. SIMPLEPLOT stores the values of the arguments, and they are only interpreted in terms of the plotting scales which exist when `BRKNFN` or `SHDEFN` is called.

In this example, the use of `FNRNGE` prevents the evaluation of `FUN2` at values which would cause the program to fail.

By default, `BRKNFN` and `SHDEFN` plot curves over the full range of the scale of the independent variable; this default can be restored by calling `FNRNGE` with equal arguments (*ie.* `START`=`STOP`).

`SHDEFN(FUNX,ISHADE)` shades the area enclosed by the curve of a used-defined function, `FUNX`. If the curve is not closed, the shaded area is constrained by the x-axis or the bottom of the picture; `SHTYPE` has no effect on `SHDEFN`.

The final argument, `ISHADE`, specifies which shading pattern is to be used (see Figures G.2 and G.3, page 157).



```
      PROGRAM PRIM05
C declare functions
      EXTERNAL FUN1,FUN2
C specify plotting scales
      CALL SCALES(0.0,4.0,1,0.0,2.0,1)
C start picture and draw pair of axes
      CALL AXES7('x','F(x)')
C draw broken (dashed) curve of function FUN1
      CALL BRKNFN(FUN1,-1)
C draw shaded curve of FUN2 avoiding X=0.0
      CALL FNRNGE(0.1,4.0)
      CALL SHDEFN(FUN2,11)
      CALL ENDPLT
      END
C first user-defined function
      FUNCTION FUN1(X)
      FUN1=2.0*ABS(SIN(X))
      END
C second user-defined function
      FUNCTION FUN2(X)
      FUN2=0.2/ABS(X)
      END
```

Example 5. Curves from functions

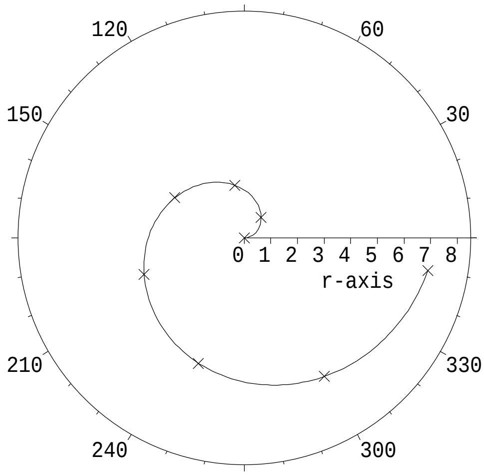
3.3 Polar plots

On a polar picture, the data plotting subroutines interpret coordinates as (r, θ) instead of (x, y); θ can be interpreted in degrees or radians.

3.3.1 A simple polar plot

This example illustrates how:

- to draw a polar chart,
- to plot a simple polar graph from a set of radial values stored in an array.




```
      PROGRAM PRIM06
      PARAMETER(RADIUS=8.5,THETA1=0.0,DTHETA=50.0,NARR=8,NSTEP=1)
      REAL RARR(NARR)
      DATA RARR/0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0/
C start picture, specify scales, draw polar framework
      CALL POLAR7(RADIUS,'r-axis')
C draw marker symbols
      CALL MARKAR(THETA1,DTHETA,RARR,NARR,4,NSTEP)
C plot solid curve of data
      CALL BRKNAR(THETA1,DTHETA,RARR,NARR,0)
      CALL ENDPLT
      END
```

Example 6. A simple polar plot

Explanation of subroutines

`BRKNAR(THETA1,DTHETA,RARR,NARR,LTYPE)` draws the curve which connects the following points:

> (`THETA1`, `RARR(1)`), (`THETA1+DTHETA`, `RARR(2)`),...
> (`THETA1+NARR*DTHETA`, `RARR(NARR)`)

These points represent a set of values from a single array where the array elements hold function values $r_i = f(\theta_i)$ corresponding to equally-spaced θ values, $\theta_i = \theta_1 + (i-1)\Delta\theta$.

`LTYPE` specifies which of the broken line patterns shown in Figure G.1 (see page 156) is to be used. 0 represents a solid line.

`MARKAR(THETA1,DTHETA,RARR,NARR,MKTYPE,NSTEP)` draws the marker symbol, `MKTYPE`, centred on each of the following points:

> (`THETA1`,`RARR(1)`), (`1+THETA1+DTHETA`, `RARR(1+NSTEP)`),...
> (`1+THETA1+2*DTHETA`, `RARR(1+2*NSTEP)`) , ...

These points represent a set of values from a single array where the array elements hold function values $r_i = f(\theta_i)$ corresponding to equally-spaced θ values, $\theta_i = \theta_1 + (i-1)\Delta\theta$. Standard SIMPLEPLOT marker symbols can be selected by `MKTYPE`=0–16, and Hershey marker symbols can be selected by `MKTYPE`=17–96 (see Figure G.5, page 159).

`POLAR7(RADIUS,CAP)` starts a new picture, sets the scales, sets the coordinate interpretation to *polar in degrees* and draws a polar framework. In this example, the maximum radial unit is specified by `RADIUS`=8.5.

`CAP` provides a caption for the horizontal axis.

Similar subroutines

`BRKNCV(RARR,THARR,NARR,LTYPE)` draws the curve which connects the following points:

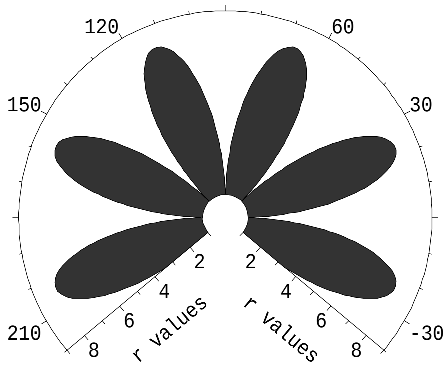
(`RARR(1)`,`THARR(1)`), (`RARR(2)`,`THARR(2)`), ...
(`RARR(NARR)`,`THARR(NARR)`)

where `THARR` holds the values of angles corresponding to the radial values in `RARR`. The final argument specifies the broken line pattern, `LTYPE`.

3.3.2 A part-cycle polar plot

This example illustrates how:

- on a polar picture, a user-defined function is interpreted as $r = f(\theta)$,
- to change coordinate interpretation,
- to specify both radial and angular scales.




```
      PROGRAM PRIM07
      PARAMETER(RSTART=1.0,RSTOP=9.0,THSTRT=320.0,THSTOP=220.0)
      PARAMETER(RADIUS=5.0)
      EXTERNAL FUN
C specify plotting scales in degrees
      CALL EQSCAL(RSTART,RSTOP,THSTRT,THSTOP,1)
C start picture and draw polar framework
      CALL POLAR7(RADIUS,'r values')
C change coordinate interpretation to polar in radians
      CALL COORDS(2)
      CALL SHDEFN(FUN,3)
      CALL ENDPLT
      END
C user-defined polar function
      FUNCTION FUN(THETA)
      FUN=7.0 * ABS(SIN(4.0*THETA)) + 1.0
      END
```

Example 7. Part cycle polar plotting

Explanation of subroutines

`COORDS(IUNITS)` changes the interpretation of coordinates for subsequent plotting subroutine arguments; `IUNITS` can take the non-negative values as described for `EQSCAL` (0=Cartesian, 1=polar with θ in degrees, 2=polar with θ in radians). `COORDS` does not affect the current plotting scales but it changes the coordinate system used to refer to existing scales. The plotting units may be switched by `COORDS`, within the plotting of a picture, without cancelling `EQSCAL`.

`EQSCAL(RSTART,RSTOP,THSTRT,THSTOP,IUNITS)` specifies proportional linear scales for Cartesian or polar plotting. Identical x and y linear scaling are needed for polar plotting and, in this example, `EQSCAL` is used to specify reduced angular and radial scales. 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
		RSTART,RSTOP	THSTRT,THSTOP
1	Polar	Radial units	Angles in degrees
2	Polar	Radial units	Angles in radians

`EQSCAL` also controls the type of any subsequent new pictures. Once `EQSCAL` is active, the first argument (`RADIUS`) of `POLAR7` is ignored in favour of `RSTOP` in the call to `EQSCAL`. 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.

Once called, `EQSCAL` is active until cancelled by another scales subroutine (*eg.* `SCALES`) or by calling `EQSCAL` with `IUNITS`$=-1$ or with one scale limit equal to the other.

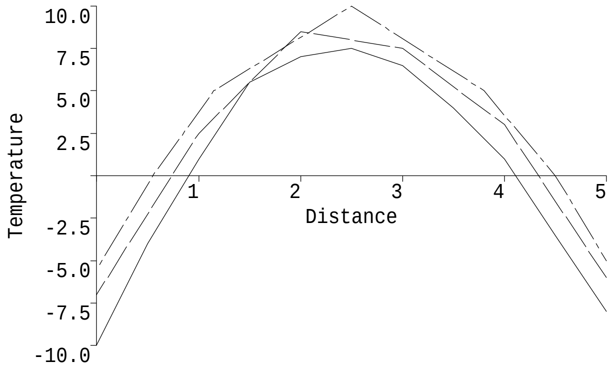
`SHDEFN(FUN,ISHADE)` evaluates the function whose name is specified as the argument, and draws the curve of the function over the range of the plotting scales. The function argument is interpreted in terms of the θ-scale of the picture, and the computed function value in terms of the r-scale. `ISHADE` specifies the type of shading pattern to be used; refer to Figure G.3, page 157, for examples of monochrome software shading patterns.

3.4 Point-by-point plotting and annotation

After a call of a *new picture* subroutine, individual data points can be plotted or annotation drawn at specified coordinates. The point-by-point plotting subroutines operate in exactly the same way as the subroutines described in sections 3.2 and 3.3 for plotting sets of data. Each subroutine has arguments representing one or two pairs of coordinates; each coordinate pair is interpreted as (x, y) or (r, θ) according to the current coordinate system, and is plotted with respect to the current plotting scales.

3.4.1 Straight lines from point to point

This example illustrates how to plot data point-by-point as each point is read from a file.




```
      PROGRAM PRIM08
      OPEN(UNIT=10,FILE='prim08.dat',STATUS='OLD')
C specify plotting scales
      CALL SCALES(0.0,5.0,1,-10.0,10.0,1)
C start picture and draw pair of axes
      CALL AXES7('Distance','Temperature')
      READ(10,*)NCURVE
C read and plot data sets
      DO 20 JCURVE=1,NCURVE
C read number of points in set
        READ(10,*)NPTS
        DO 10 I=1,NPTS
C read pair of coordinates
          READ(10,*)X,Y
C draw straight line to point
          CALL BRKNPT(X,Y,JCURVE-1)
 10     CONTINUE
C raise pen after curve is complete
        CALL BREAK
 20   CONTINUE
      CALL ENDPLT
      CLOSE(10)
      END
```

Example 8. Point-by-point plotting

Explanation of subroutines

BREAK causes the next call of BRKNPT to be treated as a first point, so that no line is drawn from the end of one piece of drawing to the beginning of the next.

`BRKNPT(X,Y,LTYPE)` draws straight lines from one data point to another; the first call of `BRKNPT` on a picture causes a move to the point without any drawing. The final argument, `LTYPE`, specifies which of the broken line patterns shown in Figure G.1 (page 156) is to be used.

The data file, `prim08.dat`, is included as part of the SIMPLEPLOT software distribution kit.

In order to plot straight lines from point to point, you must establish a sequence using successive calls to `BRKNPT`. For example, to join two points using a `BRKNPT`/`BREAK` sequence, call `BRKNPT` to move to the first point, call `BRKNPT` to draw a straight line to the second point and then call `BREAK` to lift the pen and break the sequence.

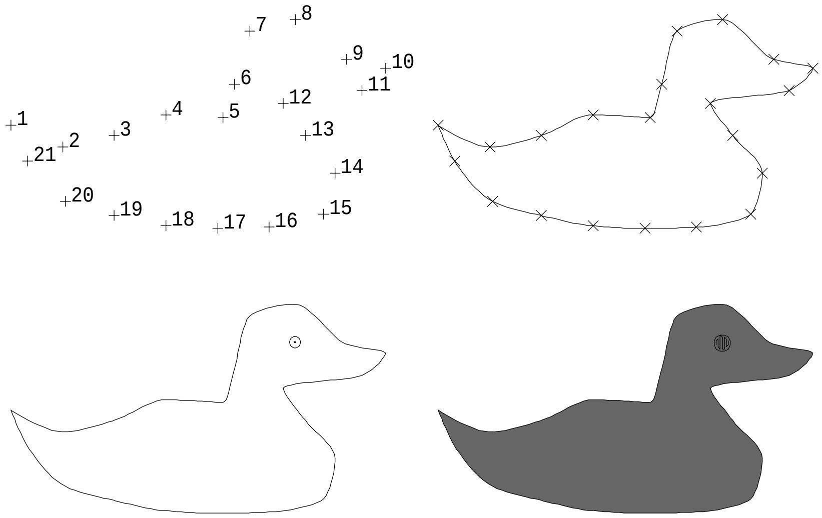
This sequence can be broken explicitly using `BREAK` or implicitly by calling any other graphics subroutine. The one exception to this rule is `MARKPT` which can be included in a point-by-point plotting sequence to draw a symbol.

For example, to draw a symbol as part of a `BRKNPT`/`BREAK` sequence, call `BRKNPT` to move to a point, call `MARKPT` to draw a symbol at the same point, call `BRKNPT` to draw a straight line to the next point *etc.*

3.4.2 Marking individual points

This example illustrates how:

- to mark points with or without captions,
- to start a picture without drawing any axis framework,
- to use proportional scales to avoid distortion.



Explanation of subroutines

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

`EQSCAL(XSTART,XSTOP,YSTART,YSTOP,IUNITS)` with `IUNITS`=0, specifies similar linear scales in x and y which avoid distortion.

`KNUMB(IVAL,&STR)` converts the integer, `IVAL`, into its equivalent `char *` representation, suitable for writing as text.

`MARKPT(X,Y,MKTYPE)` draws the marker symbol, `MKTYPE` at (`X`,`Y`). Standard SIMPLEPLOT symbols can be selected by `MKTYPE`=0–16, and Hershey symbols can be selected by `MKTYPE`=17–96 (see Figure G.5, page 159).

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

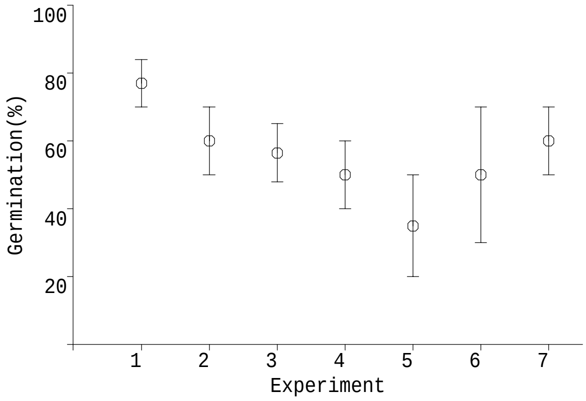
Similar subroutines

`CP7LB(X,Y,CAP)` draws the caption, `CAP`, to the right of the point (`X`,`Y`).

```
      PROGRAM PRIM09
      PARAMETER(NPTS=22,XEYE=22.0,YEYE=14.0)
      CHARACTER*10 STR
      REAL XDUCK(NPTS),YDUCK(NPTS)
      DATA XDUCK/0.0,4.0,8.0,12.0,16.4,17.3,18.5,22.0,26.0,
     +      29.0,27.2,21.1,22.8,25.1,24.2,20.0,16.0,
     +      12.0,8.0,4.2,1.3,0.0/
      DATA YDUCK/8.8,7.1,8.0,9.6,9.4,12.0,16.1,17.0,13.9,
     +      13.2,11.5,10.5,8.0,5.1,1.9,0.9,0.8,
     +      1.0,1.8,2.9,6.0,8.8/
C set up layout and scales
      CALL GROUP(2,2)
      CALL EQSCAL(-2.0,31.0,1.0,17.0,0)
C start 1st picture, mark points with symbol and caption
      CALL NEWPIC
      DO 10 I=1,NPTS-1
        CALL KNUMB(I,STR)
        CALL CP7PT(XDUCK(I),YDUCK(I),3,STR)
 10   CONTINUE
C start 2nd picture, mark points with symbol and draw curve
      CALL NEWPIC
      DO 20 I=1,NPTS-1
        CALL MARKPT(XDUCK(I),YDUCK(I),4)
 20   CONTINUE
      CALL BRKNCV(XDUCK,YDUCK,NPTS,0)
C start 3rd picture, draw curve through points, mark eye
      CALL NEWPIC
      CALL BRKNCV(XDUCK,YDUCK,NPTS,0)
      CALL MARKPT(XEYE,YEYE,58)
C start 4th picture, shade area and mark eye
      CALL NEWPIC
      CALL SHDECV(XDUCK,YDUCK,NPTS,5)
      CALL MARKPT(XEYE,YEYE,35)
      CALL ENDPLT
      END
```

Example 9. Marking individual points

3.4.3 Range plotting




```
      PROGRAM PRIM10
      PARAMETER(NEXP=7)
      REAL YARR1(NEXP),YARR2(NEXP)
      DATA YARR1/70.0,50.0,48.0,40.0,20.0,30.0,50.0/
      DATA YARR2/84.0,70.0,65.0,60.0,50.0,70.0,70.0/
C specify plotting scales
      CALL SCALES(0.0,REAL(NEXP+0.5),1,0.0,100.0,1)
C start picture and draw pair of axes
      CALL AXES7('Experiment','Germination(%)')
C plot range and mean for each experiment
      DO 100 I=1,NEXP
        YMEAN=(YARR1(I)+YARR2(I))/2
        CALL MARKPT(REAL(I),YMEAN,1)
        CALL RANGE(REAL(I),YARR1(I),REAL(I),YARR2(I))
 100  CONTINUE
      CALL ENDPLT
      END
```

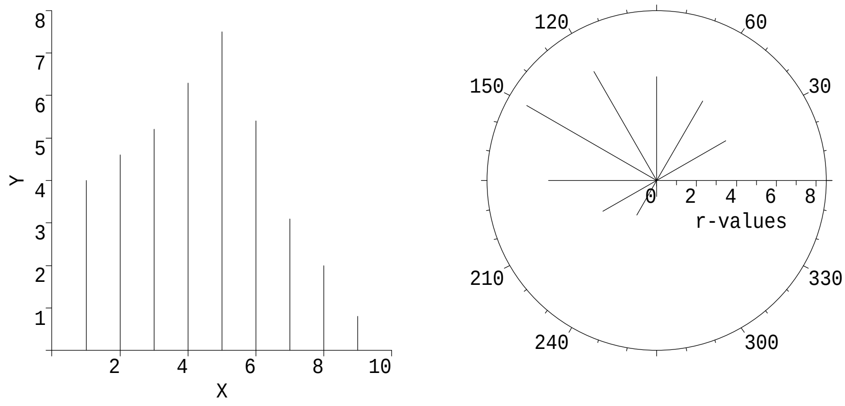
Example 10. Range plotting

Explanation of subroutines

`RANGE(X1,Y1,X2,Y2)` draws a line between (`X1`,`Y1`) and (`X2`,`Y2`) representing a range of values, with short perpendicular lines across the ends. The line joining the two points is a linear function, which may be curved when plotted on non-linear scales or polar plots.

3.4.4 Line diagrams

This example illustrates how to draw line diagrams (and rose diagrams).




```
      PROGRAM PRIM11
      PARAMETER(NPTS=9)
      REAL DARR(NPTS)
      DATA DARR/4.0,4.6,5.2,6.3,7.5,5.4,3.1,2.0,0.8/
      CALL GROUP(2,1)
C specify plotting scales, start first picture and draw axes
      CALL SCALES(0.0,10.0,1,0.0,8.0,1)
      CALL AXES7('X','Y')
C plot data as vertical lines
      DO 10 I=1,NPTS
        CALL DRAWLN(REAL(I),DARR(I))
 10   CONTINUE
C start second picture, specify scale, draw polar framework
      CALL POLAR7(8.5,'r-values')
C plot data as radial lines
      DO 20 I=1,NPTS
        CALL DRAWLN(DARR(I),REAL(I)*30.0)
 20   CONTINUE
      CALL ENDPLT
      END
```

Example 11. Line diagrams

Explanation of subroutines

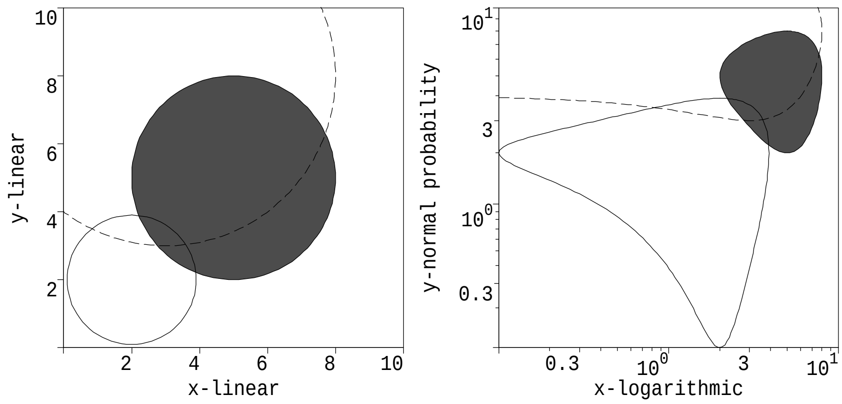
`DRAWLN(X,Y)` draws a straight line between the specified point and zero. On a Cartesian picture, the straight line is the vertical line between (x, y) and $(x, 0.0)$; on a polar picture, a radial line is drawn from (r, θ) to $(0.0, 0.0)$.

3.5 Extra plotting subroutines

In addition to the subroutines for plotting data, either as a set or as individual points, SIMPLEPLOT includes some more subroutines which interpret scales and the coordinate system in the same way as for all Cartesian and polar plotting.

3.5.1 Circles

This example illustrates how circles are plotted on similar linear scales and on non-linear scales.




```
      PROGRAM PRIM12
      CALL GROUP(2,1)
C specify similar linear plotting scales
      CALL EQSCAL(0.0,10.0,0.0,10.0,0)
C start first picture and draw axes
      CALL AXES7('x-linear','y-linear')
C draw one shaded circle and two unshaded circles
      CALL SHDECL(5.0,5.0,3.0,4)
      CALL BRKNCL(2.0,2.0,1.9,0)
      CALL BRKNCL(3.0,8.0,5.0,-2)
C draw perimeter
      CALL PERIM
C specify non-linear scales
      CALL SCALES(0.1,10.0,2,0.1,10.0,3)
C start second picture and draw axes
      CALL AXES7('x-logarithmic','y-normal probability')
C draw one shaded circle and two unshaded circles
      CALL SHDECL(5.0,5.0,3.0,4)
      CALL BRKNCL(2.0,2.0,1.9,0)
      CALL BRKNCL(3.0,8.0,5.0,-2)
C draw perimeter
      CALL PERIM
      CALL ENDPLT
      END
```

Example 12. Circles

Explanation of subroutines

`BRKNCL(XCENT,YCENT,RADIUS,LTYPE)` draws the circle (or the part of the circle which lies within the range of the plotting scales) centred at point (`XCENT`,`YCENT`), with radius `RADIUS`, using broken line pattern `LTYPE`. The resulting shape is only geometrically circular when both scales are equal and linear.

The last argument, `LTYPE`, specifies which of the broken line patterns shown in Figure G.1 (see page 156) is to be used; 0 represents a solid line.

`EQSCAL(XSTART,XSTOP,YSTART,XSTOP,IUNITS)` with `IUNITS`=0, specifies similar linear scales for Cartesian plotting. When the $x : y$ proportions of scales specified by `EQSCAL` differ from the picture proportions, identical x and y scaling is achieved by making one of the scales extend beyond the requested values at both ends.

In these circumstances, the default extent of axes drawn is the requested ranges rather than the extended ranges used to fit the picture size; thus, if `PERIM` is used to draw a box around the current picture, the box may be drawn beyond the limits defined by the axes. The next example, Example 13, illustrates how to draw a box at the scale limits.

`PERIM` draws a box around the current picture.

`SHDECL(XCENT,YCENT,RADIUS,ISHADE)` draws the shaded circle (or the part of the circle which lies within the range of the plotting scales) centred at point (`XCENT`,`YCENT`) with radius, `RADIUS`. The resulting shape is only geometrically circular when both scales are equal and linear.

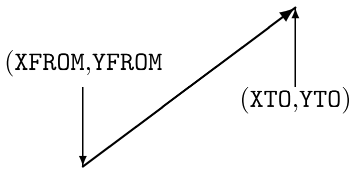
The final argument, `ISHADE`, specifies which shading pattern is to be used (see Figures G.2 and G.3, page 157).

3.5.2 Boxes and arrows

This example illustrates how to draw boxes and arrows on both polar and x-y plots.

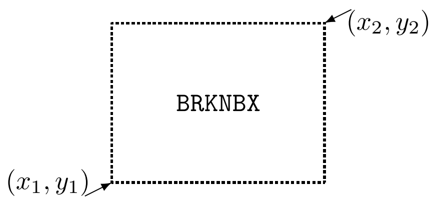
Explanation of subroutines

`ARROW(XFROM,YFROM,XTO,YTO)` draws a line from (`XFROM`,`YFROM`) to (`XTO`,`YTO`) with an arrow head at (`XTO`,`YTO`). The position and direction of the arrow are completely defined by the four arguments which specify the coordinates of the end and the head of the arrow in terms of the current plotting scales:



If `XFROM`=`XTO` and `YFROM`=`YTO`, nothing is drawn. An arrow shaft is always physically straight even on polar plots.

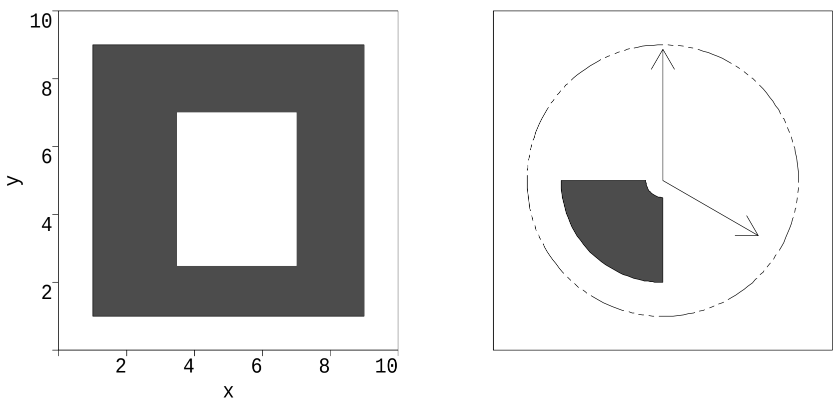
`BRKNBX(X1,Y1,X2,Y2,LTYPE)` draws a box extending from x_1 to x_2 by y_1 to y_2; if $x_1 = x_2$ or $y_1 = y_2$, nothing is drawn. The points (x_1, y_1) and (x_2, y_2) are specified in units of the current plotting scales':



The last argument, `LTYPE`, specifies which of the broken line patterns shown in Figure G.1 (see page 156) is to be used.

`SHDEBX(X1,Y1,X2,Y2,ISHADE)` draws a shaded box extending from x_1 to x_2 by y_1 to y_2. The final argument, `ISHADE`, specifies which shading pattern is to be used (see Figures G.2 and G.3, page 157).

The second shaded box is drawn using shading pattern, `ISHADE`=0. This pattern can be used to create a rub-out effect on those devices which can achieve it (as illustrated here); on other devices, shading is omitted. For further details about this effect please refer to section 5.10, page 87.



```
      PROGRAM PRIM13
      CALL GROUP(2,1)
C specify proportional Cartesian scales
      CALL EQSCAL(0.0,10.0,0.0,10.0,0)
C start first picture and draw axes
      CALL AXES7('x','y')
C draw shaded box
      CALL SHDEBX(1.0,1.0,9.0,9.0,4)
C draw shaded box using rubout (if available)
      CALL SHDEBX(7.0,7.0,3.5,2.5,0)
C draw box at scale limits
      CALL BRKNBX(0.0,0.0,10.0,10.0,0)
C specify polar scales and start second picture
      CALL EQSCAL(0.0,10.0,0.0,360.0,1)
      CALL NEWPIC
C draw circle and two arrows
      CALL BRKNCL(0.0,0.0,8.0,4)
      CALL ARROW(0.0,0.0,7.75,90.0)
      CALL ARROW(0.0,0.0,6.5,-30.0)
C draw shaded polar 'box'
      CALL SHDEBX(1.0,180.0,6.0,270.0,4)
C draw box at scale limits
      CALL COORDS(0)
      CALL BRKNBX(-10.0,-10.0,10.0,10.0,0)
      CALL ENDPLT
      END
```

Example 13. Boxes and arrows

4. Special Charts

This chapter describes SIMPLEPLOT's facilities for drawing special charts:

4.1 Introduction to special charts

The subroutines to draw these special charts differ from those for x-y plots and polar plots given in Chapter 3 with respect to plotting scales, starting a new picture and the data structures that can be accommodated.

4.1.1 Data structures

Only one subroutine is available for drawing each special chart; therefore your data must be in a form suitable for the subroutine. This is unlike the wide range of subroutines available for plotting different data structures on x-y plots and polar plots described in Chapter 3. Data must be in the following form:

Bar charts are drawn by `BARCHT` which requires a 2-D array containing one data set after another; if only one data set is to be plotted, a 1-D array can be used.

Histograms are drawn by `HSTGRM` which requires a single 1-D array containing the raw data values (*eg.* exam scores) not pre-calculated frequency values: SIMPLEPLOT evaluates the frequencies internally.

Pie charts are drawn by `PIECHT` which requires a single 1-D array containing the single data set to be plotted. These data values need not be fractions of the whole as SIMPLEPLOT evaluates the proportions internally.

4.1.2 New pictures and framework

All special charts are plotted on the current picture; this means that a call to `NEWPIC` is normally needed before any call of `BARCHT`, `HSTGRM` or `PIECHT`. The axis framework (on histograms and bar charts) is drawn as part of the chart. An additional axis can be added using `AXIS7`.

4.1.3 Plotting scales

Plotting scales for special charts differ from those for general graph plotting both in the way they are specified and the default scales used. By default, scales for special charts are automatically set according to the limits of the data to be represented such that the scales run from the minimum data value to the maximum. The limits and direction of these linear scales for bar charts and histograms can be adjusted using the following subroutines:

`BARDIR` – specify the direction of bar chart scales

`BARRNG` – specify the range of bar chart numerical scale

`HSTDIR` – specify direction of scales for histograms drawn by `HSTGRM`

`HSTRNG` – specify ranges of scales for histograms drawn by `HSTGRM`

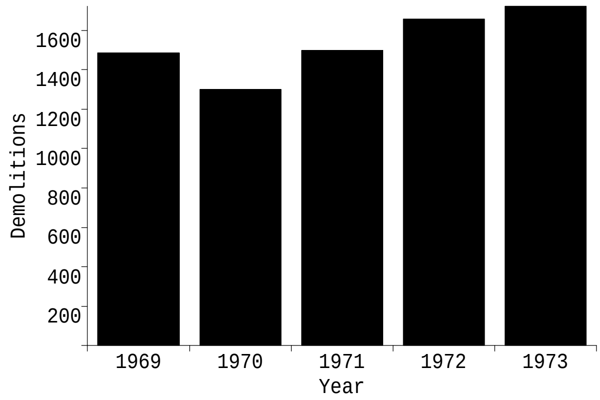
The plotting scales used for general graph plotting (as described in Chapter 3) are ignored by the subroutines described in this chapter but not reset. Similarly, `BARDIR`, `BARRNG`, `HSTDIR` and `HSTRNG` do not affect scales for general graph plotting. Please note that any subsequent plotting using general graph plotting subroutines, on a picture containing a special chart, should be preceded by the appropriate *specify scales* subroutine.

As with all subroutines which specify scales, the default ranges of scales for bar charts or histograms can be restored by calling `BARRNG` or `HSTRNG` with one scale limit equal to the other.

4.2 Bar charts

A SIMPLEPLOT bar chart has one quantitative scale and can be used to compare discontinuous data which can be grouped into one category (*eg.* date, place, type of flora/fauna, *etc.*). A SIMPLEPLOT bar chart draws bars which are proportional in height to the data values. The quantitative scale is known as the *numerical scale* and the other scale is known as the *label scale*.

4.2.1 A simple bar chart




```
      PROGRAM PRIM14
      PARAMETER(NARR=5,NSETS=1)
      REAL D2ARR(NARR,NSETS)
      CHARACTER LABARR(NARR)*4
      DATA LABARR/'1969','1970','1971','1972','1973'/
      DATA D2ARR /1486.0,1301.0,1499.0,1658.0,1723.0/
C start new picture
      CALL NEWPIC
C draw bar chart
      CALL BARCHT(D2ARR,LABARR,NARR,NSETS,'Demolitions','Year')
      CALL ENDPLT
      END
```

Example 14. A simple bar chart

Explanation of subroutines

`BARCHT(D2ARR,LABARR,NARR,NSETS,CAPN,CAPL)` draws a complete bar chart on the current picture representing the data values held in `D2ARR`.

`BARCHT` draws a pair of axes with the specified captions – `CAPN` for the numeric axis and `CAPL` for the label axis; bars are shaded with a different pattern for each data set. The label axis is annotated with the text labels given in `LABARR` which describe the corresponding values in each data set.

`D2ARR` is normally a two-dimensional array of size `NARR` × `NSETS` where `NSETS` is the number of data sets to be displayed and `NARR` is the number of values in each data set (*ie.* the number of bars to be drawn).

As `BARCHT` draws a bar chart on the current picture, it must normally be preceded by a call to `NEWPIC`.

The details of format, orientation, scales and type of bar chart *etc.* can be controlled by prior calls of the `BAR*` subroutines. In the absence of such calls, defaults are used:

- the range of the numerical scale is from 0.0 to the maximum value in the data array;
- the numerical axis is vertical and the label axis is horizontal;

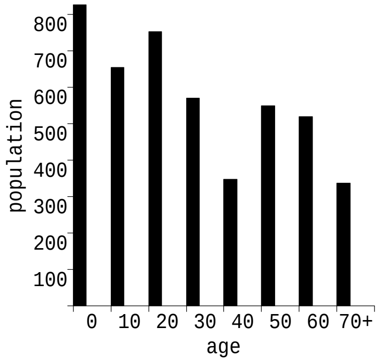
- multiple data sets are represented by bars drawn side-by-side (clustered);
- bars occupy 80% of the space available and are centred within this space.

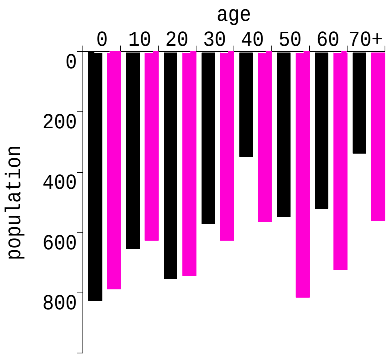
The bars representing the values of a single data set are identified by a shading pattern; the sequence of shading patterns can be altered by `SHPATT` or `SQSHAD`. A key to the current sequence of shading patterns can be drawn using `SHKEYS`; refer to Example 32, page 109, for further details.

The numerical and label scales are represented by axes referred to as the *numerical* axis and the *label* axis; the appearance of these axes can be controlled by the subroutines which specify axis characteristics; Refer to section 5.6 and Example 32 for further details.

4.2.2 Alternative bar chart formats

This example illustrates the use of BARFMT to produce a bar chart representing two data sets in alternative formats, BARRNG to extend the range of the numerical scale, and BARDIR to reverse the direction of a scale.






```
      PROGRAM PRIM15
      PARAMETER(NARR=8,NSETS=2)
      REAL POP(NARR,NSETS)
      CHARACTER AGE(NARR)*3
      DATA AGE/'0','10','20','30','40','50','60','70+'/
      DATA POP/826.1,654.3,752.3,570.9,348.0,548.4,519.5,337.2,
     +         787.2,625.6,742.3,625.2,564.0,815.5,723.9,559.6/
      CALL GROUP(2,1)
C start new picture, specify details and draw first bar chart
      CALL NEWPIC
      CALL BARFMT('P',35)
      CALL BARCHT(POP,AGE,NARR,1,'population','age')
C start new picture, specify details and draw second bar chart
      CALL NEWPIC
      CALL BARFMT('F',70)
      CALL BARRNG(0.0,1000.0)
      CALL BARCHT(POP,AGE,NARR,NSETS,'population','age')
      CALL BARDIR(.TRUE.,.FALSE.)
      CALL ENDPLT
      END
```

Example 15. Bar chart variations (1)

BARFMT(JCHAR,IVAL) specifies the justification of the bars within the space available:

JCHAR	*Justification of bars*
'P'	to the Preceding edge of the space – left for vertical bars, bottom for horizontal bars
'C'	Centred within the space (default)
'F'	to the Following edge of the space – right for vertical bars, top for horizontal bars

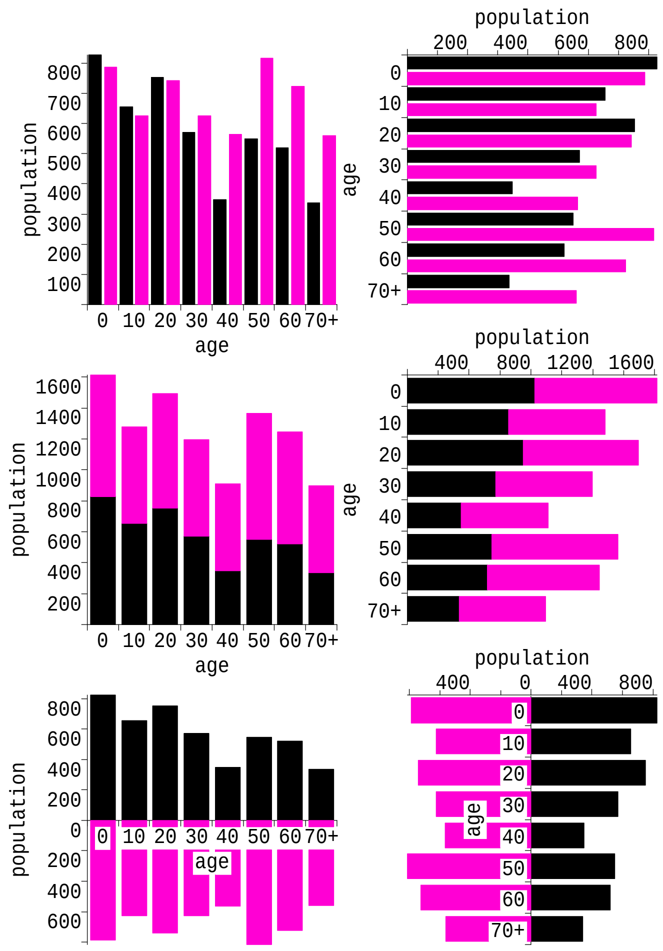
IVAL controls the size of the bars as a percentage of space available for each bar (the default is 80%).

`BARRNG(VMIN,VMAX)` specifies the range of scale for the numerical axis on subsequent bar charts. The default range of the numerical scale can be restored by calling `BARRNG` with one scale limit equal to the other.

`BARDIR(NBTORF,LBTORF)` specifies whether to reverse the direction of the numerical and label scales respectively (the default is `.FALSE.` for both `NBTORF` and `LBTORF`).

Examples 32 and 33 in Chapter 6 also illustrate the use of bar charts.

4.2.3 Alternative types of bar chart



BARTYP(OCHAR,ITYPE) specifies the orientation of the bars:

OCHAR	*Orientation of bars*
'V'	Vertical bars (default)
'H'	Horizontal bars

and the type of bar chart required:

ITYPE	*Type of bar chart*
1	Clustered bars – data sets side-by-side (default)
2	Stacked bars – data sets accumulated on same bars
3	Pyramid bars – data sets alternating

Horizontal bars are ordered from the top to the bottom of the picture.

```
      PROGRAM PRIM16
      PARAMETER(NARR=8,NSETS=2)
      REAL POP(NARR,NSETS)
      CHARACTER AGE(NARR)*3
      DATA AGE/'0','10','20','30','40','50','60','70+'/
      DATA POP/826.1,654.3,752.3,570.9,348.0,548.4,519.5,337.2,
     +         787.2,625.6,742.3,625.2,564.0,815.5,723.9,559.6/
      CALL GROUP(2,3)
C default - vertical, clustered
      CALL NEWPIC
      CALL BARCHT(POP,AGE,NARR,NSETS,'population','age')
C horizontal, clustered
      CALL NEWPIC
      CALL BARTYP('H',1)
      CALL BARCHT(POP,AGE,NARR,NSETS,'population','age')
C vertical, stacked
      CALL NEWPIC
      CALL BARTYP('V',2)
      CALL BARCHT(POP,AGE,NARR,NSETS,'population','age')
C horizontal, stacked
      CALL NEWPIC
      CALL BARTYP('H',2)
      CALL BARCHT(POP,AGE,NARR,NSETS,'population','age')
C vertical, pyramid
      CALL NEWPIC
      CALL BARTYP('V',3)
      CALL BARCHT(POP,AGE,NARR,NSETS,'population','age')
C horizontal, pyramid
      CALL NEWPIC
      CALL BARTYP('H',3)
      CALL BARCHT(POP,AGE,NARR,NSETS,'population','age')
      CALL ENDPLT
      END
```

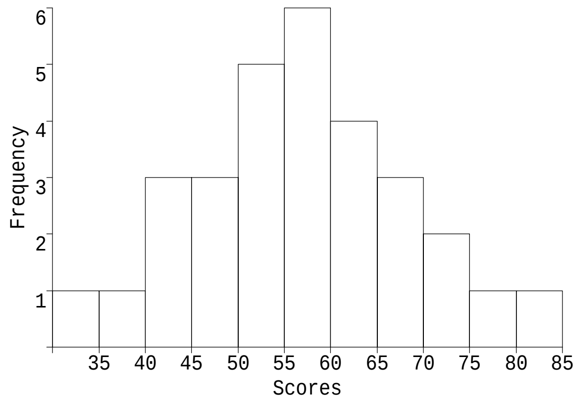
Example 16. Bar chart variations (2)

4.3 Histograms

A SIMPLEPLOT histogram has two quantitative scales and can be used to represent the frequencies of data occurring over a continuous range (*eg.* time, weight, size, height *etc.*).

4.3.1 A simple histogram

This example illustrates the simplest use of `HSTGRM` to produce a histogram (or frequency chart) from an array of raw data.




```
      PROGRAM PRIM17
      PARAMETER(NDARR=30,X1=0.0,DX=0.0)
      REAL DARR(NDARR)
      DATA DARR/31.0,56.0,63.0,45.0,56.0,73.0,80.0,54.0,48.0,51.0,
     +          63.0,67.0,52.0,58.0,75.0,47.0,42.0,60.0,50.0,55.0,
     +          65.0,39.0,42.0,55.0,57.0,66.0,64.0,71.0,44.0,54.0/
C start new picture
      CALL NEWPIC
C draw axis framework and histogram
      CALL HSTGRM(DARR,NDARR,X1,DX,'Scores','Frequency')
      CALL ENDPLT
      END
```

Example 17. A simple histogram

Explanation of subroutines

`HSTGRM(DARR,NDARR,X1,DX,CAPX,CAPY)` draws an axis framework and histogram on the current picture.

`HSTGRM` draws a pair of Cartesian axes labelled with the specified captions, `CAPX` and `CAPY`. Frequency values are calculated from the raw data in `DARR` which contains `NDARR` values. The third argument, `X1`, is an offset value for calculating subdivisions, x_1, such that interval subdivisions are placed at all values of the form $x_1 + n\,\Delta x$ (where n is any `int` and Δx is the frequency interval) within the range of values in the array `DARR`. The fourth argument, `DX`, is the frequency interval, Δx; if `DX`=0.0, SIMPLEPLOT calculates a suitable interval. Blocks are drawn to represent data falling in the ranges x_1 (inclusive) to $x_1 + \Delta x$ (exclusive), $x_1 + \Delta x$ (inclusive) to $x_1 + 2\,\Delta x$ (exclusive), *etc.*

As `HSTGRM` draws a histogram and axes on the current picture, it should normally be preceded by a call to `NEWPIC`.

4.3.2 Histogram variations

By default, a histogram displaying frequencies is drawn and the blocks are unshaded; the scales are calculated to fit the data exactly and both scales run in the natural direction of the corresponding axis. These default settings can be altered by the following subroutines:

`HSTDIR(XTORF,FTORF)` specifies whether to reverse the direction of the scales of the x and y-axes (the default is `.FALSE.` for both `XTORF` and `YTORF`). By default, Cartesian axes intersect at the origin or at the point nearest zero, therefore the use of `HSTDIR` may cause an axis to change location (see Example 19).

`HSTRNG(XMIN,XMAX,FMIN,FMAX)` specifies the range of scales used on subsequent x and y-axes when drawn on histograms. The default range of scales for histograms can be restored by calling `HSTRNG` with one scale limit equal to the other.

`HSTSHD(ISHADE)` specifies the shading pattern to be used for the blocks which make up a histogram (the default is `ISHADE`$=-1$). Refer to Figures G.2 and G.3, page 157, for suitable values of `ISHADE`.

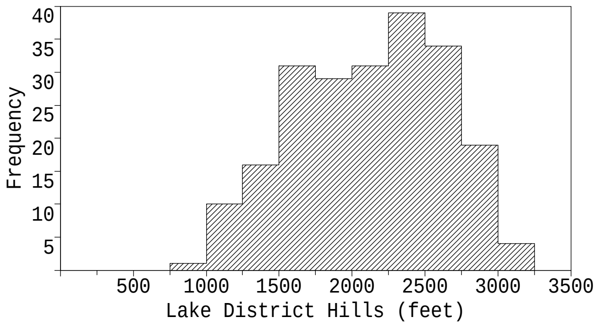
`HSTTYP(ITYPE)` specifies the type of histogram:

`ITYPE`	*Type of histogram*
1	Standard histogram (default)
2	Cumulative histogram
3	Complementary (or inverse) cumulative histogram

The next two examples illustrate different forms of histograms representing the same data set. The data file, `prim18.dat`, which is used in Examples 18 and 19 is included in the SIMPLEPLOT software distribution kit.

4.3.3 Shading and the range of histogram scales

This example illustrates the use of `HSTRNG` to control the x-scale and the frequency scale, and `HSTSHD` to select a shading pattern for the histogram.

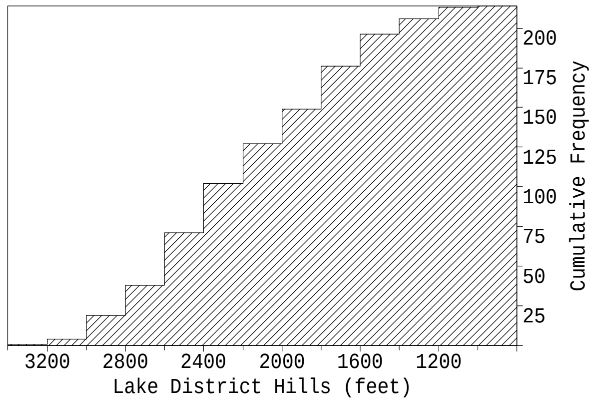



```
      PROGRAM PRIM18
      PARAMETER(NARR=214,X1=0.0,DX=250.0)
      REAL PEAKS(NARR)
C read frequency data from file
      OPEN(UNIT=10,FILE='prim1.dat',STATUS='OLD')
      READ(10,*)PEAKS
      CLOSE(10)
C start new picture and draw box around it
      CALL NEWPIC
      CALL PERIM
C specify range of histogram scales and shading pattern
      CALL HSTRNG(0.0,3500.0,0.0,40.0)
      CALL HSTSHD(4)
C draw histogram
      CALL HSTGRM(PEAKS,NARR,X1,DX,'Lake District Hills (feet)',
     +                             'Frequency')
      CALL ENDPLT
      END
```

Example 18. Histogram variations (1)

4.3.4 Alternative histograms and the direction of scales

This example illustrates the use of HSTDIR to control the direction of scales and HSTTYP to select an alternative type of histogram.




```
      PROGRAM PRIM19
      PARAMETER(NARR=214,X1=0.0,DX=200.0)
      REAL PEAKS(NARR)
C read frequency data from file
      OPEN(UNIT=10,FILE='prim18.dat',STATUS='OLD')
      READ(10,*)PEAKS
      CLOSE(10)
C start new picture and draw box around it
      CALL NEWPIC
      CALL PERIM
C specify type of histogram, direction of scales and shading pattern
      CALL HSTTYP(3)
      CALL HSTDIR(.TRUE.,.FALSE.)
      CALL HSTSHD(8)
C draw histogram
      CALL HSTGRM(PEAKS,NARR,X1,DX,'Lake District Hills (feet)',
     +                             'Cumulative Frequency')
      CALL ENDPLT
      END
```

Example 19. Histogram variations (2)

4.4 Pie charts

Pie charts differ from other graphs since they do not employ any framework (*eg.* axes). They are used to display proportional data, the whole quantity being divided into its component parts.

A circle which represents the whole quantity is divided into segments, each of which is proportional to the size of the components. Individual segments are drawn clockwise from an upward vertical radius; these segments are shaded and labelled.

The details of labelling, shading patterns, tilt of pie, positions of separate segments *etc.* can all be controlled by prior calls of `PI*` or `SG*` subroutines; in the absence of such calls, defaults are used:

- labels are boxed;
- the diameter of the pie is as large as can be accommodated with the required configuration of labels and exploded segments, within the picture size;
- labels contain only the segment name as supplied in `LABARR`;
- labels are positioned such that they are equally-spaced down the sides of the picture;
- pie charts are circular (no tilt);
- each data value is represented by a segment in the pie chart and every segment is displayed;
- the displacement of each segment is zero.

These default characteristics of a pie chart can be changed by the specification subroutines described in section 4.4.2.

The sequence of shading patterns can be altered by the following subroutines:

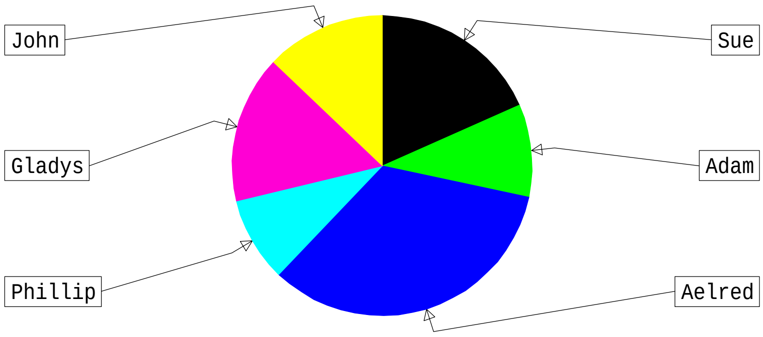
`SHPATT` – specify one of a sequence of shading patterns

`SQSHAD` – specify sequence of shading patterns

A key to the current sequence of shading patterns can be drawn using `SHKEYS`; refer to Example 32, page 109, for further details.

4.4.1 A simple pie chart

To draw a pie chart, you need one numeric data set for the segment values and one text data set for the segment labels.




```
      PROGRAM PRIM20
      PARAMETER(NSEGS=6)
      CHARACTER*7 LABARR(NSEGS)
      REAL DARR(NSEGS)
      DATA LABARR/'Sue','Adam','Aelred','Phillip','Gladys','John'/
      DATA DARR  / 83.0, 45.0,  152.0,   41.0,     72.0,    58.0 /
C start new picture
      CALL NEWPIC
C draw pie chart
      CALL PIECHT(DARR,LABARR,NSEGS)
      CALL ENDPLT
      END
```

Example 20. A simple pie chart

Explanation of subroutines

`PIECHT(DARR,LABARR,NARR)` draws a pie chart on the current picture showing the distribution of data among the `narr` values in array `DARR`.

As `PIECHT` draws a pie chart on the current picture, it must normally be preceded by a call to `NEWPIC`.

4.4.2 Pie chart variations

The default characteristics of a pie chart can be controlled by the following subroutines:

`PIBOXL(TORF)` specifies whether pie chart labels are drawn with a box around them (the default is `TORF=.TRUE.`).

`PIDIAM(CMS)` specifies the diameter of the pie, in centimetres.

`PIINCL(LTORF,VTORF,PTORF)` specifies which components are to be included in the labels:

- `LTORF` – segment label (the default is `.TRUE.`)
- `VTORF` – value (the default is `.FALSE.`)
- `PTORF` – percentage (the default is `.FALSE.`)

Percentages are specified to two decimal places and all lines contained in the label are left-justified. These two characteristics can be modified by `FIGFMT` and `ADDJST` respectively.

`PIPOSL(IPOS)` specifies the position of pie chart labels relative to the pie itself:

`IPOS`	*Label position*
1	Labels equally-spaced down sides of picture, with arrows if there is room (default)
2	Labels alongside segments, down sides of picture, with arrows if there is room
3	Labels next to segments (no arrows)
4	Labels inside segments, or next to segment when there is not enough space inside (no arrows)

`PITILT(IWDTH,IHGHT)` specifies the tilt of pie charts. Tilted pie charts are drawn by adjusting the *width* : *height* ratio to produce elliptical pies. `IWDTH` and `IHGHT` must be positive integers, and only values with `IWDTH` > `IHGHT` are accepted; any other values restore the default.

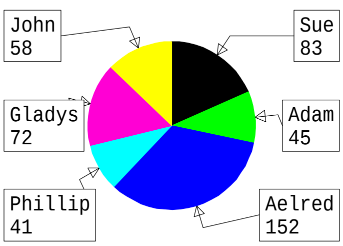
`SGEXPL(IVAL,ISEG)` specifies the amount (as a percentage of the radius) by which segment number `ISEG` is to be displaced from the centre of the pie, along the radius bisecting its angle.

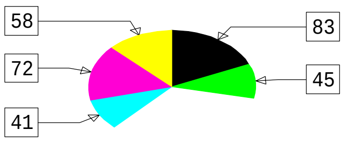
`SGOMIT(TORF,ISEG)` specifies whether or not the `ISEG`th segment of the pie is to be omitted from the picture; all segments are always used in the calculation of pie chart segments.

A key to the sequence of shading patterns can be included if required – see `SHKEYS`.

4.4.3 Pie chart label contents and omitting segments

This example illustrates the use of `PIPOSL`, `PIINCL`, `PITILT` and `SGOMIT`.



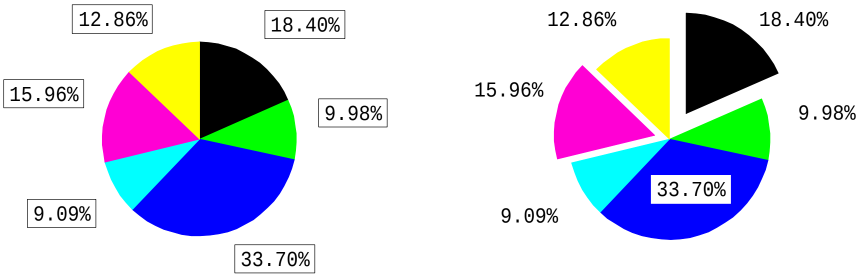


```
      PROGRAM PRIM21
      PARAMETER(NSEGS=6)
      CHARACTER*7 NAMES(NSEGS)
      REAL FREQ(NSEGS)
      DATA NAMES/'Sue','Adam','Aelred','Phillip','Gladys','John'/
      DATA FREQ / 83.0, 45.0,  152.0,   41.0,     72.0,    58.0 /
      CALL GROUP(2,1)
C start new picture, specify details and draw first pie chart
      CALL NEWPIC
      CALL PIINCL(.TRUE.,.TRUE.,.FALSE.)
      CALL PIECHT(FREQ,NAMES,NSEGS)
C start new picture, specify details and draw second pie chart
      CALL NEWPIC
      CALL PIPOSL(2)
      CALL PITILT(3,2)
      CALL SGOMIT(.TRUE.,3)
      CALL PIINCL(.FALSE.,.TRUE.,.FALSE.)
      CALL PIECHT(FREQ,NAMES,NSEGS)
      CALL ENDPLT
      END
```

Example 21. Pie chart variations (1)

4.4.4 Pie chart labels and exploding segments

This example illustrates the use of `PIBOXL`, `PIDIAM`, `PIINCL` and `SGEXPL`.




```
      PROGRAM PRIM22
      PARAMETER(NSEGS=6)
      CHARACTER*7 NAMES(NSEGS)
      REAL FREQ(NSEGS)
      DATA NAMES/'Sue','Adam','Aelred','Phillip','Gladys','John'/
      DATA FREQ / 83.0, 45.0,  152.0,   41.0,     72.0,    58.0 /
      CALL GROUP(2,1)
C start new picture, specify details and draw first pie chart
      CALL NEWPIC
      CALL PIPOSL(3)
      CALL PIINCL(.FALSE.,.FALSE.,.TRUE.)
      CALL PIECHT(FREQ,NAMES,NSEGS)
C start new picture, specify details and draw second pie chart
      CALL NEWPIC
      CALL PIPOSL(4)
      CALL PIDIAM(3.0)
      CALL PIBOXL(.FALSE.)
      CALL SGEXPL(30,1)
      CALL SGEXPL(15,5)
      CALL PIECHT(FREQ,NAMES,NSEGS)
      CALL ENDPLT
      END
```

Example 22. Pie chart variations (2)

5. Additional Facilities

This chapter describes SIMPLEPLOT's facilities for adding extra information to your pictures:

5.1 How to change page layout – groups of pictures – page and picture size – margins – surrounding boxes

5.2 How to use descriptive positions

5.3 How to add titles – more than one line of title

5.4 How to add keys and captions – defining areas – adding key and caption entries – surrounding boxes – composite keys

5.5 How to modify text – alternative character sets – size of text and marker symbols – masking text and marker symbols – long titles – vertical line spacing in titles, keys and captions

5.6 How to draw axes – axis direction – axis range, position and intersection – axis subdivision and grid lines – axis annotation

5.7 How to manipulate data – missing data – data limits – polynomial functions– number formatting

5.8 How to use shading – hardware and software shading patterns – shading sequences

5.9 How to use colour/pens – pen selection – shading in colour

5.10 How to omit drawing and shading

5.11 How to thicken lines

There are no example programs in this chapter, but Chapter 6 uses most of the subroutines described here in combination with the facilities described in Chapters 3 and 4.

5.1 How to change page layout

SIMPLEPLOT draws pictures which fit within the device plotting space and makes 'sensible' decisions about layout within that space. This section describes the default layout decisions and how they can be controlled or overridden.

5.1.1 Changing layout

The various components of SIMPLEPLOT page layout can be changed by the following subroutines:

- Page orientation and size – `PAGVW`, `PAGE`.
- Picture size – `PICSIZ`.
- Picture margins – `MARGIN`, `MARGDV`.
- Page peripheries (or page margins) – `PERIPH`.
- Groups of pictures – `GROUP`.

These different components are illustrated in Figure 5.1 (for a 2×2 grouping) and in Figure 2.1 (for a single picture). The *specify layout* subroutines are described below:

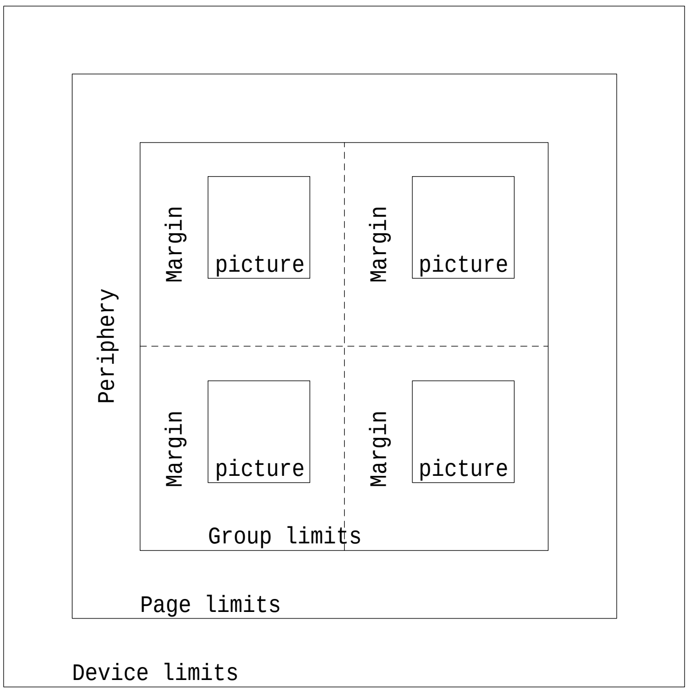


Figure 5.1 SIMPLEPLOT page layout – a 2×2 group of pictures

`PAGVW(ITYPE)` specifies the orientation of the default SIMPLEPLOT page on a *fixed page* device:

`ITYPE`	*Page orientation*
0	Default mode, determined by device
1	Portrait (if page can be turned)
2	Landscape (if page can be turned)
3	Portrait
4	Landscape
5	Upside-down portrait
6	Upside-down landscape

ITYPE=1 and 3, or ITYPE=2 and 4, are equivalent on devices which can be used in both portrait and landscape modes (*eg.* some laser printers). ITYPE=3 and 4 results in a rotated page even on devices which would not normally turn the SIMPLEPLOT page (*eg.* a screen).

PAGE(XCMS,YCMS) specifies the dimensions (in centimetres) of the SIMPLEPLOT page. If necessary, SIMPLEPLOT shrinks the page to fit device plotting area. On fixed page devices, the arguments of PAGE are not used unless the device is capable of turning the plotting area through 90°; in this case, the orientation is chosen to be similar to that indicated by the arguments of PAGE, but the device plotting area is used as page size. CALL PAGE(0.0,0.0) cancels the effect of PAGE.

PAGVW is overridden by calling PAGE to define page dimensions; if the default page is subsequently restored, any previous setting of PAGVW becomes active again.

PICSIZ(XCMS,YCMS) specifies the width and height (in centimetres) of subsequent pictures. Constraints caused by device limits, page sizes and picture groupings can cause the picture to be shrunk to fit in, but the proportions specified by PICSIZ are retained. CALL PICSIZ(0.0,0.0) cancels the effect of PICSIZ.

MARGIN(CMS) specifies the size (in centimetres) of the margin around each picture (see Figure 5.1). This margin is typically used for axis annotation. CALL MARGIN(-1.0) cancels the effect of MARGIN.

MARGDV(PICTYP) specifies the distribution of picture margins.

PICTYP	*Picture type*	*Margin distribution*
'D'	Default	Large left and bottom, small right and top
'C'	Cartesian	Large on 'XC' and 'YC' axis sides, small on other
'B'	Bar chart	Large on 'LB' and 'NB' axis sides, small on other
'O'	Other (polar or pie)	Equal distribution on all sides.

Refer to Example 33 for an illustration of MARGDV.

PERIPH(CMS) specifies the size (in centimetres) of the gap between a group of pictures and the edge of the SIMPLEPLOT page (see Figure 5.1). CALL PERIPH(-1.0) cancels the effect of PERIPH.

GROUP(NHORIZ,NVERT) specifies that subsequent pictures are to be grouped 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. There is no explicit limit to the number of pictures in a group but there may be practical constraints due to other elements of the layout. CALL GROUP(1,1) restores the default grouping of one picture per page.

The order of calling the layout subroutines is not significant; the arguments of each call are stored and then come into use when a new picture (or group) is started. However, they should be called *before* the new picture, group or page which they affect is started, otherwise unexpected results may be produced.

5.1.2 Over-specification

If all layout options are specified, contradictory requests can easily occur and SIMPLEPLOT has to disregard something. The order of priority is as follows:

1. Page size,
2. Picture size,
3. Picture margins and
4. Peripheral margins.

First, the page is shrunk to fit the device space; the shrinkage factor required to achieve this is then applied to all layout requests such that pictures and picture margins are shrunk to fit the page; finally, the periphery is also shrunk.

5.1.3 The SIMPLEPLOT page

The SIMPLEPLOT page is the area on the device within which a group of one or more pictures is positioned; all plotting is done within the SIMPLEPLOT page. The default SIMPLEPLOT page depends on the graphics device:

- On packed page devices which have no inherent page size (*eg.* large drum plotters), each picture/group is positioned within a page whose size relates to the total available on the device; square dimensions near 20cm $\times$ 20cm are chosen such that the total area is divided into a whole number of pages.
- On fixed page devices whose display cannot be turned (*eg.* graphics terminals), the device plotting area is treated as a single SIMPLEPLOT page with its natural orientation.
- On a fixed page device which can be used turned through 90° (*eg.* A4 printer or plotter), the default orientation depends on whether groups and picture size have been defined:
  - If neither picture size nor grouping has been specified, default orientation is portrait (width<height).
  - If a group of pictures has been specified with default picture sizing, orientation is chosen to produce the squarest picture possible. So a group of two pictures across and one down results in a landscape page, but a group of two pictures across and three down results in a portrait page.
  - If the picture size has been specified, the group dimension is calculated and the most suitable orientation is used.

5.1.4 Surrounding boxes

The following subroutines can be called just once each to specify that subsequent pages, groups and/or pictures are to have a line drawn around the boundary:

`BOXGRP(TORF)` specifies whether subsequent pages are drawn with a box around the group of pictures; by default none is drawn.

`BOXPAG(TORF)` specifies whether subsequent pages are drawn with a box around the limits of the SIMPLEPLOT page; by default none is drawn.

`BOXPIC(TORF)` specifies whether subsequent new pictures are to be drawn with a box around; by default none is drawn.

5.2 How to use descriptive positions

Descriptive positions enable you to describe the general location of titles, keys or caption areas without being concerned about their exact position. A descriptive position is specified by two arguments, `VCHAR` and `HCHAR`, which describe the vertical and horizontal positions respectively. These arguments are single character (`char *`) mnemonics which relate to the SIMPLEPLOT page, a group of pictures or a picture (see Table 5.1).

Page positions – `'N'`orth, `'E'`ast, `'S'`outh, `'W'`est.

Group positions – `'H'`igher than group and `'L'`ower than group.

Picture positions – `'T'`op, `'O'`ver, `'U'`nder, `'B'`ottom, `'P'`receding, `'L'`eft, `'R'`ight, `'F'`ollowing and `'C'`entre.

`'L'`eft, `'R'`ight and `'C'`entre refer to locations relative to the picture, group or page depending on the value of the other argument. For example, `VCHAR='C'` and `HCHAR='C'` positions the first line of title in the centre of the current picture; `'C'`, `'W'` places it in the centre of the page, on the far left.

Similarly, `'P'`receding and `'F'`ollowing refer to positions relative to the picture or group. This interrelationship is summarized in Table 5.2 and illustrated for titles in Figures 5.2–5.4, and for keys and caption areas in Figures 5.5 and 5.6.

5.2.1 Size of text in titles, keys and captions

The descriptive position determines both the size of text used and the horizontal limit of a title, key or caption area.

Page: The text size relates to the page size; the maximum width of a title, key or caption area is equal to the width of the SIMPLEPLOT page.

Group: The text size relates to page size; the maximum width of a title, key or caption area is equal to the width of the group.

Picture: The text size relates to picture size; the maximum width of a title, key or caption area is equal to the width of the picture (excluding margins).

Table 5.1 Descriptive positions of titles, keys and captions

`VCHAR`	*Vertical position*
`'N'`	North (highest)
`'H'`	Higher (above group)
`'O'`	Over picture/group
`'T'`	Top of picture/group
`'C'`	Centre of picture/page
`'B'`	Bottom of picture/group
`'U'`	Under picture/group
`'L'`	Lower than group
`'S'`	South (lowest)

`HCHAR`	*Horizontal position*
`'W'`	West (far left of page)
`'P'`	Preceding picture/group
`'L'`	to the Left of picture/group
`'C'`	Centre of picture/group/page
`'R'`	to the Right of picture/group
`'F'`	Following picture/group
`'E'`	East (far right of page)

Table 5.2 Relative sizes of titles, keys and captions

	`'W'`est	`'P'`receding `'L'`eft	`'C'`entre	`'F'`ollowing `'R'`ight	`'E'`ast
`'N'`orth	*page*	*group*	*page*	*group*	*page*
`'H'`igher	*page*	*group*	*group*	*group*	*page*
`'O'`ver	*page*	*picture*	*picture*	*picture*	*page*
`'T'`op	*page*	*picture*	*picture*	*picture*	*page*
`'C'`entre	*page*	*picture*	*picture*	*picture*	*page*
`'B'`ottom	*page*	*picture*	*picture*	*picture*	*page*
`'U'`nder	*page*	*picture*	*picture*	*picture*	*page*
`'L'`ower	*page*	*group*	*group*	*group*	*page*
`'S'`outh	*page*	*group*	*page*	*group*	*page*

The relationship between size and position is summarized in Table 5.2; see also Figure 5.1.

If variable-sized text is used, the size of text in titles is larger than that used for keys and captions which itself is larger than that used for general annotation and labelling.

5.3 How to add titles

Titles can be added in positions relative to the picture, group of pictures or the SIMPLEPLOT page.

`TITLE7(VCHAR,HCHAR,CAP)` draws the text string, `CAP`, as a title in the position described by the first two arguments, `VCHAR` and `HCHAR`.

The positional descriptors, `VCHAR` and `HCHAR`, can take any of the standard values. For titles, the interpretation of these positions is illustrated in Figures 5.2–5.4. `'N'`orth and `'H'`igher both position titles as high as possible on a page but differ in their horizontal positioning when centred: `'N'`orth titles use the full width of the page and are centred within the limits of the page whereas `'H'`igher titles use the limits of the group.

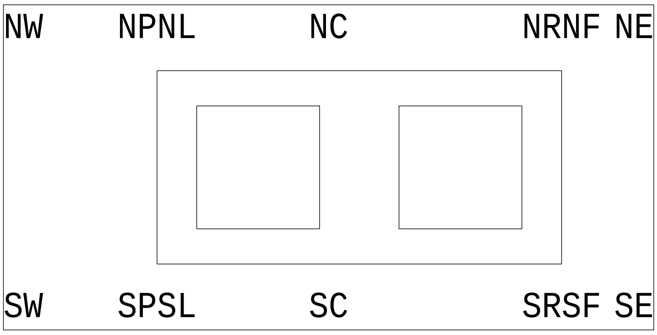


Figure 5.2 Titles – page positions

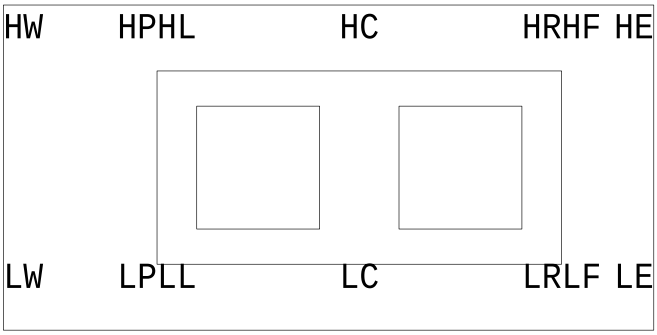


Figure 5.3 Titles – group positions

5.3.1 More than one line of title

Several lines of text can be drawn by successive calls of `TITLE7` in the same position. Such sequences of titles may be aimed anywhere and other graphics operations can be performed between calls without breaking the sequence but not other titles. This is because SIMPLEPLOT only remembers the position of the last title written; therefore, if a sequence of titles is interrupted by another title in a different position, the original sequence is automatically terminated as soon as the interrupting title is written and any attempt to continue it will start at the top of the sequence again.

Titles are only omitted when a sequence reaches the bottom of the page; every available title position can accommodate at least one title.

Refer to section 6.3 and 5.5.5 for details of how SIMPLEPLOT deals with titles which are too long and how to control the vertical spacing between multi-line titles.

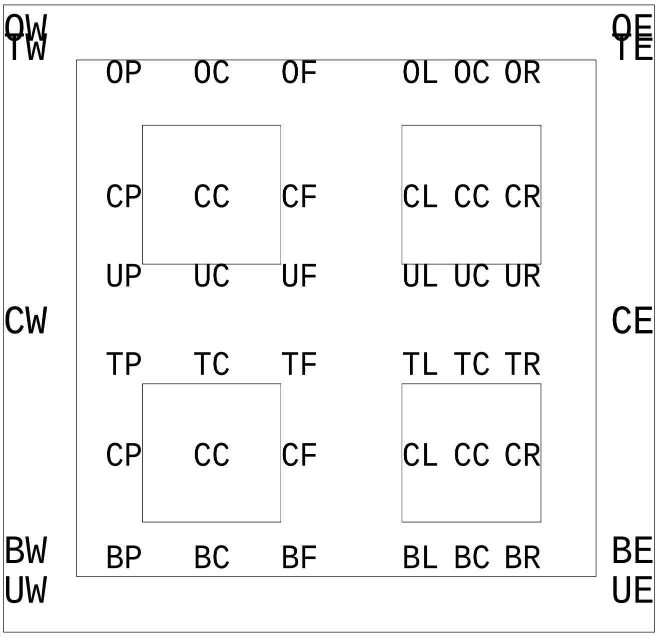


Figure 5.4 Titles – picture positions

5.4 How to add keys and caption areas

In order to add a key to your picture you must define an area to be used for keys and then add the entries line-by-line. Caption areas can be defined in a similar way and used interchangeably with keys.

5.4.1 Defining an area for keys or captions

Key and caption areas can be defined on the current picture as follows:

`DEFKEY(ITYPE,VCHAR,HCHAR,NROWS,NCOLMS)` defines an area for a key according to the value of the first argument, `ITYPE`:

`ITYPE`	*Key or caption area*
1	Area can be over-drawn
2	Area is masked

The size of the key area is high enough for the specified number of lines (`NROWS`) and wide enough to accommodate the specified number of characters per entry (`NCOLMS`) when written alongside a 12 character key sample.

`DEFCAP(ITYPE,VCHAR,HCHAR,NROWS,NCOLMS)` defines an area for captions in a similar manner to `DEFKEY`. The only difference is that the specified width, `NCOLMS`, is the full width of the caption area.

`VCHAR` and `HCHAR` can take any of the standard values. For key and caption areas, the interpretation of these positions is illustrated in Figures 5.5 and 5.6. The position of a key/caption area also determines its horizontal limit – the width of the page, the width of the group or the width of the picture.

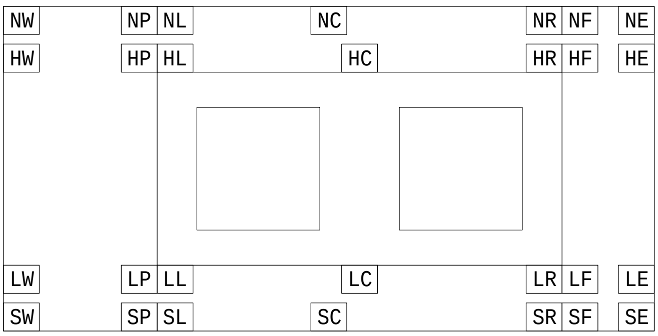


Figure 5.5 Key and caption areas – page and group positions

5.4.2 Adding key and caption entries

Captions or key entries are added to the next line of the current key or caption area; both key and caption areas are filled from the top downwards.

`ADDCP7(CAP)` enters the caption, `CAP`, in a pre-defined key or caption area. Refer to Examples 24 and 30 for illustrations of `ADDCP7`.

`BLNKKY` leaves the next entry in a key or caption area blank. Refer to Example 30 for an illustration of `BLNKKY`.

`BOTHK7(LTYPE,MKTYPE,CAP)` enters the key caption, `CAP`, alongside a sample of the broken line pattern, `LTYPE`, superimposed with the marker symbol, `MKTYPE`. Refer to Example 27 for an illustration of `BOTHK7`.

`LINEK7(LTYPE,CAP)` enters the key caption, `CAP`, alongside a sample of the broken line pattern, `LTYPE`.

`MARKK7(MKTYPE,CAP)` enters the key caption, `CAP`, alongside a sample of the marker symbol, `MKTYPE`.

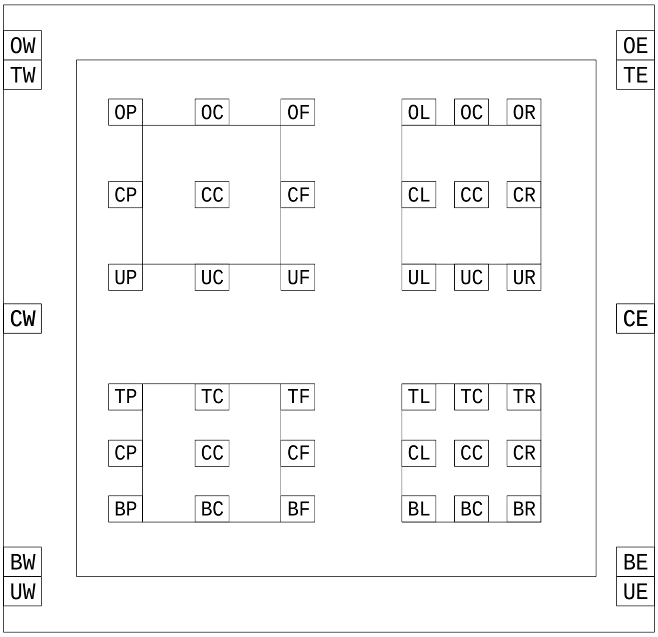


Figure 5.6 Key and caption areas – picture positions

`PUTCP7(IROW,ICOLM,CAP)` writes the caption, `CAP`, in the specified position within a caption area defined by `DEFCAP`. The position is determined as line `IROW` character position `ICOLM` (1,1 is top left). Refer to Example 31 for an illustration of `PUTCP7`.

`SHDEK7(ISHADE,CAP)` enters the key caption, `CAP`, and a sample of the shading pattern, `ISHADE`. Refer to Example 24 for an illustration of `SHDEK7`.

5.4.3 Modifying keys and captions

The following subroutines can be called to modify the behaviour of the subroutines described above:

`ADDJST(JCHAR)` specifies the justification of captions and key entries within the space available:

`JCHAR`	*Horizontal justification*
`'L'`	Left justified – default
`'C'`	Centre justified (centred)
`'R'`	Right justified

`ADDJST` also affects the justification of pie chart labels. Refer to Examples 24 and 30 for illustrations of `ADDJST`.

`BOXCAP(TORF)` specifies whether subsequent caption areas are to be drawn with a box around; by default, no box is drawn. Refer to Examples 30 and 31 for illustrations of `BOXCAP`.

`BOXKY(TORF)` specifies whether subsequent key areas are to be boxed; by default, a box is drawn. Refer to Example 35 for an illustration of `BOXKY`.

`DEFKYW(NCHARS)` specifies that samples in a key box are to be `NCHARS` characters wide. Refer to Example 35 for an illustration of `DEFKYW`.

5.4.4 Composite keys

A complete key to a sequence of shading patterns can be drawn using a call to a single subroutine, `SHKEYS`.

`SHKEYS(VCHAR,HCHAR,LABARR,NARR,CAP)` draws a complete key to shading patterns. A masked key box is defined, the caption, `CAP`, is entered at the top and the set of labels from the array, `LABARR`, is added alongside samples of each shading pattern in the current sequence. `VCHAR` and `HCHAR`, can take any of the values described in Table 5.1 (see also Figures 5.4 and 5.5). Refer to Example 32 for an illustration of `SHKEYS`.

5.5 How to modify text

By default, SIMPLEPLOT aims to produce text which is both clear and well proportioned. If the characteristics of text are to be modified, specifications affecting the size and position of a new picture should be fixed before the picture is started. Specifications which change the size of text can indirectly affect the size of margins around the picture and the periphery around the group.

5.5.1 The style of text

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, various proportionally spaced fonts (Hershey characters) and an adjustable fixed-width font are available.

`CHSET(ISET)` specifies an alternative source of characters:

`ISET`	*Font selected*
≤ -1	Alternative hardware/firmware sets (if available)
0	The default hardware font for the device (if available, otherwise as `ISET`=1)
1	A simple, fixed-width software font
2–19	Hershey fonts (see Figure G.4)
20	A solid software font
21	An outline font
22	A set of Hershey mathematical symbols
23	Large Hershey mathematical symbols
51	An adjustable software font
52	An adjustable software font including “£”

Refer to Figure G.4 (page 158) for examples of the different character sets and to Example 30 for an illustration of `CHSET`.

5.5.2 The size of text

The size of text used is chosen to relate to the physical size of the picture, to be appropriate to its purpose (*eg.* title, key entry, axis annotation) and suitable for the device. This choice can be affected by the following subroutines:

`TEXTMG(FACTMG)` specifies a magnification factor for all text (`FACTMG`=0.0 restores default).

`TEXTMN(CMSMIN)` specifies a minimum nominal character width (in centimetres) for all text (`CMSMIN`=0.0 restores default).

`TEXTSZ(CMSSIZ)` specifies a fixed nominal character width (in centimetres) for all text (`CMSSIZ`=0.0 restores default).

SIMPLEPLOT evaluates the actual size of text using the following procedure:

1. Calculate a *target* nominal text width based on the current page size, the current picture size and the text usage:
   - Normal usage – labels, axis annotation *etc.*
   - Keys/captions relating to pictures
   - Keys/captions relating to pages
   - Titles relating to pictures
   - Titles relating to pages
2. The target width is adjusted according to user calls:

- After `CALL TEXTMG(FACTMG)`, the target width is multiplied by `FACTMG`.
- After `CALL TEXTSZ(CMSSIZ)`, the value of `CMSSIZ` is used in place of all target text widths.
- The target width is always forced to be no less than some minimum value. After `CALL TEXTMN(CMSMIN)`, `CMSMIN` is used in place of all smaller targets. When `TEXTMN` has not been called, the minimum set in the device driver is used; if no minimum is set in the device driver, 0.14 centimetres is used.

`TEXTSZ` overrides any call to `TEXTMG` until cancelled.

3. For *hardware* text, this adjusted target width is passed to the device driver which returns the size of text nearest to the target. Hardware fonts have different capabilities – some devices can reproduce text at any chosen size, some have a few discrete sizes, and some can only draw one size of text.
4. For the simple *software* font, a similar process is performed within SIMPLEPLOT in order to produce readable text on low-resolution devices. The font is encoded in steps on a small grid (*eg.* 7×11 pixels); the target step size is calculated by dividing the target text width by the number of horizontal steps in the character grid; on pixel devices, the step size is adjusted to the size of a whole number of pixels.

For the *proportional spaced* fonts (`ISET`=2–23) the nominal text width is set to the width of digits which is the same for all digits within a font.

5.5.3 The size of marker symbols

By default, the size of marker symbols is controlled by the same mechanism that controls the size of text – `TEXTSZ`, `TEXTMN` and `TEXTMG` normally affect the size of both characters and symbols. The size of marker symbols can also be controlled independently:

`MKSIZE(SWIDTH)` specifies the size (in cm) of marker symbols independent of text size. Refer to Example 29 for an illustration of `MKSIZE`.

5.5.4 Masking text and symbols

By default, SIMPLEPLOT plots anywhere within the picture limits unless an area has been designated as a *blanked* or *masked area*.

After `CALL CHMASK(.TRUE.)`, text is masked. This means that subsequent drawing over the text is blanked as it crosses an area surrounding the text. Similarly, after a call of `MKMASK`, marker symbols are masked and subsequent drawing over the symbols is omitted as it crosses the area around each symbol.

`CHMASK(TORF)` specifies whether or not text is to be masked. Refer to Example 25 for an illustration of `CHMASK`.

`MKMASK(TORF)` specifies whether or not marker symbols are to be masked. Refer to Example 29 for an illustration of `MKMASK`.

There is a limit to the number of masked areas (symbols and text). When this limit is reached the following diagnostic is issued:

`(Maximum no. of masked areas reached)`[1]

Subsequent masked areas take the place of the oldest areas and overdrawing of these may occur.

5.5.5 Line spacing

Successive lines of a multi-line title and entries in keys or caption areas are spaced vertically using an adjustable line spacing.

`TEXTLF(FACTOR)` specifies the space between successive lines of text. `FACTOR` can be any `float` value greater than or equal to 1.0; by default, `FACTOR`=1.2 is used. Refer to Example 25 for an illustration of `TEXTLF`.

5.6 How to draw axes

SIMPLEPLOT can draw axes and their associated labelling in different ways:

- By using a few composite tasks, *eg.* `AXES7`, `POLAR7`, `BARCHT` or `HSTGRM`.
- By using many individual tasks *eg.* `AXIS7` for the axis and its caption, `AXTXT7` for text labels, *etc.*

The subroutines which actually do the drawing, composite or individual, can be affected by changes made to the settings of the various axis characteristics. The axis specification subroutines affect *all* axis drawing including axes drawn by `AXES7`, `BARCHT`, `HSTGRM` and `POLAR7`.

5.6.1 Type of axis

The default characteristics of an axis and the orientation of axis are determined by the *axis type*, *eg.* Cartesian x-axis, polar angular axis, polar radial axis. For each of the `AX*` subroutines (except `AXES7`) the first argument, `CHAXIS`, is used to identify the type of axis; `CHAXIS` is a two-character (`char *`) mnemonic:

- `'XC'` and `'YC'` for *Cartesian* plots, including histograms;
- `'RP'` and `'AP'` for *radial* and *angular* axes on polar plots;
- `'NB'` and `'LB'` for *numerical* and *label* axes on bar charts.

Wildcards can be used in the specification of axis type in order to specify the characteristics of more than one type of axis. For example, `'*C'` affects both Cartesian axes, `'XC'` and `'YC'`; `'**'` affects *all* axes.

5.6.2 Axis direction

The direction of an axis can be defined for all axis types, independently of the direction of the scale which the axis displays, as listed in Table 5.3. `'P'`receding and `'F'`ollowing are positions relative to the direction of an axis and describe positions unambiguously on *all* types of axis. For example, on Cartesian axes, `'P'` refers to the left of the x-axis and the bottom of the y-axis; `'F'` refers to the right of the x-axis and the top of the y-axis.

5.6.3 Drawing an axis

An individual axis can be drawn on the current picture using `AXIS7`:

`AXIS7(CHAXIS,CAP)` draws the type of axis specified by `CHAXIS` (Cartesian axes on x-y plots or histograms, polar axes or bar chart axes) with the caption, `CAP`; the type of axis must be suitable for the type of picture.

Table 5.3 Axis direction

`CHAXIS`	*Direction of axis*
`'XC'`	Left to right of picture
`'YC'`	Bottom to top of picture
`'RP'`	Centre to outside of polar chart
`'AP'`	Anti-clockwise from 3 o'clock (see `POLZER`)
`'NB'/'LB'`	As x-axis or y-axis depending on orientation of bars (see `BARTYP`)

5.6.4 Axis range, position and intersection

By default, an axis is drawn over the full range of the plotting scale. The intersection of one axis with another depends on the type of axis: a Cartesian or radial axis is positioned such that it intersects the other axis at (0,0) or at the scale value nearest the origin; an angular axis intersects the radial axis at the outer extreme of the radial scale.

`AXCRSS(CHAXIS,CROSS)` describes the point of intersection of the specified axis with another axis where `CROSS` is a `float` value specified in terms of the plotting scale of the *other* axis. `AXCRSS` does not apply to bar chart axes. `CALL AXCRSS(CHAXIS,0.0)` restores the default except for an angular axis for which you must specify the outer radial value or `CALL AXLOCN('AP','D')`. Refer to Example 28 for an illustration of `AXCRSS`.

`AXLOCN(CHAXIS,LCHAR)` specifies the location of the specified axis relative to the picture:

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

In order to specify `'P'`receding or `'F'`ollowing, you must know the *direction* of the axis (see Table 5.3). Refer to Examples 27 and 33 for illustrations of `AXLOCN`.

`AXRNGE(CHAXIS,START,STOP)` specifies a sub-range of the plotting scale over which to draw the axis. The default can be restored by calling `AXRNGE` with a zero range (*ie.* `START`=`STOP`). Refer to Example 27 for an illustration of `AXRNGE`.

5.6.5 Axis annotation labels

Axis annotation is drawn at all axis subdivisions or, if there is not enough room, alternate labels are omitted; by default, annotation is omitted at the axis intersection to avoid ambiguity. The labels represent the scale value at the point to which they refer; this is usually at the nearby tick mark. Axis annotation is written on whichever side of the axis is nearer to the outside of the picture.

Extra annotation labels can be drawn using the following subroutines:

`AXLAB7(CHAXIS,W,CAP)` draws the individual annotation label, `CAP`, at position `W` on the specified axis. `W` is interpreted in terms of the current plotting scale for the axis specified by `CHAXIS`.

`AXTXT7(CHAXIS,W1,STEP,LABARR,NARR)` draws the set of axis annotation labels in `LABARR` at regular intervals on the specified axis. Axis annotation labels are drawn such that `LABARR[0]` is positioned at `W1`, `LABARR[1]` is positioned at `W1+STEP` *etc.*

To draw an axis with non-numeric labels:

1. Suppress the normal numerical annotations by calling `AXCLR` before the axis is drawn (either as part of composite task or separately by `AXIS7`).
2. Call `AXTXT7` to draw a set of alternative annotations.

Refer to Examples 29 and 34 for illustrations of `AXTXT7`.

The position and appearance of axis annotation labels can be controlled by the following specification subroutines:

`AXCLR(CHAXIS,ILEVEL)` specifies the level of annotation to be added to an axis when it is drawn:

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

Refer to Examples 29 and 34 for illustrations of `AXCLR`.

`AXLBGP(CHAXIS,ILEVEL)` specifies the level of annotation to be drawn at the intersection of one axis with another:

`ILEVEL`	*Effect*
0	All annotation drawn
1	Annotation omitted near intersection (default)

Refer to Examples 23 and 32 for illustrations of `AXLBGP`.

`AXLBJS(CHAXIS,JCHAR)` specifies the position of an annotation label relative to the tick mark:

`JCHAR`	*Justification of annotations*
`'P'`	Preceding the tick mark (default for most axes)
`'F'`	Following the tick mark
`'C'`	Centred on the tick mark (default for horizontal axis on a histogram)
`'B'`	Between tick marks such that value refers to midpoint (default for label axis on a bar chart)
`'D'`	Default position

Refer to Examples 27, 29, 32 and 34 for illustrations of `AXLBJS`.

`AXLBLV(CHAXIS,ILEVEL)` specifies the level of additional annotation on non-linear Cartesian axes and angular polar axes:

`ILEVEL`	*Non-linear Cartesian axes annotation*
0	Determined by extent of scales (default)
1	Powers of 10 only
2	3
3	2,5
4	1, 2, 3, 4, 5, 6, 7, 8, 9

`ILEVEL`	*Angular polar axis annotation*
0	30°, 60°, 120°, 150°, 210°, 240°, 300°, 330° (default)
1	at all 30° intervals
2	at all 10° intervals

Refer to Example 23 for an illustration of `AXLBLV`.

5.6.6 Axis subdivision and grid lines

By default, SIMPLEPLOT chooses an interval which gives between five and ten subdivisions over the range of the axis; tick marks are drawn at all subdivisions but no grid lines are drawn.

`AXSBDV(CHAXIS,OFFSET,DELTA)` specifies the interval for axis subdivisions and an offset value such that all other subdivisions are drawn at multiples of the interval from this point (*ie.* at `OFFSET`, `OFFSET`+`DELTA`, `OFFSET`+(2×`DELTA`), *etc*). `CALL AXSBDV(CHAXIS,0.0,0.0)` can be used to return to default subdivision. Refer to Examples 23 and 28 for illustrations of `AXSBDV`.

`AXGRID(CHAXIS,ILEVEL,LTYPE)` specifies level and style of grid lines. For Cartesian plotting, grid lines are straight and, for polar plotting, grid lines drawn with the angular axis are straight radial lines whereas those drawn with the radial axis are circular. `ILEVEL` specifies the type of grids:

`ILEVEL`	*Type of grid*
0	No grid (default)
1	Partial grids – grid lines at every subdivision of a linear axis, at powers of 10 on a non-linear axis, and at 30° intervals on an angular axis.
2	Full grids – as for partial grids, but with additional lines at unit intervals on a non-linear axis, and at 10° intervals on an angular axis.

LTYPE specifies the type of broken line pattern to be used for grid lines; this argument has the same interpretation as the broken line argument in BRKNCV *etc*. (see Figure G.1, page 156). Refer to Examples 23, 32 and 34 for illustrations of AXGRID.

5.7 How to manipulate data

SIMPLEPLOT includes a number of subroutines which perform no graphics function but can be used to manipulate data values. This section describes the facilities available for recognising missing data values, finding and using maximum and minimum values of a data set, evaluating polynomial functions, and formatting numbers.

5.7.1 Missing data

SIMPLEPLOT recognizes a single `float` value to represent missing data. By default, this value is -1.0×10^{20} (`-1.0E20`) which is an extreme value not normally found in data; `NODATA` can be called to specify an alternative value:

`NODATA(RVAL)` specifies the value, `RVAL`, which SIMPLEPLOT is to interpret as a no-data value. Refer to Example 28 for an illustration of `NODATA`.

No-data values affect *all* plotting and related activities such as conversion or function evaluation. Before SIMPLEPLOT draws anything, it checks whether any of the coordinates of the item to be drawn corresponds to the no-data value; if any coordinate is *missing*, the item is not drawn. As this checking occurs internally at a fairly low level, it affects everything that is drawn at *user coordinates* – marker symbols, text captions, curves joining points, boxes, axes, *etc.* – any item whose position can be specified by the user.

5.7.2 Data limits

The following subroutines enable data limits to be evaluated and then adjusted in order to produce suitable scale limits:

`LIMEXC(DARR,NARR,&VARMIN,&VARMAX)` evaluates the minimum and maximum values in a `float` array; these values are returned in `VARMIN` and `VARMAX` respectively. `LIMEXC` ignores any values which correspond to the current missing data value.

The third and fourth arguments of `LIMEXC` must always be pointers to variables, because SIMPLEPLOT assigns values to them.

`LIMINC(DARR,NARR,&VARMIN,&VARMAX)` evaluates the minimum and maximum values in a `float` array. If the minimum is less than `VARMIN`, `VARMIN` is set to this new value; similarly, if the maximum is greater than `VARMAX`, `VARMAX` is set to the new maximum. `LIMINC` ignores any values which correspond to the current missing data value.

The third and fourth arguments of `LIMINC` must always be the names of variables which have already been assigned values, because SIMPLEPLOT uses their values and may need to assign new values to them.

`KSCALE(START,STOP,DIV,&VSTART,&VSTOP,&VDIV)` converts scale limits such that they span whole axis (scale) subdivisions.

`KSCALE` takes the values `START`, `STOP` and `DIV` which describe the scale requirements, and converts them to `VSTART`, `VSTOP` and `VDIV` which satisfy the requirements and, if necessary, extend the scale limits to coincide with whole multiples of the scale subdivision.

`DIV` is the requested subdivision; if `DIV`=0.0, SIMPLEPLOT allocates a subdivision size related to the range of the scale (*ie.* the subdivision which would be used by default when an axis is drawn). The value of this subdivision is returned in `VDIV` which can then be used to add whole or half subdivisions to extend the scale still further. If `DIV` is non-zero, `VDIV` is given the magnitude of `DIV` and the sign of $\texttt{STOP} - \texttt{START}$.

`VSTART` is a whole multiple of `VDIV` such that $\texttt{START} - \texttt{VDIV} < \texttt{VSTART} \leq \texttt{START}$. If `START` is a whole multiple of `VDIV`, $\texttt{VSTART} = \texttt{START}$.

Similarly, `VSTOP` is a whole multiple of `VDIV` such that $\texttt{STOP} + \texttt{VDIV} > \texttt{VSTOP} \geq \texttt{STOP}$. If `STOP` is a whole multiple of `VDIV`, $\texttt{VSTOP} = \texttt{STOP}$.

KSCALE does *not* affect scales directly, but it computes values which may be used to set scales or axis subdivisions as arguments for AXSBDV, EQSCAL, SCALES, XSCALE or YSCALE.

If VSTART = START or VSTOP = STOP, a data value may be on the exact limits of the scale; in this case VDIV can be used to extend the new scale limits beyond the data limits. For example, to extend scales by at least one interval beyond data, finishing on an axis subdivision:

```
CALL KSCALE(DATMIN,DATMAX,0.0,&VSTART,&VSTOP,&VDIV)
CALL YSCALE(VSTART-VDIV,VSTOP+VDIV,1)
CALL AXSBDV('YC',VSTART,VDIV)
```

Note that KSCALE knows nothing about axis length. Therefore, if the physical length of an axis is too small for annotation labels to be drawn at all multiples of VDIV, the annotation interval is a multiple of VDIV; tick marks are still drawn at all subdivisions.

Example 28 (see page 101) illustrates not only how missing data values can be ignored but also how auxiliary subroutines can be used to automate the allocation of sensible scale limits.

5.7.3 Polynomial functions

The following subroutines enable polynomial functions to be calculated and, if required, to use internally generated coefficients from regression curves.

POLIN(PARR,NP) inputs the polynomial coefficients in PARR before using the SIMPLEPLOT function POLY to calculate polynomial function values for plotting with SHDEFN or BRKNFN.

POLOUT(&PARR) obtains the values of polynomial coefficients held within SIMPLEPLOT. When a curve through data points has been plotted (*eg.* using BRKNCV, SHDEAR *etc.*) with curve type 4 (linear regression) or 5 (quadratic regression) the regression coefficients are held in the locations reserved for polynomial coefficients. The only argument, PARR, is the name of an array into which the coefficients are copied, in ascending polynomial powers, starting from the constant term.

POLY(XVAL) is a FORTRAN function defined within SIMPLEPLOT which calculates the polynomial function determined by the coefficients stored by POLIN.

Examples 25 and 26 illustrate the use of these subroutines.

5.7.4 Formatting numbers

When float numbers are drawn on pictures (*eg.* in pie chart labels) or converted to text strings, the format is chosen to produce the smallest number of characters while still representing the value of the number.

FIGFMT(CHFORM,NUMINT,NDEC) specifies the format of float values when they are either converted to text strings using KREAL or used in pie chart labels (see PIINCL).

CHFORM corresponds to FORTRAN format specifiers:

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

The second argument, NUMINT, gives the number of digits before the decimal point in floating-point numbers. The final argument, NDEC, specifies the number of decimal places.

By default, the format of float numbers is chosen to produce the smallest number of characters while still representing the value of the number; this default can be restored by CALL FIGFMT('G',1,-2).

Refer to Examples 31 and 33 for illustrations of FIGFMT.

`FIGSGN(SIGN,ESIGN)` specifies the sign format of positive components of `float` numbers. The first argument, `SIGN`, refers to the sign of the number, the second, `ESIGN`, to the sign of the exponent. Both are single character (`char *`) arguments

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`

Negative components are *always* preceded by a minus sign. Refer to Example 31 for an illustration of `FIGSGN`.

`KREAL(RVAL,&STR)` converts the `float` number, `RVAL`, into a text string, `STR`. `STR` should be declared as a `char *` variable long enough to represent `RVAL`; `char[30]` should be sufficient for most `floats`.

Refer to Example 31 for an illustration of `KREAL`.

`KNUMB(IVAL,&STR)` converts the integer, `IVAL`, into its equivalent `char *` representation, suitable for writing as text. `STR` should be declared as a `char *` variable long enough to represent `IVAL`; `char[10]` should be sufficient for most `ints`.

Refer to Example 9 for an illustration of `KNUMB`.

5.8 How to use shading

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. An individual shading pattern can be passed to the appropriate subroutine using an `int` number, `ISHADE`=-1, 0, 1, 2, 3, ... *etc.* The pattern is selected from the current set of shading patterns. By default SIMPLEPLOT selects hardware shading patterns if they are available on your device.

`SHSET(ISET)` specifies an alternative set of shading patterns:

`ISET`	*Polygon Area-fill Source*
0	Hardware/firmware area fill, if available (default)
1	Device independent software shading

Refer to Examples 24 and 34 for illustrations of `SHSET`.

5.8.1 Hardware shading patterns

The availability of hardware shading and the number of patterns depends on the device you are using. In order to make full use of each device, the pattern corresponding to each shading pattern number may differ between devices.

On all devices, SIMPLEPLOT resorts to software shading patterns when the hardware patterns have been exhausted; the sequence continues with the next software shading pattern. For example, if a device has two hardware shading patterns and the number of shading colours in use is seven, then shading pattern numbers 1–7 select all the colours with the first hardware pattern, numbers 8–14 select all the colours with the second hardware pattern, and numbers 15–21 select all the colours of the third *software* pattern. The actual appearance of a particular software pattern depends upon the number of colours, the number and direction of hatching angles, and the minimum hatching gap, all of which may be controlled by the user.

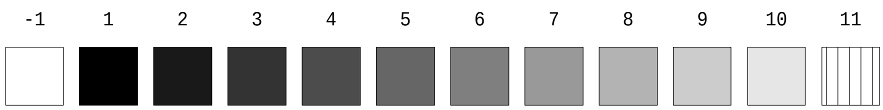


Figure 5.7 Typical hardware shading patterns on a monochrome device

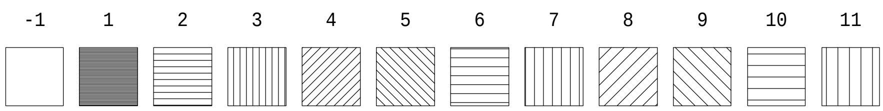


Figure 5.8 Software shading patterns on a monochrome device

Figure 5.7 illustrates the sequence of shading patterns on a monochrome device with ten hardware shading patterns.

5.8.2 Software shading patterns

SIMPLEPLOT also provides software shading patterns using a powerful shading algorithm which can shade areas of any complexity on any device. By default, the software shading patterns use the available colours combined with a sequence of four hatching angles with increasing line separation. The sequence for a monochrome device is illustrated in Figure 5.8 and is described 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).

- 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 (set using `SHDESC`).
- 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.

And so on. If very large pattern numbers are chosen, very large line separation is used.

Each individual pattern can be selected by `int` numbers −1, 0, 1, 2 ... *etc* in subroutines which require a shading pattern as an argument. The positive numbered patterns (*ie.* 1, 2, 3...) depend on:

- how many colours are in use,
- how many hatching angles are in use,
- the minimum separation of hatching lines.

These can all be specified using `SHDESC`:

`SHDESC(IDEG1,IDEG2,IDEG3,IDEG4,NCOLS,GAPCMS)` specifies the characteristics of software shading patterns where `IDEG1`, `IDEG2`, `IDEG3` and `IDEG4` are the four hatching angles, `NCOLS` is the number of colours to be used for shading (see section 5.9.2), and `GAPCMS` is the initial separation of shading lines. `CALL SHDESC(0,90,45,135,0,0.0)` restores the default.

5.8.3 Shading sequences

By default, subroutines which shade a sequence of areas to distinguish them from each other (*eg.* `BARCHT`, `PIECHT`, `SHKEYS`) use the shading patterns in numerical order, 1, 2, 3, *etc.* This default shading sequence can be changed by using either of the following subroutines:

`SHPATT(ISHADE,IPOS)` specifies an alternative pattern, `ISHADE`, to be associated with the `IPOS`th member of the shading sequence. `CALL SHPATT(`*n*`,`*n*`)` restores the default for each position, *n*.

`SQSHAD(IARR,NARR)` specifies an alternative sequence of pattern numbers. `CALL SQSHAD(IARR,0)` restores the default.

Refer to Example 33 for an illustration of `SHPATT` and `SQSHAD`.

The number of shading patterns is unlimited but only 32 patterns can be defined as a sequence. This restricts the sequence controlled by `SQSHAD` and `SHPATT` which is then used for pie charts, bar charts and composite shaded keys.

5.9 How to use colour

SIMPLEPLOT controls the use of colour with *pens*. On your graphics device, a pen may correspond to an actual physical plotting pen or to one of a series of available colours. Pens are identified by positive `int`s but zero can be specified to indicate that background colour should be selected or −1 to indicate that part of the plotting is to be omitted.

5.9.1 Pen selection

Two different methods of pen selection are offered using individual pen selection or pen pointers. SIMPLEPLOT holds four *pen pointers* which are all set initially to refer to the same default pen. These pointers can be reset (as often as required) to any combination of available pens using `SETPNS`, or all to the same pen using `PEN`.

`SETPNS(IPEN1,IPEN2,IPEN3,IPEN4)` specifies the pens associated with the four pen pointers ([1], [2], [3] and [4]) which are used for different types of plotting. The pens are identified by `int` code numbers:

Pen pointer	*Pen usage*
[1]	All general line plotting and axis lines
[2]	All text
[3]	Symbols and partial (major) grid lines
[4]	Additional lines in full grids

`SETPNS` enables the control of pen usage within composite tasks such as axis drawing. For complete details of particular pen usages by any specific subroutine, Refer to the individual subroutine specifications in the *SIMPLEPLOT Reference manual*.

`PEN(IPEN)` specifies which pen is to be used for all subsequent drawing; composite drawing from a single subroutine is all performed with the same pen.

The pen used for any plotting operation is determined by the most recent call of either `PEN` or `SETPNS`; `CALL SETPNS(1,1,1,1)` or `CALL PEN(1)` restores the default.

5.9.2 Shading in colour

The sequence of shading patterns used when more than one colour is available is exactly the same as those used for a monochrome device, but each pattern is repeated in each available colour before the next pattern is used. The number of colours used is normally set to the maximum available on the device, but `SHDESC` can be used to select the use of `NCOLS` colours. For example, after calling `SHDESC` with `NCOLS`=3, shading patterns use colours 1, 2 and 3. On a device with fewer colours than the number selected, the pattern form is retained, even though the colours are not realizable, *ie.* if `NCOLS`=3 on a device with only two colours, patterns are still drawn in groups of three – the third colour for each sequence being drawn with *pen 1*.

If monochrome shading is required on a colour device, calling `SHDESC` with `NCOLS`=1 always selects the patterns in colour number 1. Monochrome shading in a different colour can be chosen by a call of `SHPEN`.

`SHPEN(IPEN)` specifies single colour shading patterns with the specified pen. `CALL SHDESC(IDEG1,IDEG2,IDEG3,IDEG4,0,GAPCMS)` restores the default.

For example, after calling `SHPEN` with `IPEN`=2, all subsequent shading is in the monochrome sequence, but drawn with *pen 2*.

5.10 How to omit drawing and shading

When any part of a picture is not wanted, drawing operations can be either omitted or (on some devices) 'rubbed out' by performing the requested operation using the background colour. Pen values of zero (to draw in background colour) or -1 (to omit drawing) can be used for pen numbers *via* `SETPNS`, `PEN` or `SHPEN`, for pattern numbers in all the available shading operations, and for use in sequences of shading patterns *via* `SHPATT` and `SQSHAD`.

On a device which does not allow plotting in background colour, *pen 0* is treated in the same way as negative values.

- General plotting, text, symbols, *etc*:
  - *Pen 0* – if the device is capable, draw in background colour to perform tasks, otherwise treat as *Pen* -1.
  - *Pen* -1 – bypass all drawing operations and (where possible) all associated computations.
- Shading
  - *Pattern 0* – if the device is capable, shade in background colour with a solid pattern in order to rub out the area described; if the device cannot shade in background colour, pattern 0 is treated in the same way as *Pattern* -1.
  - *Pattern* -1 – no shading is attempted, but if the shading operation would have included a drawn boundary, the boundary is drawn using the colour set by `SHPEN`, or if `SHPEN` has not been called, the pen pointed to by the first pen pointer.

Refer to Example 13 for an illustration of rub-out.

5.11 How to thicken lines

The lines used for general line plotting, software marker symbols and software text can be thickened but, as with colour, the availability of line thickening depends on the capabilities of the device and the device driver. `THCKMG` can be used to control the thickness of lines:

`THCKMG(CHTYPE,FACTOR)` specifies the magnification, `FACTOR`, of the 'normal' line thickness for the type of graphics described by `CHTYPE`:

`CHTYPE`	*Type of plotting*
`'L'`	Line drawing
`'T'`	Selected software text
`'M'`	Software marker symbols
`'*'`	All of the above

Software shading lines are not affected by `THCKMG`.

Note that you may not get what you ask for; not just because the device is not capable of thickening, but because one request is indistinguishable from another request – *eg.* multiples of 2.25 and 1.75 may both result in a multiple of 2.0.

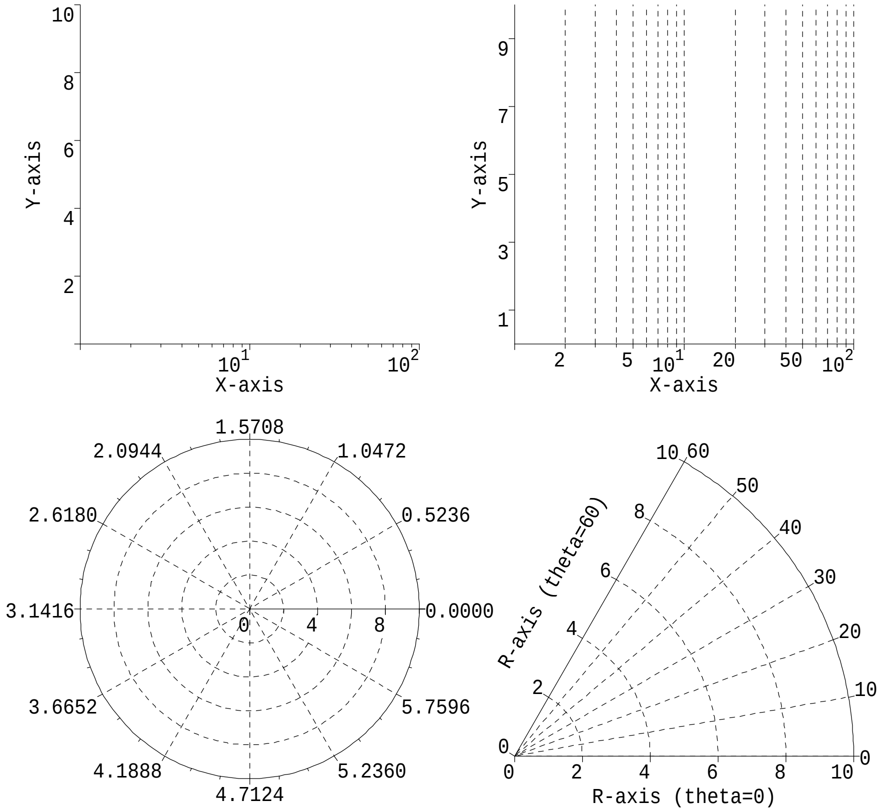
6. Cookbook

This chapter is made up of a collection of examples which use features of SIMPLEPLOT described in Chapters 3, 4 and 5 in combination with each other.

6.1 Drawing axes

This example demonstrates how:

- to use different methods of drawing and annotating axes,
- to use wild cards to change more than one type of axis at once.



Explanation of subroutines

`AXGRID(CHAXIS,ITYPE,LTYPE)` specifies that grid lines are to be drawn at annotated axis subdivisions drawn by composite axis drawing subroutines or `AXIS7`, `AXLAB7`, `AXTXT7`. The final argument, `LTYPE`, specifies the type of broken line from Figure G.1 (see page 156) to be used.

`AXIS7(CHAXIS,CAP)` draws an individual axis on an existing picture. The last argument provides an axis caption.

`AXLBGP(CHAXIS,ILEVEL)` specifies the level of annotations drawn where two axes intersect.

`AXLBLV(CHAXIS,ILEVEL)` specifies the level of additional labelling on the angular axis of a polar chart or on a non-linear Cartesian axis.

`AXSBDV(CHAXIS,OFFSET,DELTA)` specifies the interval at which an axis is to be subdivided and annotated.

`COORDS(IUNITS)` with `IUNITS`=2, specifies polar interpretation of coordinates (r, θ) with θ in radians. `COORDS` also affects the type of new pictures and the axis annotation of the angular polar axis.

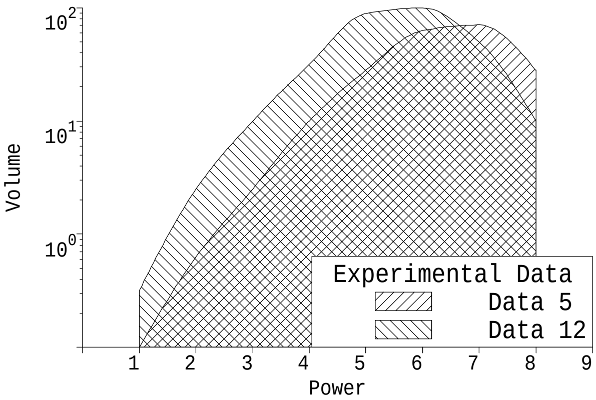
```
      PROGRAM PRIM23
      PARAMETER(RADIUS=10.0)
      CALL GROUP(2,2)
C non-linear Cartesian axes
      CALL SCALES(1.0,100.0,2, 0.0,10.0,1)
      CALL AXLBLV('XC',1)
      CALL AXES7 ('X-axis','Y-axis')
C individual Cartesian axes
      CALL NEWPIC
      CALL AXLBLV('XC',3)
      CALL AXGRID('XC',2,-1)
      CALL AXIS7 ('XC','X-axis')
      CALL AXSBDV('YC',1.0,2.0)
      CALL AXIS7 ('YC','Y-axis')
C polar axes using COORDS and POLAR7
      CALL COORDS(2)
      CALL CHMASK(.TRUE.)
      CALL AXLBGP('AP',0)
      CALL AXGRID('*P',1,-1)
      CALL AXLBLV('AP',1)
      CALL POLAR7(RADIUS,' ')
C polar axes using EQSCAL and AXES7
      CALL EQSCAL(0.0,RADIUS,0.0,60.0,1)
      CALL AXLBLV('AP',2)
      CALL AXLBGP('RP',0)
      CALL AXES7 ('R-axis (theta=0)','R-axis (theta=60)')
      CALL ENDPLT
      END
```

Example 23. Axis drawing

6.2 Drawing keys and captions

This example demonstrates how:

- to select software shading,
- to change the justification of key and caption entries,
- to use a masked key area for captions and key entries.




```
      PROGRAM PRIM24
      PARAMETER(NPTS=8,X1=1.0,DX=1.0)
      REAL YARR1(NPTS),YARR2(NPTS)
      DATA YARR1/0.10,0.63,2.50,10.00,28.18,63.10,70.79,28.18/
      DATA YARR2/0.32,2.51,10.00,31.62,89.13,100.00,50.00,10.00/
C specify software shading and its characteristics
      CALL SHSET(1)
      CALL SHDESC(45,45,135,135,1,0.05)
C specify linear vs logarithmic plotting scales
      CALL SCALES(0.0,9.0,1,0.1,100.0,2)
C start picture and draw pair of axes
      CALL AXES7('Power','Volume')
C draw key box for 3 lines of 7 characters (max)
      CALL DEFKEY(2,'B','R',3,7)
C plot curve of X vs YARR1 with shading pattern 2
      CALL SHDEAR(X1,DX,YARR1,NPTS,2)
C plot curve of X vs YARR2 with shading pattern 3
      CALL SHDEAR(X1,DX,YARR2,NPTS,3)
C enter caption in key area
      CALL ADDJST('C')
      CALL ADDCP7('Experimental Data')
C write keys to shading patterns 2 and 3
      CALL ADDJST('L')
      CALL SHDEK7(2,'Data 5')
      CALL SHDEK7(3,'Data 12')
      CALL ENDPLT
      END
```

Example 24. Keys and captions

Explanation of subroutines

`ADDCP7(CAP)` adds the caption `CAP` in the next line of the defined key area.

`ADDJST(JCHAR)` with `JCHAR='C'`, specifies that captions and key entries are to be `'C'`entred within the space available.

`DEFKEY(ITYPE,VCHAR,HCHAR,NROWS,NCOLMS)` defines an area for a key which is used according to the value of the first argument, `ITYPE`:

`ITYPE`	*Key or caption area*
1	Area can be over-drawn
2	Area is masked

The key is positioned according to `VCHAR` and `HCHAR`, in this case, at the `'B'`ottom, `'R'`ight of the picture. The positional descriptors, `VCHAR` and `HCHAR`, can take any of the standard values described in Table 5.1 (see also Figures 5.4 and 5.5).

`SHDEK7(ISHADE,CAP)` enters the key caption, `CAP`, alongside a sample of the shading pattern, `ISHADE`. Refer to Figure G.2, page 157 for examples of software shading patterns.

`SHDESC(IDEG1,IDEG2,IDEG3,IDEG4,NCOLS,GAPCMS)` specifies the characteristics of software shading patterns. `IDEG1`=`IDEG2`=45 and `IDEG3`=`IDEG4`=135 describe just two hatching angles; `NCOLS`=1 specifies that only one colour is to be used; `GAPCMS` is the initial separation of shading lines (in cm).

`SHSET(ISET)` with `ISET`=1, specifies that software shading patterns are to be used even if hardware patterns are available on the device; these patterns consist of hatching lines. Where shaded areas overlap a third pattern emerges.

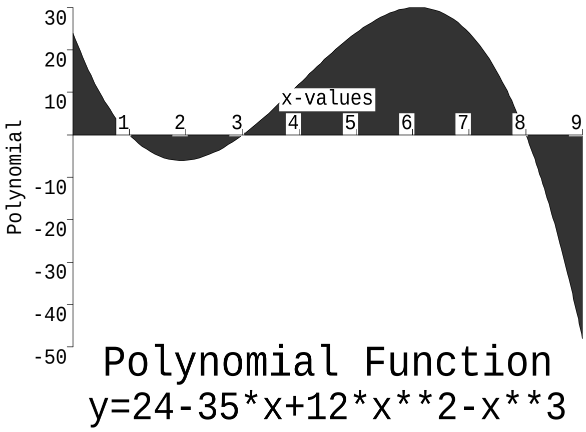
6.3 Multi-line titles and polynomials

This example demonstrates how:

- to mask text,
- to draw a multi-line title,
- to draw a function from the SIMPLEPLOT-defined polynomial function.

This example also illustrates some of the default characteristics of axes:

- Cartesian axes intersect at the origin or, if (0,0) does not lie within the plotting scales, at the point nearest to the origin.
- axis annotation is omitted at the axis intersection to avoid ambiguity.
- axis annotation is written on whichever side of the axis is nearest to the outside of the picture.




```
      PROGRAM PRIM25
      REAL PARR(4)
C declare function
      EXTERNAL POLY
      DATA PARR/24.0,-35.0,12.0,-1.0/
C specify character masking
      CALL CHMASK(.TRUE.)
C specify plotting scales
      CALL SCALES(0.0,9.0,1, -50.0,30.0,1)
C start picture and draw pair of axes
      CALL AXES7('x-values','Polynomial')
C pass polynomial coefficients to SIMPLEPLOT
      CALL POLIN(PARR,4)
C draw shaded curve of polynomial function
      CALL SHDEFN(POLY,3)
C write two lines of title under the picture, centred
      CALL TEXTLF(1.1)
      CALL TITLE7('U','C','Polynomial Function')
      CALL TITLE7('U','C','y=24-35*x+12*x**2-x**3')
      CALL ENDPLT
      END
```

Example 25. Multi-line titles

Explanation of subroutines

CHMASK(TORF) with TORF=.TRUE. specifies that text is to be masked.

POLIN(PARR,NP) inputs the polynomial coefficients in PARR before using the SIMPLEPLOT function POLY to calculate polynomial function values for plotting with SHDEFN or BRKNFN.

`POLY(XVAL)` is a FORTRAN function defined within SIMPLEPLOT which calculates the polynomial function determined by the coefficients stored by `POLIN`.

`TEXTLF(FACTOR)` with `FACTOR`=1.1, specifies a reduced space left between successive lines of text – in this case, in a multi-line title. `TEXTLF` can be called with any `float` value greater than or equal to 1.0; by default, `FACTOR`=1.2 is used.

`TITLE7(VCHAR,HCHAR,CAP)` draws the text string, `CAP`, as a title.

The title is positioned according to `VCHAR` and `HCHAR` which, in this example, is `'C'`entred, `'U'`nder the picture; refer to page 69 for suitable values of `VCHAR` and `HCHAR`.

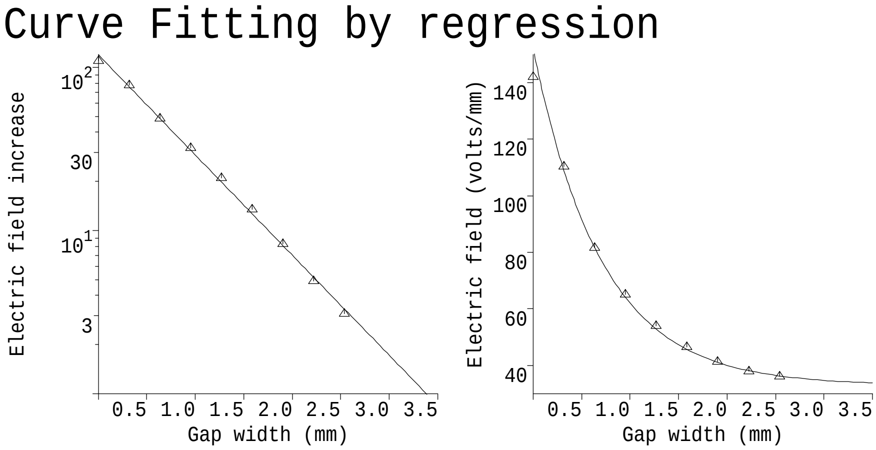
If a title is too long to fit within the limits determined by its descriptive position, it is either shrunk, truncated or both:

- If variable-sized text is in use (*ie.* `TEXTSZ` has not been called), the size of text to be used for the title is reduced until it fits or until the minimum text size is reached.
- If fixed-sized text is in use, or the minimum text size has been reached, the title is truncated, character by character, until it fits.

In this way, successive lines of a multi-line title may be written with varying text size if lines are too long to fit.

6.4 Curve fitting by regression

This example illustrates how SIMPLEPLOT can be used to initiate the traditional method of curve fitting by plotting on a logarithmic plot. The regression coefficients of a straight line fitted to data on a linear *vs.* logarithmic plot are derived and then used to set up an exponential approximation to the data.



Explanation of subroutines

`CVTYPE(ITYPE)` with `ITYPE`=4, specifies that a linear regression curve is to be plotted by `BRKNCV`.

`POLOUT(&PARR)` obtains the values of polynomial coefficients held within SIMPLEPLOT. When a curve through data points has been plotted (*eg.* using `BRKNCV`, `SHDEAR` *etc.*) with curve type 4 (linear regression) or 5 (quadratic regression) the regression coefficients are held in the locations reserved for polynomial coefficients. The only argument, `PARR`, is the name of an array into which the coefficients are copied, in ascending polynomial powers, starting from the constant term.

`YSCALE(YSTART,YSTOP,IYTYPE)` specifies the y plotting scale only, leaving the x scale unchanged. The scale is defined in the same way as for `SCALES` – initial scale value, final scale value and scale type (`IYTYPE`=0 for natural scaling (centimetres), 1 for linear, 2 for logarithmic or 3 for normal probability (%)).

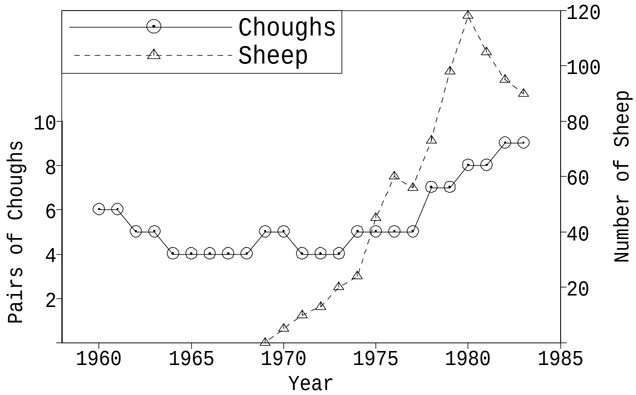
```
      PROGRAM PRIM26
      PARAMETER(NPTS=9,YMIN=33.0)
      REAL XARR(NPTS),Y1ARR(NPTS),Y2ARR(NPTS),PARR(2)
C declare function
      EXTERNAL YFUN
      DATA XARR /0.0,0.3175,0.6350,0.9525,1.270,1.5875,1.9050,
     +           2.2225,2.54/
      DATA Y1ARR/142.0,110.4,81.6,65.0,54.0,46.5,41.3,37.9,36.1/
C specify group of pictures - two across and one down
      CALL GROUP(2,1)
C specify plotting scales linear vs logarithmic
      CALL SCALES(0.0,3.5,1,1.0,120.0,2)
C start picture and draw pair of axes
      CALL AXES7('Gap width (mm)','Electric field increase')
C subtract saturation constant (YMIN) and plot
      DO 10 I=1,NPTS
        Y2ARR(I)=Y1ARR(I)-YMIN
 10   CONTINUE
      CALL MARKCV(XARR,Y2ARR,NPTS,2,1)
C compute and draw linear regression on log scale
      CALL CVTYPE(4)
      CALL BRKNCV(XARR,Y2ARR,NPTS,0)
C get linear regression coefficients
      CALL POLOUT(PARR)
C change vertical scale to linear for original data
      CALL YSCALE(30.0,150.0,1)
C start second picture and draw pair of axes
      CALL AXES7('Gap width (mm)','Electric field (volts/mm)')
C draw the curve of function YFUN
      CALL BRKNFN(YFUN,0)
C mark data points with symbols
      CALL MARKCV(XARR,Y1ARR,NPTS,2,1)
C draw title and terminate plotting
      CALL TITLE7('H','L','Curve Fitting by regression')
      CALL ENDPLT
      END
C user-defined function in terms of SIMPLEPLOT function POLY
      FUNCTION YFUN(X)
      YFUN=33.0+10.0**POLY(X)
      END
```

Example 26. Curve fitting by regression

6.5 Multiple scales on an x-y plot

This example demonstrates how:

- to combine plotting using different scales on the same picture,
- to draw individual axes showing the different scales,
- to draw a key for both marker symbols and broken line patterns,
- to select a curve made up from straight lines from point to point.




```
      PROGRAM PRIM27
      PARAMETER(NCH=24,NSH=15)
      REAL CHOUGH(NCH),SHEEP(NSH)
      DATA CHOUGH/6.0,6.0,5.0,5.0,4.0,4.0,4.0,4.0,4.0,5.0,5.0,4.0,
     +            4.0,4.0,5.0,5.0,5.0,5.0,7.0,7.0,8.0,8.0,9.0,9.0/
      DATA SHEEP /0.0,5.0,10.0,13.0,20.0,24.0,45.0,60.0,56.0,73.0,
     +            98.0,118.0,105.0,95.0,90.0/
C specify scales and axis characteristics
      CALL SCALES(1958.0,1985.0,1,0.0,15.0,1)
      CALL AXLBJS('*C','C')
      CALL AXRNGE('YC',0.0,10.0)
C start picture and draw axes
      CALL AXES7('Year','Pairs of Choughs')
C specify straight lines and plot 'curves'
      CALL CVTYPE(3)
      CALL BRKNAR(1960.0,1.0,CHOUGH,NCH,0)
      CALL MARKAR(1960.0,1.0,CHOUGH,NCH,58,1)
C specify new y-scale and y-axis characteristics
      CALL YSCALE(0.0,120.0,1)
      CALL AXLOCN('YC','F')
      CALL AXRNGE('YC',0.0,0.0)
C draw second y-axis and plot second curve
      CALL AXIS7('YC','Number of Sheep')
      CALL BRKNAR(1969.0,1.0,SHEEP,NSH,-1)
      CALL MARKAR(1969.0,1.0,SHEEP,NSH,2,1)
C define key area and add entries
      CALL DEFKEY(1,'T','L',2,7)
      CALL BOTHK7(0,58,'Choughs')
      CALL BOTHK7(-1,2,'Sheep')
C draw perimeter and terminate plotting
      CALL PERIM
      CALL ENDPLT
      END
```

Example 27. Multiple scales on an x-y plot

Explanation of subroutines

`AXLBJS(CHAXIS,JCHAR)` specifies the position of axis annotation labels relative to the tick marks. In this example, Cartesian axis labels (`CHAXIS`='XC' or 'YC') are centred.

`AXLOCN(CHAXIS,LCHAR)` specifies the location of the axis with respect to the picture boundary.

`AXRNGE(CHAXIS,START,STOP)` with `CHAXIS='YC'`, specifies the sub-range over which y-axis is to be drawn as an alternative to the full range of the scale.

`BOTHK7(LTYPE,MKTYPE,CAP)` enters the key caption, `CAP`, alongside a sample of the broken line pattern, `LTYPE`, with the marker symbol, `MKTYPE`, centred on the sample line.

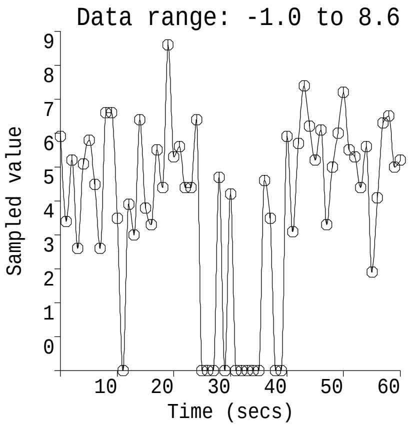
Refer to Figure G.1 (see page 156) for examples of the different line patterns available, and to Figure G.5 (see page 159) for examples of the available marker symbols.

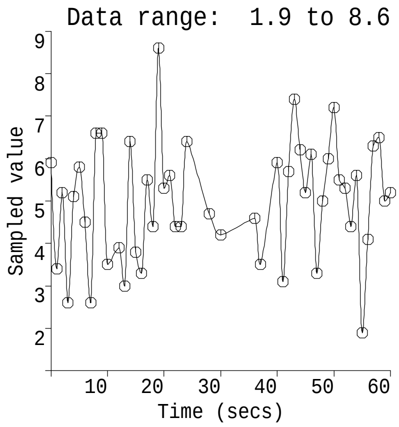
`CVTYPE(ITYPE)` with `ITYPE`=3, specifies that straight lines from point-to-point are to be plotted by `BRKNCV`.

6.6 Missing data

This example demonstrates how:

- missing data values can be ignored,
- auxiliary subroutines can be used to automate the allocation of sensible scale limits.





Explanation of subroutines

`AXCRSS(CHAXIS,CROSS)` specifies the intersection of the x-axis with the y-axis. In this case, `CROSS` is given in terms of the y plotting scale.

`KSCALE(START,STOP,DIV,&VSTART,&VSTOP,&VDIV)` converts scale limits such that they span whole axis subdivisions. `KSCALE` does not affect the scales directly but the values returned are then used in `YSCALE` to change the y-scale.

`LIMEXC(RARR,NARR,&VARMIN,&VARMAX)` (LIMits EXClusive) scans the values in an array (ignoring no-data values) to find the minimum and maximum values.

`NODATA(RVAL)` specifies a value which SIMPLEPLOT is to interpret as a missing data value; by default, this value is -1.0×10^{20} (`-1.0E20`).

`XSCALE(XSTART,XSTOP,IXTYPE)` specifies the x-scale only, leaving the y-scale unchanged. The scale is defined in the same way as for `SCALES` – initial scale value, final scale value, and scale type (`IXTYPE`=0 for natural scaling (centimetres), 1 for linear, 2 for logarithmic or 3 for normal probability (%)).

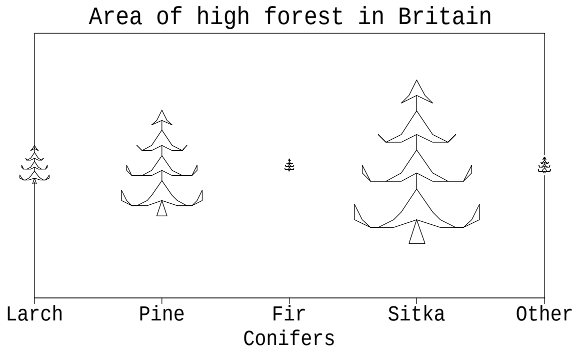
```
      PROGRAM PRIM28
      PARAMETER(NPTS=61,XSTART=0.0,XSTOP=60.0)
      REAL YARR(NPTS)
      DATA YARR/5.9,3.4,5.2,2.6,5.1,5.8,4.5,2.6,6.6,6.6,3.5,-1.0,3.9,
     +          3.0,6.4,3.8,3.3,5.5,4.4,8.6,5.3,5.6,4.4,4.4,6.4,-1.0,
     +         -1.0,-1.0,4.7,-1.0,4.2,-1.0,-1.0,-1.0,-1.0,-1.0,4.6,
     +          3.5,-1.0,-1.0,5.9,3.1,5.7,7.4,6.2,5.2,6.1,3.3,5.0,6.0,
     +          7.2,5.5,5.3,4.4,5.6,1.9,4.1,6.3,6.5,5.0,5.2/
C specify layout
      CALL GROUP(2,1)
C specify X scale
      CALL XSCALE(XSTART,XSTOP,1)
C plot curve including 'missing data' points
      CALL PIC(YARR,NPTS)
      CALL NODATA(-1.0)
C plot curve excluding 'missing data' points
      CALL PIC(YARR,NPTS)
      CALL ENDPLT
      END
C subroutine to draw picture
      SUBROUTINE PIC(YARR,NARR)
      REAL YARR(NARR)
      CHARACTER*20 CAP
C find limits of data
      CALL LIMEXC(YARR,NARR,DATMIN,DATMAX)
C convert data limits to whole subdivisions
      CALL KSCALE(DATMIN,DATMAX,0.0,YSTART,YSTOP,YDIV)
C specify y-scale and axis characteristics
      CALL YSCALE(YSTART,YSTOP,1)
      CALL AXSBDV('YC',1.0,YDIV)
      CALL AXCRSS('XC',YSTART)
C start new picture and draw axes
      CALL AXES7('Time (secs)','Sampled value')
C plot data and title
      CALL BRKNAR(0.0,1.0,YARR,NARR,0)
      CALL MARKAR(0.0,1.0,YARR,NARR,1,1)
      WRITE(CAP,'(F4.1,A3,F4.1)') DATMIN,' to',DATMAX
      CALL TITLE7('T','R','Data range: '//CAP)
      END
```

Example 28. Missing data

6.7 Marker symbols

This example demonstrates how:

- to draw marker symbols at varying sizes to represent numeric values,
- to mask marker symbols in order to prevent the picture boundary being drawn through the outer symbols,
- to add a set of non-numeric axis labels,
- to use a caption area as an alternative to a title.




```
      PROGRAM PRIM29
      PARAMETER(NE=5)
      REAL EVRGRN(NE)
      CHARACTER*8 ENAMES(NE)
      DATA EVRGRN/151.76,415.36,47.39,642.75,63.71/
      DATA ENAMES/'Larch','Pine','Fir','Sitka','Other'/
C specify scales and start new picture
      CALL SCALES(1.0,REAL(NE),1, 0.0,1.0,1)
      CALL NEWPIC
C specify axis characteristics, draw axis and add labels
      CALL AXLBJS('XC','C')
      CALL AXCLR('XC',1)
      CALL AXIS7('XC','Conifers')
      CALL AXTXT7('XC',1.0,1.0,ENAMES,NE)
C draw marker symbols
      CALL MKMASK(.TRUE.)
      DO 10 I=1,NE
        CALL MKSIZE(EVRGRN(I)/500.0)
        CALL MARKPT(REAL(I),0.5,54)
 10   CONTINUE
C define caption area and add caption as a title
      CALL DEFCAP(1,'O','C',1,30)
      CALL ADDCP7('Area of high forest in Britain')
C draw perimeter and terminate plotting
      CALL PERIM
      CALL ENDPLT
      END
```

Example 29. Marker symbols

Explanation of subroutines

`AXCLR(CHAXIS,ILEVEL)` with `ILEVEL`=1, prevents the automatic drawing of subdivision marks (tick marks) and axis annotation.

`AXTXT7(CHAXIS,W1,STEP,LABARR,NARR)` draws a set of axis annotation labels, tick marks and grid lines (if specified by `AXGRID`) at regular intervals along the specified axis.

To draw an axis with non-numeric labels, suppress the normal numerical annotations by calling `AXCLR` before the axis is drawn; then call `AXTXT7` to draw a set of alternative labels such that `LABARR[0]` is positioned at `W1`, `LABARR[1]` is positioned at `W1+STEP` *etc.*

`DEFCAP(ITYPE,VCHAR,HCHAR,NROWS,NCOLMS)` with `ITYPE`=1, defines an area for captions such that other plotting may occur within the area. The caption area is positioned according to `VCHAR` and `HCHAR`, `'C'`entred, `'O'`ver the picture. `VCHAR` and `HCHAR`, can take any of the standard values described in Table 5.1 (see also Figures 5.4 and 5.5).

`MKMASK(.TRUE.)` specifies that all future symbols are to be masked, preventing any subsequent plotting being drawn over the symbol.

`MKSIZE(CMS)` specifies the size of marker symbols. In this example, the single argument of `MKSIZE` is set to be proportional to the relevant data value, thus using the symbol to show quantity.

`PERIM` draws a box around the current picture avoiding masked areas.

Similar subroutines

`AXLAB7(CHAXIS,W,CAP)` draws the individual annotation label, `CAP`, at position `W` on the axis specified by `CHAXIS`. `W` is interpreted in terms of the current plotting scale.

6.8 Text in pictures – alternative character sets

This example demonstrates how:

- to select different character sets,
- to use captions to draw a picture made up entirely of text strings.

Explanation of subroutines

`ADDCP7(CAP)` enters the caption, `CAP`, in the next line of the defined area.

`ADDJST(JCHAR)` specifies the justification of captions within the caption area.

`BLNKKY` leaves the next entry in the caption area blank.

`BOXCAP(TORF)` with `TORF=.TRUE.` specifies that a box is to be drawn around all new caption areas; by default, no box is drawn.

`CHSET(ISET)` specifies an alternative source of character sets (or fonts).

`THCKMG(CHTYPE,FACTOR)` with `CHTYPE='T'`, specifies that software `'T'`ext is to be thickened by a factor, `FACTOR`.

```
      PROGRAM PRIM30
C start picture
      CALL NEWPIC
C specify caption characteristics and define caption area
      CALL BOXCAP(.TRUE.)
      CALL ADDJST('C')
      CALL DEFCAP(1,'C','C',20,40)
C add blank lines and captions
      CALL BLNKKY
      CALL CHSET(7)
      CALL THCKMG('T',2.5)
      CALL ADDCP7('*** Today''s menu ***')
      CALL BLNKKY
C first course
      CALL THCKMG('*',1.0)
      CALL ADDCP7('-- APPETISERS --')
      CALL BLNKKY
      CALL CHSET(8)
      CALL ADDCP7('Whitebait & tartar sauce')
      CALL ADDCP7('or Tortilla chips & cheese dip')
      CALL ADDCP7('or Garlic mushrooms')
      CALL BLNKKY
C second course
      CALL CHSET(7)
      CALL ADDCP7('-- MAIN COURSE --')
      CALL BLNKKY
      CALL CHSET(8)
      CALL ADDCP7('Trout')
      CALL ADDCP7('or Pork in cider')
      CALL ADDCP7('served with ratatouille & parsnips')
      CALL BLNKKY
C third course
      CALL CHSET(7)
      CALL ADDCP7('-- SWEET --')
      CALL BLNKKY
      CALL CHSET(8)
      CALL ADDCP7('Toffee pudding')
      CALL ADDCP7('Cheese & biscuits')
C terminate plotting
      CALL ENDPLT
      END
```

Example 30. Text

6.9 Text in pictures – controlling number format

This example demonstrates how:

- to position captions within a defined area by row and column,
- to select alternative number formats,
- to convert `float` numbers into text strings.

x	x*x	sqrt(x)	exp(x)	sin(x)	cos(x)	sin(x)+cos(x)
0.000	0	0.00E0	1.00	+0.000	1.000	1.000
1.000	1	1.00E0	2.72	+0.841	0.540	1.382
2.000	4	1.41E0	7.39	+0.909	-0.416	0.493
3.000	9	1.73E0	20.09	+0.141	-0.990	-0.849
4.000	16	2.00E0	54.60	-0.757	-0.654	-1.410
5.000	25	2.24E0	148.41	-0.959	0.284	-0.675
6.000	36	2.45E0	403.43	-0.279	0.960	0.681
7.000	49	2.65E0	1096.63	+0.657	0.754	1.411
8.000	64	2.83E0	2980.96	+0.989	-0.146	0.844
9.000	81	3.00E0	8103.08	+0.412	-0.911	-0.499
10.000	100	3.16E0	2.20E+4	-0.544	-0.839	-1.383
11.000	121	3.32E0	5.99E+4	-1.000	0.004	-0.996
12.000	144	3.46E0	1.63E+5	-0.537	0.844	0.307
13.000	169	3.61E0	4.42E+5	+0.420	0.907	1.328
14.000	196	3.74E0	1.20E+6	+0.991	0.137	1.127
15.000	225	3.87E0	3.27E+6	+0.650	-0.760	-0.109

Explanation of subroutines

`FIGFMT(CHFORM,NUMINT,NDEC)` specifies the fixed-point or floating-point representation of `REAL` numbers. `CHFORM` specifies the format; `NUMINT` gives the number of digits before the decimal point in floating-point numbers and `NDEC` specifies the number of decimal places.

`FIGSGN(SIGN,ESIGN)` specifies the format of signs of positive components. `SIGN` applies to the sign of the number, `ESIGN` to the sign of the exponent; both are `char[1]` arguments – `'+'` for a plus sign, `''` for a space, and `'D'` to restore the default (neither a sign nor a space).

`KREAL(RVAL,&STR)` converts the `float` number, `RVAL`, into a text string, `STR`. `STR` should be declared as a `char *` variable long enough to represent `RVAL`; `char[30]` should be sufficient for most `float`s.

`PUTCP7(IROW,ICOLM,CAP)` writes the caption, `CAP`, in the specified position within the caption area defined by `DEFCAP`. The position is determined as line `IROW` (counting down from `IROW`=1 at top), character position `ICOLM` (counting across from `ICOLM`=1 at left).

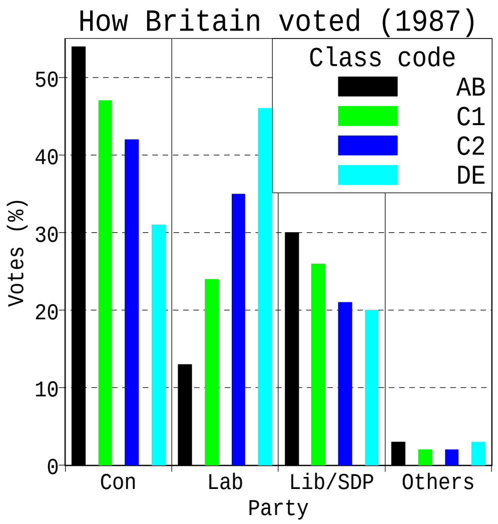
```
      PROGRAM PRIM31
      CALL BOXCAP(.TRUE.)
      CALL NEWPIC
      CALL DEFCAP(1,'C','C',20,65)
C add column headings
      CALL PUTCP7(2,4,'x')
      CALL PUTCP7(2,11,'x*x')
      CALL PUTCP7(2,17,'sqrt(x)')
      CALL PUTCP7(2,26,'exp(x)')
      CALL PUTCP7(2,36,'sin(x)')
      CALL PUTCP7(2,45,'cos(x)')
      CALL PUTCP7(2,52,'sin(x)+cos(x)')
C enter numbers row-by-row
      DO 10 NROW=4,19
        X = REAL(NROW-4)
        CALL ENTRY('F',1,3,'D','D',NROW, 2,X)
        CALL ENTRY('F',3,0,'D','D',NROW,11,X*X)
        CALL ENTRY('E',1,2,'D','D',NROW,17,SQRT(X))
        CALL ENTRY('G',2,2,' ','+',NROW,26,EXP(X))
        CALL ENTRY('F',1,3,'+','+',NROW,36,SIN(X))
        CALL ENTRY('F',1,3,' ',' ',NROW,45,COS(X))
        CALL ENTRY('F',1,3,'D','D',NROW,55,SIN(X)+COS(X))
 10   CONTINUE
      CALL ENDPLT
      END
C subroutine to draw table entry
      SUBROUTINE ENTRY(CHFM,NINT,NDEC,SIGN,ESIGN,IROW,ICOL,XVAL)
      CHARACTER CHFM,SIGN,ESIGN,STRING*20
C specify format of number and sign
      CALL FIGFMT(CHFM,NINT,NDEC)
      CALL FIGSGN(SIGN,ESIGN)
C convert number to string and enter in table
      CALL KREAL(XVAL,STRING)
      CALL PUTCP7(IROW,ICOL,STRING)
      END
```

Example 31. Tabulated captions

6.10 Bar chart

This example demonstrates how:

- to draw a bar chart from 2-dimensional data,
- to draw grids on a bar chart,
- to draw a composite key to the sequence of shading patterns,
- to select an alternative format of individual bars,
- to use AX* subroutines to control bar chart axes.



Explanation of subroutines

BARCHT(D2ARR,LABARR,NARR,NSETS,CAPN,CAPL) draws a bar chart on the current picture representing data values held in the 2-D array, D2ARR.

BARFMT(JCHAR,IVAL) specifies a reduced width of bars from 80% (default) to 50% of the width available; justification of the bars is centred.

BARRNG(VMIN,VMAX) specifies an extended range of the numerical scale from the data maximum (54.0) to 55.0.

BOXPIC(TORF) with TORF=.TRUE. specifies that a box is to be drawn around all new pictures; by default, no box is drawn.

SHKEYS(VCHAR,HCHAR,LABARR,NARR,CAP) draws a complete key in a masked key box with the specified caption, CAP, at the top; it then adds a set of labels, LABARR, with samples of the first NARR shading patterns in the current sequence.

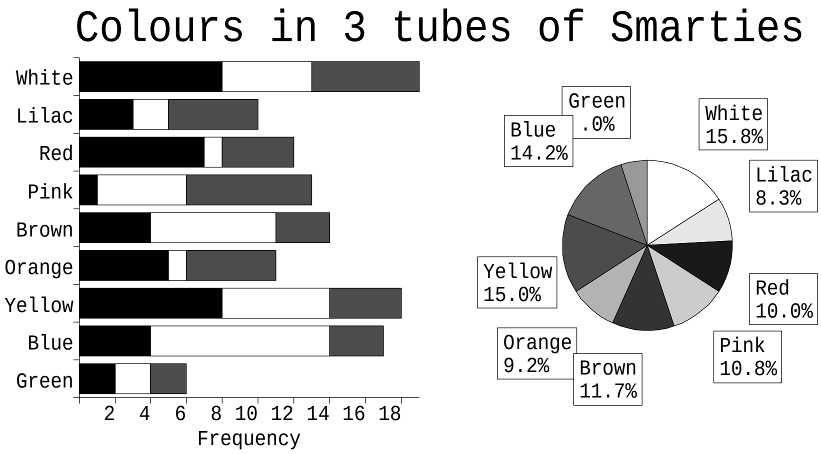
```
      PROGRAM PRIM32
      PARAMETER(NCLASS=4,NPARTY=4)
      REAL VOTES(NPARTY,NCLASS)
      CHARACTER*20 CLSLAB(NCLASS),PARTY(NPARTY)
      DATA VOTES/54.0,13.0,30.0,3.0,
     +           47.0,24.0,26.0,2.0,
     +           42.0,35.0,21.0,2.0,
     +           31.0,46.0,20.0,3.0/
      DATA CLSLAB/'AB','C1','C2','DE'/
      DATA PARTY/'Con','Lab','Lib/SDP','Others'/
C specify picture and axis characteristics
      CALL BOXPIC(.TRUE.)
      CALL AXLBGP('NB',0)
      CALL AXLBJS('NB','C')
      CALL AXGRID('NB',1,-1)
      CALL AXGRID('LB',1,0)
C specify bar chart characteristics
      CALL BARFMT('C',50)
      CALL BARRNG(0.0,55.0)
C start new picture
      CALL NEWPIC
C draw key to shading patterns in a blanked area
      CALL SHKEYS('T','R',CLSLAB,NCLASS,'Class code')
C draw bar chart
      CALL BARCHT(VOTES,PARTY,NPARTY,NCLASS,'Votes (%)','Party')
C add title and terminate plotting
      CALL TITLE7('T','C','How Britain voted (1987)')
      CALL ENDPLT
      END
```

Example 32. Bar chart and key

6.11 Bar chart and pie chart

This example demonstrates how:

- to adjust the margin distribution for different types of picture,
- to change the sequence of shading patterns,
- to control the format of numbers in pie chart labels.




```
      PROGRAM PRIM33
      PARAMETER(NCOLS=9,NTUBES=3)
      CHARACTER*6 NAMES(NCOLS)
      REAL FREQ(NCOLS,NTUBES),TOTAL(NCOLS)
      INTEGER ICOLS(NCOLS)
      DATA NAMES/'White','Lilac','Red','Pink','Brown','Orange',
     +           'Yellow','Blue','Green'/
      DATA FREQ/8.0,3.0,7.0,1.0,4.0,5.0,8.0,4.0,2.0,
     +          5.0,2.0,1.0,5.0,7.0,1.0,6.0,10.0,2.0,
     +          6.0,5.0,4.0,7.0,3.0,5.0,4.0,3.0,2.0/
      DATA TOTAL/19.0,10.0,12.0,13.0,14.0,11.0,18.0,17.0,6.0/
      DATA ICOLS/-1,10,2,9,3,8,4,5,7/
      CALL GROUP(2,1)
C specify bar chart characteristics
      CALL AXLOCN('NB','P')
      CALL BARTYP('H',2)
      CALL SHPATT(-1,2)
C start picture and draw bar chart
      CALL NEWPIC
      CALL BARCHT(FREQ,NAMES,NCOLS,3,'Frequency',' ')
C specify pie chart characteristics
      CALL MARGDV('O')
      CALL FIGFMT('F',2,1)
      CALL SQSHAD(ICOLS,NCOLS)
      CALL PIPOSL(3)
      CALL PIINCL(.TRUE.,.FALSE.,.TRUE.)
C start picture and draw pie chart
      CALL NEWPIC
      CALL PIECHT(TOTAL,NAMES,NCOLS)
C draw title and terminate plotting
      CALL TITLE7('H','C','Colours in 3 tubes of Smarties')
      CALL ENDPLT
      END
```

Example 33. Bar and pie charts

Explanation of subroutines

`BARTYP(OCHAR,ITYPE)` specifies horizontal bars (`OCHAR='H'`) which are to be stacked together (`ITYPE=2`).

`FIGFMT(CHFORM,NUMINT,NDEC)` specifies the format of numbers included in the pie chart labels.

`MARGDV(PICTYP)` specifies the picture margin distribution. `PICTYP='O'` distributes margins equally, suitable for a circular chart such as a pie chart.

`PIPOSL(IPOS)` with `IPOS=3`, specifies that pie chart labels are to be positioned next to the segments (with no arrows).

`PIINCL(LTORF,VTORF,PTORF)` specifies that pie chart labels are to contain the segment label (`LTORF=.TRUE.`), the percentage (`PTORF=.TRUE.`) but not the data value (`VTORF=.FALSE.`).

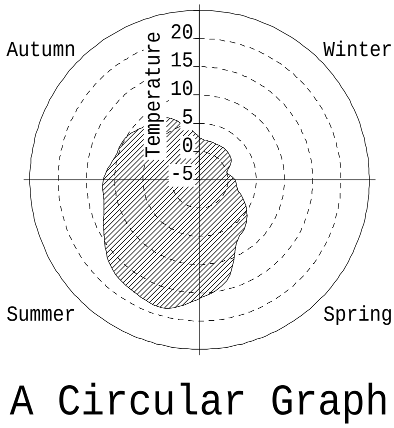
`SHPATT(ISHADE,IPOS)` specifies an alternative pattern, `ISHADE`, to be associated with the `IPOS`th element in the sequence of shading patterns.

`SQSHAD(IARR,NARR)` specifies an alternative sequence of shading pattern numbers for composite shading tasks.

6.12 Circular chart

This example demonstrates how:

- to adapt polar plots into more general circular charts,
- to change the convention for plotting polar coordinates,
- to form a closed curve for shading.




```
      PROGRAM PRIM34
      PARAMETER(NARR=25)
      REAL TEMP(NARR)
      CHARACTER SEASON(4)*6
      DATA TEMP/2.5,2.0,2.0,2.0,1.5,0.0,1.25,2.0,4.5,6.5,8.0,13.5,16.0,
     +     18.5,18.0,17.0,14.5,12.5,12.0,10.5,10.0,8.5,7.75,5.0,0.0/
      DATA SEASON/'Winter','Spring','Summer','Autumn'/
C create a closed curve
      TEMP(NARR)=TEMP(1)
C specify scales and polar coordinate convention
      CALL EQSCAL(-5.0,25.0,0.0,360.0,1)
      CALL POLZER(-5.0,0,90.0,-1)
C start new picture
      CALL NEWPIC
C specify axis characteristics and draw each axis
      CALL CHMASK(.TRUE.)
      CALL AXGRID('RP',1,-1)
      CALL AXLBJS('RP','F')
      CALL AXIS7 ('RP','Temperature')
      CALL AXCLR ('AP',1)
      CALL AXGRID('AP',1,0)
      CALL AXIS7 ('AP',' ')
C add angular axis labels
      CALL AXLBJS('AP','B')
      CALL AXTXT7('AP',0.0,90.0,SEASON,4)
C specify curve type and software shading
      CALL CVTYPE(2)
      CALL SHSET(1)
C draw curve and shade area enclosed by it
      CALL SHDEAR(0.0,15.0,TEMP,NARR,4)
C add title and terminate plotting
      CALL TITLE7('B','C','A Circular Graph')
      CALL ENDPLT
      END
```

Example 34. Circular chart

Explanation of subroutines

`AXLBJS(CHAXIS,JCHAR)` with `CHAXIS='AP'` and `JCHAR='B'`, specifies that the position of annotation labels on the angular axis is to be `'B'`etween the tick marks.

`CVTYPE(ITYPE)` with `ITYPE`=2, specifies that a loose-fitting curve is to be plotted by `SHDEAR`.

`POLZER(RZERO,IRDIR,THZERO,ITHDIR)` specifies an alternative convention for polar coordinates, (r, θ).

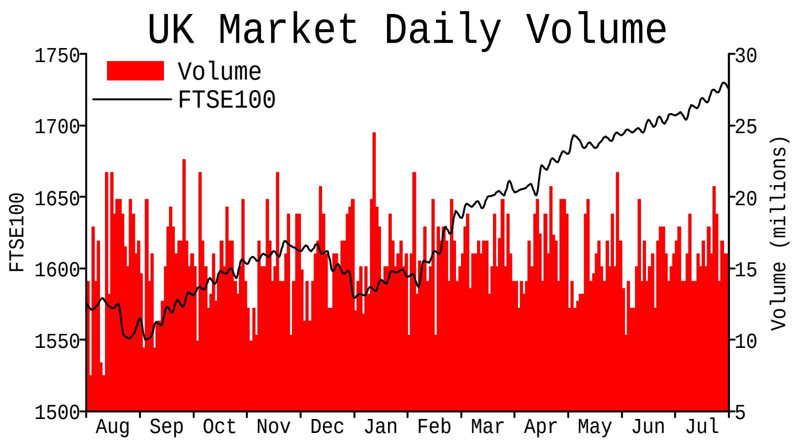
`IRDIR`	*Direction of radial increase*	`ITHDIR`	*Direction of angular increase*
-ve	Decreasing outwards	-ve	Clockwise
0	Increasing outwards (default)	0	Either way depending on directions of change of x & y (default)
+ve	Increasing outwards	+ve	Anticlockwise

By default, SIMPLEPLOT assumes that radial units increase from zero at the Cartesian origin, (0.0,0.0), and that angles increase in an anti-clockwise direction from zero at a fixed horizontal line at 3 o'clock. In this example, radial units increase outwards from -5.0 and angular units increase in a clockwise direction from 12 o'clock. The default can be restored by `CALL POLZER(0.0,0,0.0,0)`

6.13 Thickened lines

This example demonstrates how:

- to thicken lines for plotting,
- to combine a bar chart and x-y plot on the same picture,
- to reduce the width of samples in a key box.



Explanation of subroutines

`BOXKY(TORF)` with `TORF=.FALSE.` specifies that subsequent key areas are to be drawn without a box; by default, a box is drawn.

`DEFKYW(NCHARS)` specifies the width of samples in a key box; by default, the space allocated for a key sample is 12 characters wide. This width is needed to illustrate standard SIMPLEPLOT software line patterns (see Figure G.1, page 156), but can appear too wide for samples of shading patterns and marker symbols. In this example, `NCHARS` is six characters.

`THCKMG(CHTYPE,FACTOR)` with `CHTYPE='L'`, specifies that `'L'`ines are to be drawn with a pen with its thickness magnified by `FACTOR`. In this example `THCKMG` is called to thicken the axis lines and tick marks, the curve and the corresponding key entry.

The data file, `prim35.dat`, is included in the SIMPLEPLOT software distribution kit.

```
      PROGRAM PRIM35
      PARAMETER(NMONTH=12,NX=20,NVOL=NMONTH*NX,N100=NMONTH*NX/2+1)
      CHARACTER MONTHS(NMONTH)*3
      REAL VOLUME(NMONTH,NX),FTSE(N100)
      DATA MONTHS/'Aug','Sep','Oct','Nov','Dec','Jan',
     +            'Feb','Mar','Apr','May','Jun','Jul'/
      OPEN(UNIT=10,FILE='prim35.dat',STATUS='OLD')
      READ(10,*)VOLUME,FTSE
      CLOSE(10)
C specify shading and general plotting characteristics
      DO 10 I=1,NX
        CALL SHPATT(2,I)
 10   CONTINUE
      CALL AXLBGP('**',0)
      CALL AXLBJS('**','C')
      CALL THCKMG('L',3.0)
C start new picture and define key
      CALL NEWPIC
      CALL DEFKYW(6)
      CALL BOXKY(.FALSE.)
      CALL DEFKEY(1,'T','L',2,8)
C draw bar chart
      CALL AXLOCN('NB','F')
      CALL AXSBDV('NB',0.0,5.0)
      CALL BARFMT('C',100)
      CALL BARRNG(5.0,30.0)
      CALL BARCHT(VOLUME,MONTHS,NMONTH,NX,'Volume (millions)',' ')
      CALL SHDEK7(2,'Volume')
C draw x-y plot with a thick line
      CALL SCALES(0.0,REAL(N100-1),1,1500.0,1750.0,1)
      CALL AXIS7('YC','FTSE100')
      CALL BRKNAR(0.0,1.0,FTSE,N100,0)
      CALL LINEK7(0,'FTSE100')
C draw title and terminate plotting
      CALL TITLE7('T','C','UK Market Daily Volume')
      CALL ENDPLT
      END
```

Example 35. Bar chart and thickened line plot

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 ADDJST (JCHAR)

Name

`ADDJST` – to specify how entries in caption areas/keys are to be justified.

Availability Section 4, released before version 2-5.

Argument

`JCHAR`	`CHARACTER*1`	Justification of key entries/captions

`JCHAR`	*Horizontal justification*
`'C'`	Centred
`'L'`	Left justified
`'R'`	Right justified
`'D'`	Default

SUBROUTINE ARROW (XFROM, YFROM, XTO, YTO)

Name

`ARROW` – to draw an arrow between two specified points.

Availability Section 4, released version 2-6.

Arguments

`XFROM`, `YFROM`	`REAL` expressions	Coordinates of the end of the arrow in units of plotting scales
`XTO`, `YTO`	`REAL` expressions	Coordinates of the head of the arrow in units of plotting scales

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 AXCRSS (CHAXIS, CROSS)

Name

`AXCRSS` – to specify the intersection of one axis with its natural pair.

Availability Section 1, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
CROSS	REAL expression	Position of axis in units of the 'other' 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 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 AXLAB7 (CHAXIS, W, CAP)
SUBROUTINE AXLAB (CHAXIS, W, CAP, NCAP)

Name

AXLAB7 – to draw a single axis annotation label and tick mark.

Availability Section 1, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
W	REAL expression	Position along axis in units of the plotting scales
CAP	STRING expression	Annotation label
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
	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$

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 AXLBLV (CHAXIS, ILEV)

Name

AXLBLV – to specify the level of additional annotation on some axes.

Availability Section 4, released version 2-11.

Arguments

CHAXIS	CHARACTER*2	Axis type
ILEV	INTEGER expression	Level 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
	$\surd^{NL}$	$\surd^{NL}$	$\times$	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

NL: non-linear scales only

ILEV	*Non-linear Cartesian axes*	*Angular polar axis*
0	determined by scales	30°,60°,120°,150°,210°,240°,300°,330°
1	powers of 10 only	plus 0°, 90°, 180°, 270°
2	3	plus 10° intervals
3	2,5	—
4	1,2,3,4,5,6,7,8,9	—

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
	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

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`
	√^{L}	√^{L}	√	√	√	√	√	√	×	√	√	√	√	√	√

L: Linear scales only

SUBROUTINE AXTXT7 (CHAXIS, W1, STEP, LABARR, NARR)
SUBROUTINE AXTXT (CHAXIS, W1, STEP, LABARR, NARR, NCAP)

Name

`AXTXT7` – to draw a set of axis annotation labels and tick marks.

Availability Section 4, released version 2-11.

Arguments

`CHAXIS`	`CHARACTER*2`	Axis type
`W1`	`REAL` expression	Position for first label
`STEP`	`REAL` expression	Interval between labels
`LABARR`	`STRING` array	Set of labels
`NARR`	`INTEGER` expression	Number of labels in `LABARR`
`NCAP`	`INTEGER` expression	Maximum number of characters in any label

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 BARCHT (D2ARR, LABARR, NARR, NSETS, CAPN, CAPL)

Name

BARCHT – to draw a bar chart on the current picture.

Availability Section 4, released version 2-11.

Arguments

D2ARR	REAL 2-D array	Data values, D2ARR(NARR,NSETS)
LABARR	STRING array	Names of items in each data set
NARR	INTEGER expression	Number of elements in each data set and in LABARR
NSETS	INTEGER expression	Number of data sets
CAPN	STRING expression	Numeric axis caption
CAPL	STRING expression	Label axis caption

SUBROUTINE BARDIR (NBTORF, LBTORF)

Name

BARDIR – to specify the directions of bar chart scales.

Availability Section 4, released version 2-11.

Arguments

NBTORF	LOGICAL expression	Whether to reverse numeric scale
LBTORF	LOGICAL expression	Whether to reverse labels scale

SUBROUTINE BARFMT (JCHAR, IVAL)

Name

BARFMT – to specify how bars in a bar chart occupy the space available.

Availability Section 4, released version 2-11.

Arguments

JCHAR	CHARACTER*1	Justification of bars
IVAL	INTEGER expression	Percentage (1–100%) of space available

JCHAR	*Justification of bars*
'P'	Preceding the space – left for vertical bars, bottom for horizontal bars
'C'	Centred within the space
'F'	Following the space – right for vertical bars, top for horizontal bars

SUBROUTINE BARRNG (VMIN, VMAX)

Name

BARRNG – to specify the range of the numerical scale for bar charts.

Availability Section 4, released version 2-11.

Arguments

VMIN	REAL expression	Minimum scale value
VMAX	REAL expression	Maximum scale value

SUBROUTINE BARTYP (OCHAR, ITYPE)

Name

`BARTYP` – to specify the type of bar chart required.

Availability Section 4, released version 2-11.

Arguments

`OCHAR`	`CHARACTER*1`	Orientation of bars
`ITYPE`	`INTEGER` expression	Type of bar chart

`OCHAR`	*Orientation of bars*
`'V'`	Vertical bars
`'H'`	Horizontal bars

`ITYPE`	*Type of bar chart*
1	Clustered bars – data sets side-by-side
2	Stacked bars – data sets accumulated on same bars
3	Pyramid bars – data sets alternating

SUBROUTINE BLNKKY

Name

`BLNKKY` – to add a blank entry in a key or caption area.

Availability Section 1, released before version 2-5.

Arguments

None.

SUBROUTINE BOTHK7 (LTYPE, MKTYPE, CAP)
SUBROUTINE BOTHKY (LTYPE, MKTYPE, CAP, NCAP)

Name

`BOTHK7` – to draw an annotated sample of a broken line pattern and marker symbol in a key.

Availability Section 1, released before version 2-5.

Arguments

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

SUBROUTINE BOXCAP (TORF)

Name

`BOXCAP` – to specify whether boxes are to be drawn around caption areas.

Availability Section 4, released before version 2-5.

Argument

`TORF`	`LOGICAL` expression	Whether captions are to be boxed

SUBROUTINE BOXGRP (TORF)

Name

`BOXGRP` – to specify whether boxes are to be drawn around groups of pictures.

Availability Section 4, released version 2-7.

Argument

`TORF`	`LOGICAL` expression	Whether groups are to be boxed

SUBROUTINE BOXKY (TORF)

Name

`BOXKY` – to specify whether boxes are to be drawn around keys.

Availability Section 4, released before version 2-5.

Argument

`TORF`	`LOGICAL` expression	Whether keys are to be boxed

SUBROUTINE BOXPAG (TORF)

Name

`BOXPAG` – to specify whether boxes are to be drawn around the boundaries of pages.

Availability Section 1, released before version 2-5.

Argument

`TORF`	`LOGICAL` expression	Whether pages are to be boxed

SUBROUTINE BOXPIC (TORF)

Name

`BOXPIC` – to specify whether boxes are to be drawn around individual pictures.

Availability Section 4, released version 2-7.

Argument

`TORF`	`LOGICAL` expression	Whether pictures are to be boxed

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 BRKNAR (X1, DX, DARR, NARR, LTYPE)

Name

`BRKNAR` – to draw a curve from an array of values, using a specified broken line pattern.

Availability Section 4, released version 2-11.

Arguments

`X1`	`REAL` expression	Value of independent variable for plotting `DARR(1)`
`DX`	`REAL` expression	Independent variable interval between function values
`DARR`	`REAL` array	Function values for plotting
`NARR`	`INTEGER` expression	Number of points
`LTYPE`	`INTEGER` expression	Type of broken line pattern

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 BRKNCL (XCENT, YCENT, RADIUS, LTYPE)

Name

`BRKNCL` – to draw a circular function with specified centre, radius and broken line pattern.

Availability Section 1, released before version 2-5.

Arguments

`XCENT`, `YCENT`	`REAL` expressions	Coordinates of the centre of the circle in units of the plotting scales
`RADIUS`	`REAL` expression	Circle radius in plotting scale units
`LTYPE`	`INTEGER` expression	Type of broken line pattern

SUBROUTINE BRKNCV (XARR, YARR, NARR, LTYPE)

Name

`BRKNCV` – to draw a curve from a set of coordinates, using a specified broken line pattern.

Availability Section 1, released before version 2-5.

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

SUBROUTINE BRKNFN (FUNX, LTYPE)

Name

`BRKNFN` – to draw a curve of a user-defined function, $y = f(x)$, using the specified broken line pattern.

Availability Section 1, released before version 2-5.

Arguments

`FUNX`	function name	`REAL` function with one `REAL` argument, also declared in an `EXTERNAL` statement
`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 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 CHSET (ISET)

Name

`CHSET` – to specify an alternative source of character sets or fonts.

Availability Section 1, released version 2-8.

Argument

`ISET`	`INTEGER` expression	Character set source

`ISET`	*Font source*
≤ -1	Alternative hardware/firmware sets (if available)
0	Hardware/firmware font (if available)
1	Simple device independent software font
2–19	Hershey fonts (SIMPLEPLOT-PLUS)
20	Solid font (SIMPLEPLOT-PLUS)
21	Outline font (SIMPLEPLOT-PLUS)
22	Mathematical symbols (SIMPLEPLOT-PLUS)
23	Large mathematical symbols (SIMPLEPLOT-PLUS)
51	Adjustable ANSI characters (Section 4)
52	Adjustable UK characters (Section 4)

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 CVTYPE (ITYPE)

Name

`CVTYPE` – to specify the type of curve to be drawn on 2-D graphs.

Availability Section 1, released before version 2-5.

Argument

`ITYPE`	`INTEGER` expression	Method for curve construction

ITYPE	*Curve type*
1	A tight fitting smooth curve through every data point, which is confined between the same limits as the data. Sudden changes in the data result in sharp corners in the curve. A Cartesian curve drawn by this method is not invariant under rotation
2	A looser fitting smooth curve through every data point, which will probably extend higher or lower than the data at turning points, but should not have sharp corners; a Cartesian curve drawn by this method is invariant under rotation
3	Straight lines from point to point through the data, even on non-linear scales and polar charts
4	Linear regression *
5	Quadratic regression *

* If `ITYPE`=4 or 5, the curve drawn covers the range of the x scale defined by `FNRNGE`.

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

Name

`DEFCAP` – to define an area for captions.

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 line of a caption

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 DEVNAM (STR)

Name

`DEVNAM` – to specify a graphics device by name.

Availability Section plus, released version 2-11.

Argument

`STR`	`STRING` expression	Site or absolute name of device

SUBROUTINE DEVNO (IDEV)

Name

`DEVNO` – to specify a graphics device by number.

Availability Section 1, released before version 2-5.

Argument

`IDEV`	`INTEGER` expression	Number to identify device

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 DRAWLN (X, Y)

Name

`DRAWLN` – to draw a straight line between a point and zero.

Availability Section 1, released before version 2-5.

Arguments

`X`, `Y`	`REAL` expressions	Coordinates of a point specified in units of the plotting scales

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 FNRNGE (START, STOP)

Name

FNRNGE – to specify a restricted x range over which to plot 2-D functions $y = f(x)$.

Availability Section 1, released before version 2-5.

Arguments

START, STOP	REAL expressions	Values of the independent variable between which function to be drawn

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 HSTDIR (XTORF, FTORF)

Name

HSTDIR – to specify the directions of scales for histograms drawn by HSTGRM.

Availability Section 4, released version 2-11.

Arguments

XTORF	LOGICAL expression	Whether to reverse direction of x scale
FTORF	LOGICAL expression	Whether to reverse direction of frequency scale

SUBROUTINE HSTGRM (DARR, NARR, X1, DX, CAPX, CAPY)

Name

HSTGRM – to draw a histogram from raw data on the current picture.

Availability Section 4, released version 2-11.

Arguments

DARR	REAL array	Raw data
NARR	INTEGER expression	Number of elements in DARR
X1	REAL expression	Offset value for subdivision
DX	REAL expression	Frequency interval
CAPX	STRING expression	x-axis caption
CAPY	STRING expression	y-axis caption

SUBROUTINE HSTRNG (XMIN, XMAX, FMIN, FMAX)

Name

`HSTRNG` – to specify the range of scales for histograms drawn by `HSTGRM`.

Availability Section 4, released version 2-11.

Arguments

`XMIN`	`REAL` expression	Minimum horizontal scale value
`XMAX`	`REAL` expression	Maximum horizontal scale value
`FMIN`	`REAL` expression	Minimum vertical scale value
`FMAX`	`REAL` expression	Maximum vertical scale value

SUBROUTINE HSTSHD (ISHADE)

Name

`HSTSHD` – to specify a shading pattern for the boxes used to draw a histogram.

Availability Section 4, released version 2-11.

Argument

`ISHADE`	`INTEGER` expression	Shading pattern number

SUBROUTINE HSTTYP (ITYPE)

Name

`HSTTYP` – to specify the type of histogram to be plotted by `HSTGRM`.

Availability Section 4, released version 2-11.

Argument

`ITYPE`	`INTEGER` expression	Type of histogram

`ITYPE`	*Type of histogram*
1	Standard histogram
2	Cumulative histogram
3	Complementary (or inverse) cumulative histogram

SUBROUTINE INITSP

Name

`INITSP` – to reset all SIMPLEPLOT defaults.

Availability Section 1, released version 2-5.

Arguments

None.

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 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 LIMINC (DARR, NARR, VARMIN, VARMAX)

Name

`LIMINC` – 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 be reset to minimum value
`VARMAX`	`REAL` variable	To be reset to maximum value

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 MARGDV (PICTYP)

Name

MARGDV – to specify the distribution of picture margins.

Availability Section 1, released version 2-12.

Argument

PICTYP	CHARACTER*1	Type of picture

PICTYP	*Picture type*	*Margin distribution*
'D'	Default	Large left-hand and bottom, small right-hand and top
'C'	Cartesian (x-y) plot	Large on 'XC' and 'YC' axis sides, small on other
'B'	Bar chart	Large on 'LB' and 'NB' axis sides, small on other
'I'	Isometric picture	Large on 'XI', 'YI', 'ZI' axis sides, small on other
'W'	Waterfall chart	Large on 'LW' and 'NW' axis sides, small on other
'O'	Other, *eg.* polar plot, pie chart	Equal distribution on all sides

SUBROUTINE MARGIN (CMS)

Name

MARGIN – to specify the overall size of the margin around individual pictures.

Availability Section 1, released before version 2-5.

Argument

CMS	REAL expression	Width of margin (in cms)

SUBROUTINE MARKAR (X1, DX, DARR, NARR, MKTYPE, NSTEP)

Name

MARKAR – to draw marker symbols at a set of values held in an array.

Availability Section 4, released version 2-11.

Arguments

X1	REAL expression	Value of independent variable for plotting DARR(1)
DX	REAL expression	Independent variable interval between function values
DARR	REAL array	Function values for plotting
NARR	INTEGER expression	Number of points
MKTYPE	INTEGER expression	Type of marker symbol
NSTEP	INTEGER expression	Interval between points marked

SUBROUTINE MARKCV (XARR, YARR, NARR, MKTYPE, NSTEP)

Name

MARKCV – to draw marker symbols at a set of coordinates.

Availability Section 1, released version 2-11.

Arguments

XARR, YARR	REAL arrays	Coordinates of points, in units of the plotting scales
NARR	INTEGER expression	Number of points
MKTYPE	INTEGER expression	Type of marker symbol
NSTEP	INTEGER expression	Interval between points marked

SUBROUTINE MARKK7 (MKTYPE, CAP)
SUBROUTINE MARKKY (MKTYPE, CAP, NCAP)

Name

MARKK7 – to draw an annotated sample of a marker symbol in a key.

Availability Section 1, released before version 2-5.

Arguments

MKTYPE	INTEGER expression	Type of marker symbol
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 MKMASK (TORF)

Name

MKMASK – to specify whether marker symbols are to be masked.

Availability Section 4, released version 2-11.

Argument

TORF	LOGICAL expression	Whether symbols are to be masked

SUBROUTINE MKSIZE (CMS)

Name

MKSIZE – to specify a fixed size of marker symbols independent of text size.

Availability Section 4, released version 2-11.

Argument

CMS	REAL expression	Size of marker symbols (in cms)

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 PAGE (WCMS, HCMS)

Name

PAGE – to specify the size of the SIMPLEPLOT page.

Availability Section 1, released before version 2-5.

Arguments

WCMS	REAL expression	Width of page (in cms)
HCMS	REAL expression	Height of page (in cms)

SUBROUTINE PAGVW (ITYPE)

Name

`PAGVW` – to specify the orientation of a page when page dimensions have not been specified.

Availability Section 4, released version 2-11.

Argument

`ITYPE` `INTEGER` expression The orientation of the page

`ITYPE`	*Page orientation*
0	Default mode, determined by device and/or `GROUP`
1	Portrait (if page can be turned)
2	Landscape (if page can be turned)
3	Portrait
4	Landscape
5	Upside-down portrait
6	Upside-down landscape

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 PERIPH (CMS)

Name

`PERIPH` – to specify the overall size of the periphery around a group of pictures.

Availability Section 4, released version 2-7.

Argument

`CMS` `REAL` expression Size of periphery (in cms)

SUBROUTINE PIBOXL (TORF)

Name

PIBOXL – to specify whether a box is to be drawn around pie chart labels.

Availability Section 4, released before version 2-5.

Argument

TORF	LOGICAL expression	Whether a box is to be drawn

SUBROUTINE PICSIZ (WCMS, HCMS)

Name

PICSIZ – to specify the size of pictures.

Availability Section 1, released before version 2-5.

Arguments

WCMS	REAL expression	Width of pictures (in cms)
HCMS	REAL expression	Height of pictures (in cms)

SUBROUTINE PIDIAM (CMS)

Name

PIDIAM – to specify a reduced diameter for pie charts.

Availability Section 4, released before version 2-5.

Argument

CMS	REAL expression	Diameter (in cms)

SUBROUTINE PIECHT (DARR, LABARR, NARR)

Name

PIECHT – to draw a pie chart on the current picture.

Availability Section 4, released version 2-11.

Arguments

DARR	REAL array	Data values
LABARR	STRING array	Names for each segment
NARR	INTEGER expression	Number of segments and number of elements in DARR and LABARR

SUBROUTINE PIINCL (LTORF, VTORF, PTORF)

Name

PIINCL – to specify what elements are to be included in pie chart segment labels.

Availability Section 4, released before version 2-5.

Arguments

LTORF	LOGICAL expression	Whether segment names are to be included in labels
VTORF	LOGICAL expression	Whether segment values are to be included in labels
PTORF	LOGICAL expression	Whether segment percentages are to be included in labels

SUBROUTINE PIPOSL (IPOS)

Name

`PIPOSL` – to specify the position of segment labels on pie charts.

Availability Section 4, released before version 2-5.

Argument

`IPOS`	`INTEGER` expression	Label position

`IPOS`	*Label position*
1	Labels equally-spaced down sides of picture, with arrows if there is room (default)
2	Labels alongside segments, down sides of picture, with arrows if there is room
3	Labels next to segments (no arrows)
4	Labels inside segments, or next to segment when there is not enough space inside (no arrows)

SUBROUTINE PITILT (IWDTH, IHGHT)

Name

`PITILT` – to specify the tilt of pie charts.

Availability Section 4, released before version 2-5.

Arguments

`IWDTH`, `IHGHT`	`INTEGER` expressions	Width to height ratio

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 POLIN (PARR, NP)

Name

`POLIN` – to transfer polynomial coefficients before plotting curves of polynomial functions.

Availability Section 1, released before version 2-5.

Arguments

`PARR`	`REAL` array	Polynomial coefficients
`NP`	`INTEGER` expression	Number of polynomial coefficients (1–20)

SUBROUTINE POLOUT (PARR)

Name

`POLOUT` – to transfer polynomial coefficients after plotting regression curves.

Availability Section 1, released before version 2-5.

Argument

`PARR`	`REAL` array	To receive polynomial coefficients

REAL FUNCTION POLY (RVAL)

Name

`POLY` – to evaluate a polynomial function.

Availability Section 1, released before version 2-5.

Argument

`RVAL`	`REAL` expression	Argument of polynomial function in units of plotting scales

SUBROUTINE POLZER (RZERO, IRDIR, THZERO, ITHDIR)

Name

`POLZER` – to specify the convention used for polar coordinates.

Availability Section 1, released version 2-12.

Arguments

`RZERO`	`REAL` expression	Radial coordinate at centre of chart
`IRDIR`	`INTEGER` expression	Direction of radial scale
`THZERO`	`REAL` expression	Angle on chart of zero angular coordinate
`ITHDIR`	`INTEGER` expression	Direction of angular scale

`IRDIR` or `ITHDIR`	*Direction of radial increase*	*Direction of angular increase*
-ve	Decreasing outwards	Clockwise
0	Increasing outwards (default)	Either way depending on directions of change of x & y (default)
+ve	Increasing outwards	Anticlockwise

SUBROUTINE PUTCP7 (IROW, ICOLM, CAP)
SUBROUTINE PUTCAP (IROW, ICOLM, CAP, NCAP)

Name

`PUTCP7` – to draw a caption in a specified position in a caption area.

Availability Section 4, released before version 2-5.

Arguments

`IROW`	`INTEGER` expression	Line number
`ICOLM`	`INTEGER` expression	Character position
`CAP`	`STRING` expression	Caption
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

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 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 SGEXPL (IVAL, ISEG)

Name

`SGEXPL` – to specify the amount by which one pie chart segment is to be exploded.

Availability Section 4, released before version 2-5.

Arguments

`IVAL`	`INTEGER` expression	Percentage explosion (0–50)
`ISEG`	`INTEGER` expression	Segment number (1–*maximum*)

SUBROUTINE SGOMIT (TORF, ISEG)

Name

`SGOMIT` – to specify whether a pie chart segment is to be omitted.

Availability Section 4, released before version 2-5.

Arguments

`TORF`	`LOGICAL` expression	Whether segment is to be omitted
`ISEG`	`INTEGER` expression	Segment number (1–*maximum*)

SUBROUTINE SHDEAR (X1, DX, DARR, NARR, ISHADE)

Name

`SHDEAR` – to draw a curve from an array of values, and shade under it.

Availability Section 4, released version 2-11.

Arguments

`X1`	`REAL` expression	Value of independent variable for plotting `DARR(1)`
`DX`	`REAL` expression	Independent variable interval between function values
`DARR`	`REAL` array	Function values for plotting
`NARR`	`INTEGER` expression	Number of points
`ISHADE`	`INTEGER` expression	Shading pattern number

SUBROUTINE SHDEBX (X1, Y1, X2, Y2, ISHADE)

Name

`SHDEBX` – to draw a shaded box.

Availability Section 4, 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
`ISHADE`	`INTEGER` expression	Shading pattern number

SUBROUTINE SHDECL (XCENT, YCENT, RADIUS, ISHADE)

Name

`SHDECL` – to draw a circular function with specified centre and radius, and to shade inside it.

Availability Section 4, released before version 2-5.

Arguments

`XCENT`, `YCENT`	`REAL` expressions	Coordinates of the centre of the circle in units of the plotting scale
`RADIUS`	`REAL` expression	Circle radius in plotting scale units
`ISHADE`	`INTEGER` expression	Shading pattern number

SUBROUTINE SHDECV (XARR, YARR, NARR, ISHADE)

Name

`SHDECV` – to draw a curve from a set of coordinates, and shade under it.

Availability Section 4, released before version 2-5.

Arguments

`XARR`, `YARR`	`REAL` arrays	Coordinates of points in units of the plotting scales
`NARR`	`INTEGER` expression	Number of points
`ISHADE`	`INTEGER` expression	Shading pattern number

SUBROUTINE SHDEFN (FUNX, ISHADE)

Name

`SHDEFN` – to draw a curve of a user-defined function, $y = f(x)$, and shade under it.

Availability Section 4, released before version 2-5.

Arguments

`FUNX`	function name	`REAL` function with one `REAL` argument, also declared in an `EXTERNAL` statement
`ISHADE`	`INTEGER` expression	Shading pattern number

SUBROUTINE SHDEK7 (ISHADE, CAP)
SUBROUTINE SHDEKY (ISHADE, CAP, NCAP)

Name

`SHDEK7` – to draw an annotated sample of a shading pattern in a key.

Availability Section 4, released before version 2-5.

Arguments

`ISHADE`	`INTEGER` expression	Shading pattern number
`CAP`	`STRING` expression	Caption
`NCAP`	`INTEGER` expression	Number of characters in `CAP`

SUBROUTINE SHDESC (IDEG1, IDEG2, IDEG3, IDEG4, NCOLS, GAP)

Name

`SHDESC` – to specify all characteristics of shading patterns.

Availability Section 4, released before version 2-5.

Arguments

`IDEG1`, `IDEG2`, `IDEG3`, `IDEG4`	`INTEGER` expressions	Angles in degrees
`NCOLS`	`INTEGER` expression	Number of colours
`GAP`	`REAL` expression	Minimum separation between shading lines (in cms)

SUBROUTINE SHKEYS (VCHAR, HCHAR, LABARR, NARR, CAP)

Name

`SHKEYS` – to draw a complete key to a sequence of shading patterns.

Availability Section 4, released version 2-11.

Arguments

`VCHAR`	`CHARACTER*1`	Vertical position of area
`HCHAR`	`CHARACTER*1`	Horizontal position of area
`LABARR`	`STRING` array	Set of captions
`NARR`	`INTEGER` expression	Number of labels in `LABARR`
`CAP`	`STRING` expression	Header caption for key

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 SHPEN (IPEN)

Name

`SHPEN` – to specify single colour shading patterns with the specified pen.

Availability Section 4, released version 2-7.

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/colours are selected as determined by the device driver

SUBROUTINE SHSET (ISET)

Name

`SHSET` – to specify an alternative source of area-fill patterns for polygon shading.

Availability Section 4, released version 2-8.

Argument

`ISET`	`INTEGER` expression	Code number for area fill source

`ISET`	*Polygon area-fill source*
≤ -1	Alternative hardware/firmware sets (if available)
0	Hardware/firmware area fill (if available)
1	Device independent software shading

SUBROUTINE SHTYPE (ITYPE)

Name

`SHTYPE` – to specify the shading boundary used for unclosed curves.

Availability Section 4, released before version 2-5.

Argument

`ITYPE`	`INTEGER` expression	Mode of shading

`ITYPE`	*Shading area for unclosed curves*
0	Area contained by joining last point of curve to first with a straight line
1	Area between curve and horizontal axis or base of graph
2	Area between curve and vertical axis or left hand edge of graph

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 TEXTLF (FACTOR)

Name

TEXTLF – to specify the vertical separation of lines of text.

Availability Section 1, released version 2-11.

Argument

FACTOR	REAL expression	Factor to multiply text height

SUBROUTINE TEXTMG (FACTOR)

Name

TEXTMG – to specify the magnification of text.

Availability Section 1, released before version 2-5.

Argument

FACTOR	REAL expression	Magnification factor for text size

SUBROUTINE TEXTMN (CMS)

Name

TEXTMN – to specify the minimum text size.

Availability Section 1, released version 2-6.

Argument

CMS	REAL expression	Minimum character width (in cms)

SUBROUTINE TEXTSZ (CMS)

Name

TEXTSZ – to specify the fixed character width of all text.

Availability Section 1, released version 2-5.

Argument

CMS	REAL expression	Character width (in cms)

SUBROUTINE THCKMG (CHTYPE, FACTOR)

Name

THCKMG – to specify the pen thickness to be used for all plotting.

Availability Section 1, released version 2-13.

Arguments

CHTYPE	CHARACTER*1	Type of plotting to be affected by FACTOR
FACTOR	REAL expression	Magnification factor for thickening

CHTYPE	*Type of plotting*
'L'	Line drawing
'T'	Selected software text
'M'	Software marker symbols
'*'	All of the above

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 XSCALE (XSTART, XSTOP, IXTYPE)

Name

`XSCALE` – to specify the Cartesian horizontal scale for all 2-D plotting.

Availability Section 1, released before version 2-5.

Arguments

`XSTART`	`REAL` expression	Horizontal scale value at left edge
`XSTOP`	`REAL` expression	Horizontal scale value at right edge
`IXTYPE`	`INTEGER` expression	Type of horizontal scale

`IXTYPE`	*Type of scale*
0	Centimetre
1	Linear
2	Logarithmic
3	Normal probability (%)

SUBROUTINE YSCALE (YSTART, YSTOP, IYTYPE)

Name

`YSCALE` – to specify the Cartesian vertical scale for all 2-D plotting.

Availability Section 1, released before version 2-5.

Arguments

`YSTART`	`REAL` expression	y scale at bottom edge
`YSTOP`	`REAL` expression	y scale value at top edge
`IYTYPE`	`INTEGER` expression	Type of vertical scale

`IYTYPE`	*Type of scale*
0	Centimetre
1	Linear
2	Logarithmic
3	Normal probability (%)

D. Device Driver Information

SIMPLEPLOT has already been interfaced to a large number of graphics devices and the range of validated device drivers is continuously being extended. Graphics devices can be addressed directly or through a separate low-level graphics systems (*eg.* GKS, X Window System) or using graphics languages (*eg.* PostScript, CGM). This appendix describes how SIMPLEPLOT communicates with graphics devices:

D.1 Device drivers and device independence

D.2 Device input and output

D.3 Device selection

D.1 Device drivers

At most sites, the SIMPLEPLOT library has been installed with a collection of device driver programs for locally available graphics devices. Each device driver is a set of subroutines, used by SIMPLEPLOT but not used directly by the user program, which control the graphics display or plotting for individual devices. SIMPLEPLOT uses the Single Entry Point (S.E.P.) device driver system which can be configured to address a wide range of different devices.

D.1.1 Variation across devices

SIMPLEPLOT provides a device independent interface for the user. It is the device driver which determines which, if any, hardware characteristics which are used by SIMPLEPLOT:

- the number of colours/pens available;
- the availability of a hardware font, the size of hardware characters and whether hardware text can be rotated;
- the number of hardware shading patterns;
- the default layout of the SIMPLEPLOT page – drivers determine whether a device is classified as a *fixed page* device or a *packed page* device and whether a fixed page can be rotated (see section 5.1).

SIMPLEPLOT uses as many of these features as possible in order to make the best use of individual devices. However, it is the use of these characteristics which causes pictures to differ when output on different devices.

D.2 Device input and output

In order to communicate successfully with a graphics device and the user program, SIMPLEPLOT must 'know' where to direct its output and whence to read its input. SIMPLEPLOT has six different channels for communication; the menu configuration channel is opened and closed once for a single SIMPLEPLOT session. Graphical input and output are opened and closed for each device used in a session, as required.

Interactive input and output are used for prompting the 'user' for additional information such as device selection or for continuing between frames (*eg.* `Press <RETURN> to continue`). In interactive applications, both channels would normally point to the users terminal.

Diagnostic output from a SIMPLEPLOT program is sent to a single destination irrespective of which kind of or how many device driver options are used. If diagnostics are sent to the users terminal, they may, in the case of single plane devices, obscure some graphics. In this case, diagnostics may be better routed to a file. Please refer to your *Host Specific Information* for details of how to do this.

A menu configuration file can be read by SIMPLEPLOT to change the appearance of the device selection menu. This menu may be displayed when no call to `DEVNO` is made or when `CALL DEVNO(0)` is used.

A device driver configuration file can be read by SIMPLEPLOT to alter the default behaviour of certain device drivers. Please note that not all device drivers can be configured.

Graphical output is the means of communication between the device driver and the device itself. Graphical output depends on the communications link (direct, indirect, spooled or none) between the device and the user's program.

Graphical input is used by SIMPLEPLOT when the device driver needs information from the device itself. The availability of graphical input depends on the device itself, its connection with the users terminal (on-line or off-line) and facilities within the device driver.

Please refer to your *Host Specific Information* for details of how these I/O channels are implemented on your system.

D.2.1 Categories of devices

SIMPLEPLOT can be configured to run with a wide range of devices but as far as communication is concerned, they fall into four distinct categories. Individual device drivers which have similar input/output requirements have been grouped together into classes which relate to the way they are connected to your computer system – Class I, interactive terminals; Class R, remote or spooled devices; Class D, directly linked devices; and Class X, exceptions to the above.

Class I – *Interactive devices* are those which are accessible *via* the default interactive device (*ie.* the user's terminal). For example,

- Interactive graphics terminals.
- Other devices which can *only* operate by sharing another device's line in eavesdrop mode or hooked on a printer port *etc* (*eg.* a graphics terminal or plotter slaved to a text terminal).

Class R – *Remote devices* (or spooled devices) are typically shared devices such as a printer or plotter, or a device that is not currently available. For this class of devices there is no graphical input and the graphical output may be sent direct to a spooled device or to a file (usually one per page) which can then be directed to the device after you have left SIMPLEPLOT.

Class D – *Directly linked devices* are typically in direct communication with the user's program but are not necessarily accessible *via* the standard input and output channels (*eg. via* an independent communications link). As bi-directional communication is not essential for the driver to function, such devices can be operated without graphical input and graphical output can be directed to the device as described for Class R devices.

Class X *– eXceptions to the above.*

Typically, these devices are connected on an independent communications link and require bi-directional communication with SIMPLEPLOT. Both graphical input and output channels must be pre-assigned if SIMPLEPLOT is to be able to send graphics output at all.

D.3 Device selection

Devices can be selected by user program or by the user at run-time. SIMPLEPLOT device drivers are based on the BUSS Single Entry Point (S.E.P.) device driver system.

D.3.1 Single entry point device drivers

The single entry point device driver system allows any combination of individual device drivers, each with one or more modes of operation (options), to be available at run-time. Each individual device driver attends to the needs of one particular device or a small group of related devices and supplies one or more *options*. These options are normally used to allow for small changes in the configuration of the device, such as whether A3 or A4 paper is placed in a plotter, or at which resolution a laser printer is required to operate.

Device driver options may be selected by number or name in the traditional way (*via* `DEVNO` or `DEVNAM`) or selected at run-time by name or number. Absolute numbers allow a specific device driver option to be selected in the same way at any site.

D.3.2 Using the S.E.P. system

The Single Entry Point device driver system creates an internal table of available options from all the individual device drivers which have been installed. The table contains the following items for each option:

- The *site number* of the option.

  Site numbers are integers in the range 1–999. Initially, each option has a site number of its position in the table (that is, the first option will have a site number of 1, the second 2, *etc.*). Site numbers may be altered to suit a site's requirements if required.

  For compatibility with old multiple device driver configurations, the site number may be assigned by the local site.

- The *absolute number* of the option.

  The absolute number is an integer in the range 1000–32767. It can be used to specify a particular option, regardless of local site configuration. The absolute number is always assigned by BUSS: it cannot be locally configured.

  For example, an application may specifically require the use of a Digital VT241. At one site, the VT241 driver may have a site number of 5, at another it may be 27. However, at all sites which support the VT241, the absolute number 6315 will always select the VT241 driver.

- The *name* of the option

  Each option has both a *site name* and an *absolute name* which correspond to the site number and absolute number respectively. By default, the site name is the same as the absolute name (which is fixed by BUSS) but the site name may be changed. It is the site name which is displayed on the device menu but absolute names can be used to select a device if the specified name is preceded by a slash character (`/`). For example,

  ```
  CALL DEVNAM('/VT241')
  ```

- A brief *description* of the option.

Neither the absolute name nor the absolute number is site configurable.

Either the *site number* or the *absolute number* can be specified using `DEVNO`; alternatively, the option *name* can be specified using `DEVNAM`. If neither `DEVNO` nor `DEVNAM` have been called before plotting is started, the system (unless altered *via* a configuration file) prompts the user to select an option – a banner appears, identifying SIMPLEPLOT as the source of the request, and you are prompted for a device name, device number or press the RETURN key to have a menu of available devices displayed. For example,

Table D.1 A selection of SIMPLEPLOT device drivers

Option Name	*Driver Name*	*Class*	*Number*
META	META	--	10000
VT240	VT240	I	6310
VT241	VT241	I	6315
VT330	VT330	I	6325
VT340	VT340	I	6330
LN03PLUS	LN03plus	R	8300
LN03PLUS_ROTATE	LN03plus	R	8301
LN03R	PostScript	R	8400
LASERWRITER	PostScript	R	1140
LPS40	PostScript	R	1160
HP7475_A4	HP7475	X	5540
HP7475_A3	HP7475	X	5545
HP7475_A3_F	HP7475	X	5546
TEK4010	TEK4010	I	7000
TEK4014	TEK4014	I	7020
TEK4105	TEK4105	I	7120
TEK4107	TEK4107	I	7140
TEK4107_SEG	TEK4107	I	7141
TEK4107_SEG_HR	TEK4107	I	7142
MG200	MG	I	6160
MG600	MG	I	6180
MG620	MG	I	6200
GOULD6320	Gould6320	X	2160
GOULD6320_A3	Gould6320	X	2162
GOULD6320_A4_M	Gould6320	X	2163
GOULD6320_A3_M	Gould6320	X	2164
LA100_NARROW	LA100	R	6020
LA100_WIDE	LA100	R	6021

```
+------------------------------------+
| SIMPLEPLOT Device Driver Selection |
+------------------------------------+

Enter NAME, NUMBER or <RETURN> for menu:
```

Alternatively, interactive selection can be requested from within your program by `CALL DEVNO(0)` or `CALL DEVNAM('')`.

G. Graphic Details

This appendix illustrates the graphical details of SIMPLEPLOT.

G.1 Broken line patterns

The BRKN* subroutines can 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, $-6 \ldots 6$, 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 G.1 illustrates the thirteen SIMPLEPLOT software broken line patterns.

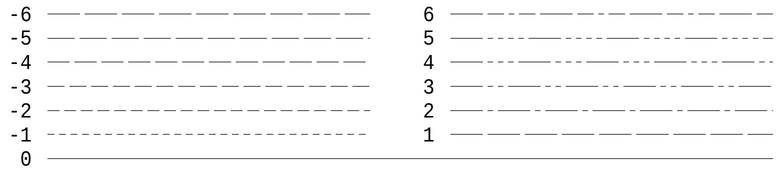


Figure G.1 SIMPLEPLOT software broken line patterns

G.2 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 G.2 illustrates hardware shading on a monochrome device with ten hardware shading patterns.

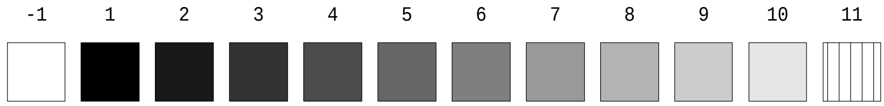


Figure G.2 Typical hardware shading patterns on a monochrome device

The patterns used for software shading consist of parallel hatching lines with adjustable angles and line separation. Figure G.3 illustrates software shading on a monochrome device. Software shading

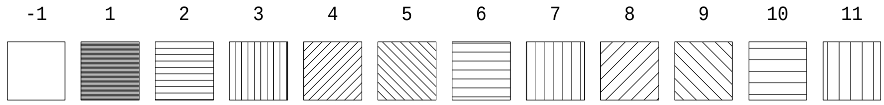


Figure G.3 Software shading patterns on a monochrome device

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).
- 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.

G.3 Fonts

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 G.4 illustrates the character sets available. `CHSELECT` is a utility program distributed with SIMPLEPLOT version 2-13 which can be used to generate font tables for any of the character sets.

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 G.4 SIMPLEPLOT character sets

G.4 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 17–96. A full range of software symbols is given in Figure G.5.

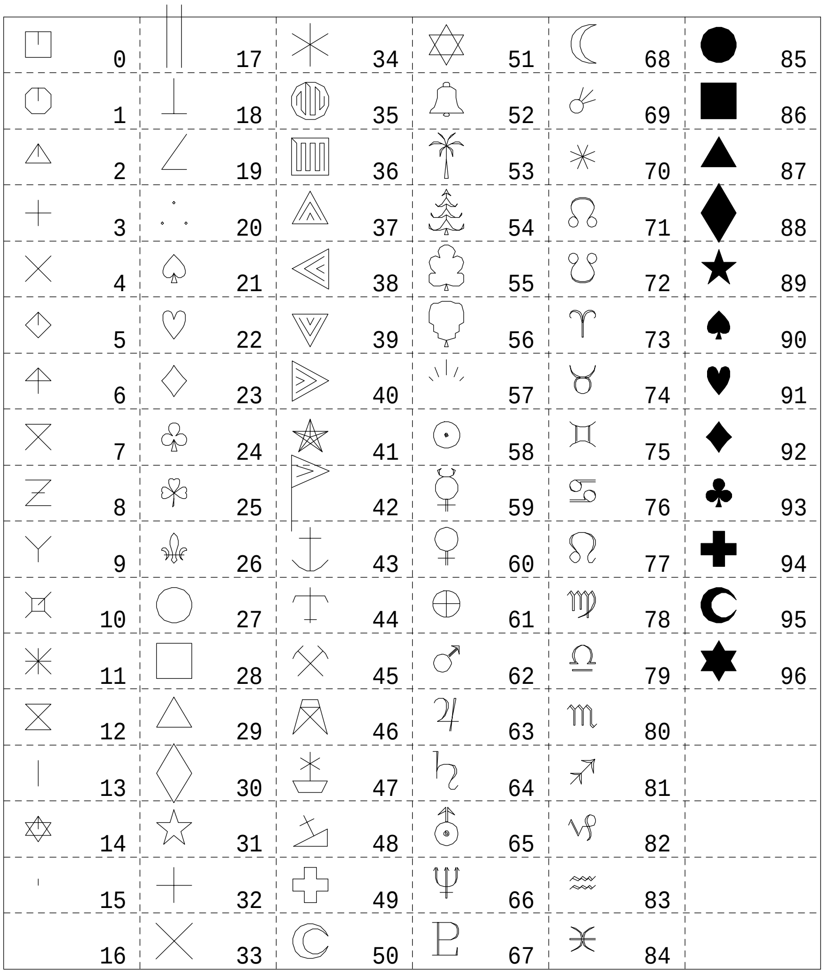


Figure G.5 SIMPLEPLOT marker symbols

M. Messages

This appendix describes the diagnostics messages which SIMPLEPLOT issues both as a result of normal operation and of error conditions.

M.1 Normal operation

In a simple program diagnostic messages are produced to monitor progress of the program. For example, the diagnostic messages produced by Example 1 (see page 8) are issued as follows:

- When the first SIMPLEPLOT subroutine is called, the first diagnostics are produced:

  `(SIMPLEPLOT Mark 2-13(000)F)`[1]

- 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`*`)`[1]

- When `ENDPLT` is called, SIMPLEPLOT closes the plotting device, triggers the output of diagnostic messages still outstanding for the last picture:

  `(END OF PICTURE)`[1]

- Finally, SIMPLEPLOT outputs the following messages:

  `(DEVICE CLOSED)`[1]
  `(SIMPLEPLOT CLOSED)`[1]

Diagnostic messages are issued through the *diagnostic channel* which is opened unless diagnostics have been explicitly suppressed by `CALL DIAGLV(0)`. The diagnostic channel is closed by `ENDPLT` along with other I/O channels used by SIMPLEPLOT. For details about how you can direct the destination of diagnostic messages, please refer to your *Host Specific Information*.

`DIAGLV` can be called to change 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 can be restored by `CALL DIAGLV(1)`.

M.2 Diagnostic messages

The different diagnostic messages fall into the following categories:

- Progress reports
- Errors in data or arguments
- Exceeding the picture or page limits
- Layout adjustment
- Plotting with no active picture or page
- Trace of subroutine calls

These are described in detail below.

M.2.1 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 2-13(`*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.

M.2.2 Errors in data or arguments

SIMPLEPLOT always attempts 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 the following error messages are issued; for example,

`(BAR CHART OMITTED: INVALID DIMENSIONS)`[1]
`(PIE CHART OMITTED: LESS THAN 1 SEGMENT)`[1]
`(PIE CHART OMITTED: -VE AND +VE VALUES)`[1]
`(POLAR OMITTED: ZERO RADIUS)`[1]
`(Range >=100; linear scale used)`[1]
`(Range through 0; linear scale used)`[1]

All these messages are of type 1 and are issued at diagnostic levels 1, 2 and 4.

M.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 incomplete tasks is given:

`(Key/caption area full)`[2]
`(Maximum no. of masked areas reached)`[2]
`(Symbol spills over boundary)`[2]
`(Title omitted below bottom)`[2]

Similarly, the following messages are output no more than once per user call:

`(Circle exceeds scales)`[2]
`(Data curve exceeds scales)`[2]
`(Function curve exceeds scales)`[2]
`(Inappropriate axis type:` *XX*`)`[2]
`(Invalid axis type:` *XX*`)`[1]
`(Null box, nothing drawn)`[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_1,y_1,x_2,y_2**)[2] – for two points.

M.2.4 Layout adjustment

When any of the layout parameters is modified due to space requirements, a diagnostic is generated:

`(Margin modified)`[1]
`(Periphery modified)`[1]

At diagnostic level 2 and 4, these are supplemented by the actual dimensions requested and used:

(v_1 `requested)`[2]
(v_2 `used)`[2]

Similarly,

`(Picture Size modified)`[1]
`(Page Size modified)`[1]

are supplemented at diagnostic level 2 or 4 by,

(x_1 y_1 `requested)`[2]
(x_2 y_2 `used)`[2]

Layout subroutines also issue diagnostic messages, at level 1, when their default behaviour is restored:

`(Grouping discontinued)`[1]
`(PAGE cancelled)`[1]
`(PICSIZ cancelled)`[1]

M.2.5 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
```

For example,

```
(12 omissions, no active picture)¹
```

Diagnostic level 2 or 4 must be specified to get more information about these incomplete tasks:

```
(Key/caption attempted with no page)²
(Title attempted with no picture)²
```

M.2.6 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 **)
(** SCALES **)
(** AXES7 **)
(** BRKNCV **)
(** ENDPLT **)
```

M.2.7 Example diagnostics

The following diagnostics were produced by example program PRIM12 at diagnostic level 2:

```
 (SIMPLEPLOT Mark 2-13(000)F)
(DEVICE OPENED: device_name)
(START OF GROUP)
 (Circle exceeds scales)
(END OF PICTURE)
 (No. of incomplete picture tasks=   1)
 (Circle exceeds scales)
(END OF PICTURE)
 (No. of incomplete picture tasks=   1)
(END OF GROUP)
(DEVICE CLOSED)
(SIMPLEPLOT CLOSED)
```

M.3 List of Messages

The following list includes all diagnostic messages which can be 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 then 999 omissions have accumulated before a page has been started.

`(>999 omissions, no active picture)`^{1} – more then 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.

`(`v_1 `requested)`^{2} – the value, v_1, of a layout parameter requested (by `MARGIN` or `PERIPH`) which has had to be modified.

`(`v_2 `used)`^{2} – the modified value, v_2, of a layout parameter requested (by `MARGIN` or `PERIPH`) which is actually used.

`(`x_1 y_1 `requested)`^{2} – the values, x_1 and y_1, of layout parameters requested (by `PAGE` or `PICSIZ`) which have had to be modified.

`(`x_2 y_2 `used)`^{2} – the modified values, x_2 and y_2, of layout parameters requested (by `PAGE` or `PICSIZ`) which are actually used.

`(BAR CHART OMITTED: INVALID DIMENSIONS)`^{1} – the 2-D data array has not been given valid dimensions.

`(Caption truncated)`^{2} – the caption added to a key or caption area (using `ADDCP7`, `BLNKKY`, `BOTHK7`, `LINEK7`, `MARKK7`, `PUTCP7` or `SHDEK7`) is too long to fit within the predefined area and has been truncated.

`(Circle exceeds scales)`^{2} – the extent of a circle (drawn using `BRKNCL` or `SHDECL`) exceeds the current picture scales.

`(Data curve exceeds scales)`^{2} – the extent of a curve (drawn using `BRKNAR`, `BRKNCV`, `SHDEAR` or `SHDECV`) exceeds the current picture scales.

`(DEVICE CLOSED)`^{1} – device closed following either a call to `ENDPLT` to terminate plotting or to `DEVNO`/`DEVNAM` to open another device.

`(DEVICE OPENED: `*`device name`*`)`^{1} – communication with the selected device (for input or output) has started.

`(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.

`(External character file not available)`^{1} – the Hershey file cannot be opened.

`(External character file unreadable)`^{1} – the Hershey file was opened but is unreadable.

`(Function curve exceeds scales)`^{2} – the extent of a function curve (drawn using `BRKNFN` or `SHDEFN`) exceeds the current picture scales.

`(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 subroutines (AXIS7, AXLAB7 or AXTXT7) has been called with an inappropriate type of axis, CHAXIS=*XX*.

(Invalid axis type: *XX*)[1] – one of the axis subroutines has been called with an unrecognized type of axis, CHAXIS=*XX*.

(Invalid external character file)[1] – the Hershey file was opened and is readable but is not in the correct format.

(Key/caption area full)[2] – there is no room for the caption/key entry (requested by ADDCP7, BLNKKY, BOTHK7, LINEK7, MARKK7 or SHDEK7).

(Key/caption attempted with no page)[2] – a key or caption area has been defined (using DEFCAP, DEFKEY or SHKEYS) 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 DEFCAP, DEFKEY or SHKEYS) 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 DEFCAP, DEFKEY or SHKEYS) has been reduced due to restrictions of the current page and/or picture size and the size of text.

(Margin modified)[1] – the margin requested (using MARGIN) has had to be modified due to layout constraints.

(Maximum no. of keys/captions reached)[2] – the number of blanked or reserved key and caption areas (defined using DEFCAP or DEFKEY with ITYPE=2, or SHKEYS) 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, BLNKKY, BOTHK7, LINEK7, MARKK7, PUTCP7 or SHDEK7).

(No room for key/caption area)[2] – the space available for a key or caption area (defined using DEFCAP, DEFKEY or SHKEYS) 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 (drawn using BRKNBX or SHDEBX) exceeds the current picture scales.

(PAGE cancelled)[1] – PAGE has been called to restore the default page size.

(PAGE ignored: invalid argument)[1] – PAGE has been called with a negative argument.

(Page Size modified)[1] – the page size requested (using PAGE) has had to be modified due to layout constraints.

(Periphery modified)[1] – the size of the periphery requested (using PERIPH) has had to be modified due to layout constraints.

(PICSIZ cancelled)[1] – PICSIZ has been called to restore default picture size.

(PICSIZ ignored: invalid argument)[1] – PICSIZ has been called with a negative argument.

(Picture Size modified)[1] – the picture size requested (using PICSIZ) has had to be modified due to layout constraints.

(PIE CHART OMITTED: LESS THAN 1 SEGMENT)[1] – a pie chart has been requested with NSEGS < 1.

`(PIE CHART OMITTED: -VE AND +VE VALUES)`[1] – `PIECHT` has been called with both negative and positive segment values.

`(POLAR OMITTED: ZERO RADIUS)`[1] – `POLAR7` has been called with zero value for maximum radial scale value.

`(Range >=100; linear scale used)`[1] – `SCALES`, `XSCALE` or `YSCALE` has been called for a normal probability scale which exceeds 100.

`(Range through 0; linear scale used)`[1] – `SCALES`, `XSCALE` or `YSCALE` has been called for a non-linear scale which includes zero.

`(Read failed on external character file)`[1] – the Hershey file has been read from but a subsequent read has failed.

`(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 2-13(`*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 `int` .

`(String too short for REAL)`[1] – `KREAL` has been called with a string variable which is not large enough to hold the converted `float` number.

`(Symbol clipped)`[2] – a marker symbol (drawn by `CP7PT`, `MARKAR`, `MARKCV` or `MARKPT`) has exceeded the current clipping window (the picture or the page).

`(Symbol spills over boundary)`[2] – a marker symbol (drawn by `CP7PT`, `MARKAR`, `MARKCV` or `MARKPT`) is centred on the edge of the current picture.

`(Text clipped)`[2] – a caption (drawn by `CP7LB` or `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).

`(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.

M.4 Common problems

This section describes some of the most commonly encountered problems.

- ***Why has my axis annotation 'flipped' to the inside of the axis?***

  In general, SIMPLEPLOT's default margins are biased towards Cartesian graphs with axes at the left and bottom. Default picture margins are unlikely to cause flipping axes to the left and bottom of the picture, but axis annotation to the right or to the top may not fit within the default margins with the default distribution.

  Moreover, any change to the text size may also affect the size of margins needed to accommodate axis annotation, and user-defined picture margins large enough to accommodate axis annotation on one device may not be adequate for another.

  If you have 'flipping axes', then reconsider any requests which affect the picture margins or the text size.

- ***Why have some of my titles been omitted?***

  Titles are only omitted when a sequence reaches the bottom of the page; every available title position can accommodate at least one title – see section 5.3.1, on page 69.

- ***Why is my text not the size I asked for?***

  Even if you have specified a fixed text size (using `TEXTSZ`), this size is only a target. Whether this target size can be achieved will depend on the current minimum text size and the capabilities of the graphics device and its device driver – see section 5.5.2, page 74.

- ***Why haven't I got the non-linear scales I asked for?***

  Care must be taken to ensure that a logarithmic (or normal probability) scale does not include zero. SIMPLEPLOT issues the following diagnostics and uses a linear scale:

  ```
  (Range through 0; linear scale used)
  ```

  Similarly a normal probability scale must not exceed 100.0.

- ***Why does my output look different on different devices?***

  Hardware characteristics vary and it is the device driver which determines which, if any, hardware characteristics which are used by SIMPLEPLOT; for example:

  - the number of colours/pens available;
  - the availability of a hardware font, the size of hardware characters and whether hardware text can be rotated;
  - the number of hardware shading patterns;
  - the default layout of the SIMPLEPLOT page and whether a fixed page can be rotated.

  SIMPLEPLOT uses as many of these features as possible in order to make the best use of individual devices. However, it is the use of these characteristics which causes pictures to differ when output on different devices.

- ***Why do I lose some of my plotting?***

  `ENDPLT` must be called at the end to finish off properly – it empties any plotting buffers which are in use, closes the plotting device, triggers the output of diagnostic messages still outstanding for the last picture, and then outputs the final diagnostic messages.

- ***Why is my layout not as I requested?***

  If you experience problems with layout, it can be useful to include calls to `BOXGRP`, `BOXPAG` and `BOXPIC` to help clarify how SIMPLEPLOT has responded to your requests.

  Remember, care is needed to ensure that specifications which affect the size and position of a *new picture* are fixed before the picture is started. In particular, specifications which change the size of text can indirectly affect the layout.

- ***Why are some of my pie chart labels obscured/missing?***

  By default, pie chart labels are positioned such that they are equally spaced down the sides of the picture; this strategy provides the best chance that a large number of labels can be clearly drawn. If `PIPOSL` has been called to specify an alternative positioning strategy for the labels, there is a risk that some labels will overlap; since all labels are masked, any label can be obscured by one drawn earlier. Also, if `PIINCL` has been called to specify additional contents of the labels, the number and size of labels will cause overcrowding.

S. Summary of Subroutines

This appendix summarizes the SIMPLEPLOT subroutines described in this manual. The subroutines represent only a subset of the complete SIMPLEPLOT library which is made up of six separate sections. The numbers marked against the subroutine name indicate in which section of the library the subroutine is included:

1. The basic package for conventional graph plotting.
2. Additional subroutines for 3-dimensional plotting.
4. Additional facilities for presentation graphics.

\+ Extra facilities which are available with a SIMPLEPLOT library that has at least Sections 1, 2 and 4.

Starting a new picture

`AXES7`^{1}	start a new picture and draw axes
`NEWPIC`^{1}	start a new picture
`POLAR7`^{1}	specify scales, start a picture, draw polar axes

Data manipulation

`LIMEXC`^{1}	find the exclusive range of values in an array
`LIMINC`^{1}	find the inclusive range of values in an array
`NODATA`^{1}	specify `REAL` value to represent missing data
`POLIN`^{1}	transfer polynomial coefficients to SIMPLEPLOT
`POLOUT`^{1}	transfer polynomial coefficients from SIMPLEPLOT
`POLY`^{1}	evaluate a polynomial function

Drawing characteristics

`CVTYPE`^{1}	specify curve drawing method for 2-D curves
`FNRNGE`^{1}	specify range for evaluating 2-D functions, $y = f(x)$
`SHTYPE`^{4}	specify shading boundary for unclosed shaded curves
`THCKMG`^{1}	specify magnification factor for line thickness

2-D coordinate system and scales

`COORDS`^{1}	change interpretation of coordinates
`EQSCAL`^{1}	specify similar linear scales for Cartesian/polar pictures
`KSCALE`^{1}	convert scale limits such that they span whole subdivisions
`POLZER`^{1}	specify convention for polar coordinates
`SCALES`^{1}	specify both horizontal and vertical scales
`XSCALE`^{1}	specify horizontal scale
`YSCALE`^{1}	specify vertical scale

Plotting data sets

`BRKNAR`[4]	draw a curve from function values in array with a broken line
`BRKNCV`[1]	draw a curve from arrays with a broken line
`BRKNFN`[1]	draw a user defined function with a broken line
`MARKAR`[4]	draw a set of symbols from function values in an array
`MARKCV`[1]	draw a set of symbols from arrays of coordinates
`SHDEAR`[4]	shade a curve from function values in an array
`SHDECV`[4]	shade curve from arrays of coordinates
`SHDEFN`[4]	shade user defined function

Point-by-point plotting

`ARROW`[4]	draw an arrow
`BREAK`[1]	force break between joined points
`BRKNBX`[1]	draw a box with a broken line
`BRKNCL`[1]	draw a circle with a broken line
`BRKNPT`[1]	draw a straight line to a specified point with a broken line
`DRAWLN`[1]	draw a straight line from a specified point to zero
`MARKPT`[1]	draw a marker symbol at a specified point
`RANGE`[1]	draw a line indicating a range of values
`SHDEBX`[4]	shade a box
`SHDECL`[4]	shade a circle

Annotation labelling at user coordinates

`CP7LB`[4]	draw a caption at a specified point
`CP7PT`[1]	draw a symbol annotated with a caption at a point
`FIGFMT`[1]	specify format for `REAL` numbers
`FIGSGN`[1]	specify format for signs of `REAL` numbers
`KNUMB`[1]	convert `INTEGER` to text string
`KREAL`[1]	convert `REAL` to text string

Bar charts, histograms and pie charts

`BARCHT`[4]	draw a bar chart from data held in a 2-D array
`BARDIR`[4]	specify the direction of bar chart scales
`BARFMT`[4]	specify how bars occupy the space available
`BARRNG`[4]	specify the range of bar chart numerical scale
`BARTYP`[4]	specify type of bar chart to be drawn by `BARCHT`
`HSTDIR`[4]	specify direction of scales for histograms drawn by `HSTGRM`
`HSTGRM`[4]	draw a histogram of raw data and axes
`HSTRNG`[4]	specify ranges of scales for histograms drawn by `HSTGRM`
`HSTSHD`[4]	specify shading pattern for histogram boxes
`HSTTYP`[4]	specify type of histogram to be drawn by `HSTGRM`
`PIBOXL`[4]	specify whether to box pie chart labels
`PIDIAM`[4]	specify reduced diameter of pie charts
`PIECHT`[4]	draw a pie chart from an array of values
`PIINCL`[4]	specify contents of pie chart labels
`PIPOSL`[4]	specify positions for pie chart labels
`PITILT`[4]	specify tilt of pie charts
`SGEXPL`[4]	specify one pie chart segment to be exploded
`SGOMIT`[4]	specify one pie chart segment to be omitted

Text and symbol attributes

`CHMASK`[4]	specify character masking
`CHSET`[1]	specify source of character fonts
`MKMASK`[4]	specify whether to mask marker symbols
`MKSIZE`[4]	specify the size of marker symbols
`TEXTLF`[1]	specify depth of line spacing for multiple lines of text
`TEXTMG`[1]	specify magnification of text
`TEXTMN`[1]	specify minimum character width
`TEXTSZ`[1]	specify fixed character width of text
`THCKMG`[1]	specify magnification factor for line thickness

Shading control

`SHDESC`[4]	specify all the details of software shading patterns
`SHPATT`[4]	specify one of a sequence of shading patterns
`SHPEN`[4]	specify pen number for monochrome shading
`SHSET`[4]	specify source of shading patterns
`SHTYPE`[4]	specify shading boundary for unclosed shaded curves
`SQSHAD`[4]	specify sequence of shading patterns

Axes

`AXCLR`[1]	specify level of automatically generated axis labels
`AXCRSS`[1]	specify the intersection of an axis
`AXES7`[1]	start a new 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
`AXLBLV`[4]	specify level of additional axis annotation
`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
`AXTXT7`[4]	draw a set of axis annotation labels
`POLAR7`[1]	specify scales, start a picture, draw polar axes

Titles, keys and captions

`ADDCP7`[1]	draw a caption in defined area
`ADDJST`[4]	specify justification of captions/key entries
`BLNKKY`[1]	leave blank line in key/caption area
`BOTHK7`[1]	draw key to broken line pattern and marker symbol
`BOXCAP`[4]	specify whether captions are to be boxed
`BOXKY`[4]	specify whether keys are to be boxed
`DEFCAP`[1]	define an area for captions
`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
`MARKK7`[1]	draw a key to a marker symbol
`PUTCP7`[4]	draw caption positioned by row and column in defined area
`SHDEK7`[4]	draw a key to a shading pattern
`SHKEYS`[4]	draw a complete key to a sequence of shading patterns
`TITLE7`[1]	draw a caption as a title to a picture, group or page

Pen/colour control

`PEN`[1]	specify single pen for all drawing
`SETPNS`[1]	specify pens for four pen pointers
`SHPEN`[4]	specify pen number for monochrome shading
`THCKMG`[1]	specify magnification factor for line thickness

Layout

`BOXGRP`[4]	specify whether boxes required around groups of pictures
`BOXPAG`[1]	specify whether boxes required around pages
`BOXPIC`[4]	specify whether boxes required around pictures
`GROUP`[1]	specify group layout of pictures
`MARGDV`[1]	specify distribution of picture margins
`MARGIN`[1]	specify overall size of picture margins
`PAGE`[1]	specify size of SIMPLEPLOT page
`PAGVW`[4]	specify orientation of page
`PERIM`[1]	draw rectangular perimeter around current picture
`PERIPH`[4]	specify overall size of peripheral margin
`PICSIZ`[1]	specify size of pictures

Conversion subroutines

`KNUMB`[1]	convert `INTEGER` to text string
`KREAL`[1]	convert `REAL` to text string
`KSCALE`[1]	convert scale limits such that they span whole subdivisions

Device and job control

`DEVNAM`[+]	specify device by absolute or site name
`DEVNO`[1]	specify device by absolute or site number
`DIAGLV`[+]	specify level of diagnostics
`ENDPLT`[1]	terminate plotting, close graphics device and SIMPLEPLOT
`INITSP`[1]	reset all defaults

Glossary

Angular axis: the circular axis on a polar plot which represents the angular (θ) scale.

Annotation: see *Axis annotation*.

Area fill: see *Shading*.

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.

Bar chart: a chart comparing magnitudes of data using rectangular blocks.

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.

Chart: a special chart, *eg.* bar chart, pie chart, histogram.

Composite plotting: plotting where a single subroutine performs a number of different operations.

Coordinate pair: a numerical description of the position of a point, (x, y) or (r, θ).

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.

Font: a set of graphic representations of the standard character set (*eg.* italic).

Framework: the axis structure of a graph.

Frequency chart: see *Histogram*.

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: is the physical display medium (*eg.* plotter or VDU).

Group: a set of pictures within a page surrounded by a periphery.

Histogram: a block graph using two quantitative scales; the area of the block represents the frequency.

Host Specific Information: the information which tells you how to execute a program which contains calls to SIMPLEPLOT on your computer system.

Justification: the position of text relative to the margins (*eg.* right justified text is positioned as close as possible to the right margin).

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.

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.

Natural scaling: by default, plotting scales are in centimetres.

New picture: the start, on a page, of a new 2-D graph, display, table, *etc.*

Origin: the fixed point, (0,0), from which coordinates are positioned.

Page: a SIMPLEPLOT page is what is actually plotted; 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 can be selected by number for drawing.

Pen pointer: one of SIMPLEPLOT's four internal indicators of pen usage; each pointer can be assigned to a *pen*.

Picture: the pictorial representation of one or more data sets; it may be an x-y plot, polar plot, histogram, bar chart, pie chart or text picture.

A picture is also the current plotting area.

Pie chart: a circular, sectored graph which can be used to display categories and proportions.

Plotting scale: see *Scale*.

Polar: a radial form of 2-D plotting.

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.

Regression: the determination of the association between dependent and independent variables.

Scale or plotting scale: the range over which your data is represented – it may be marked on an axis.

Shading: the filling of an area using a distinguishable combination of pattern and colour.

SIMPLEPLOT page: see *Page*.

Title: a text string used as the major title of a picture or group of pictures.

Units: the units which SIMPLEPLOT assumes your data to be in, either Cartesian (x, y) points or polar (r, θ) where θ may be given in radians or degrees.

User coordinates: coordinate pairs – (x, y) or (r, θ) – interpreted in terms of the current plotting scales.

Wildcard: `*` can be used to represent one or more characters describing the type of an axis.

x**-axis:** in Cartesian plotting, the horizontal axis representing the x scale.

y**-axis:** in Cartesian plotting, the vertical axis representing the y scale.

Index