simplot – a plot package for C including an interpreter

Wybo Dekker
wybo@dekkerdocumenten.nl

May 4, 2021

Contents

simplot library - overview of available routines

Description

Simplot is a plot package, especially designed for C programmers. It also contains a simplot program for use by those having no programming experience at all. A very simple program, let's call it simple.c, looks like this:

```
#include <simplot.h>
int main(void) {
   plinit(X,"",200,200,15,15,"","");
   plloop();
   exit(0);
}
```

This program simply displays an X window of 200 x 200 millimeters, which has the origin at 15 mm from the left side and 15 mm from the bottom. It makes use of the default `plevent` routine, which recognises only a few interactions: the key stroke `q` ends your program, and the left mouse button displays cross hairs and their `(x,y)` position. The window also shows one button 'Quit', which you can click with obvious results. Assuming that you installed Simplot in the default location, `/simplot`, you can compile this program with:

```
cc -I/usr/openwin/include \
   -I/simplot/include \
   -c -o simple.o simple.c
cc -L/simplot/lib \
   -L/usr/openwin/lib \
   -L/usr/X11R6/lib \
   -lsimplot -lxview -lolgx -lX11 -lm \
   -o simple simple.o \
```

You can make your own `plevent` routine to have more functionality. Here is an example:

```
#include <simplot.h>
void plevent(float x, float y, int key) {
   static float xx,yy;
   static int leftdown=FALSE;
   switch(key) {
      case 'q':
         plend();
         exit(0);
         break;
      case MouseMiddleDown:
         plmess("%g %g",x,y);
         plhairs(x,y,TRUE,TRUE);
         break;
      case MouseLeftDown:
         leftdown=TRUE;
         plmess("%g %g",x,y);
         plu(xx=x,yy=y);
         break;
      case MouseDrag:
         plmess("%g %g",x,y);
         if (leftdown) {
        plu(xx,yy);
   pld(xx=x,yy=y);
         } else {
           plhairs(x,y,TRUE,TRUE);
         }
         break;
```

```
      case MouseLeftUp:
      case MouseMiddleUp:
         leftdown=FALSE;
         plhairs(x,y,FALSE,FALSE);
         plmess("");
         break;
   }
}
int main(void) {
   plinit(X,"",200,200,15,15,"","");
   plloop();
   exit(0);
}
```

This program shows the same cross hairs as the previous example, but in this case on pressing the middle mouse button. The left button now lets you draw free-hand curves. Here is an overview of all Simplot's routines; click on the routine names for full detail:

Overview

a	plarc	xcenter,ycenter,radius,start_angle,arc ,line_style
		plot an arc using user units
	plarcm	xcenter,ycenter,radius,start_angle,arc ,line_style
		plot an arc using user units
	plarcr	dxcenter,dycenter,arc ,line_style
		plot an arc relative to the current position, in user units
	plarcrm	dxcenter,dycenter,arc ,line_style
		plot an arc relative to the current position, in millimeter units
	plarrow	dx,dy,len_pointer,ang_small,ang_large ,both_ends ,open,filled,perpendicular ,*text
		plot an annotated arrow
	plaxes	xleft,ybot,xright,ytop,xlen,ylen ,*xtext,*ytext,*text
		plot x– and y–axes, more or less automatically
	plaxfit	*x,*y ,n ,dpercent,xlen,ylen ,*xtext,*ytext,*text
		plot axes, fitted to an (x,y) array
b	plblock	x1,y1,z1,x2,y2,z2
		plot a 3D-block in user units
	plblockm	x1,y1,z1,x2,y2,z2
		plot a 3D-block in millimeter units
	plbox	x1,y1,x2,y2,radius
		plot a box with rounded corners in user units
	plboxm	x1,y1,x2,y2,radius
		plot a box with rounded corners in millimeter units
	plbutton	*label ,(*func)()
		create a button in an X-display
c	plclip	x1,y1,x2,y2
		set clipping rectangle in user units
	plclipm	x1,y1,x2,y2
		set clipping rectangle in millimeter units
	plcont	*xy,*x,*y ,nx,ny,nxa ,height ,*ident ,line_style ,*flag
		plot contours in an (x,y) array
	plcolor	red,green,blue
		set the current color
d	pld	x,y
		draw a straight line to a position in user units
	pldm	x,y
		draw a straight line to a position in millimeter units
	pldot	x,y
		plot a dot with diameter equal to current linewidth in user units
	pldotm	x,y

		plot a dot with diameter equal to current linewidth in millimeter units
	pldraw	(void)
		the user's drawing routine; (re)draws everything
e	plend	(void)
		end plotting session
	plevent	xpointer,yointer ,event_code
		handle X-events
	plexit	*message
		print a message to standard error and exit
	pleye	xeye,yeye,zeye,xfocus,yfocus,zfocus
		define location and focus of the eye in a 3D plot in user units
	pleyem	xeye,yeye,zeye,xfocus,yfocus,zfocus
		define location and focus of the eye in a 3D plot in millimeter units
f	plfill	fill_red,fill_green,fill_blue,stroke_red,stroke_green,stroke_blue
		fill and/or stroke the current path
	plfcolor	fill_red,fill_green,fill_blue
		set fill_color for filled objects
	plfont	(void)
		toggle between primary and alternate font
	plformat	left_right,up_down ,*format...
		plot text according to extended C-format string with justification
	plframe	distance ,radius
		draw a (rounded) box around current plot
	plfunc	(*func)(xf),xmin,ymin,xmax,ymax ,line_style
		plot a 2D function
g	plget	name
		return the value of a global variable
h	plhairs	x,y ,horizontal,vertical
		(re)display or clear horizontal and/or vertical hairs in an X-application
	plhiss	*x,*y ,n,type,baseline ,ybase ,line_style ,ang,dis ,icshad
		plot shaded histogram
	plhist	*x,*y ,n,type,baseline ,ybase
		plot histogram or stick plot
i	plinit	drvr ,*file ,xsize,ysize ,*main_font,*alternate_font
		initialize the plot-system
l	plloop	(void)
		start event loop
m	plmess	*format...
		display a formatted message
	plmvorg	x,y
		move the origin in user units
	plmvorgm	x,y
		move the origin in millimeter units
o	plot	x,y ,line_style
		move the pen to a new 2D position in user units
	plotm	x,y ,line_style
		move the pen to a new 2D position in millimeter units
	plot3	x,y,z ,line_style
		move the pen to a new 3D position in user units
	plotm3	x,y,z ,line_style
		move the pen to a new 3D position in millimeter units
	plotr	x,y ,line_style
		move the pen to a new relative 2D position in user units
	plotrm	x,y ,line_style
		move the pen to a new relative 2D position in millimeter units
	plotr3	x,y,z ,line_style
		move the pen to a new relative 3D position in user units
	plotrm3	x,y,z ,line_style
		move the pen to a new relative 3D position in millimeter units
p	plpage	(void)
		feed a new page / clear the screen

	plpgon	*x,*y ,n,filltype ,spacing,angle ,line_style,last
		plot a (filled) polygon
	plpie	**label ,n,lablen ,radius,x,y ,*toptext ,topfac ,*bottext ,botfac
		plot an annotated pie
	plpline	*x,*y ,n ,*marker ,dash
		plot a 2D polyline
	plpline3	*x,*y,*z ,n,isym
		plot a 3D polyline
	plpolar	size,angle ,pen
		plot e vector in polar coordinates in user units
	plpolarm	size,angle ,pen
		plot e vector in polar coordinates in millimeter units
	plpolc	*p
		retrieve the coefficients of the current polynomial
	plpolf	*x,*y ,n,kk
		calculate a polynomial
	plpolv	x
		evaluate the current polynomial in x
	plpoly	*x,*y ,n,kk ,xmin,xmax ,line_style
		calculate and plot a polynomial
r	plrect	x1,y1,x2,y2
		plot a (filled) rectangle in user units
	plrectm	x1,y1,x2,y2
		plot a (filled) rectangle in millimeter units
	plrectr	dx,dy
		plot a (filled) relative rectangle in user units
	plrectrm	dx,dy
		plot a (filled) relative rectangle in millimeter units
	pldraw	(void)
		user-defined routine redrawing the canvas in X-applications
	plreserv	xrg,yrg,yhigh
		move origin for concatenation of sub-plots
	plrotate	dx,dy
		rotate user coordinate system
s	plsave	(void)
		save current graphics context
	plscale3	xmin,ymin,zmin,xmax,ymax,zmax,xlen,ylen,zlen
		define 3D scaling factors
	plset	name ,value
		set global values
	plshade	*xvert,*yvert ,nvert ,dir,wmesh ,line_style
		hatch polygon
	plsize	*s
		return the length of a string plotted in the current font
	plsmooth	*a ,n,k
		perform k-point smoothing on (x,y) array
	plsort	*x,*y ,n
		sort (x,y) array
	plsymbol	isym
		plot a symbol
t	pltext0	*text
		plot a string, returning to position
	pltext	*s
		plot a string, leave pen at end
	pltrace	(*f)(x,y),xmin,ymin,xmax,ymax,resol,grid ,*ident ,line_style
		track an (implicit) 2D function
u	plu[m]	x,y
		move to position without plotting in user units
	plu[m]	x,y
		move to position without plotting in user units
	plunclip	(void)

		remove clipping
	plunsave	(void)
		restore graphics context
x	plxbar	v
		plot error bar in x-direction
y	plybar	v
		plot error bar in x-direction

Predefined names are:

A0	A1	A2	A3
A4	A5	A6	A7
ANGLE	ArrowDown	ArrowLeft	ArrowRight
ArrowUp	CROSS	DASH	DASHUNIT
DOWN	Delete	EVENT_ASCII	EVENT_BUTTON
EVENT_DOWN	EVENT_FUNCTION	EVENT_UP	End
Escape	F1	F10	F11
F12	F2	F3	F4
F5	F6	F7	F8
F9	GXcopy	GXxor	HEIGHT
Home	Insert	LEVENTS	MouseDrag
MouseLeftDown	MouseLeftUp	MouseMiddleDown	MouseMiddleUp
MouseRightDown	MouseRightUp	OPAQUETEXT	PDF
PDFL	PENDIA	PLOTMODE	PLOTTER
PRECISION	PS	PSL	
PageDown	PageUp	RECTFILL	RECTSTROKE
RESOLUTION	UP	X	XANGLE
XCROSS	XGRID	XIN	XLOG
XMARK	XPEN	XSIZE	XSKIP
XSQUAR	XU2M	XU2P	YANGLE
YCROSS	YGRID	YIN	YLOG
YMARK	YPEN	YSIZE	YSKIP
YSQUAR	YU2M	YU2P	

simplot - Simplot interpreter program

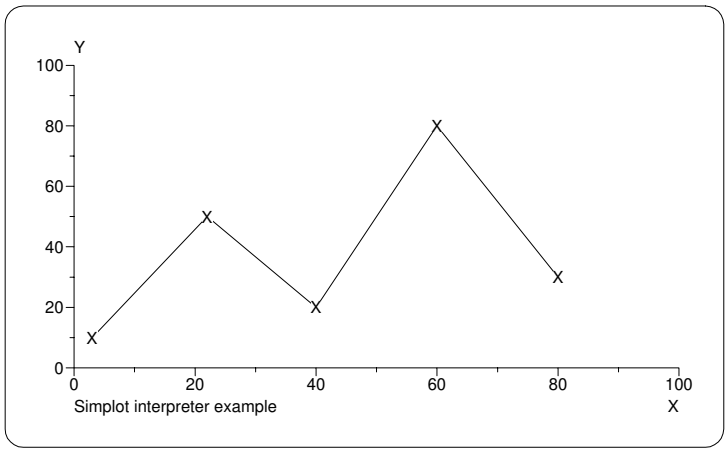
simplot [inputfile]

Description

simplot is a standalone program, which reads an input file, with .sim extension, (or standard input) and interprets the simplot-commands it contains. A very simple example of an input file is the following:

```
plinit PS simplot1 A4 50 50 "" ""
xl=100 yl=xl/2 plset HEIGHT xl/50
plaxes 0 0 100 100 xl yl X Y "Simplot interpreter example"
n=5 plrdxy n 3 10 22 50 40 20 60 80 80 30
plpline n X DOWN plframe 5 3
```

This example generates the following picture:



This example illustrates most properties of the simplot interpreter:

- simplot subroutines are called by their standard name, followed by all parameters needed, separated with white-space.
- strings may be entered without double quotes unless white-space is to be included.
- array parameters are omitted; the program knows about three arrays x, y and z, which may be filled using special subroutines plrdx, plrdy, plrdz and plrdxy. These routines have one parameter, which tells how many values or (for plrdxy) pairs of values will follow.
- Numerical variables may be created with an assignment statement of the form **var**=x; any undefined variables used will be created with a value of 0.
- predefined variables (both numerical and string) like HEIGHT, PS, A4P and are recognized by the interpreter; they may be used in expressions.
- instead of numbers, simple expressions may be used; legal operators are +, -, *, and /. Parentheses may be used to any depth.
- in this program, plformat has two limitations:

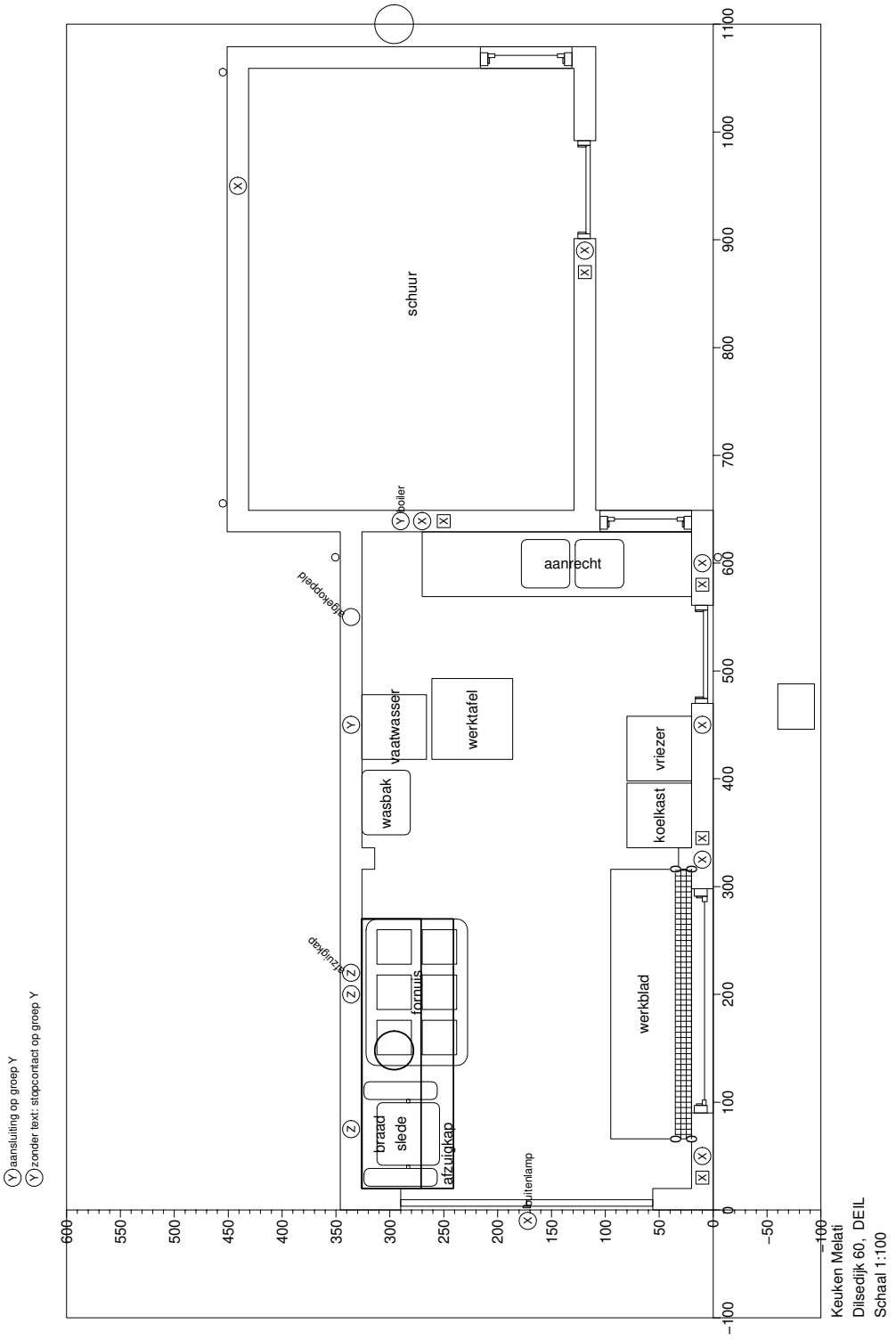
Limitations

Some facilities can not be used in the interpreter:

- the 's' and '[]' formats may not be used; strings to be plotted must be incorporated in the format-string.
- the maximum number of parameters is 11, including the format-string.
- subroutines like plfunc which need a function parameter can not be used.
- functions like plget and plpolc can be used; they report their result to standard error, but the result can not be used or assigned to a variable.

Examples

This manual contains many examples of applications of the simplot interpreter. By using a good editor, quite complex inputfiles can be created, especially if many repetitive items have to be plotted. An example of such a plot (without the corresponding 180-line inputfile) is shown below:



xyplot - produce x-y plot for binary or ASCII data

Usage

```
\this [options] [file]
```

Description

xyplot is a program which is practically useful and at the same time a good example for the use of the simplot routines. The source can be a convenient starting point for your own applications.

xyplot reads a file (or standard input by default) containing coordinates of datapoints and perhaps markers for those points, and plots it. Data must be sorted on x, or part of the data may be lost on expansion. This is because **xyplot** otherwise would not be able to efficiently handle expansion of large amounts of data. The file may either be an ascii file with 1 to 3 columns, or it may be a binary file.

Ascii files may have comment lines starting with # and comments (not necessarily starting with a #) may be added after the two numbers on a line.

Without one of the options -w, -o or -n, an ascii input file is assumed, and **xyplot** reads the columns that are plotted in an x-y frame. The number of columns in an ASCII file depends on the options -1 and -m:

(no options) two columns, with x and y values; a third column, with markers, may be present, but will not be used.
-1 one column, with y-values; the corresponding x-values are assumed to be 0, 1, 2, ... etcetera.
-m three columns: x-value, y-value, and a marker which is plotted instead of a dot.
-1 -m two columns: y-value and a marker.

A marker may have up to 7 characters. The last one of these characters may be given a special significance: If a file .xycolors exists in the current directory, and its first column contains this last character, the marker, less its last character, will be plotted in the color defined in columns 2 (red), 3 (green) and 4 (blue) of this file. If the marker becomes empty after removal of its last character, a dot will be plotted in that color. So if .xycolors contains:

```
X 1 0 0
```

then a marker *testX* will be plotted as *test* in red. A marker *X* will generate a red dot.

Options

Options -w, -o and -n cause the input file to be considered a binary file. In that case binary words are read and each word is considered a y-value. The corresponding x-value is set to the next available integer, starting at 0. Thus the first y-value will have x=0. Numbers quoting datalength or offset may contain a trailing k,K,M,or m, which multiplies the number with 1024 or 1024x1024. Especially useful for offsets is a trailing w, setting the offset count in units of a word, or a trailing D, setting the offset count in units of displayed datalength.

-s	sets stick mode (see Section Keystrokes)
-c	sets connect mode (see Section Keystrokes)
-g	sets grid on
-w *wordlength*	sets the length of the word to be read in bytes; default: 4 bytes
-o *offset*	sets the number of bytes to be skipped at the start of the file
-n *datalength*	sets the length of the data in words. Used to cut off the end of the file
-x *xtext*	sets the string plotted along the x-axis
-y *ytext*	sets the string plotted along the y-axis
-t *text*	sets the string plotted in the lower left corner; default: the filename
-1	(default for binary data): the datafile contains only y-data, x's are set to 0,1,2,3,...
-m	the dataset has an extra column containing a marker of up to 7 characters. These markers are used instead of bullets to mark the datapoints, as described before.
-X *Xsize*	sets the axes lengths (both x an y) in mm; default: 120
-Y *Ysize*	sets the y-axis length (only y) in mm; default: same as x-axis length.

-D *dotsize*	sets the dot size in mm; default: 1
-d *degree*	sets the polynomial degree; default: no polynomial
-H *symbol height*	sets the symbol height; default 3, minimum .8, maximum 8
-l *count*	makes for count digits left of the y-axis; useful if you generate several pictures and them to be all the same width.
-b	no X display presented, PDF plot generated
-B	no X display presented, PDF plot generated and sent to the plotter

Keystrokes

Several keystrokes are available that influence the display interactively. Some of these make use of a number (the current number) that has been typed before hitting the key:

F1	Help. Shows the available keystrokes. Pressing F1 again returns the xy display.
c	Toggle connection lines between points. Initially, points are shown without interconnecting lines.
d	Set polynomial degree to current number. Initially, no polynomial is shown. Typing a number (including 0), followed by d, shows the best fit polynomial of degree number through the points. A space, instead of a number, can be used to clear drawing a polynomial. The r-key may be used to show the residuals of the polynomial fit. This may be useful when looking for a good polynomial representation for a set of datapoints.
D	Set dot size. Initially points are shown as 1 mm dots. This may be changed with the D key, preceded by a number. A zero number or no number at all removes the dots.
e	Toggle plotting of even points only.
f	Fourier transform the data or (if current number is non-zero) transform them back
g	Toggle grid-plotting.
I	differentiate.
i	integrate.
l	toggle labeling.
m	Print the pointer's current x and y to stdout.
M	Print all x and y to stdout.
o	Toggle plotting of odd points only.
O	Output differences datafile. The y-differences of the datapoints are written to a file with the same name as the input file (or xyplot, if input was from standard input) with a .txt extension. This is of course only useful for equidistant data.
p	Plot to a PDF file.
P	Plot to the plotter (an appropriate PDF conversion filter is assumed to be installed).
q	Quit
r	Toggle residual plotting. The differences between the y-values of the datapoints and the y-values for the current polynomial are shown instead of the datapoints themselves. Pressing r again returns to the datapoint display.
s	Toggle stick mode. In stick mode, datapoints are shown as sticks at position x with height y. This mode is especially useful for the detection of clustering in large amounts of data. The option m may be useful to output the x-values separating the clusters to standard output.
S	smooth, using current number as smoothing range.
w	Write out 4-byte integer binary. The y-values of the datapoints are written to a file with the same name as the input file (or xyplot, if input was from standard input) with a .out extension. This is of course only useful for equidistant data.
z	Zoom out by 10%.
[0-9.-]	Used for entering numbers.
Escape	Clear current number.
Left	Move left by half a screen.
Right	Move right by half a screen.
MouseLeftdrag	Expand. A rectangle dragged with the left mouse button is expanded. Normally one presses the button on the lower left corner and releases it at the upper right corner. Starting and ending in other corners inverts the x- and/or y-axes. Pressing the left button and releasing at in the same position undoes the expansion, that is: the axes are fitted to the extremes of the data.

MouseMiddleDrag Track position and integrate. Dragging the mouse with the middle button pressed shows the current x,y position in the upper left corner, followed by the sum of the y-values of the datapoints that are between the initial position and the current position.

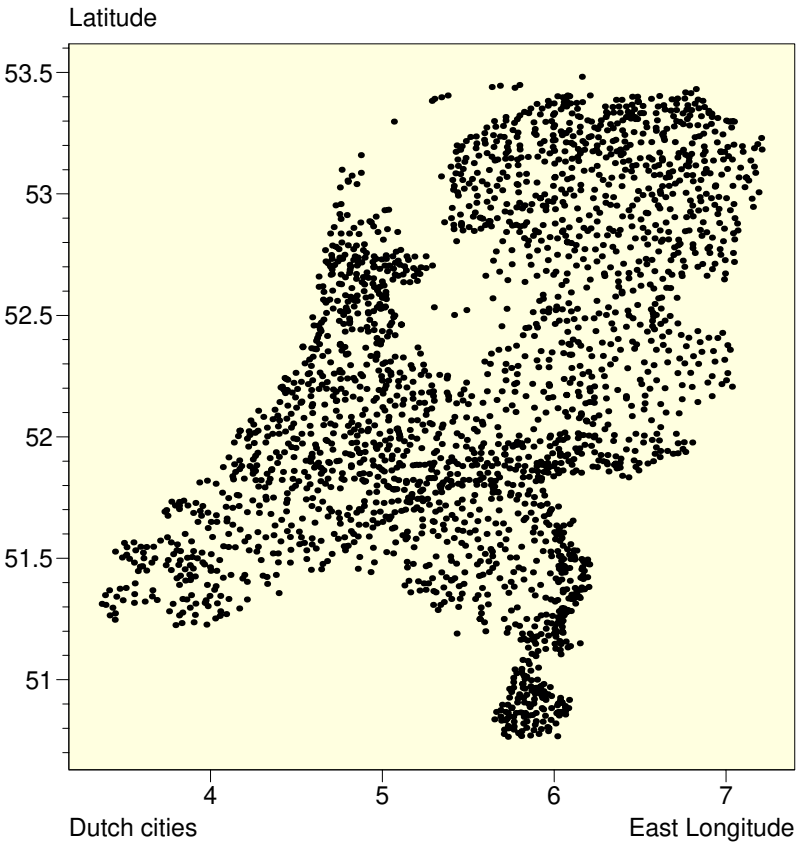
Examples

```
xyplot -o 1024 file             # wordlength 4, offset 1024 bytes
xyplot -o 1k   file             # wordlength 4, offset 1024 bytes
xyplot -o 1K   file             # wordlength 4, offset 1024 words = 4096 bytes
xyplot -w 2 -n 33 -o 10D        # wordlength 2, display only 33 words,
                                # offset 10*33*2=660 bytes
xyplot -w 2 -n 33 -o 11D        # look at the next chop of 33 words
```

The SIMPLOT distribution contains a data file xy containing the coordinates and names of all Dutch cities and villages. You can display these by running xyplot:

```
xyplot xy    # displays a dot for each entry
xyplot -m xy # displays the name for each entry
```

The first of the two shows this picture:



plarc - plot an arc using user units

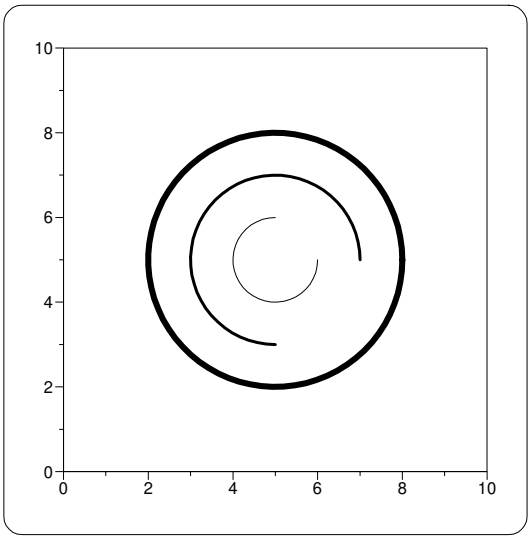
parameter	type	units	description
xcentr,ycentr	float	uu	position of the center of the arc
radius	float	uu	radius of the arc
angl	float	degr	angle of radius to the start relative to x-axis
arc	float	degr	size of the arc
linestyle	int	—	see plot
returns:	void		

Description

plarc plots an arc, center position and radius in user units. For positive **arc,** the arc is plotted anti-clockwise. If **arc** is larger than 360 degrees it is truncated to that value.

Examples

```
plinit PS plarc A4 50 50 "" ""
plaxes 0 0 10 10 70 70 "" "" ""
plrect 0 0 10 10
plset PENDIA 1  plarc 5 5 3  0 400 DOWN
plset PENDIA .5 plarc 5 5 2  0 270 DOWN
plset PENDIA .1 plarc 5 5 1 90 270 DOWN
plframe 5 3
```



See also

plarcr, plarcm, plarcrm

See also

simplot program plfill plarcr plformat

plarcm - plot an arc in mm units

parameter	type	units	description
xcentr,ycentr	float	mm	position of the center of the arc
radius	float	mm	radius of the arc
angl	float	degr	angle of radius to the start relative to x-axis
arc	float	degr	size of the arc
linestyle	int	—	see plot
returns:	void		

Description

plarcm plots an arc, center position and radius in millimeter units. For positive **arc,** the arc is plotted anti-clockwise. If **arc** is larger than 360 degrees it is truncated to that value.

See also

plarc, plarcr, plarcrm

plarcr - plot arc from current position, relative center in user units

parameter	type	units	description
dx,dy	float	uu	center of the arc, relative to the current position
arc	float	degr	length of the arc
linetype	int	—	see plot
returns:	void		

Description

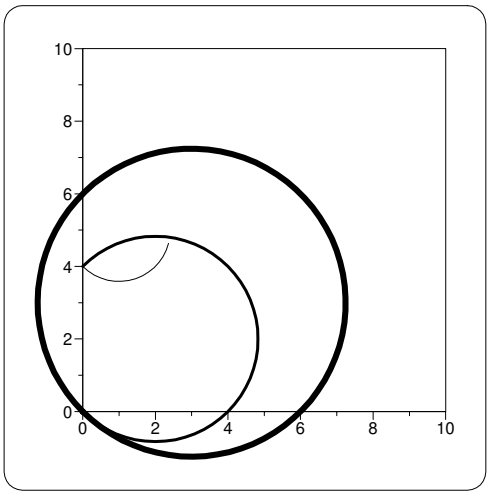
plarcr plots an arc starting at the current position. The center position is defined user units relative to the current position. For positive **arc** the arc is plotted anti-clockwise. The pen is left at the end of the arc.

See also

plarcrm, plarcm, plarc

Examples

```
plinit PS plarcr A4 50 50 "" ""
plaxes 0 0 10 10 60 60 "" "" ""
plrect 0 0 10 10
plu 0 0
plset PENDIA 1  plarcr 3 3 400 DOWN
plset PENDIA .5 plarcr 2 2 270 DOWN
plset PENDIA .1 plarcr 1 1 120 DOWN
plframe 5 3
```



plarcrm - plot arc from current position, relative center in mm

parameter	type	units	description
dx,dy	float	mm	center of the arc, relative to the current position
arc	float	degr	length of the arc
linetype	int	—	see plot
returns:	void		

Description

plarcrm plots an arc starting at the current position. The center position is defined millimeter units relative to the current position. For positive **arc** the arc is plotted anti-clockwise. The pen is left at the end of the arc.

See also

plarc, plarcm, plarcr

plarrow - plot an arrow

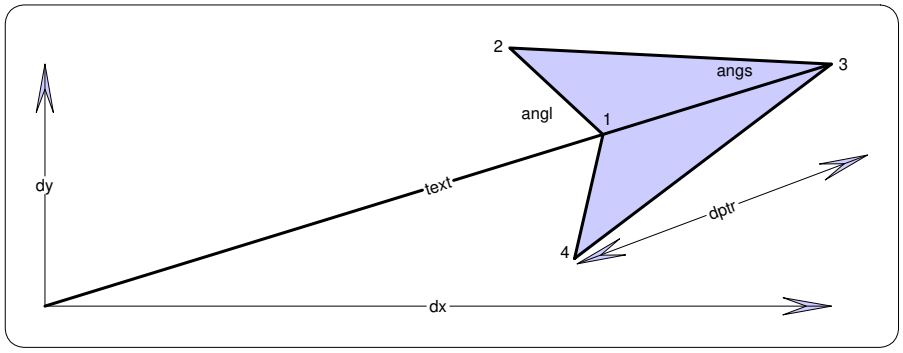
parameter	type	units	description
dx,dy	float	uu	displacement of the arrow
dptr	float	mm	length of the pointer
angs	float	degr	small angle of the pointer
angl	float	degr	large angle or the pointer
both	int	—	if TRUE, both ends will have a pointer
open	int	—	if TRUE, the pointer is open
filled	int	—	if TRUE, the pointer is filled
perp	int	—	if TRUE, text in the shaft is perpendicular to it
text	char*	—	text in the middle of the shaft
returns:	void		

Description

plarrow plots an arrow, starting at the current position, extending to dx,dy. If dptr=0, the current symbol height will be used. Filling of the pointer is performed inside the polygon 12341 (see the example below). An open pointer misses lines 1–2 and 4–1.

Examples

```
plinit PS plarrow A4 30 30 "" ""
# set some constants
dx=130 dy=40 dptr=50 angs=20 angl=60

# plot the big arrow
plfcolor 0.8 0.8 1
plset PENDIA .5 plarrow dx dy dptr 20 60 0 0 1 0 text

# plot instructive arrows
plset PENDIA .1 angs=10 angl=20 dptr=8
plarrow dx 0 dptr angs angl 0 0 1 0 dx
plarrow 0 dy dptr angs angl 0 0 1 1 dy
plot 88 7 UP plarrow 48 18 dptr angs angl 1 0 1 0 dptr

# mark some points in the big arrow
plot 84 33 UP plformat -.5 -.5 angl
plot 117 40 UP plformat -.5 -.5 angs
plot 93 31 UP plformat 0 0 1
plot 75 43 UP plformat 0 0 2
plot dx+2 dy UP plformat 0 0 3
plot 86 9 UP plformat 0 0 4

# frame around all
plframe 5 3
```



See also

plpolv

plarrowm - plot an arrow

parameter	type	units	description
dx,dy	float	mm	displacement of the arrow
dptr	float	mm	length of the pointer
angs	float	degr	small angle of the pointer
angl	float	degr	large angle or the pointer
both	int	—	if TRUE, both ends will have a pointer
open	int	—	if TRUE, the pointer is open
filled	int	—	if TRUE, the pointer is filled
perp	int	—	if TRUE, text in the shaft is perpendicular to it
text	char*	—	text in the middle of the shaft
returns:	void		

Description

plarrowm plots an arrow, starting at the current position, extending to dx,dy, in mm If dptr=0, the current symbol height will be used. Filling of the pointer is performed inside the polygon 12341 (see the example below). An open pointer misses lines 1–2 and 4–1.

plaxes - plots x- and/or y-axes and scales the user units

parameter	type	units	description
xleft,ybot	float	uu	value at the left/low end of the axes
xright,ytop	float	uu	value at the right/high end of the axes
xlen,ylen	float	mm	lengths of the axes
xtext,ytext	char*	—	strings plotted at the right/upper end of the axes
text	char*	—	string plotted at the left end of the x-axis
returns:	void		

Description

plaxes plots x- and y-axes and sets scale factors and user origin The placement of axes and numbers along the axes is governed by global variables set by **plset** (T=TRUE, F=FALSE):

global	default	value	description
X/YIN	F/F	T	number at the right/above of the x/y axis
		F	at the left/under
X/YSQUAR	T/F	T	number square to the x/y-axis
		F	number in the direction of the x/y-axis
X/YANGLE	0/90	**val**	directions of the axes relative to the initial x-axis; any angle is permitted
CROSS	F	T	axes cross at user unit set by X/YCROSS
		F	axes cross at xleft,ybot
X/YCROSS	xleft/ybot	**val**	y/x axis crosses the x/y axis at **val** user units if CROSS has been set to a non-zero value
X/YMARK	0/0	**val**	x/y axis scale marks are placed every **val** user units
		0	x/y axis scale marks are placed automatically
X/YSKIP	0	**val**	x/y axis numbers are placed at every **val** scale mark
		0	x/y axis numbers are placed automatically
X/YGRID	F	T	grid lines at plotted through the numbered scale marks of the y/x axis
		F	no grid lines are plotted

Bugs

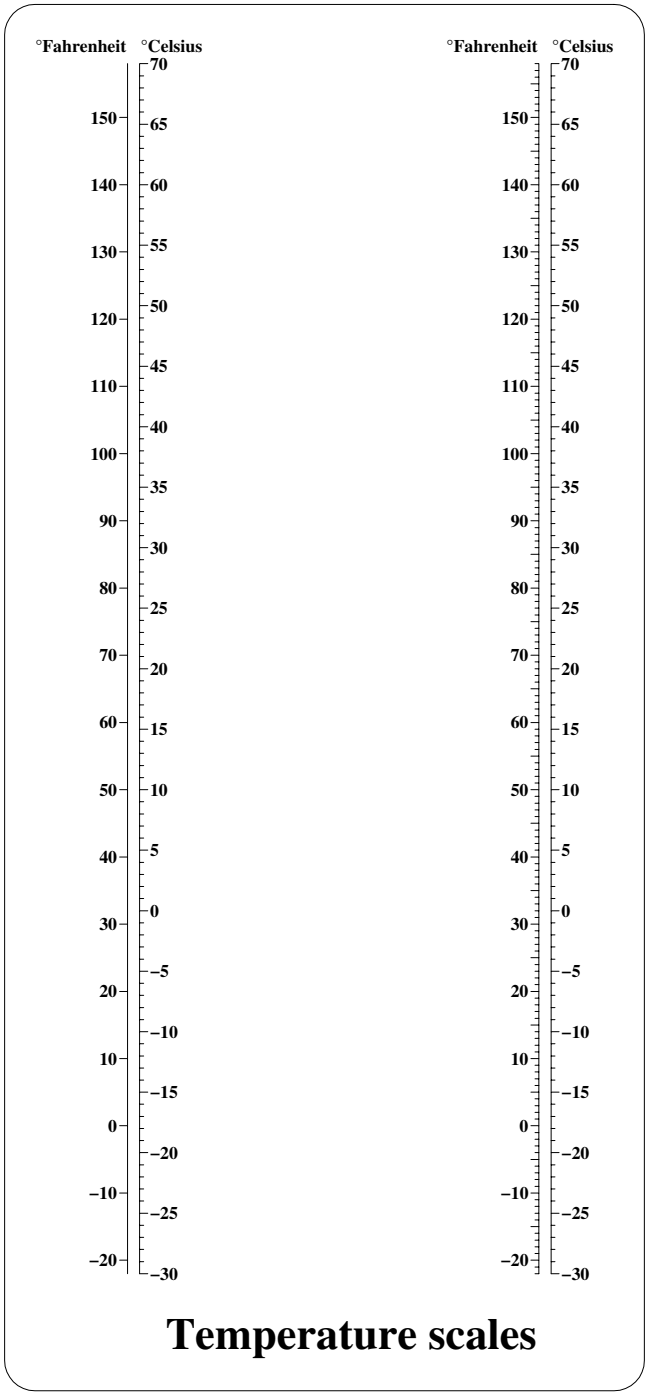
Plaxes plots in GXcopy mode, even if the user has set another mode.

Examples

several features: plotting of a single axis by setting xl=0 or xleft=xright, plotting of scale ticks and number to the left or to the right of the axis, and the usage of YMARK:

```
#include <simplot.h>
#include <stdio.h>
int main() {
   int i,Fmin=-22,Fmax=158,Cmin=-30,Cmax=70;

   plinit(PS,"",A4,50,50,"TB","");

   for (i=0;i<=66;i+=66) {
      plmvorgm(i,0);
      // note that we need tricks to put the Celsius and the Fahrenheit
      // on the correct side. Celsius is put above the Fahrenheit axis
      // and vice versa, and spaces are used for shifting

      plaxes(0,Fmin,0,Fmax,0,200,"","   #0#Celsius","");
```

```
      // plots vertical scale; x-axis not used
      plset(YIN,TRUE);
      // numbers will be plotted "inside", that is: right of y-axis
      plmvorgm(2,0);
      // moves origin 2 mm to the right
      plaxes(0,Cmin,0,Cmax,0,200," ","#0#Fahrenheit   "," ");
      // plots celsius scale 2 mm right of fahrenheit scale
      plset(YIN,FALSE);
      // resets plotting 'outside' that is: left of y-axis
      plset(YMARK,TRUE);
      // causes plotting of scale marks at each degree
   }
   plset(HEIGHT,5);
   plotm(-33,-10,UP);
   plformat(0,0,"Temperature scales");
   plframe(5,5);
   exit(0);
}
```



Temperature scales

See also

plpoly plarc plfont plpline plhist plarcr plformat plloop plmvorg simplot program plfunc plpolar plrotate pltrace plpolv

plaxfit - plot axes fitted to a set of datapoints

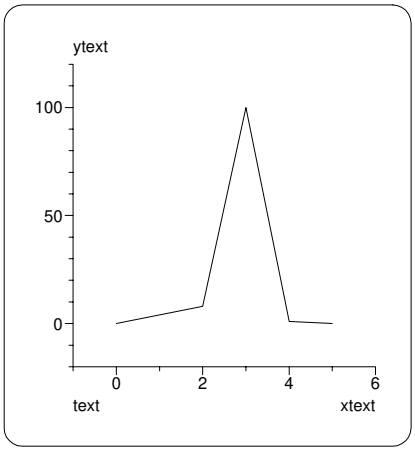
parameter	type	units	description
*x	float*	uu	pointer to an array of x-values
y	float*	uu	pointer to an array of y-values
n	int	—	size of the x- and y-arrays
dproc	float	—	empty space between the plotted points and the axes
xl	float	mm	x-axis length
yl	float	mm	y-axis length
xtext	char*	—	text along the x-axis (lower right)
ytext	char*	—	text on top of the y-axis (upper left)
text	char*	—	text under the picture (lower left)
returns:	void		

Description

plaxfit plots axes such that the data in **x** and **y,** if plotted by plpline, do not come closer than **dproc** percent of **x**/yl to the x/y-axis. All other parameters and variables are as in plaxes.

Examples

```
plinit PS plaxfit A4 50 50 "" ""
plrdxy 5 0 0 2 8 3 100 4 1 5 0
plaxfit 5 20 50 50 xtext ytext text
plpline 5 "" DOWN
plframe 5 3
```



See also

plsmooth

plblock - plots a block (3D rectangle) using user units

parameter	type	units	description
x1,y1,z1	float	uu	position of a corner of the block
x2,y2,z2	float	uu	position of the diagonally opposite corner
returns:	void		

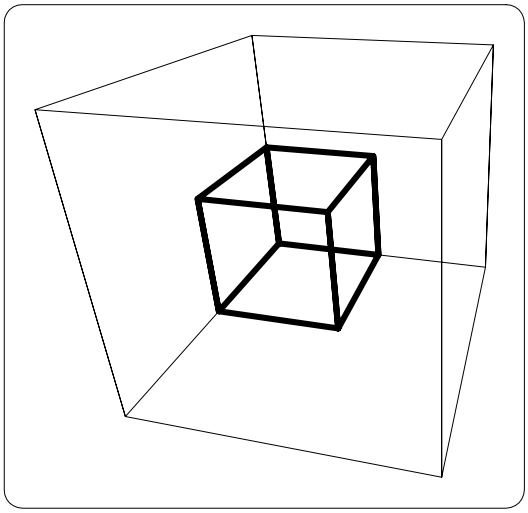
Description

plblock plots a block (3D rectangle) between (x1,y2,z1) and (x2,y2,z2).

Examples

The following program plots two blocks, the first in millimeters, the second, using thick lines, in user units:

```
plinit PS plblock A4 100 100 "" ""
plscale3 0 0 0 50 50 50 80 80 80
pleye 100 60 70 50 50 50
plblock 0 0 0 50 50 50
plframe 5 3
plset PENDIA 1
plblock 0 0 0 25 25 25
```



plblockm - plots a block (3D rectangle) in mm

parameter	type	units	description
x1,y1,z1	float	mm	position of a corner of the block
x2,y2,z2	float	mm	position of the diagonally opposite corner
returns:	void		

Description

plblockm plots a block (3D rectangle) between (x1,y2,z1) and (x2,y2,z2).

Example

See plblock

plbox - plot a rectangle with rounded corners

parameter	type	units	description
x1,y1	float	uu	position of a corner of the box
x2,y2	float	uu	position of the opposite corner
r	float	mm	radius of the rounded corner
returns:	void		

Description

plbox plots a rectangle with rounded corners, radius **r**. **x,y** in user units, radius in millimeters

Examples

```
plinit PS plbox A4 15 15 H H
plset HEIGHT 5
plu 40 15 plformat 0 0 "Box in a Rectangle"
plrect 0 0 80 30
plbox 1 1 79 29 3
```

See also

simplot program

plboxm - plot a rectangle with rounded corners in mm

parameter	type	units	description
x1,y1	float	mm	position of a corner of the box
x2,y2	float	mm	position of the opposite corner
r	float	mm	radius of the rounded corner
returns:	void		

Description

plboxm plots a rectangle with rounded corners, radius **r**. all units in millimeters.

Examples

```
plinit PS plbox A4 15 15 H H
plset HEIGHT 5
plu 40 15 plformat 0 0 "Box in a Rectangle"
plrect 0 0 80 30
plbox 1 1 79 29 3
```

See also

simplot program

plbutton - create button in X window

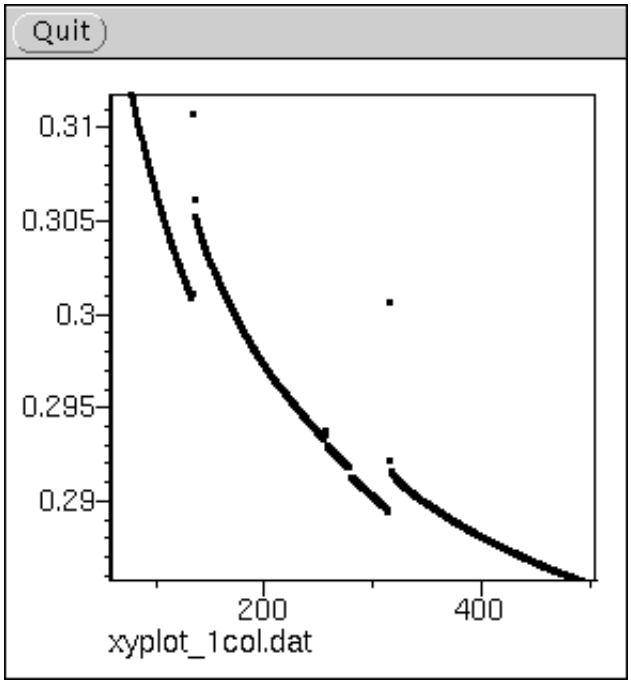
parameter	type	units	description
*label	char*	—	label displayed by the button
func(void)	void*	—	function associated with the button
returns:	void		

Description

plbutton, if the plotter driver is X, creates a button on the panel, labeled with the string in **label**. When the button is pressed, the user-defined routine **func** will be called. If PS is the plotter driver **plbutton** has no effect.

Examples

Every X-windows Simplot display comes with one initial button, labeled 'Quit'. It calls plend when pressed and is created by the statement: plbutton("Quit",plend); This is an example of a Simplot X-window showing this button:



plclip - set clipping rectangle in user units

parameter	type	units	description
x1,y1	float	uu	position of one corner of the clipping rectangle
x2,y2	float	uu	position of the opposite corner
returns:	void		

Description

plclip sets the clipping rectangle in user units.

Examples

The following example draws a rectangle, sets clipping to this rectangle and then plots two long strings, one in the center of the rectangle and one centered on the bottom line.

See also

plclipm plunclip

```
plinit PS plclip 60 60 15 15 "" ""
plset HEIGHT 3
plrect 0 0 30 30 plclip 0 0 30 30
plu 15 15 plformat 0 0 "STRING CENTERED IN THE CLIPPING RECTANGLE"
plu 15 0  plformat 0 0 "STRING CENTERED ON THE BOTTOM LINE OF THE CLIPPING RECTANGLE"
```

See also

plpoly plreserv plloop pltrace plpolv

plclipm - set clipping rectangle in mm

parameter	type	units	description
x1,y1	float	mm	position of one corner of the clipping rectangle
x2,y2	float	mm	position of the opposite corner
returns:	void		

Description

plclipm sets the clipping rectangle in millimeter units.

See also

plclip

plcolor - change color

parameter	type	units	description
red,green,blue	float	—	rgb components (0..1)
returns:	void		

Description

plcolor changes the color for subsequent plotting operations. The parameters give the red, green and blue fraction, such that black is represented by 0,0,0 and white by 1,1,1. For many colors, macros have been predefined, thus permitting the replacement of the three parameters with one predefined constant. Thus white may also be set with plcolor(White) and black with plcolor(Black). The following table shows the predefined colors; a red printed color indicates that 1, 2, 3 or 4 may be appended, where 1 is equivalent to nothing appended, and 2, 3 and 4 represent darker colors (100, 93, 80 and 55% respectively).

AliceBlue	AntiqueWhite	Aquamarine	Azure
Beige	Bisque	Black	BlanchedAlmond
Blue	BlueViolet	Brown	Burlywood
CadetBlue	Chartreuse	Chocolate	Coral
CornflowerBlue	Cornsilk	Cyan	DarkGoldenrod
DarkGreen	DarkKhaki	DarkOliveGreen	DarkOrange
DarkOrchid	DarkSalmon	DarkSeaGreen	DarkSlateBlue
DarkSlateGray	DarkTurquoise	DarkViolet	DeepPink
DeepSkyBlue	DimGray	DodgerBlue	Firebrick
FloralWhite	ForestGreen	Gainsboro	GhostWhite
Gold	Goldenrod	Gray	Green
GreenYellow	Honeydew	HotPink	IndianRed
Ivory	Khaki	Lavender	LavenderBlush
LawnGreen	LemonChiffon	LightBlue	LightCoral
LightCyan	LightGoldenrod	LightGoldenrodYellow	LightGray
LightPink	LightSalmon	LightSeaGreen	LightSkyBlue
LightSlateBlue	LightSlateGray	LightSteelBlue	LightYellow
LimeGreen	Linen	Magenta	Maroon
MediumAquamarine	MediumBlue	MediumOrchid	MediumPurple
MediumSeaGreen	MediumSlateBlue	MediumSpringGreen	MediumTurquoise
MediumVioletRed	MidnightBlue	MintCream	MistyRose
Moccasin	NavajoWhite	Navy	NavyBlue
OldLace	OliveDrab	Orange	OrangeRed
Orchid	PaleGoldenrod	PaleGreen	PaleTurquoise
PaleVioletRed	PapayaWhip	PeachPuff	Peru
Pink	Plum	PowderBlue	Purple
Red	RosyBrown	RoyalBlue	SaddleBrown
Salmon	SandyBrown	SeaGreen	Seashell
Sienna	SkyBlue	SlateBlue	SlateGray
Snow	SpringGreen	SteelBlue	Tan
Thistle	Tomato	Turquoise	VioletRed
Violet	Wheat	WhiteSmoke	White
Yellow	YellowGreen		

Examples

See also

plfill plsmooth plloop simplot program plfunc pltrace

plcont - plot contours in 2D array

parameter	type	units	description
*xy	float*	—	
x	float*	—	
y	float*	—	
nx	int	—	
ny	int	—	
nxa	int	—	
height	float	—	
*ident	char**	—	
linetype	int	—	
*flag	char**	—	
returns:	void		

Description

plcont draws contour line through matrix xy[nxa,ny] at height height coordinates of the matrix are in x[nxa],y[ny] they need not be equidistant, but should ascend or descend ident contains a string to be plotted somewhere in the contour line linetype: see plot flag[nxa*ny] is a scratch character array written by l.c. willemsens, iwis-tno, the hague, december 1983

Examples

pld - draw a straight line to a position

parameter	type	units	description
x,y	float	uu	position to which the line is drawn
returns:	void		

Description

pld moves the pen to (**x,y**) in user units, drawing a straight line. It is equivalent to plot(**x,y,DOWN**).

See also

plot, plu

Examples

See also

plreserv plfill pldot plloop plmvorg simplot program plfunc pltrace

pldm - draw a straight line to a position

parameter	type	units	description
x,y	float	mm	position to which the line is drawn
returns:	void		

Description

pldm moves the pen to (**x,y**) in mm, drawing a straight line. It is equivalent to plotm(**x,y,DOWN**).

See also

plotm, pld

pldot - draw a point (zero length line)

parameter	type	units	description
x,y	float	uu	position where the point is drawn
diameter	float	mm	diameter of the dot
returns:	void		

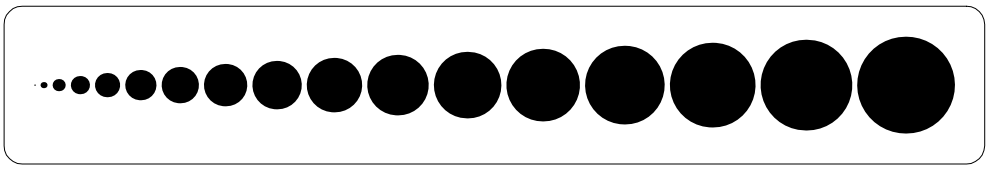
Description

pldot draws a point (that is: a zero length line) with diameter **diameter** in the position (**x,y**) user units. Normally, plotting a zero length line would not do any actual plotting, but pldot forces the pen down on the paper.

Examples

The following program plots a series of dots of increasing size:

```
#include <simplot.h>
int main() {
   float x=50;
   int i;
   plinit(PS,"pldot",A4,0,0,"","");
   for (i=0;i<17;i++) {
      x+=i+.5;
      pldot(x,50,i);
   }
   plframe(5,3);
   exit(0);
}
```



See also

pldotm

See also

plreserv plloop plmvorg

pldotm - draw a point (zero length line)

parameter	type	units	description
x,y	float	mm	position where the point is drawn
returns:	void		

Description

pldotm draws a point (that is: a zero length line) in the position (**x,y**) millimeter units. Normally, plotting a zero length line would not do any actual plotting, but pldot forces the pen down on the paper. The size of the point is determined by the current pen diameter, as set by plset(PENDIA,...).

See also

pldot

pldraw - is the user's drawing routine

parameter	type	units	description
—	void	—	—
returns:	void		

Description

pldraw is the user's drawing routine. The standard one, delivered with the SIMPLOT-package does nothing at all. The user should use a copy of **pldraw** and change it to his needs.

Examples

See plloop for a more complex example of pldraw.

See also

plloop xyplot

plend - end plotting session

parameter	type	units	description
—	void	—	—
returns:	void		

Description

plend ends a plotting session, flushing buffers and closing the plotfile. This routine is normally called automatically when the program exits, but must be called when in one program run another call of plinit is used to start a new plotting session. This may happen, for example, if one wants to print a plot built up under X-windows. An other application would be to make room by closing buffers and allocated memory if plotting is ready but the program continues.

Examples

see the example for plevent

See also

plformat plsmooth plloop pltrace

plevent - is the user's interrupt routine

parameter	type	units	description
xpointer,ypointer	float	uu	position of the mouse cursor at the moment of the interrupt
event_code	int	—	value of the code that caused the interrupt
returns:	void		

Description

plevent is the user's interrupt routine. The standard one, delivered with the SIMPLOT-package only shows the position of the mouse cursor, until the q-key is hit which then causes the application to exit. The user should use a copy of **plevent** and change it to his needs.

Possible values for **event**_code are the ASCII-values of keys and values predefined in simplot.h with self-explaining names:

ArrowDown	F1	MouseDrag
ArrowLeft	F2	MouseLeftDown
ArrowRight	F3	MouseLeftUp
ArrowUp	F4	MouseMiddleDown
Delete	F5	MouseMiddleUp
End	F6	MouseRightDown
Escape	F7	MouseRightUp
Home	F8	
Insert	F9	
PageDown	F10	
PageUp	F11	
	F12	

Except for MouseLeftUp, MouseMiddleUp and MouseRightUp, the values are positive and are generated by key/button press actions. Key/button releases generate the opposite values. One can thus discard key/button release actions by discarding negative event codes. Several event classes may be distinguished using plget:

- plget(EVENT_ASCII) is non-zero if the event was caused by an ASCII key
- plget(EVENT_BUTTON) is non-zero if the event was caused by a mouse button
- plget(EVENT_DOWN) is non-zero if the event was caused by a key or button being pressed
- plget(EVENT_FUNCTION) is non-zero if the event was caused by a function key
- plget(EVENT_UP) is non-zero if the event was caused by a key of button release

Examples

Here is the simplest example, using the default versions of plevent and pldraw, It shows an X window with a 'Quit' button and, with one of the mouse buttons pressed, horizontal and vertical hairlines plus the

```
#include <simplot.h>
int main () {
   plinit(X,"",100,100,10,10,"","");
   plloop();
}
```

See plloop for a more complex example program using several event codes.

See also

plloop

plexit - print a message to stderr and exit

parameter	type	units	description
*message	char*	—	message to be printed
returns:	void		

Description

plexit prints the message to standard error and exits the program

pleye - define location and focus of the eye in 3D plot in user units

parameter	type	units	description
xeye,yeye,zeye	float	uu	position of the eye
xfoc,yfoc,zfoc	float	uu	position of the eye's focus
returns:	void		

Description

pleye set eye-position and focus position in user units.

Examples

See also

plblock

pleyem - define location and focus of the eye in 3D plot in mm

parameter	type	units	description
xeye,yeye,zeye	float	mm	position of the eye
xfoc,yfoc,zfoc	float	mm	position of the eye's focus
returns:	void		

Description

pleyem set eye-position and focus position in millimeter units.

plfcolor - change the filling color

parameter	type	units	description
red,green,blue	float	—	rgb components (0..1)
returns:	void		

Description

plfcolor changes the color for subsequent filling operations. The parameters give the red, green and blue fraction, such that black is represented by 0,0,0 and white by 1,1,1. For many colors, macros have been predefined, thus permitting the replacement of the three parameters with one predefined constant. Thus white may also be set with plfcolor(White) and black with plfcolor(Black). See plcolor for a table of recognised color names.

A special case is the color NoColor: after plfcolor(NoColor) filling is inhibited. This is the default case as plinit calls plfcolor(NoColor).

See also

plcolor, plrect, plrectm

Examples

See also

plfill

plfft - fourier transform an array

parameter	type	units	description
x	float*	—	positions of points to be transformed
n	int	—	number of points
dir	int	—	direction of the transform (0=forward)
returns:	void		

Description

plfft fourier transforms the data in **x**. **n** is the number of elements in **x**. For **dir**=0 the transformation in in forward direction, with other values a back transform is applied.

Examples

See also

xyplot

plfill - fill and/or stroke current path

parameter	type	units	description
—	void	—	—
returns:	void		

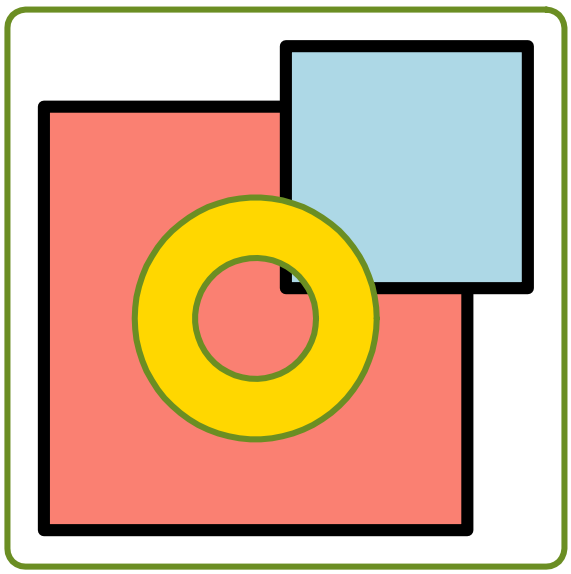
Description

plfill fills the current path using the current fill color and then strokes it using the current stroke color. For filling of complex paths, the non-zero winding rule is applied. The current path is the most recent uninterrupted series of pen movements, either with the pen up or with the pen down. Interruptions in that path are normally caused by many simplot calls such as color changes, (non-stroked) text plotting or program exit. Those interruptions cause the current path to be stroked.

Examples

The following example plots two stroked rectangles, each with its own fill color and then plots two circles in opposite directions but as a single path. By opposing the directions, the path is effectively a ring. For the rectangles, filling is performed using plset(RECTFILL,TRUE). Plfill cannot be used here, since rectangle are plotted using special PostScript commands, not by explicit pen movements.

```
#include <simplot.h>

void pldraw(void) {
   plcolor(OliveDrab);
   plmess("plfill\n");
   if (plget(PLOTTER)==X) {
      plcolor(Magenta);
      plu(-5,-5);
      plformat(0.5,-.5,"Type q to quit");
   }
   plcolor(Black);
   plfcolor(Salmon);
   plrect(0,0,70,70);
   plfcolor(LightBlue);
   plrect(40,40,80,80);
   plset(PENDIA,1);
   plfcolor(Gold);
   plcolor(OliveDrab);
   plarc(35,35,20,0,-360,DOWN);
   plarc(35,35,10,0,360,DOWN);
   plfill();
   plframe(5,3);
}

int main(int argc,char *argv[]) { // without an argument, make a PS file and quit; use X otherwise
   plinit(argc==1 ? PS : X,"plfill",105,105,15,15,"","");
   plset(HEIGHT,4);
   plset(PENDIA,2);
   plset(RECTFILL,TRUE);
   plloop();
   exit(0);
}
```



See also

plcolor, plfcolor, plrect

Postscript Language Reference Manual, page 163.

Bugs

Plfill does not yet work for X programs.

plfont - toggle between primary and alternative font

parameter	type	units	description
—	void	—	—
returns:	void		

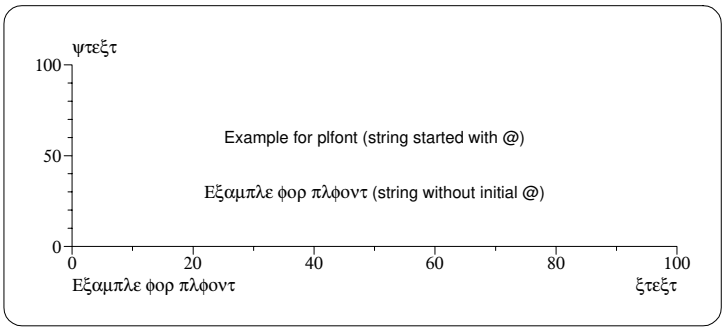
Description

plfont toggles between primary and alternative font. Toggling between fonts may also be performed by using the @-character in strings plotted with plformat, but there, after leaving the routine, the original font is restored. Thus plfont actually sets the default font for plformat.

Examples

The following program plots axes using the alternative font (Symbol), then a text in the normal font by switching font within the text. Finally, a text in alternate font is plotted, showing that font switching within a text is only

```
plinit PS plfont A4 50 50 "" ""
plfont
plaxes 0 0 100 100 100 30 xtext ytext "Example for plfont"
plu 50 60 plformat 0 0 "@Example for plfont (string started with @@)"
plu 50 30 plformat 0 0 "Example for plfont @(string without initial @@)"
plframe 5 3
```



See also

plformat

plformat - plot text by extended C-formatstring with various justifications

parameter	type	units	description
leftright	float	text	horizontal shift
updown	float	text	vertical shift
format	char*	—	any standard C-format
returns:	void		

Description

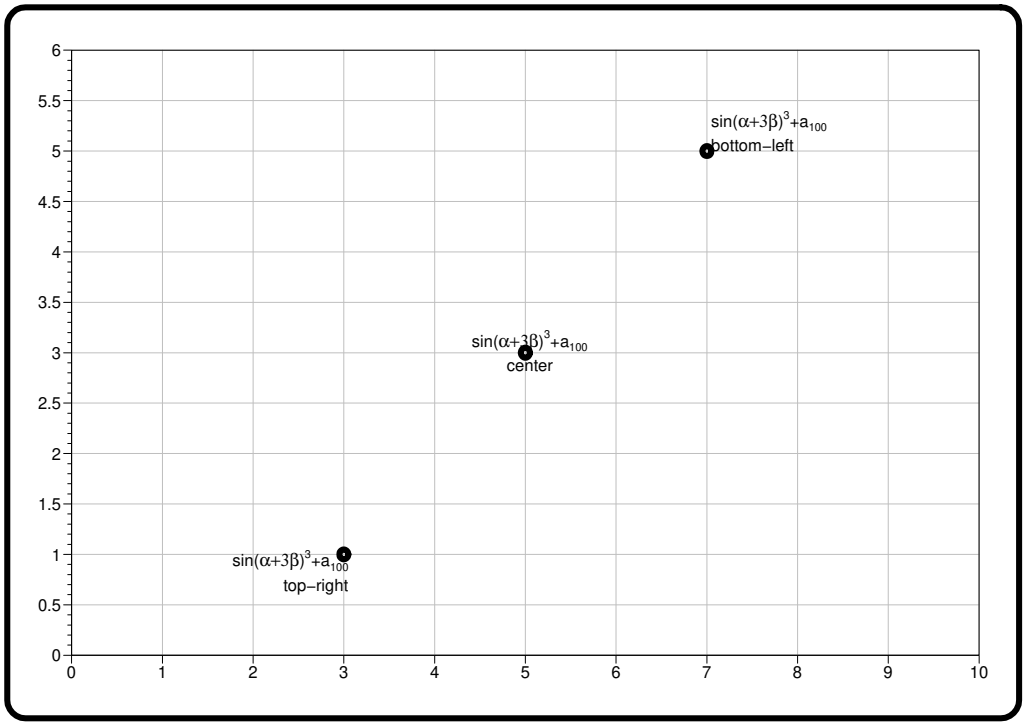
plformat plots formatted data centered at the current position, shifted by (**leftright,updown**) text unit relative to the center. Text units are the length and height of the text. Thus **leftright** =**updown** =0,5 places the lower left of the text box at the current position. **format** can be any valid C-language format string. As expected, the sequence
n inserts a newline, that is: a new line of text is started below the initial position. Some symbols: @ ^ _ # and % will be output only if they are doubled. For the latter two this is obvious as it is imposed by the C-compiler. The first four have a special significance:

@ **(at sign)** toggles between initial and alternate font (see plinit)
^ **(uparrow)** toggles between normal/subscript and superscript
_ **(underscore)** toggles between normal/superscript and subscript
 # (hash mark) toggles between lower and upper ASCII table

Examples

The following script shows how text can be justified relative to the current point:

```
plinit PS plformat A4 15 15 "" ""
plset XGRID 1 plset YGRID 1
plaxes 0 0 10 6 150 100 "" "" ""
plrect 0 0 10 6
plset PENDIA 1 s=.5 a=360 r=.05
plot 7 5 UP plarc 7 5 r 0 a DOWN
plformat  s  s sin(@a+3b@)^3^+a_%d_\nbottom-left 90
plot 5 3 UP plarc 5 3 r 0 a DOWN
plformat  0  0 sin(@a+3b@)^3^+a_%d_\ncenter 90
plot 3 1 UP plarc 3 1 r 0 a DOWN
plformat -s -s sin(@a+3b@)^3^+a_%d_\ntop-right 90
plframe 5 3
```



The following example plots symbol tables showing the Ascii characters in a given font. Both the normal (lower Ascii) and the shifted (upper Ascii) characters are shown. Tables are shown here for the fonts Helvetica, Symbol, ZapfChancery-MediumItalic and ZapfDingbats.

```
#include <stdio.h>
#include <string.h>
#include <simplot.h>
int main() {
   int i,j,k,nf=4,ifont;
   float x,y;
   char *font[]={"H","S","ZCMI","ZD"};
   char *fontname[]={"Helvetica",
                     "Symbol",
                     "ZapfChancery-MediumItalic",
                     "ZapfDingbats"};

   for (ifont=0;ifont<nf;ifont++) {
      plinit(PS,font[ifont],A4,25,269,"C",font[ifont]);
      plset(HEIGHT,5);

      plfont();   // switch to alt font
      plformat(.5,.5,"%s - %s",font[ifont],fontname[ifont]);
      plfont();   // back to normal font
      plmvorgm(0,-15);

      plset(HEIGHT,3);
      for (i=32;i<128;i+=32) {
         x=(i-32)*1.7;
         for (j=0;j<32;j++) {
            y=-j*6;
            k=i+j;
            plu(x,y);
            if (strchr("@^_#",k))   // these chars must be doubled
               plformat(.5,.5,"%3d (%c%c)@   %c%c  # %c%c",k,k,k,k,k,k,k);
            else
               plformat(.5,.5,"%3d (%c)@   %c # %c",k,k,k,k);
            // # must be followed by space, else we get # instead of shifted #
```

```
        }
      }
      plframe(5,3);
      plend();
   }
   exit(0);
}
```

H – Helvetica

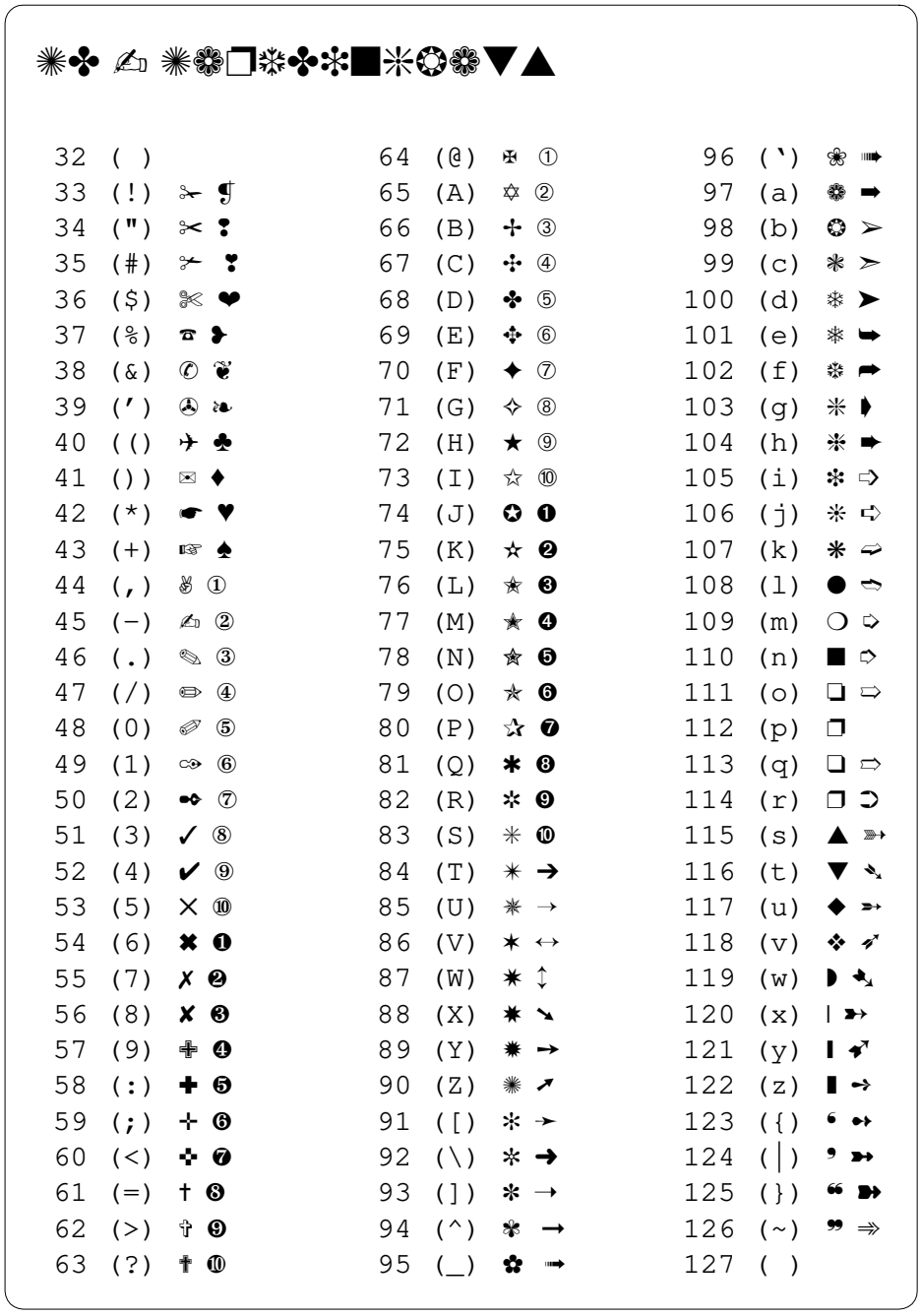
32	( )		64	(@)	@ À	96	(`) ‘ à
33	(!)	! ¡	65	(A)	A Á	97	(a)	a á
34	(")	" ¢	66	(B)	B Â	98	(b)	b â
35	(#)	# £	67	(C)	C Ã	99	(c)	c ã
36	($)	$ ¤	68	(D)	D Ä	100	(d)	d ä
37	(%)	% ¥	69	(E)	E Å	101	(e)	e å
38	(&)	& ¦	70	(F)	F Æ	102	(f)	f æ
39	(’)	’ §	71	(G)	G Ç	103	(g)	g ç
40	(()	(¨	72	(H)	H È	104	(h)	h è
41	())	) ©	73	(I)	I É	105	(i)	i é
42	(*)	* ª	74	(J)	J Ê	106	(j)	j ê
43	(+)	+ «	75	(K)	K Ë	107	(k)	k ë
44	(,)	, ¬	76	(L)	L Ì	108	(l)	l ì
45	(−)	− -	77	(M)	M Í	109	(m)	m í
46	(.)	. ®	78	(N)	N Î	110	(n)	n î
47	(/)	/ ¯	79	(O)	O Ï	111	(o)	o ï
48	(0)	0 °	80	(P)	P Ð	112	(p)	p ð
49	(1)	1 ±	81	(Q)	Q Ñ	113	(q)	q ñ
50	(2)	2 ²	82	(R)	R Ò	114	(r)	r ò
51	(3)	3 ³	83	(S)	S Ó	115	(s)	s ó
52	(4)	4 ´	84	(T)	T Ô	116	(t)	t ô
53	(5)	5 µ	85	(U)	U Õ	117	(u)	u õ
54	(6)	6 ¶	86	(V)	V Ö	118	(v)	v ö
55	(7)	7 ·	87	(W)	W ×	119	(w)	w ÷
56	(8)	8 ¸	88	(X)	X Ø	120	(x)	x ø
57	(9)	9 ¹	89	(Y)	Y Ù	121	(y)	y ù
58	(:)	: º	90	(Z)	Z Ú	122	(z)	z ú
59	(;)	; »	91	([)	[Û	123	({)	{ û
60	(<)	< ¼	92	(\)	\ Ü	124	(|)	| ü
61	(=)	= ½	93	(])	] Ý	125	(})	} ý
62	(>)	> ¾	94	(^)	^ Þ	126	(~)	~ þ
63	(?)	? ¿	95	(_)	_ ß	127	( )	ÿ

Σ – Σψμβολ

32	( )		64	(@)	≅	ℵ	96	(`)		◊
33	(!)	!	ϒ	65	(A)	A	ℑ	97	(a)	α	⟨
34	(")	∀	′	66	(B)	B	ℜ	98	(b)	β	®
35	(#)	#	≤	67	(C)	X	℘	99	(c)	χ	©
36	($)	∃	/	68	(D)	Δ	⊗	100	(d)	δ	™
37	(%)	%	∞	69	(E)	E	⊕	101	(e)	ε	∑
38	(&)	&	*f*	70	(F)	Φ	∅	102	(f)	ϕ	⎛
39	(′)	∍	♣	71	(G)	Γ	∩	103	(g)	γ	⎜
40	(()	(	♦	72	(H)	H	∪	104	(h)	η	⎝
41	())	)	♥	73	(I)	I	⊃	105	(i)	ι	⎡
42	(*)	*	♠	74	(J)	ϑ	⊇	106	(j)	φ	⎜
43	(+)	+	↔	75	(K)	K	⊄	107	(k)	κ	⎣
44	(,)	,	←	76	(L)	Λ	⊂	108	(l)	λ	⎡
45	(−)	−	↑	77	(M)	M	⊆	109	(m)	μ	⎨
46	(.)	.	→	78	(N)	N	∈	110	(n)	ν	⎝
47	(/)	/	↓	79	(O)	O	∉	111	(o)	ο	⎜
48	(0)	0	°	80	(P)	Π	∠	112	(p)	π	
49	(1)	1	±	81	(Q)	Θ	∇	113	(q)	θ	⟩
50	(2)	2	″	82	(R)	P	®	114	(r)	ρ	⌠
51	(3)	3	≥	83	(S)	Σ	©	115	(s)	σ	⎛
52	(4)	4	×	84	(T)	T	™	116	(t)	τ	⎢
53	(5)	5	∝	85	(U)	Y	∏	117	(u)	υ	⎞
54	(6)	6	∂	86	(V)	ς	√	118	(v)	ϖ	⎠
55	(7)	7	•	87	(W)	Ω	·	119	(w)	ω	⎜
56	(8)	8	÷	88	(X)	Ξ	¬	120	(x)	ξ	⎞
57	(9)	9	≠	89	(Y)	Ψ	∧	121	(y)	ψ	⎤
58	(:)	:	≡	90	(Z)	Z	∨	122	(z)	ζ	⎜
59	(;)	;	≈	91	([)	[	⇔	123	({)	{	⎟
60	(<)	<	…	92	(\)	∴	⇐	124	(|)	|	⎟
61	(=)	=	⏐	93	(])	]	⇑	125	(})	}	⎬
62	(>)	>	—	94	(^)	⊥	⇒	126	(~)	~	⎞
63	(?)	?	↵	95	(_)	_	⇓	127	( )		

ZCMI - ZapfChancery-MediumItalic

32	( )		64	(@)	@	96	(`)	‘
33	(!)	! ¡	65	(A)	A `	97	(a)	a Æ
34	(")	" ¢	66	(B)	B ´	98	(b)	b
35	(#)	# £	67	(C)	C ^	99	(c)	c ª
36	($)	$ /	68	(D)	D ~	100	(d)	d
37	(%)	% ¥	69	(E)	E ¯	101	(e)	e
38	(&)	& f	70	(F)	F ˘	102	(f)	f
39	(’)	’ §	71	(G)	G ˙	103	(g)	g
40	(()	(¤	72	(H)	H ¨	104	(h)	h £
41	())	) '	73	(I)	I	105	(i)	i Ø
42	(*)	* “	74	(J)	J °	106	(j)	j Œ
43	(+)	+ «	75	(K)	K ¸	107	(k)	k º
44	(,)	, ‹	76	(L)	L	108	(l)	l
45	(−)	- ›	77	(M)	M ˝	109	(m)	m
46	(.)	. fi	78	(N)	N ˛	110	(n)	n
47	(/)	/ fl	79	(O)	O ˇ	111	(o)	o
48	(0)	0	80	(P)	P —	112	(p)	p
49	(1)	1 –	81	(Q)	Q	113	(q)	q æ
50	(2)	2 †	82	(R)	R	114	(r)	r
51	(3)	3 ‡	83	(S)	S	115	(s)	s
52	(4)	4 ·	84	(T)	T	116	(t)	t
53	(5)	5	85	(U)	U	117	(u)	u ı
54	(6)	6 ¶	86	(V)	V	118	(v)	v
55	(7)	7 •	87	(W)	W	119	(w)	w
56	(8)	8 ‚	88	(X)	X	120	(x)	x ł
57	(9)	9 „	89	(Y)	Y	121	(y)	y ø
58	(:)	: ”	90	(Z)	Z	122	(z)	z œ
59	(;)	; »	91	([)	[	123	({)	{ ß
60	(<)	< ...	92	(\)	\	124	(|)	|
61	(=)	= ‰	93	(])	]	125	(})	}
62	(>)	>	94	(^)	^	126	(~)	~
63	(?)	? ¿	95	(_)	_	127	( )	



32	( )		64	(@)	✠	①	96	(`)	❀	➠
33	(!)	✁	¶	65	(A)	✡	②	97	(a)	❁	➡
34	(")	✂	❢	66	(B)	✢	③	98	(b)	✪	➢
35	(#)	✂	❣	67	(C)	✣	④	99	(c)	✱	➢
36	($)	✄	❤	68	(D)	✤	⑤	100	(d)	❄	➤
37	(%)	☎	❥	69	(E)	✥	⑥	101	(e)	✱	➥
38	(&)	✆	❦	70	(F)	✦	⑦	102	(f)	❄	➦
39	(’)	✇	❧	71	(G)	✧	⑧	103	(g)	✳	➧
40	(()	✈	♣	72	(H)	★	⑨	104	(h)	❇	➨
41	())	✉	♦	73	(I)	☆	⑩	105	(i)	❇	➩
42	(*)	☛	♥	74	(J)	✪	❶	106	(j)	✳	➪
43	(+)	☞	♠	75	(K)	✯	❷	107	(k)	✱	➫
44	(,)	✌	①	76	(L)	✰	❸	108	(l)	●	➬
45	(−)	✍	②	77	(M)	✭	❹	109	(m)	❍	➭
46	(.)	✎	③	78	(N)	✮	❺	110	(n)	■	➮
47	(/)	✏	④	79	(O)	✯	❻	111	(o)	❑	➯
48	(0)	✐	⑤	80	(P)	✰	❼	112	(p)	❒	
49	(1)	✑	⑥	81	(Q)	✱	❽	113	(q)	❑	➱
50	(2)	✒	⑦	82	(R)	✲	❾	114	(r)	❒	➲
51	(3)	✓	⑧	83	(S)	✳	❿	115	(s)	▲	➳
52	(4)	✔	⑨	84	(T)	✴	➔	116	(t)	▼	➴
53	(5)	✕	⑩	85	(U)	✵	→	117	(u)	◆	➵
54	(6)	✖	❶	86	(V)	✶	↔	118	(v)	❖	➶
55	(7)	✗	❷	87	(W)	✷	↕	119	(w)	◗	➷
56	(8)	✘	❸	88	(X)	✸	↘	120	(x)	❙	➸
57	(9)	✙	❹	89	(Y)	✹	➙	121	(y)	❚	➹
58	(:)	✚	❺	90	(Z)	✺	➚	122	(z)	❯	➺
59	(;)	✛	❻	91	([)	✻	➛	123	({)	❮	➻
60	(<)	✜	❼	92	(\)	✼	➜	124	(|)	❰	➼
61	(=)	✝	❽	93	(])	✽	→	125	(})	❱	➽
62	(>)	✞	❾	94	(^)	✾	➝	126	(~)	❲	➾
63	(?)	✟	❿	95	(_)	✿	➞	127	( )		

See also

plfill plclip plbox plframe plarrow plfunc plrotate pltrace plaxes plhiss plreserv plfont plloop plsmooth simplot program plpolar plpolv plot

plframe - draw a box around current plot

parameter	type	units	description
dist	float	mm	distance between the box and the extremes of the plot
radius	float	mm	radius of the corners
returns:	void		

Description

plframe plots a box with rounded corners (radius=**radius**) at a distance of dist mm of what was plotted since either the start of the plot or since the last call of **plframe** with dist $¡$ 0. If **dist** $¡$ 0, the limits are initialized, but nothing is plotted.

Examples

```
plinit PS plframe A4 0 0 "" ""
plu 50 50 plformat 0 0 tekst1 plframe 2 0 plframe 1 0 plframe -1 0
plu 50 30 plformat 0 0 tekst2 plframe 2 0 plframe 1 0
```

See also

plformat plarrow plfunc plrotate pltrace plaxes plreserv plhiss plhist plsmooth simplot program plaxfit plpie plot

plfunc - plot a two-dimensional function

parameter	type	units	description
func(float)	float*	uu	function of x to be plotted
xmin,ymin,xmax,ymax	float	uu	window in which the function is plotted
linetype	int	—	see plot
returns:	void		

Description

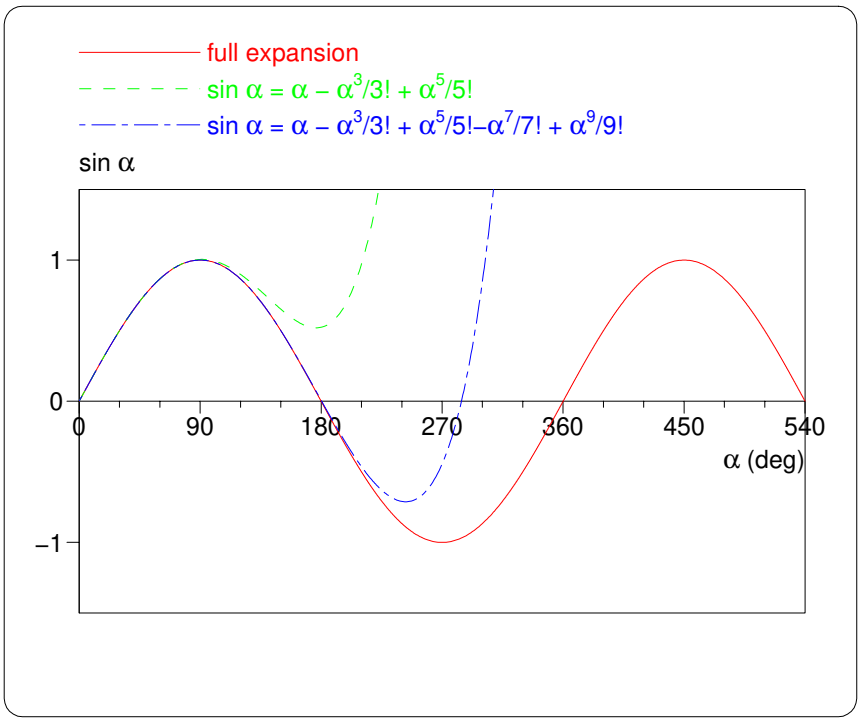
plfunc plfunc plots the function **func**(x) as far as it is within the window **xmin,** ymin, xmax, ymax. Stepsize is adjusted depending on earlier changes in the curvature. As a result, functions with sudden changes may give problems.

Examples

see program sine.c on the next page

```
// plotting sinusoidal curves, using various degrees of series expansion
//
// features: plotting of coloured and dashed curves
//           suppression of automatic scaling
//           crossing axes
//           drawing of text with greek letter and superior numbers

#include <simplot.h>
#include <math.h>
#define Deg2Rad (M_PI/180)

void plosin(int textshft,float red, float green, float blue,
            float (*func)(float),int def, char *text)
// plots the function y=func(x). def defines dashed line
{ plcolor(red,green,blue);
   plot(0,1.7,UP);
   plotrm(0,textshft*plget(HEIGHT),UP);
   plotrm(20,0,def);
   plformat(.5,0,text);
   plfunc(func,0,-1.5,540,1.5,def);
   plcolor(Black);
}

float sinx(float x) {
   x*=Deg2Rad;
   return sin(x);
}
float sin5(float x) {
   x*=Deg2Rad;
   return x-pow(x,3)/6+pow(x,5)/120;
}
float sin9(float x) {
   x*=Deg2Rad;
   return x-pow(x,3)/6+pow(x,5)/120-pow(x,7)/5040+pow(x,9.)/362880.;
}

void pldraw(void) {
   plaxes(0,-1.5,540,1.5,120,70,"@a@ (deg)","sin @a@"," ");
   plrect(0,-1.5,540,1.5);
```

```
   plosin(6,Red,      sinx,DOWN,
          " full expansion");
   plosin(4,Green,sin5,2222,
          " sin @a@ = @a@ - @a@^3^/3! + @a@^5^/5!");
   plosin(2,Blue,     sin9,1141,
          " sin @a@ = @a@ - @a@^3^/3! + @a@^5^/5!-@a@^7^/7! + @a@^9^/9!");
   plframe(5,3);
}

int main() {
   plinit(PS,"plfunc",160,150,30,80,"","");
   plset(HEIGHT,3);
   plset(PLOTMODE,GXcopy);
   plset(XMARK,30);     // forces scale marks every 30 degrees
   plset(XSKIP,3);      // forces number at axis every 90 degrees
   plset(CROSS,1);      // axes P_cross in (0,0)
   plloop();
   exit(0);
}
```



plget - return the value of a global variable

parameter	type	units	description
name	int	—	alias for an internal variable to be retrieved
returns:	float		

Description

plget delivers current internal values to the user. For **name** the following values are predefined in `simplot.h`:

global	description
ANGLE	symbol plotting direction in degrees
HEIGHT	symbol height in mm
LINESKIP	distance between lines in plformat, in units of symbolheight
OPAQUETEXT	TRUE if text is plotted opaquely, FALSE if text is transparant
PENDIA	linewidth in mm
PLOTMODE	plot mode (GXxor or GXcopy)
PLOTTER	plotter-name
RESOLUTION	resolution in plits per millimeter
VERSION	simplot version number (print with format %.2f)
XANGLE	direction of the x-axis in degrees
YANGLE	direction of the y-axis in degrees
XCROSS	x-coordinate for the intersection of axes to be plotted
YCROSS	y-coordinate for the intersection of axes to be plotted
XLOG	true if an x-axis will be defined logarithmically, else false
YLOG	true if an y-axis will be defined logarithmically, else false
XPEN	x-coordinate of the pen in user units
YPEN	y-coordinate of the pen in user units
XU2M	mm-scaling factor in the x-direction (size of the user unit in mm)
YU2M	mm-scaling factor in the y-direction (size of the user unit in mm)
XU2P	plit-scaling factor in the x-direction (size of the user unit in plits)
YU2P	plit-scaling factor in the y-direction (size of the user unit in plits)
XSIZE	paper size in mm in the x-direction
YSIZE	paper size in mm in the y-direction
EVENT_ASCII	TRUE if the last X-event was caused by an ASCII key
EVENT_BUTTON	TRUE if the last X-event was caused by a mouse button
EVENT_DOWN	TRUE if the last X-event was caused by a key or button being pressed
EVENT_FUNCTION	TRUE if the last X-event was caused by a function key
EVENT_UP	TRUE if the last X-event was caused by a key or button release

Examples

ang=plget(ANGLE)

height=plget(HEIGHT)

See also

plfill plfunc plsmooth plloop

plhairs - draw horizontal and/or vertical hairlines

parameter	type	units	description
x,y	float	uu	position at which the hairs are drawn
hor,vert	int	—	if TRUE, draw horizontal/vertical hair
returns:	void		

Description

plhairs draws horizontal and/or vertical hairlines, depending on the values of **hor** and **vert**. **plhairs** is normally used in event-catching routines, such as plevent.

See also

plevent

Examples

See also

xyplot

plhiss - plot shaded histogram

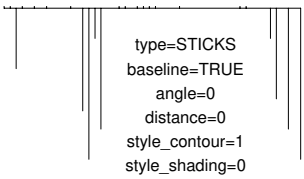
parameter	type	units	description
x,y	float*	uu	arrays defining the histogram
n	int	—	number of elements in the arrays
type	int	—	histogram type: BARS, ENVELOPE or STICKS
baseline	int	—	if TRUE, a baseline is drawn
ybase	float	uu	y-position of the baseline, if any
style_contour	int	—	line style for the histogram contours
angle	float	degr	angle for hatching
distance	float	mm	distance between hatching lines
style_shade	int	—	line style for hatching lines
returns:	void		

Description

plhiss plots a histogram using values in arrays **x** (position) and **y** (height+ybase). **x** must be in ascending order. If **baseline** is TRUE a baseline is drawn at y=**ybase**. **type** determines the histogram type and can have one of three predefined values: BARS, ENVELOPE and STICKS. See the example for further explanation. The histogram contours are dashed according to the parameter **style**_contour the histogram may also be hatched by setting **distance** to a non-zero value.

Examples

```
#include <simplot.h>
int main() {
   float x[] = { 1, 2, 3, 4, 6, 8,12,14,15,16,17,18,20,
                 21,23,24,25,26,30,40,41,45,46,48,50};
   float y[] = { 5,15,-10,15,25,20,15,-17,-25,-5,-20,0,
                 6,7,8,9,10,9,8,7,6,-5,-15,-20,-25};
   int type[] = {BARS, BARS,  ENVELOPE, STICKS};
   int base[] = {TRUE, TRUE,  FALSE,    TRUE};
   int das[]  = {DOWN, DOWN,  DOWN,     DOWN};
   int ang[]  = {0,    45,    0,        0};
   int dis[]  = {1,    1,     0,        0};
   int sda[]  = {0,    1111,  0,        0};
   char *s[5]={"FALSE","TRUE","STICKS","ENVELOPE","BARS"};
   int i;

   plinit(PS,"plhiss",200,150,0,0,"","");
   for (i=0;i<4;i++) {
      plreserv(30,25,55);
      plhiss(x,y,25,type[i],base[i],0,das[i],ang[i],dis[i],sda[i]);
      plot(31,-5,UP);
      plformat(0,-.5,"type=%s\n"
               "baseline=%s\n"
               "angle=%d\n"
               "distance=%d\n"
               "style__contour=%d\n"
               "style__shading=%d",
               s[type[i]],s[base[i]],ang[i],dis[i],das[i],sda[i]);
   }
   exit(0);
}
```


type=STICKS
baseline=TRUE
angle=0
distance=0
style_contour=1
style_shading=0

plhist - plot histogram or stick plot

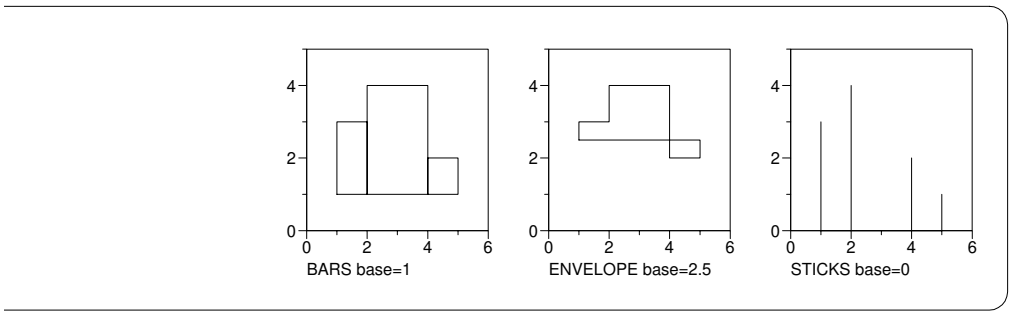
parameter	type	units	description
*x,*y	float*	uu	arrays defining the histogram
n	int	—	number of elements in the arrays
type	int	—	histogram type: BARS, ENVELOPE or STICKS
baseline	int	—	if TRUE, a baseline is drawn
ybase	float	uu	y-position of the baseline, if any
returns:	void		

Description

plhist plots a histogram using values in arrays **x** (position) and **y** (height). **x** must be in ascending order. If **baseline** is TRUE a baseline is drawn at y=**ybase**. **type** determines the histogram type and can have one of three predefined values: BARS, ENVELOPE and STICKS. See the example for further explanation.

Examples

```
n=4 plrdx n 1 2 4 5 plrdy n 3 4 2 1

plinit PS plhist  A4 50 50 "" ""
plaxes 0 0 6 5 30 30 "" "" "BARS base=1"
plrect 0 0 6 5
       plhist n BARS TRUE 1   plmvorg 8 0
plaxes 0 0 6 5 30 30 "" "" "ENVELOPE base=2.5"
plrect 0 0 6 5
       plhist n ENVELOPE TRUE 2.5 plmvorg 8 0
plaxes 0 0 6 5 30 30 "" "" "STICKS base=0"
plrect 0 0 6 5
       plhist n STICKS TRUE 0
plframe 5 3
```



plinit - initialize the system

parameter	type	units	description
driver	int	—	plotter driver, currently PS, PSL, PDF, PDFL or X
file	char*	—	plotfile name, stdout if empty, not used when driver is X
xsize,ysize	float	mm	paper size
xorg,yorg	float	mm	position of the origin relative to lower left of the paper
initial_font	char*	—	initial font name
alternate_font	char*	—	alternate font name
returns:	void		

Description

plinit (re)initializes plot package. It chooses a new plotter driver. Currently three drivers are available:

PS portrait oriented PostScript
PSL landscape oriented PostScript
PDF portrait oriented PDF
PDFL landscape oriented PDF
X for plotting on the screen (X-windows or VGA).

plinit resets all internal variables, including the origin and scaling. Scaling is reset to millimeter units in both x- and y-direction, and the origin is reset to **xorg,yorg** millimeters relative to the lower left corner of the paper. The parameter **file** is a name for the plotfile. It is not used when plotting in X-windows; otherwise, if it is empty then the PostScript output will be sent to standard output and if it is non-empty, then the plotfile will be `file.eps`. The pair **width,height** determines the paper/screen sizes in mm; it may also be replaced by one of the following macros:

A0	841	x	1189
A1	594	x	841
A2	420	x	594
A3	297	x	420
A4	210	x	297
A5	146	x	210
A6	105	x	146
A7	74	x	105

Using PSL or PDFL for the plotter driver automatically exchanges paper width and height. Thus A4 is the normal papersize, both for PS and for PSL. For PostScript plotting, the papersize given should be the actual papersize of the plotter. It may be tempting to use other papersizes instead, for example to influence the behaviour of plreserv. However, plclip should be used for such purposes. The two font parameters make two fonts available, the initial font and an alternate font. The latter becomes effective if either plfont is called or, in plformat, a @–character is found in the format string. Available Postscript fonts have names constructed from the capital letters of their standard Postscript names. Their geometries are found in files with those names in the directory `/simplot/fonts`. Currently available fonts are:

AGB	AvantGarde-Book	HNO	Helvetica-Narrow-Oblique
AGBO	AvantGarde-BookOblique	HO	Helvetica-Oblique
AGD	AvantGarde-Demi	NCSB	NewCenturySchlbk-Bold
AGDO	AvantGarde-DemiOblique	NCSBI	NewCenturySchlbk-BoldItalic
BD	Bookman-Demi	NCSI	NewCenturySchlbk-Italic
BDI	Bookman-DemiItalic	NCSR	NewCenturySchlbk-Roman
BL	Bookman-Light	PB	Palatino-Bold
BLI	Bookman-LightItalic	PBI	Palatino-BoldItalic
C	Courier	PI	Palatino-Italic
CB	Courier-Bold	PR	Palatino-Roman
CBO	Courier-BoldOblique	S	Symbol
CO	Courier-Oblique	TB	Times-Bold
H	Helvetica	TBI	Times-BoldItalic
HB	Helvetica-Bold	TI	Times-Italic

HN	Helvetica-Narrow	TR	Times-Roman
HNB	Helvetica-Narrow-Bold	ZCMI	ZapfChancery-MediumItalic
HNBO	Helvetica-Narrow-BoldOblique	ZD	ZapfDingbats

If an empty string is given, Helvetica ("H") is used for the **initial**_font and Symbol ("S") for the **alternate**_font For plotting under X-windows, any available X-font definition string may be given. Default values are

`"-*-helvetica-medium-r-*-*-10-*-*-*-*-*-*-*"`

for the **initial**_font and

`"-*-symbol-medium-r-*-*-10-*-*-*-*-*-*-*"`

for the **alternate**_font

Examples

See also

plpoly plshade plfill pldot plclip plarcr plbox plformat plmvorg plframe plblock plarrow plfunc plrotate pltrace plaxes plpgon plarc plfont plreserv plhiss plpline plhist plsmooth plloop simplot program plaxfit plpolar plpie plpolv plevent plot

pllabels - draw an array of labels

parameter	type	units	description
x,y	float*	uu	positions of labels to be plotted
lbels	char**	—	pointers to the labels
n	int	—	number of labels
dx,dy	float	—	offsets for plformat
angle	float	deg	text angle
transparant	int	—	true for transparant, false for opaque drawing
returns:	void		

Description

pllabels draws the labels in **labels** at the positions in **x** and **y,**

plloop - start event loop

parameter	type	units	description
—	void	—	-
returns:	void		

Description

plloop, if the plotter driver is X, starts an event loop, testing whether a key or a mouse button was hit. If so, plevent is called, with information about the key that was hit and the current position of the mouse pointer. The user is responsible for delivering the plevent routine. See plevent. If the plotter driver is PostScript (PS or PSL) then plloop is equivalent to calling pldraw and plend. The user is responsible for supplying the pldraw-routine. This facility has been built in for users who want to temporarily convert an X-windows program into a PostScript plotting program without having to make it dynamically switchable .

Examples

The following program is a relatively simple example of the use of plloop, yet it is very illustrative for many features of simplot. It plots MIRA-fractals for which the user can change the parameters interactively. The program sets up a loop recognizing several events:

F1 shows help information about the key recognized
q quits the program
p plots the current fractal to the plotter
0-9.- these keys enter a number. The current number becomes visible in the top centre of the window
n set the number of iterations to the current number and redraws the fractal
a sets fractal parameter a to the current number and draws the new fractal
b sets fractal parameter b to the current number and draws the new fractal
mouse left button may be pressed and dragged to expand a subrectangle
mouse right button undoes the expansion, returning the fractal to its original dimensions
mouse middle button displays two crosshairs, annotated with the x,y position of their intersection

The following page shows the initial plot (a=0.4, b=0.999).

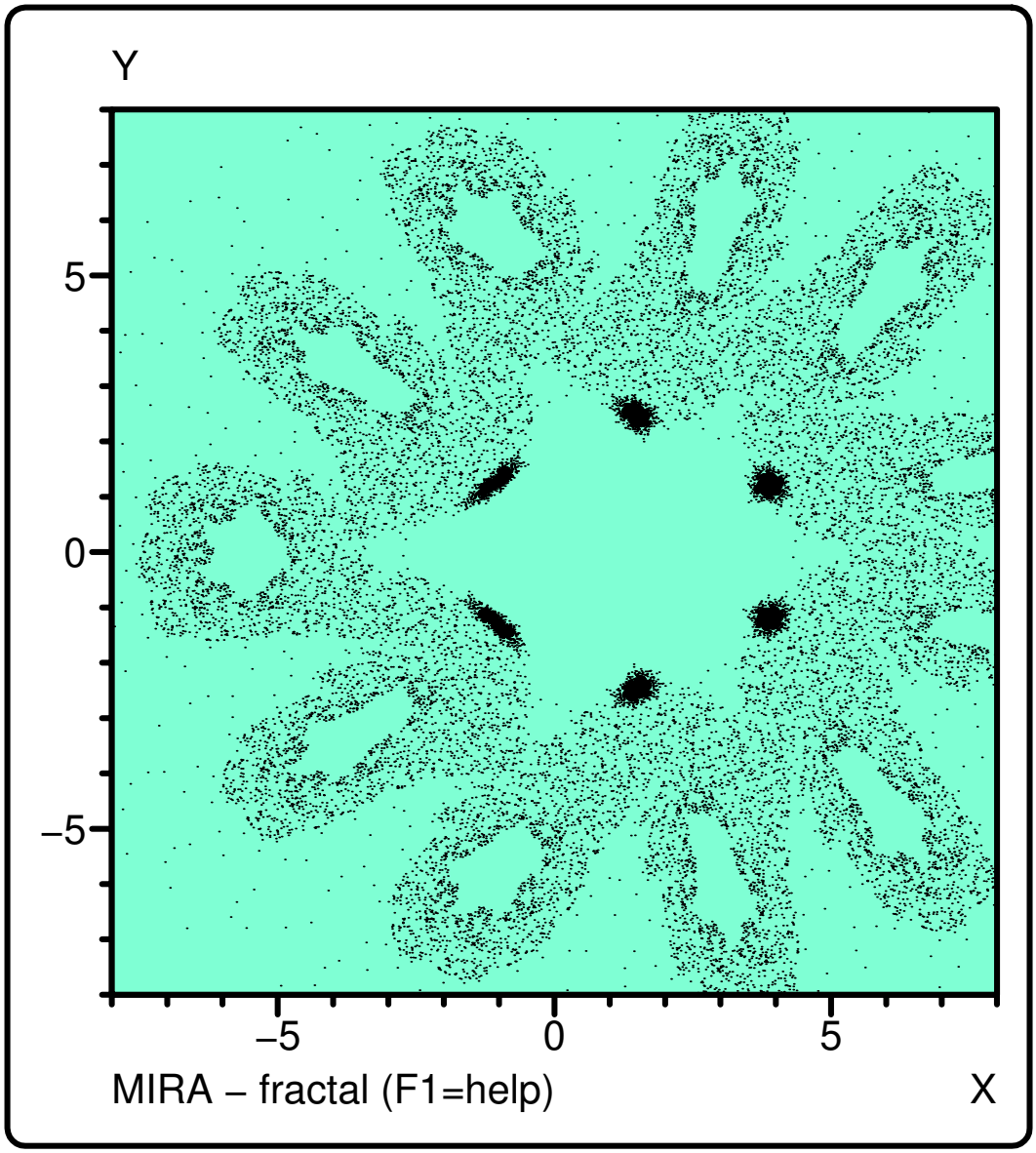
```
// plloop example - basic X-windows program plotting fractals

#include <stdio.h>
#include <simplot.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
#define NONE 0
#define EXPANDING 1
#define MARKING 2
#define BW 16
#define SIZE 150

float number=0.,a=.40002,b=.999;
//float number=0.,a=.399,b=.999;
float ymin=-8,ymax=8,xmin=-8,xmax=8;
float xmn=-8,ymn=-8,xmx=8,ymx=8;
float xl=SIZE,yl=SIZE;
int num=100000,mode=NONE;
float dot=.3;

void pldraw(void) {
   int i;
```

```
   double w,x,y,z;
   float c;
   if (plget(PLOTTER)==X)
      plpage();      // clear the window
   plaxes(xmn,ymn,xmx,ymx,xl,yl,"X","Y","MIRA - fractal (F1=help)");
   if (plget(PLOTTER)!=X)
      plframe(5,3);
   plset(RECTFILL,TRUE);
   plrect(xmn,ymn,xmx,ymx);
   plset(RECTFILL,FALSE);

   x=12;
   y=0;
   c=2*(1-a);
   w=a*x+c*x*x/(1+x*x);
   if (plget(PLOTTER)==X)
      plcolor(Red);
   for (i=1;i<num;i++) {                 // plot the fractal
      z=x;
      x=b*y+w;
      w=a*x+c*x*x/(1+x*x);
      y=w-z;
      if (x<xmn)
         continue;
      if (x>xmx)
         continue;
      if (y<ymn)
         continue;
      if (y>ymx)
         continue;
      pldot(x,y,dot);
   }
   plcolor(Blue);
   plmess("MIRA fractal  %d iterations  a=%g  b=%g\n",num,a,b);
   plcolor(Black);
}

void expand(float xmi, float ymi, float xma, float yma)
// expand a rectangle opened with the mouse
{ xmn=xmi;
   ymn=ymi;
   xmx=xma;
   ymx=yma;
   pldraw();
}

void unexpand(void) {
   expand(xmin,ymin,xmax,ymax);
}

void plevent(float xpointer,float ypointer,int key) {
   static int numpos=0,help=TRUE;
   static char numstring[100]="";
   int toplotter=FALSE;

   // for non-mouse events look at key down only:
   if (!plget(EVENT_BUTTON))
      if (key<0)
         return;

   switch(key) {
```

```
case 'a':
   a=number;
   number=numpos=0;
   unexpand();
   break;
case 'b':
   b=number;
   number=numpos=0;
   unexpand();
   break;
case 'd':
   dot=number;
   number=numpos=0;
   pldraw();
   break;
case 'n':
   num=(int)number;
   number=numpos=0;
   pldraw();
   break;
case F1:    // F1 function key: help
   if (!help)
      pldraw();
   else {
      plpage();
      plu(xmn,ymx);
      plformat(.5,-1, "F1 : help (press again to continue)\n"
               "[0-9.-] : build number\n"
               "a : set parameter (now %g) a to current number\n"
               "b : set parameter (now %g) a to current number\n"
               "d : set dotsize in mm\n"
               "n : set number of iterations (now %d)\n"
               "P : print to plotfile and plotter\n"
               "p : print to plotfile only\n"
               "q : quit\n"
               "Mouse:\n"
               "Leftdrag : expand\n"
               "Right    : unexpand\n"
               "Middle   : track position\n",a,b,num);
   }
   help=!help;
   break;
case '0':
case '1':
case '2':
case '3':
case '4':
case '5': // build number
case '6':
case '7':
case '8':
case '9':
case '-':
case '.':
   numstring[numpos++]=key;
   numstring[numpos]=0;
   sscanf(numstring,"%g",&number);
   plmess("current number: %g",number);
   return;
case Delete:
   if (numpos)
```

```
      numstring[--numpos]=0;
   sscanf(numstring,"%g",&number);
   plmess("current number: %g",number);
   return;
case Escape:
   number=numpos=0;
   plmess("");
   break;
case 'P':                          // plot to plotter
   toplotter=TRUE;
case 'p': // plot to file only
   plmess("");
   plend();
   plinit(PS,"plloop",A4,30,30,"","");
   pldraw();
   plend();
   if (toplotter)
      system("plot plloop&");
   plinit(X,"",SIZE+20,SIZE+20,10,10,"","");
   plloop();
   break;
case 'q':                          // Quit
   plend();
   exit(0);
case MouseLeftDown:                // initialize rectangle
   plmess("expanding");
   mode=EXPANDING;
   xmn=xpointer>xmin?xpointer:xmin;
   ymn=ypointer>ymin?ypointer:ymin;
   xmx=xmn;
   ymx=ymn;
   plset(PLOTMODE,GXxor);
   break;
case MouseLeftUp:                  // expand part in rect
   plset(PLOTMODE,GXcopy);
   if ((xmx-xmn)*plget(XU2P)<10||(ymx-ymn)*plget(YU2P)<10)
      unexpand();
   else
      expand(xmn,ymn,xmx,ymx);
   mode=NONE;
   break;
case MouseMiddleDown:
   plhairs(xpointer,ypointer,TRUE,TRUE);
   mode=MARKING;
   break;
case MouseMiddleUp:
   mode=NONE;
   plhairs(xpointer,ypointer,FALSE,FALSE);   // erase last hairs
   plmess("MIRA fractal  %d iterations  a=%g  b=%g\n",num,a,b);
   break;
case MouseRightUp:                  // restore full plot
   unexpand();
   break;
case MouseDrag:                            // change rectangle
   if (xpointer<xmn)
      break;
   if (ypointer<ymn)
      break;
   if (xpointer>xmax)
      break;
   if (ypointer>ymax)
```

```
            break;
         switch(mode) {
            case EXPANDING:
               plrect(xmn,ymn,xmx,ymx);              // replot previous rect, erasing it
               xmx=xpointer;
               ymx=ypointer;
               plrect(xmn,ymn,xmx,ymx);              // plot new rect
               break;
            case MARKING:
               plmess("x=%g y=%g",xpointer,ypointer);
               plhairs(xpointer,ypointer,TRUE,TRUE);
               break;
         }
         break;
   }
}

int main(int argc,char *argv[]) { // without an argument, just make a PostScript plotfile and quit
   // with an argument, run under X
   if (argc==1) {
      plinit(PS,"plloop",A4,50,50,"","");
      plset(PENDIA,1);
      plset(HEIGHT,5);
      plfcolor(Aquamarine);
      pldraw();
      plend();
   } else {
      plinit(X,"",SIZE+50,SIZE+50,25,25,"","");
      plset(HEIGHT,5);
      plset(PENDIA,.3);
      plfcolor(Aquamarine);
      plloop();
   }
   exit(0);
}
```



See also

plfill plfunc plevent

plmess - display a message

parameter	type	units	description
format	char*	—	format creating the message from the optional parameters
returns:	void		

Description

plmess creates a message formatted using the format-string and the optional parameters. If PostScript is the plotter driver the message is sent to standard error. In X, the message is displayed in the top center of the display window. If the message contains a BELL-character (ASCII 7) it is removed and sent to standard error.

Examples

See also

plfill plloop

plmvorg - move origin in user units

parameter	type	units	description
x,y	float	uu	position to which the origin is moved
returns:	void		

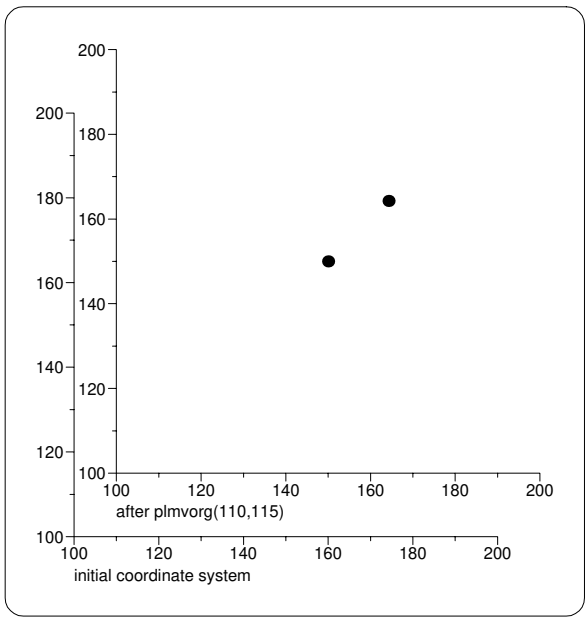
Description

plmvorg moves the origin (the position (0,0)) of the user units coordinate system to the position (**x,y**) in the current user units coordinate system (plmvorg).

Examples

The following program draws axes, then moves the origin and plots the same axes again; as the axes cross in the origin (that is: in the point (0,0)) in millimeters, the new axes are shifted with respect to the initial axes. Finally, a dot is plotted in the new coordinate system, before as well as after moving the origin with plmvorgm.

```
plinit PS plmvorg A4 50 50 "" ""
plaxes 100 100 200 200 70 70 "" "" "initial coordinate system"
plmvorg 110 115
plaxes 100 100 200 200 70 70 "" "" "after plmvorg(110,115)"
pldot 150 150 2 plmvorgm 10 10 pldot 150 150 2
plframe 5 3
```



See also

plformat plaxes plpgon plreserv plpline plhist simplot program plot

plmvorgm - move origin in mm

parameter	type	units	description
x,y	float	mm	position to which the origin is moved
returns:	void		

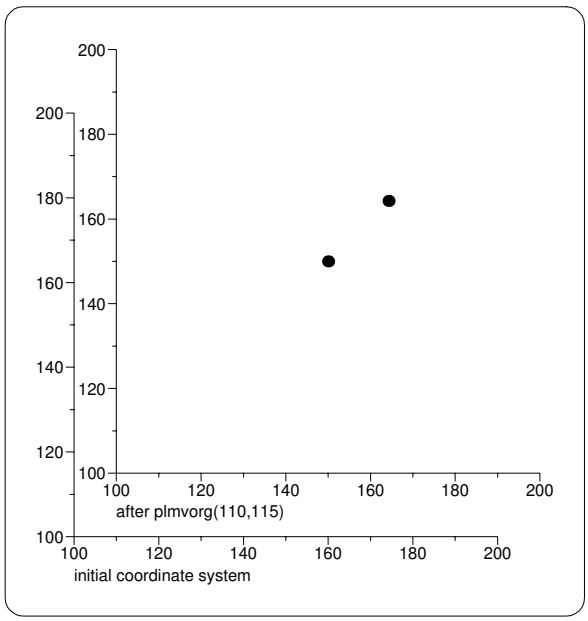
Description

plmvorgm moves the origin (the position (0,0)) of the millimeter coordinate system to the position (**x,y**) in the current millimeter coordinate system.

Examples

The following program draws axes, then moves the origin and plots the same axes again; as the axes cross in the origin (that is: in the point (0,0)) in millimeters, the new axes are shifted with respect to the initial axes. Finally, a dot is plotted in the new coordinate system, before as well as after moving the origin with plmvorgm.

```
plinit PS plmvorg A4 50 50 "" ""
plaxes 100 100 200 200 70 70 "" "" "initial coordinate system"
plmvorg 110 115
plaxes 100 100 200 200 70 70 "" "" "after plmvorg(110,115)"
pldot 150 150 2 plmvorgm 10 10 pldot 150 150 2
plframe 5 3
```



See also

plformat plmvorg plaxes plpgon

plot - move pen to new position in user units

parameter	type	units	description
x,y	float	uu	new pen position
linetype	int	—	format of the line drawn
returns:	void		

Description

plot moves the pen to (**x,y**) in user units. The format of the line produced (if any) is determined by the **linetype** parameter:

DOWN plot with the pen down, drawing a solid line with the current pen diameter (see plset)
UP plot with the pen up, so nothing is plotted, the pen is only moved
DASH plot a dashed line, alternately 2 dash units DOWN and 2 dash units UP
any other value is used as a format to create a dashed line; see the section Dash

Dash format

The third argument of **plot** is interpreted as a 4-digit number, each digit representing the number of dash units by which the pen is alternately going up and down. Alternately, a leading 1 may be added to the 4-digit number, resulting in the dash unit to be set to 0.1 mm temporarily. The lengths over which the pen went up and down is remembered between successive calls of **plot** or any other routine producing dashed lines, thus allowing building up a dashed line from many small sections.

The dash unit, by default, is 1 millimeter, but this may changed with plset

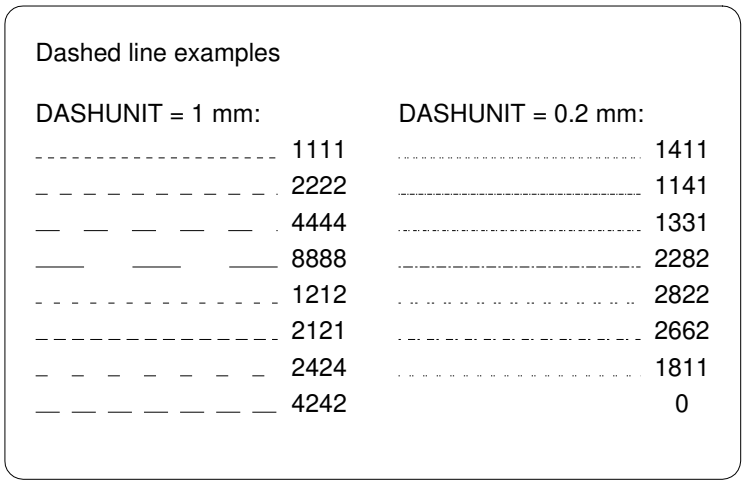
See also

plotm, plotr, plotrm, plu, plum

Examples

```
plinit PS plot A4 50 160 "" ""
plset HEIGHT 3
plformat .5 .5 "Dashed line examples"
plmvorg 0 -10 pltext0 "DASHUNIT = 1 mm:" plotr 0 -6 UP
d=1111 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=2222 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=4444 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=8888 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=1212 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=2121 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=2424 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=4242 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
plset DASHUNIT .2
plmvorg 60 0
pltext0 "DASHUNIT = 0.2 mm:" plotr 0 -6 UP
d=1411 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=1141 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=1331 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=2282 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=2822 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=2662 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
d=1811 plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
```

```
d=0    plotr 40 0 d plformat .5 .5 %6.0f d plotr -40 -6 UP
plframe 5 3
```



See also

plfill pldot plclip plarcr plbox plformat plmvorg plframe plblock plfunc plpolc pltrace plaxes plarc plfont plreserv plaxfit plpolv plpoly plshade plarrow plrotate plpgon plhiss plhist plpline plloop plsmooth simplot program plpie plpolar plevent

plot3 - move pen to new 3-D position in user units

parameter	type	units	description
x,y,z	float	uu	new pen position
linetype	int	—	see plot
returns:	void		

Description

plot3 moves the pen to (**x,y,z**) in user units. The format of the line produced (if any) is determined by the **linetype** parameter:

DOWN plot with the pen down, drawing a solid line with the current pen diameter (see plset)
UP plot with the pen up, so nothing is plotted, the pen is only moved
DASH plot a dashed line, alternately 2 dash units DOWN and 2 dash units UP
any other value is used as a format to create a dashed line; this number is interpreted as a 4-digit number, each digit representing the number of dash units[1] by which the pen is alternatingly going up and down. Alternatively, a leading 1 may be added to the 4-digit number, resulting in the dash unit to be set to 0.1 mm temporarily. The lengths over which the pen went up and down is remembered between successive calls of **plot3** or any other routine producing dashed lines, thus allowing building up a dashed line from many small sections.

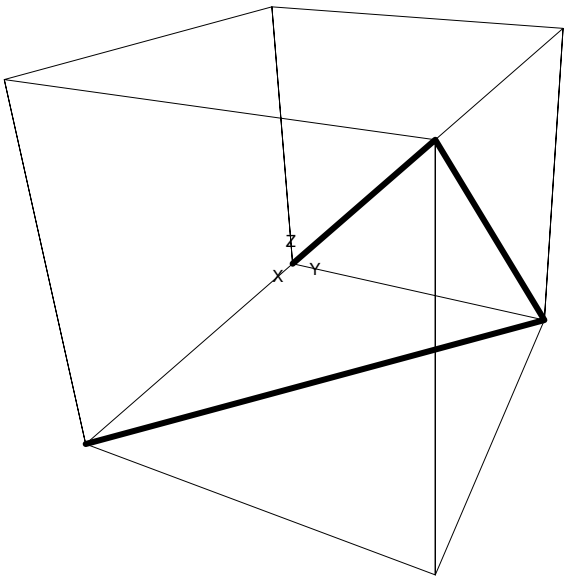
[1] by default, a dash unit is 1 millimeter, but this may changed with plset

See also

plotm3, plotr3, plotrm3

Example

```
plinit PS plot3 A4 100 100 "" ""
plscale3 0 0 0 50 50 50 100 100 100
pleye 100 70 70 50 50 50
plotm3 0 0 10 UP plformat 0 0 Z
plotm3 0 10 0 UP plformat 0 0 Y
plotm3 10 0 0 UP plformat 0 0 X
plblock 0 0 0 50 50 50
plset PENDIA 1
plot3 0 0 0 UP plot3 50 50 50 DOWN #3D-diagonal through origin
plot3 0 50 0 DOWN #2D-diagonal in y=50 plane
plot3 50 0 0 DOWN #2D-diagonal in z=0 plane
```


Z
X
Y

plotm - move pen to new position in millimeters

parameter	type	units	description
x,y	float	mm	new pen position
linetype	int	—	see plot
returns:	void		

Description

plotm moves the pen to (**x,y**) in millimeters. The format of the line produced (if any) is determined by the **linetype** parameter - see plot.

See also

plot, plotr, plotrm, plu, plum

Examples

See also

plaxes plsmooth

plotm3 - move pen to new 3-D position in millimeters

parameter	type	units	description
x,y,z	float	mm	new pen position
linetype	int	—	see plot
returns:	void		

Description

plotm3 moves the pen to (**x,y,z**) in millimeters. The format of the line produced (if any) is determined by the **linetype** parameter - see plot.

See also

plot3, plotr3, plotrm3

plotr - move pen in user units, relative to current position

parameter	type	units	description
x,y	float	uu	relative new pen position
linetype	int	—	see plot
returns:	void		

Description

plotr moves the pen by (**x,y**) user units, relative to the current position. The format of the line produced (if any) is determined by the **linetype** parameter - see plot.

See also

plot, plotm, plotrm, plu, plum

Examples

See also

plfunc plot

plotr3 - move pen in 3-D user units, relative to current position

parameter	type	units	description
x,y,z	float	uu	relative new pen position
linetype	int	—	see plot
returns:	void		

Description

plotr3 moves the pen by (**x,y,z**) user units, relative to the current position. The format of the line produced (if any) is determined by the **linetype** parameter - see plot.

See also

plot3, plotm3, plotrm3

plotrm - move pen in millimeters, relative to current position

parameter	type	units	description
x,y	float	mm	relative new pen position
linetype	int	—	see plot
returns:	void		

Description

plotrm moves the pen by (**x,y**) millimeters, relative to the current position. The format of the line produced (if any) is determined by the **linetype** parameter - see plot.

See also

plot, plotm, plotr, plu, plum

Examples

See also

plfunc

plotrm3 - move pen 3-D in millimeters, relative to current position

parameter	type	units	description
x,y,z	float	mm	relative new pen position
linetype	int	—	see plot
returns:	void		

Description

plotrm3 moves the pen by (**x,y,z**) millimeters, relative to the current position. The format of the line produced (if any) is determined by the **linetype** parameter - see plot.

See also

plot3, plotm3, plotr3

plpage - eject a page or clear the window

parameter	type	units	description
—	void	—	—
returns:	void		

Description

plpage ejects a page (Postscript) or clears the plotting window (X-windows) and resumes plotting on a new page or in an empty window. All settings are reset to the values they had immediately after calling plinit, so if changed the linewidth, the scaling, clipping, character height etcetera, you'll have to set those again.

Examples

See also

plloop

plpgon - plot (filled) polygon

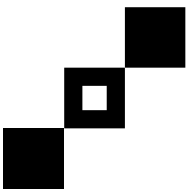
parameter	type	units	description
x,y	float*	uu	arrays with datapoint coordinates in user units
n	int	-	number of datapoints
linetype	int	-	line format used for outline (see plot)
returns:	void		

Description

plpgon plots a filled and outlined polygon.

Examples

```
plinit PS plpgon A4 50 50 "" ""

plrdx 4 0 10 10 0
plrdy 4 0 0 10 10
plpgon 4 DOWN 1

plmvorgm 10 10
plpgon 4 DOWN 0
plrdx 4 3 3 7 7
plrdy 4 3 7 7 3
plpgon 4 DOWN 1

plmvorgm 10 10
plrdx 4 0 10 10 0
plrdy 4 0 0 10 10
plpgon 4 DOWN 0
plrdx 4 3 7 7 3
plrdy 4 3 3 7 7
plpgon 4 DOWN 1
```



plpie - plot a simple pie diagram

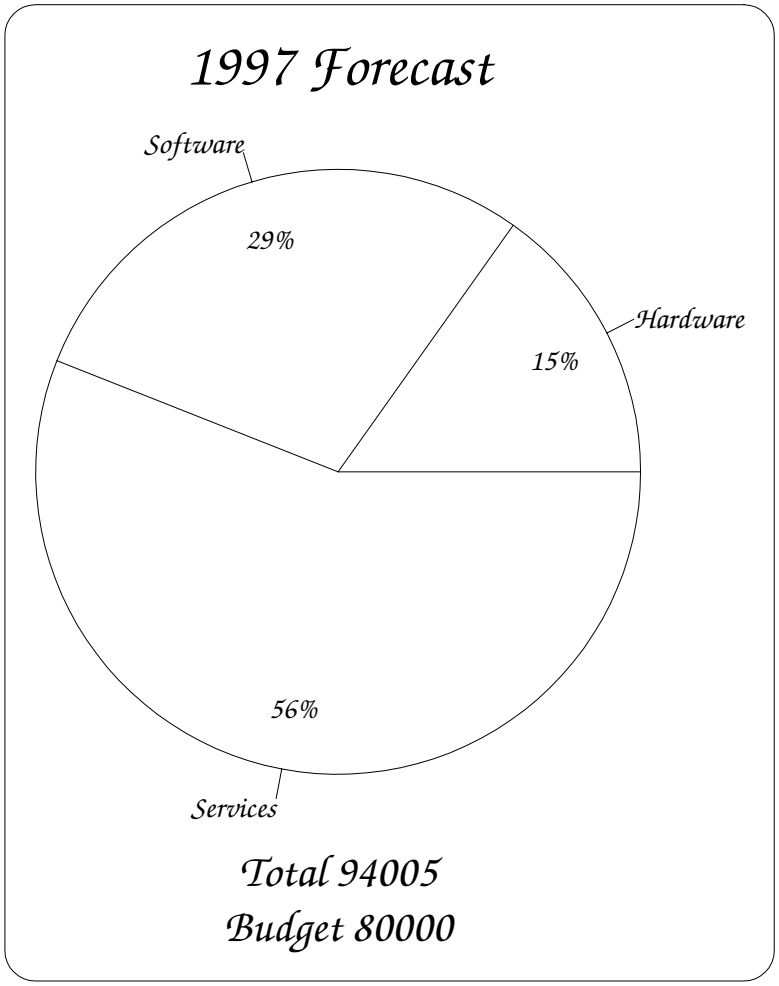
parameter	type	units	description
values	float*	any	relative sizes of the segments
n	int	-	number of segments to be plotted
labels	char**	-	array of 16-character strings, for labelling the segments
r	float	uu	pie radius
x,y	float	uu	position of the center of the pie
toptext	char*	-	text to be plotted on top of the pie
topfac	float	-	enlargement factor for the top text symbol size
bottext	char*	-	text to be plotted under the pie
botfac	float	-	enlargement factor for the bottom text symbol size
returns:	void		

Description

plpie plots a pie showing the relative contributions of values in **values** to their total. The segments are labeled with string of up to 16 characters stored in **labels** and two texts of variable size are placed over and under the pie. Usage in a simplot script needs a special treatment for the segment labels: these are not read like other arrays, but exactly **n** strings must be placed between the arguments **n** and **r**.

Examples

```
plinit PS plpie A4 0 0 ZCMI ""
n=3 plrdx n 123 234 454
plset HEIGHT 3
plpie n
      Hardware Software Services
      50
      100 100
      "1997 Forecast" 2
      "Total 94005\nBudget 80000" 1.5
plframe 5 5
```


1997 Forecast
Software
29%
Hardware
15%
56%
Services
Total 94005
Budget 80000

plpline - plot polyline

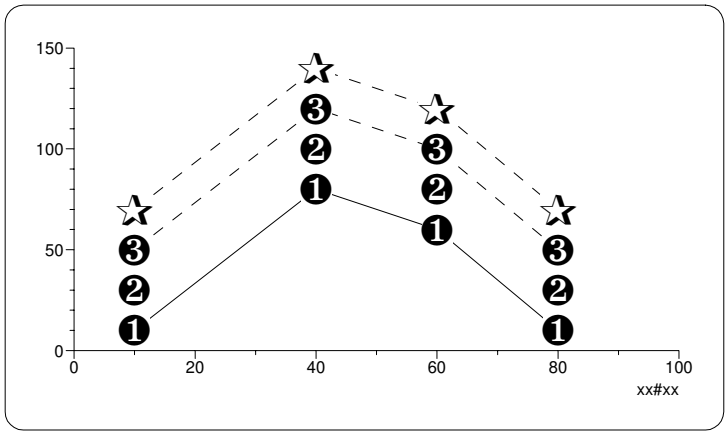
parameter	type	units	description
x,y	float*	uu	positions of points to be plotted
n	int	—	number of points
label	char*	—	string to be centered around each point
linetype	int	—	see plot
returns:	void		

Description

plpline plots the datapoints in x and y, by marking them with the centered string label. As Simplot centers strings in vertical direction assuming that at least one fullsized character such as capital X is present, lower case characters are normally not suitable for use in label strings. Also, asymmetric characters would look like not nicely centered. One should therefore use characters like X, O, or characters from the ZapfDingbats font.

Examples

```
plinit PS plpline A4 50 50 H ZD
plaxes 0 0 100 150 100 50 "xx##xx" "" ""
plset HEIGHT 5
n=4 plrdxy n 10 10 40 80 60 60 80 10
plpline n @#6 DOWN plmvorg 0 20
plpline n @#7 UP   plmvorg 0 20
plpline n @#8 DASH plmvorg 0 20
plpline n @P DASH
plframe 5 3
```



See also

plaxfit plpolv plpoly plshade plsmooth simplot program

plpline3 - plot 3-D polyline

parameter	type	units	description
x,y,z	float*	uu	positions of points to be plotted
n	int	—	number of points
label	char*	—	string to be centered around each point
linetype	int	—	see plot
returns:	void		

Description

plpline3 plots the datapoints in x, y and z, by marking them with the centered string label. As Simplot centers strings in vertical direction assuming that at least one fullsized character such as capital X is present, lower case characters are normally not suitable for use in label strings. Also, asymmetric characters would look like not nicely centered. One should therefore use characters like X, O, or characters from the ZapfDingbats font.

plpolar - plot a vector in polar coordinates in user units

parameter	type	units	description
size	float	uu	size (length) of the vector
angle	float	degr	direction of the vector with respect to the x-axis
linetype	int	—	see plot
returns:	void		

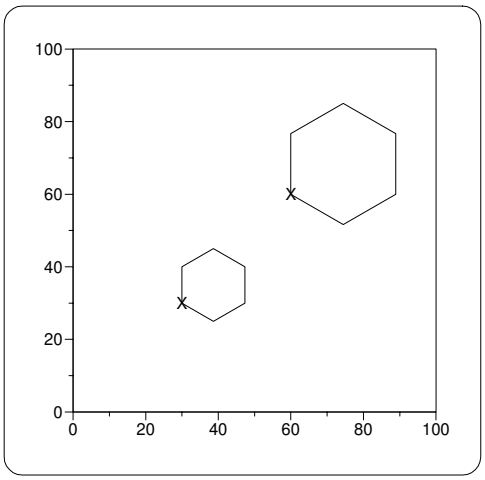
Description

plpolar draws a line of length **size** user units in the direction that makes an angle of **angle** degrees with respect to the current x-axis.

Examples

The following example draws two hexagons, one in user units, the other in millimeters:

```
plinit PS plpolar A4 50 50 "" ""
plaxes 0 0 100 100 60 60 "" "" ""
plrectm 0 0 60 60

# in user units:
plu 30 30 plformat 0 0 X
plpolar 10 90 DOWN plpolar 10 30 DOWN plpolar 10 -30 DOWN
plpolar 10 -90 DOWN plpolar 10 -150 DOWN plpolar 10 150 DOWN

# same one in millimeters and shifted:
plu 60 60 plformat 0 0 X
plpolarm 10 90 DOWN plpolarm 10 30 DOWN plpolarm 10 -30 DOWN
plpolarm 10 -90 DOWN plpolarm 10 -150 DOWN plpolarm 10 150 DOWN
plframe 5 3
```



plpolarm - plot a vector in polar coordinates in mm

parameter	type	units	description
size	float	mm	size (length) of the vector
angle	float	degr	direction of the vector with respect to the x-axis
linetype	int	—	see plot
returns:	void		

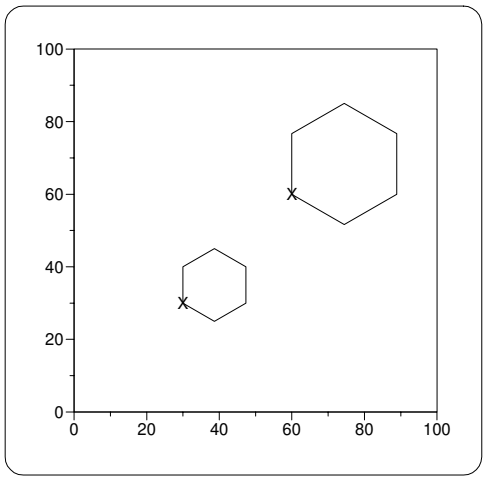
Description

plpolarm draws a line of length **size** millimeter units in the direction that makes an angle of **angle** degrees with respect to the current x-axis.

Examples

The following example draws two hexagons, one in user units, the other in millimeters:

```
plinit PS plpolar A4 50 50 "" ""
plaxes 0 0 100 100 60 60 "" "" ""
plrectm 0 0 60 60

# in user units:
plu 30 30 plformat 0 0 X
plpolar 10 90 DOWN plpolar 10 30 DOWN plpolar 10 -30 DOWN
plpolar 10 -90 DOWN plpolar 10 -150 DOWN plpolar 10 150 DOWN

# same one in millimeters and shifted:
plu 60 60 plformat 0 0 X
plpolarm 10 90 DOWN plpolarm 10 30 DOWN plpolarm 10 -30 DOWN
plpolarm 10 -90 DOWN plpolarm 10 -150 DOWN plpolarm 10 150 DOWN
plframe 5 3
```



See also

plpolar

plpolc - find the coefficients of the polynomial calculated by plpolf/plpoly

parameter	type	units	description
coeff	double*	—	array receiving polynomial coefficients
returns:	void		

Description

plpolc returns, in **coeff** the normal coefficients of a polynomial calculated by plpoly or plpolf. Note that the array of coefficients is to be declared *double.*

See also

plpoly plpolf

Examples

The following Simplot-program does not call plinit, because there is nothing to plot. It calculates the same polynomial as the example in the manual page

```
n=4
plrdx n 1991 1991.5 1992 1992.1
plrdy n 800 700 1600 2200
plpolf n 2
plpolc
```

prints the following:

```
exp coeff
  0 9801503565.8875
  1 -9844259.1106
  2 2471.8004
```

which tells that the equation of the polynomial is:

y = 9801503565.8875 - 9844259.1106 x + 2471.8004 x^2

plpolf - calculate a polynomial through a set of points

parameter	type	units	description
x,y	float*	—	datapoints
n	int	—	number of datapoints
k	int	—	degree of the polynomial
returns:	void		

Description

plpolf fits a polynomial of degree **k** through **n** points stored in the arrays **x** and **y**. If the NULL pointer is given instead of **x** or **y,** the values 0..n-1 are used instead. This routine is equivalent to plpoly, except that does not plot anything. After executing it, the function plpolv may be used to get the value of the polynomial in any x-position, or the function plpolc may be used to find the coefficients.

See also

plpoly, plpolc, plpolv, plpolc G.E.Forsythe, "Generation and use of orthogonal polynomials for data-fitting with a digital computer" SIAM Journal on Numerical Analysis, vol 5, page 74

Examples

See also

plpolc

plpolv - calculate y-value for an x-value of a polynomial

parameter	type	units	description
x	float	uu	x-value for which polynomial value is to be returned
returns:	float		

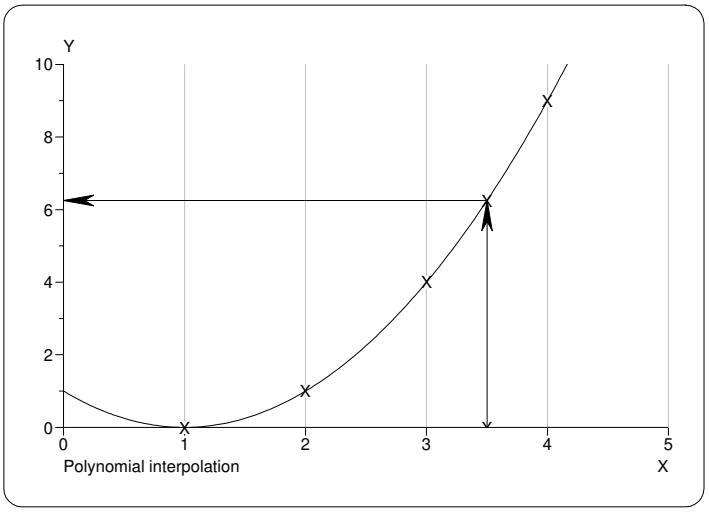
Description

plpolv returns the value at **x** of the polynomial calculated earlier by plpoly/plpolf.

Examples

The following program plot a polynomial and then instructs you how to interpolate on it by hand. The program itself interpolates by using:

```
#include <simplot.h>
int main() {
   float x[]={1,2,3,4};
   float y[]={0,1,4,9};
   float xmin=0,xmax=5,ymin=0,ymax=10,xpol=3.5,ypol;

   plinit(PS,"plpolv",A4,50,50,"","");
   plset(XGRID,TRUE);
   plaxes(xmin,ymin,xmax,ymax,100,60,"X","Y","Polynomial interpolation");
   plframe(5,3);
   plpline(x,y,4,"X",UP);
   plclip(xmin,ymin,xmax,ymax);
   plpoly(x,y,4,2,xmin,xmax,DOWN);
   ypol=plpolv(xpol);
   plot(xpol,ymin,UP);
   plformat(0,0,"X");
   plarrow(0,ypol,5,10,30,FALSE,FALSE,TRUE,FALSE,"");
   plot(xpol,ypol,UP);
   plformat(0,0,"X");
   plarrow(-xpol,0,5,10,30,FALSE,FALSE,TRUE,FALSE,"");
   plunclip();
   plset(PENDIA,2);
   plpscomt("start plframe");
   exit(0);
}
```


Y
10
8
6
4
2
0
0
1
2
3
4
5
X
Polynomial interpolation

plpoly - calculate and plot a polynomial through a set of points

parameter	type	units	description
x,y	float*	uu	datapoints
n	int	—	number of datapoints
k	int	—	degree of the polynomial
xmin,xmax	float	uu	plotrange
linetype	int	—	format of the line drawn
returns:	void		

Description

plpoly fits a polynomial of degree **k** through **n** points stored in the arrays **x** and **y,** and then plots it between **xmin** and **xmax**. If the NULL pointer is given instead of **x** or **y,** the values 0..n-1 are used instead. After executing it, the function plpolv may be used to get the value of the polynomial in any x-position, or the function plpolc may be used to find the coefficients.

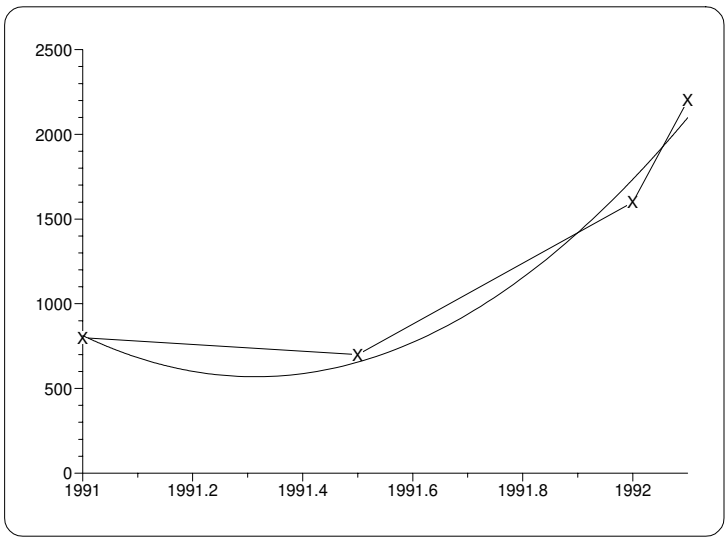
See also

plpolf, plpolc, plpolv

G.E.Forsythe, ”Generation and use of orthogonal polynomials for data-fitting with a digital computer” SIAM Journal on Numerical Analysis vol 5, page 74

Examples

```
n=4 xmin=1991 xmax=1992.1 ymin=0 ymax=2500
plinit PS plpoly A4 50 50 "" ""
plrdx n 1991 1991.5 1992 1992.1
plrdy n 800 700 1600 2200
plaxes xmin ymin xmax ymax 100 70 "" "" ""
plpline n X DOWN
plframe 5 3
plclip xmin ymin xmax ymax
plpoly n 2 xmin xmax DOWN
plpolc
```



Because of the plpolc statement this program prints:

exp	coeff
0	9801916765.45
1	-9844674.11426
2	2471.90463773

which tells that the equation of the polynomial is:

$y = 9801916765.45 - 9844674.11426\ x + 2471.90463773\ x^2$

Examples

See also

plpolv

plpscomt - insert a comment in the PostScript outputDfile

parameter	type	units	description
comment	char*	—	comment to be written in the PostScript output
returns:	void		

Description

plpscomt inserts a comment in the PostScript output. This is useful if one wants to do something with the PostScript output other than merely plot it. Just returns if plotter driver is X.

plrect - plot a (filled) rectangle in user units

parameter	type	units	description
x1,y1	float	uu	one vertex of the rectangle
x2,y2	float	uu	diagonally opposite vertex
returns:	void		

Description

plrect, if border-drawing is set (see plset), plots a rectangle between (x1,y1) and (x2,y2) in user units and, if filling is set (see plset), fills it with the current fill color. This is the default case since plinit sets the fill color to NoColor.

See also

plfcolor plset

Examples

See also

plfill plclip plarcr plformat plbox plfunc pltrace plarc plreserv plhist plloop simplot program plpolar

plrectm - plot a (filled) rectangle in mm

parameter	type	units	description
x1,y1	float	mm	one vertex of the rectangle
x2,y2	float	mm	diagonally opposite vertex
returns:	void		

Description

plrectm, if border-drawing is set (see plset), plots a rectangle between (x1,y1) and (x2,y2) in millimeter units and, if filling is set (see plset), fills it with the current fill color. This is the default case since plinit sets the fill color to NoColor.

See also

plfcolor plset

Examples

See also

plpolar

plrectr - plot relative rectangle

parameter	type	units	description
dx	float	uu	width of the rectangle
dy	float	uu	height of the rectangle
returns:	void		

Description

plrectr draws a rectangle from the current position to (current x+**dx,current** y+**dy**) in user units and fills it with the current fill color. Setting the fill color to NoColor inhibits filling.

See also

plfcolor

plrectrm - plot relative rectangle in mm

parameter	type	units	description
dx	float	mm	width of the rectangle
dy	float	mm	height of the rectangle
returns:	void		

Description

plrectrm draws a rectangle from the current position to (current x+**dx,current** y+**dy**) in millimeter units and fills it with the current fill color. Setting the fill color to NoColor inhibits filling.

See also

plfcolor

plreserv - move origin for concatenating sub-plots

parameter	type	units	description
xorg,yorg	float	mm	relative origin in a sub-plot
yhigh	float	mm	reserved height for next sub-plot
returns:	void		

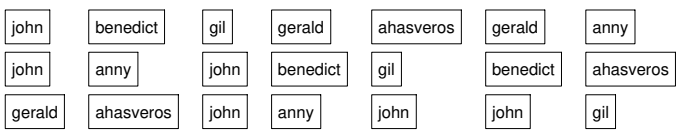
Description

plreserv is used to generate sub-plots and arranging these in the available space. It moves the origin so that the next sub-plot is plotted below previous one. If the sub-plot doesn't fit below previous one, a new column of sub-plots will be started. The coordinates of the origin in the reserved space will be at **xorg,yorg** millimeters realtive to the lower left corner of that space. The height of the reserved space will be **yhigh** millimeters. Use plclip or plclipm to arrange for blank margins around the plotting field.

Examples

The following example arranges variable width sub-plots in columns. In order to prevent the plots to overlap it is important that their contents do not reach x-positions more than xorg millimeters to the left of the x-position of the origin – preferably plotting should occur only to the right of the origin. Thus centering the strings this example would cause overlap:

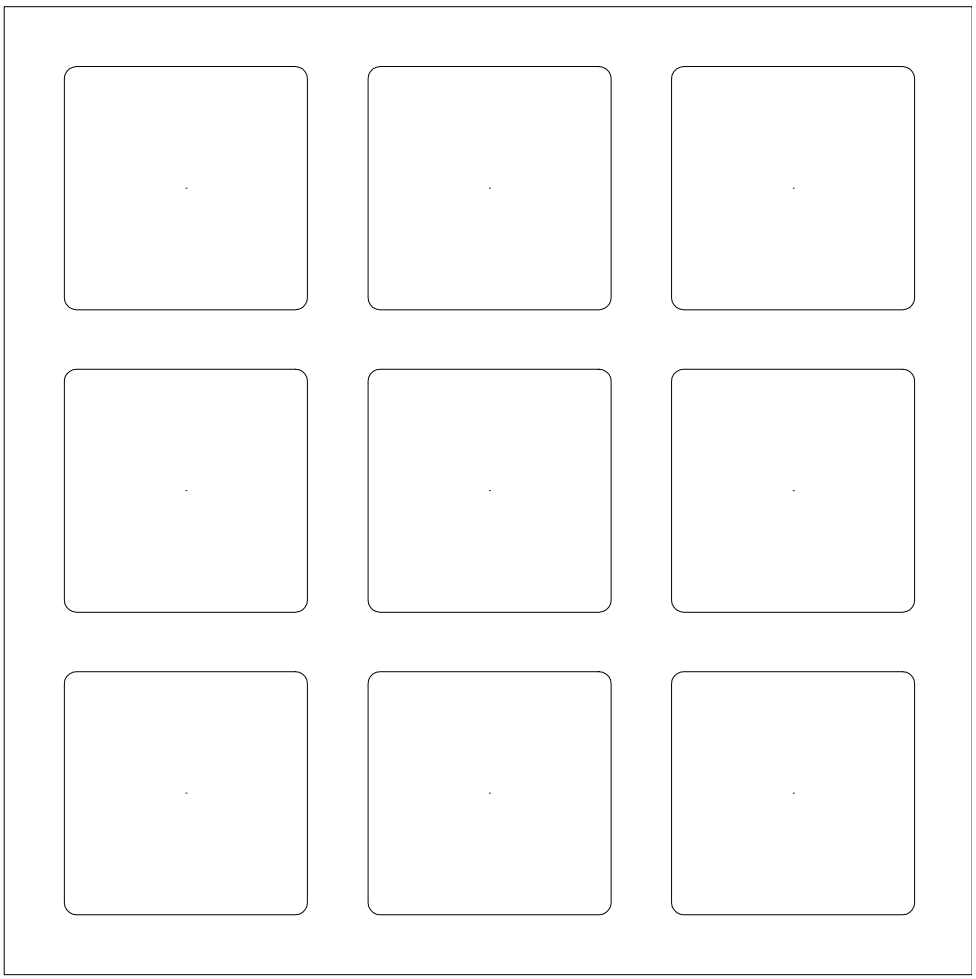
```
plinit PS plreserv1 A4 0 0 "" ""
plmvorg 10 10 plclip 0 0 130 25
plframe -1 0
plreserv 5 0 7 pltext john       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext john       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext gerald     plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext benedict   plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext anny       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext ahasveros  plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext gil        plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext john       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext john       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext gerald     plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext benedict   plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext anny       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext ahasveros  plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext gil        plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext john       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext gerald     plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext benedict   plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext john       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext anny       plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext ahasveros  plframe 1 0 plframe -1 0
plreserv 5 0 7 pltext gil        plframe 1 0 plframe -1 0
```



The next example fills available space regularly with boxes; note that the origin and the clipping window are marked afterwards – doing it first would make the plotting space unavailable for use by plreserv:

```
#include <simplot.h>
```

```
int main() {
   int i;
   plinit(PS,"plreserv2",A4,50,50,"","");

   plsave(); // save current origin etcetera
   plclip(0,0,160,160);
   for (i=0;i<9;i++) {
      plreserv(30,20,50);
      pldot(0,0,0);
      plframe(20,2);
      plframe(-1,0);
   }
   plunsave(); // restore original origin etcetera
   plformat(0,0,"X"); // mark origin
   plunclip();
   plrect(0,0,160,160); // plot clipping window
   exit(0);
}
```



See also

pltrace plhiss

plrotate - rotate user coordinate system

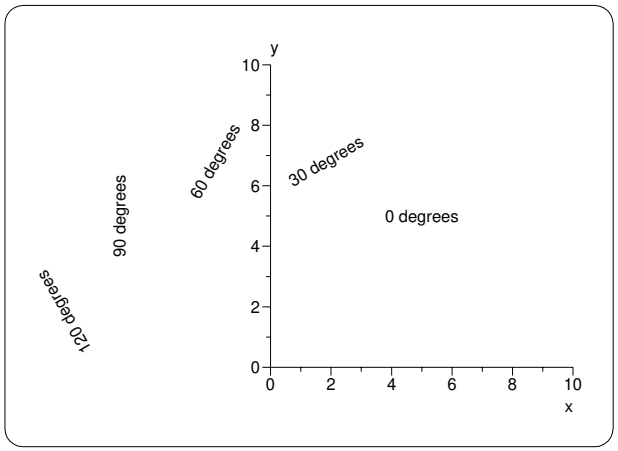
parameter	type	units	description
angle	float	degr	rotation angle
returns:	void		

Description

plrotate rotates the user coordinate system by **angle** degrees in counterclockwise direction.

Examples

```
plinit PS plrotate A4 50 50 "" ""
plaxes 0 0 10 10 50 50 x y ""
            plot 5 5 UP plformat 0 0 "0 degrees"
plrotate 30 plot 5 5 UP plformat 0 0 "30 degrees"
plrotate 30 plot 5 5 UP plformat 0 0 "60 degrees"
plrotate 30 plot 5 5 UP plformat 0 0 "90 degrees"
plrotate 30 plot 5 5 UP plformat 0 0 "120 degrees"
plframe 5 3
```



plsave - save current graphics context

parameter	type	units	description
—	void	—	—
returns:	void		

Description

plsave Save current position, origin and scaling parameters; use plunsave to return them.

Examples

See also

plreserv

plscale3 - defines scaling factors and origin for 3-D coordinate system

parameter	type	units	description
xmin,ymin,zmin	float	uu	value at the left/low/bottom end of the axes
xmax,ymax,zmax	float	uu	value at the right/high/top end of the axes
xlen,ylen,zlen	float	mm	lengths of the axes
returns:	void		

Description

plscale3 defines scaling factors and user origin for a 3-D coordinate system.

Examples

See also

plblock

plset - set new global values

parameter	type	units	description
name	int	—	global variable to bet set
value	float	any	new value for the variable
returns:	void		

Description

plset changes global values. The following table lists the names, default value and description of the global values:

global	default	description
ANGLE	0	plotting direction in degrees relative to the x-axis for symbols and texts. Changing the direction of the x-axis thus also changes the direction for plotting symbols.
CAPSTYLE	CapRound	set the linecapping style. Choices are CapNotLast (X only), CapButt, CapRound and CapProjecting.
CROSS	FALSE	if TRUE, axes will cross in (XCROSS,YCROSS) (in user units, see below), otherwise axes will cross in (xmin,ymin) (see plaxes).
DASHUNIT	1	unit in line_style definitions. If set to 0.1, then plot(...,...,1111) plot 0.1 mm down, 0.1 mm up, 0.1 mm down, 0.1 mm up, etcetera.
HEIGHT	2	set the symbol height in mm.
JOINSTYLE	JoinRound	set linejoining style. Choices are JoinMiter, JoinBevel and JoinRound.
LEVENTS	FALSE	in X, if TRUE, list events on standard output
LINESKIP	2	distance between lines in plformat in units of symbol height
OPAQUETEXT	FALSE	if TRUE, the background will be cleared before plotting text
PENDIA	0.1	set the pen diameter. For pen plotters PENDIA should be set to the physical pen diameter, for PostScript printers a value of 0 mm sets the linewidth to the minimum displayable linewidth. In X, a value of $¡=$ the pixel size draw lines with a width equal to the pixel size. This is usually much faster than larger widths
PLOTMODE	GXcopy	set the plotting mode. Can be GXcopy (the default and only possibility for PS) or GXxor (the default for X). Effective only for X-windows.
PRINT	FALSE	if TRUE, plmess will print its message to stdout in addition to its normal behaviour
RECTFILL	FALSE	if TRUE, rectangles will be filled with the current fill color
RECTSTROKE	TRUE	If TRUE, outlines of rectangles are drawn.
XANGLE	0	x-axis direction in degrees relative to initial x-direction.
YANGLE	90	y-axis direction in degrees relative to initial x-direction.
XCROSS	0	x-position in user units where y-axis crosses the x-axis. Effective only if CROSS!=0
YCROSS	0	y-position in user units where x-axis crosses the y-axis. Effective only if CROSS!=0
XIN	FALSE	if TRUE, scale marks and texts are plotted below the x-axis, if zero, they are plotted above the x-axis
YIN	FALSE	if TRUE, scale marks and texts are plotted to the left of the y-axis, if zero, they are plotted to the right of the y-axis
XGRID	FALSE	if TRUE, gridlines are plotted through number bearing scale marks of the x-axis
YGRID	FALSE	similarly for the y-axis
XLOG	FALSE	if TRUE, an x-axis with logarithmic divisions will be plotted and the coordinate system will be defined logarithmically in the x-direction.
YLOG	FALSE	similarly for the y-axis
XSKIP	0	if nonzero, numbers are placed along the x-axis every **value** scale marks; if zero, numbers are placed automatically.
YSKIP	0	similarly for the y-axis
XMARK	0	if nonzero, scale marks are placed every **value** user units along the x-axis. If zero, scale marks are placed automatically.

YMARK	0	similarly for the y-axis
XSQUAR	FALSE	if TRUE, numbers are plotted perpendicularly to the x-axis, if zero, they are placed parallel to the x-axis.
YSQUAR	TRUE	similarly for the y-axis

Examples

See also

plfill plclip plarcr plbox plformat plblock plfunc plaxes plarc plpolv plot plarrow plpline plsmooth plloop simplot program plpie

plshade - hatch polygon

parameter	type	units	description
*xvert,*yvert	float*	uu	vertices of the polygon
nvert	int	—	number of vertices
dir	float	degr	direction of shading lines
wmesh	float	mm	distance between shading lines
linetype	int	dash	see plot
returns:	void		

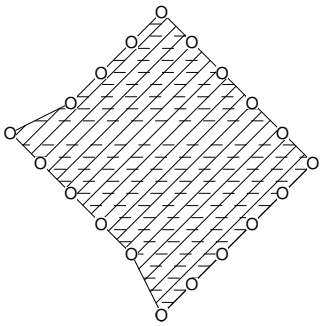
Description

plshade shades a polygon determined by **nvert** vertices whose coordinates (in user units) are stored in **xvert** and **yvert**. The polygon outline is not drawn: use plpline if you need it. The direction of the shading lines is **dir** degrees with respect to the current x-axis. The distance between the lines is **wmesh** millimeters. **linetype** (see plot) determines the form of the shading lines.

Examples

The following script draws a polygon with plpline and hatches it with solid as well as dashed lines in two directions.

```
n=20
plrdx n 0  5 10 15 20 25 20 15 10  5  0  -5 -10 -15 -25 -20 -15 -10  -5  0
plrdy n 0  5 10 15 20 25 30 35 40 45 50  45  40  35  30  25  20  15  10  0
plinit PS plshade A4 50 50 "" ""
plshade n 45 2 DOWN
plshade n  0 2 DASH
plpline n "O" DOWN
```



Bugs

Concave polygons give unpredictable results when shading lines cross the border more than twice.

See also

simplot program

plsize - find length of a string plotted in the current font

parameter	type	units	description
s	char*	—	characterstring
plsize	float	mm	length of **s**
returns:	float		

Description

plsize returns the length in millimeters of string **s,** if plotted in the current font. This is the distance between initial and final penposition when the string would be plotted with pltext.

Examples

plsmooth - performs a k-point smoothing on an array

parameter	type	units	description
array	float*	uu	array to be smoothed
n	int	—	length of array
k	int	—	number of points of the smooth (must be odd)
returns:	void		

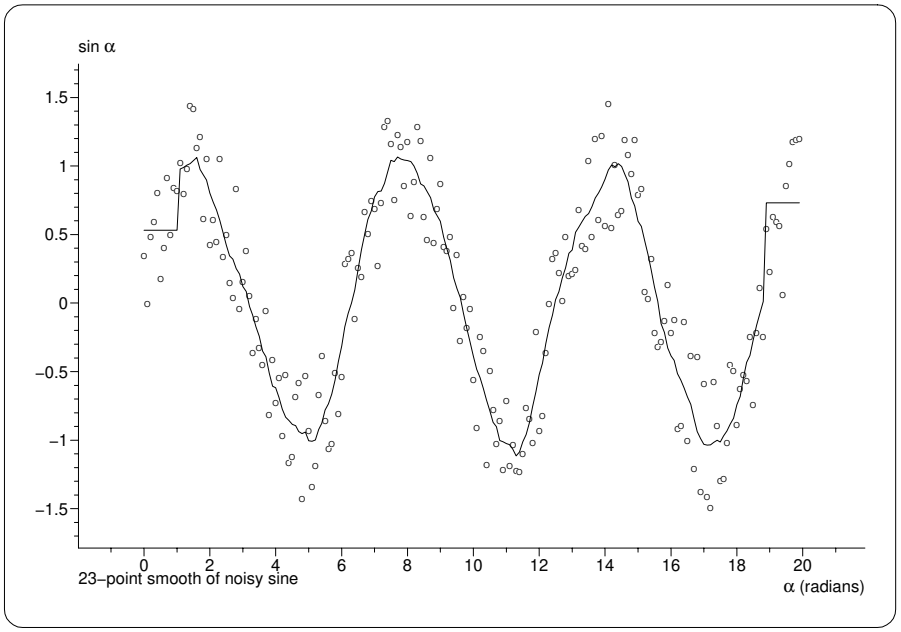
Description

plsmooth applies a **k**-point (**k** must be odd and $¡$ **n**) quartic-cubic smooth to the **n** values in **array**. The first and last **k**/2 points in a are unaffected.

see c.g.enke, t.a.nieman, anal. chem. vol 48, page 705a

Examples

```
/* __________ plsmooth __________
  plotting a noise modulated sine with and without smoothing
  features: use of plaxfit, plpline, plformat
            smoothing
*/
#include <stdlib.h>
#include <math.h>
#include <simplot.h>
#define NPOINTS 200
#define RANGE 23
int main() {
   float x[NPOINTS],y[NPOINTS];
   int i;
   plinit(PS,"plsmooth",A4,50,50,"","");

   for (i=0;i<NPOINTS;i++)     // loop generates the noise-modulated sine data
   { x[i]=i/10.;
      y[i]=sin(x[i])+(float)rand()/RAND_MAX-.5;
   }
   plaxfit(x,y,NPOINTS,10,130,80,"@a@ (radians)","sin @a@"," ");
   // fits axes to the data
   plotm(0,0,UP);
   plformat(.5,-.5,"\n%d-point smooth of noisy sine",RANGE);
   // report smoothing range
   plset(HEIGHT,1);                // make symbols small giving dots
   plpline(x,y,NPOINTS,"O",UP);    /* plot unsmoothed data.
                                        mark with O's, no connecting line */
   if (plget(PLOTTER)!=PS)
      plcolor(Red);
   plsmooth(y,NPOINTS,RANGE);      // smooth the data
   plpline(x,y,NPOINTS,"",DOWN);   // and plot again. no marking symbols
   plframe(5,3);
   plend();
   exit(0);
}
```


sin α
1.5
1
0.5
0
−0.5
−1
−1.5
0
2
4
6
8
10
12
14
16
18
20
23–point smooth of noisy sine
α (radians)

plsort - sort two arrays with xy data

parameter	type	units	description
x	float*	—	array to be sorted
y	float*	—	array that follows
n	int	—	number of elements
returns:	void		

Description

plsort sorts x in ascending (n $¿$ 1) or descending (n $¡$ -1) order y is sorted concurrently with x

plsymbol - draw a symbol

parameter	type	units	description
isym	int	—	ASCII value of symbol to be drawn
returns:	void		

Description

plsymbol plots symbol with ASCII value isym with its reference point at the current position.

pltext - plot text, leaving the pen at the end

parameter	type	units	description
string	char*	—	string to be drawn
returns:	void		

Description

pltext is equivalent to plformat(.5,.5,**string**) except that

- **string** may not contain format specifications that need extra parameters
- the pen is left at the end of the string

Examples

See also

plreserv plot

pltext0 - plot text like pltext, but return to position

parameter	type	units	description
string	char*	—	string to be drawn
returns:	void		

Description

pltext0 plots **string** like pltext, but returns to the starting position

Examples

See also

plot

pltrace - traces an (implicit) 2-dimensional function

parameter	type	units	description
ident	char*	—	marker to be plotted with the curve
func(float,float)	float*	uu	function of x and y to be traced
xmin,ymin,xmax,ymax	float	uu	in this window
resol	float	%	resolution relative to visible field, default 0 = 1%
grid	float	%	search grid's line distance relative to visible field, default 0 = 1%
linetype	int	—	see plot
returns:	void		

Description

pltrace traces a function in the rectangle xmin,ymin,xmax,ymax. For example:

func(x,y) = x^2 + y^2 - 25

plots a circle with a radius of 5 user units and the centre at the origin. But, of course, pltrace is not the most efficient method for plotting a circle. **resol** is the resolution of the plot in mm, that is: the curve will be plotted in steps of **resol** mm long. It can not be set to less than 0.05 mm. **grid** is the line-distance of an imaginary grid placed over the plotting area. It is used for searching starting points of (sub)curves. If a (sub)curve happens to lie completely within a "pixel" of the grid, then it will not be found.

Examples

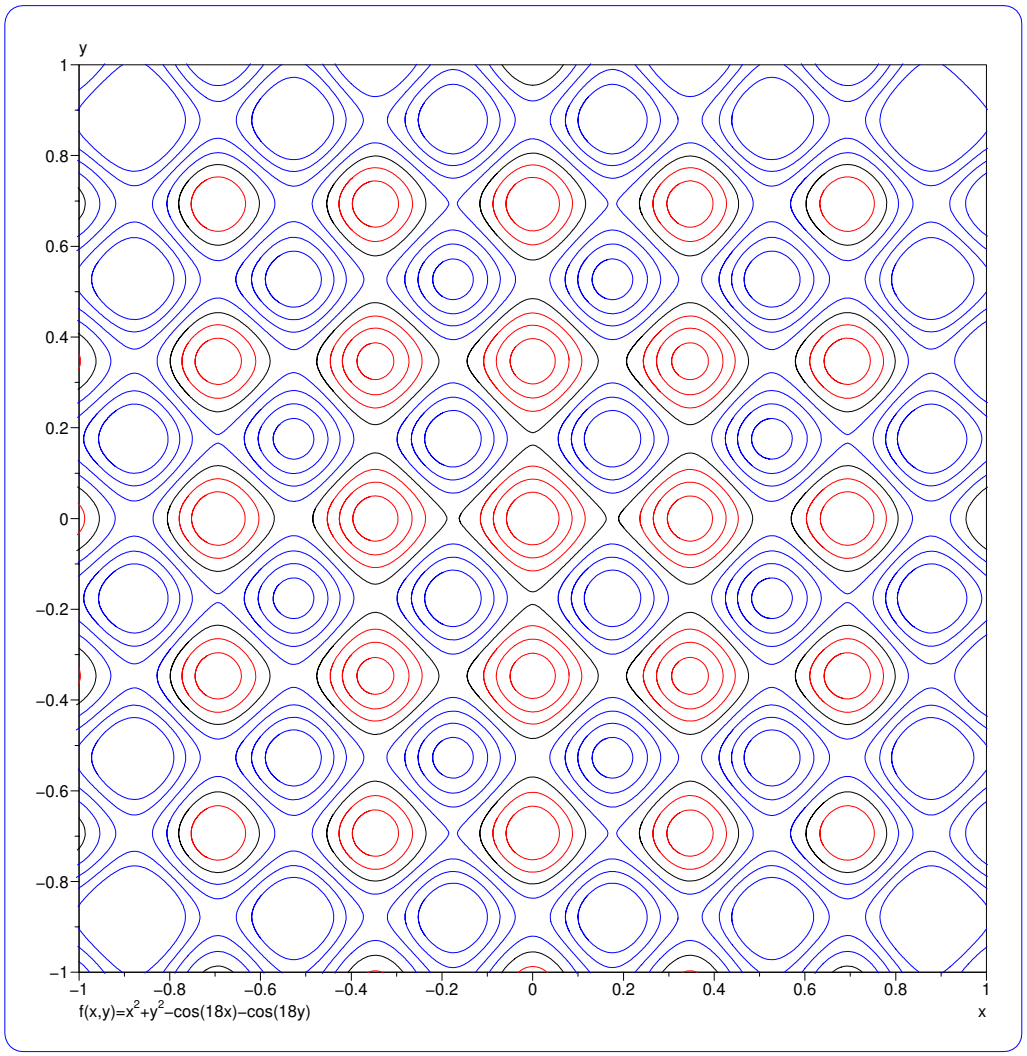
The program pltrace1.c plots the function

f(x,y)=x^2+y^2-cos(18x)-cos(18y)

By adjusting FIRST, LAST and STEP you can easily adapt it to plot other cross-sections above and below the plane of the paper.

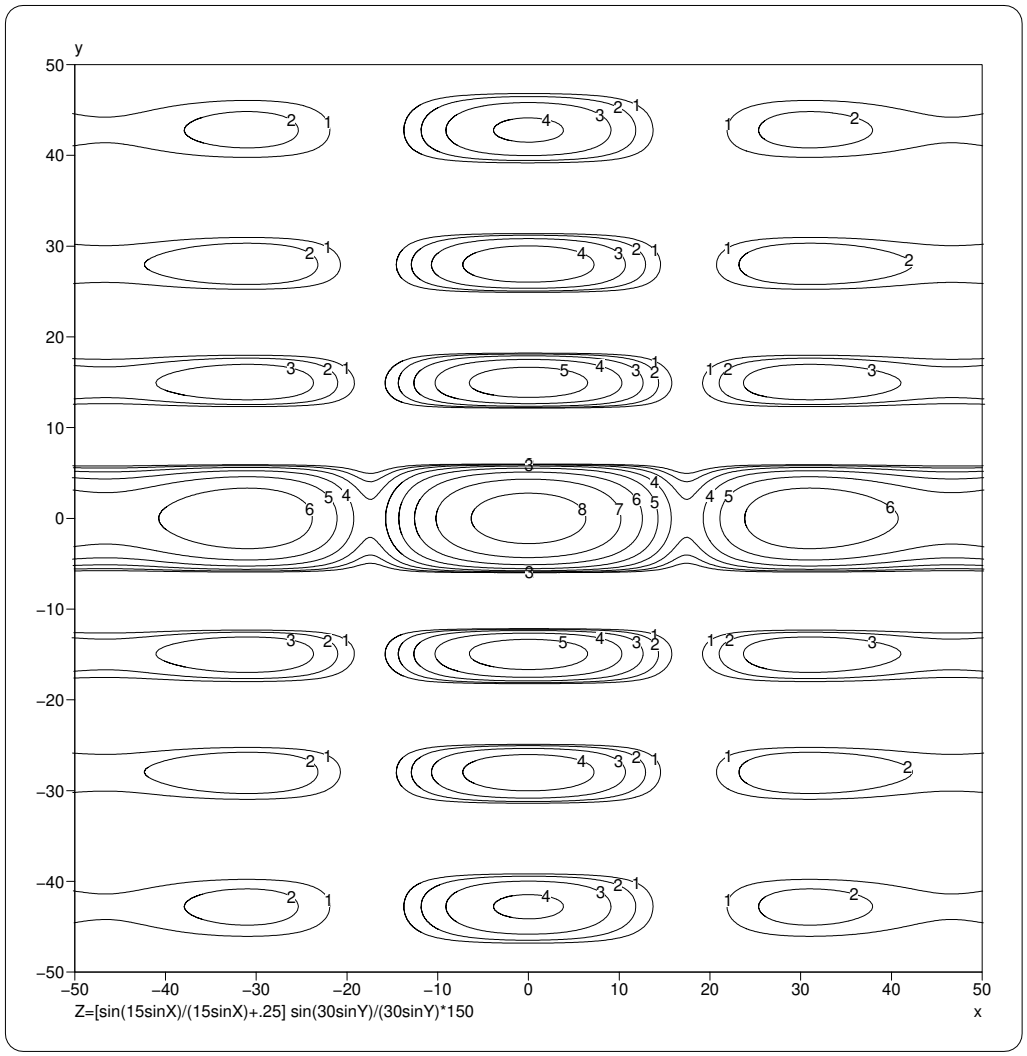
```
// pltrace1 - traces contours of the function:
// f(x,y)=x^2+y^2-cos(18x)-cos(18y)

#include <stdlib.h>
#include <math.h>
#include <simplot.h>
#define RESOL .3
#define GRID 2
#define FIRST -2
#define LAST 2
#define STEP .5
float z;

float func(float x,float y) {
   float r = x*x+y*y-cos(18*x)-cos(18*y)-z;
   return r;
}

int main() {
   extern float func(float,float);

   plinit(PS,"pltrace1",A4,30,30,"","");
   plaxes(-1,-1,1,1,150,150,"x","y",
          "f(x,y)=x^2+y^2-cos(18x)-cos(18y)");
   plrect(-1,-1,1,1);
```

```
   for (z=FIRST;z<=LAST;z+=STEP) {
      if (z<-0.1)
         plcolor(Red);
      else
         if (z>0.1)
            plcolor(Blue);
         else
            plcolor(Black);
      pltrace("",func,-1,-1,1,1,RESOL,GRID,DOWN);
   }
   plframe(5,3);
   exit(0);
}
```



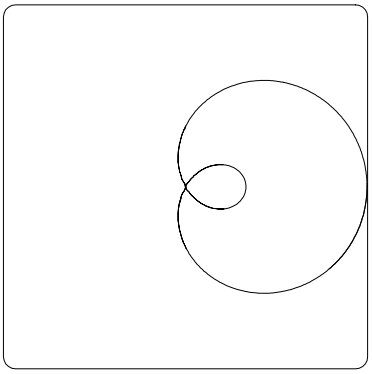

```
// pltrace2 - traces contours of the function:

//      sin(15*sin(x))          sin(30*sin(y))
//     (-------------- + .25) * -------------- * 150
//          15*sin(x)               30*sin(y)

#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <simplot.h>
#define XMIN -50
#define YMIN -50
```

```
#define XMAX +50
#define YMAX +50
#define XL   150
#define YL   150
#define RESOL .3
#define GRID   2
float z;

float efsin(float xx,float yy) {
   float x,y;
   x = 15*sin(xx*M_PI/180);
   if (fabs(x)<.1)
      x = .1;
   y = 30*sin(yy*M_PI/180);
   if (fabs(y)<.1)
      y = .1;
   return (sin(x)/x+.25)*sin(y)/y*150-z;
}

int main() {
   char ident[2]=" ";
   extern float efsin(float,float);
   int i;

   plinit(PS,"pltrace2",A4,30,30,"","");
   plaxes(XMIN,YMIN,XMAX,YMAX,XL,YL,"x","y",
          "Z=[sin(15sinX)/(15sinX)+.25] sin(30sinY)/(30sinY)*150");
   plrect(XMIN,YMIN,XMAX,YMAX);
   plset(OPAQUETEXT,TRUE);
   for (i=0,z=1;i<=8;i++) {
      ident[0]=i+'1';
      pltrace(ident,efsin,XMIN,YMIN,XMAX,YMAX,RESOL,GRID,DOWN);
      z *= 2;
   }
   plframe(5,3);
   plend();
   exit(0);
}
```

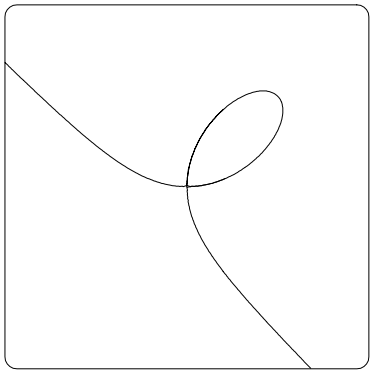



```
//  ___________ pltrace3 ___________
// plotting functions with pltrace
// each curve is placed in a separate square frame
// features: distribution of plots over the plot-page with plreserv
//           use of plframe and plclip

float a,b,asq,bsq;
#include <values.h>
#include <string.h>
#include <simplot.h>
#include <math.h>
#define S(x) ((x)*(x))

float snail (float x,float y) {
   return S(x*x+y*y-a*x)-bsq*(x*x+y*y);
}
float cassin(float x,float y) {
   return S(x*x+y*y+asq)-4*asq*x*x-bsq*bsq;
}
float folium(float x,float y) {
   return x*x*x+y*y*y-3*a*x*y;
}
float astrid(float x,float y) {
   return pow((x*x),.333333)+pow((y*y),.333333)-pow((a*a),.333333);
}
float concho(float x,float y) {
```

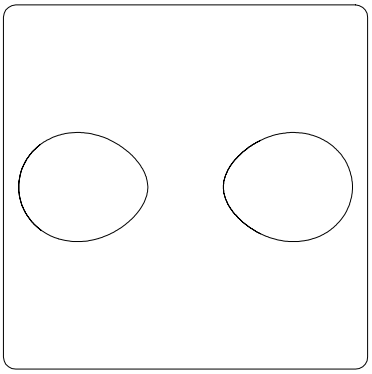
```
   return S(y-a) * (x*x+y*y)-bsq*y*y;
}
float sinus (float x,float y) {
   return y-a*sin(b*x);
}

void curve(float (*func)(float,float),float p,float q,char *name,char *form) {
   a=p;
   b=q;
   asq=a*a;
   bsq=b*b;
   plframe(-1,0);
   plreserv(55,50,85);
   plu(0,0);
   pld(0,0);
   plframe(30,2);           //box around origin
   plu(0,-35);
   plformat(0,-.5,"%s\n%s\na=%g b=%g",name,form,a,b);
   pltrace("",func,-30,-30,30,30,0,0,DOWN);
}

int main() {
   plinit(PS,"pltrace3",A4,0,0,"","");
   plclip(0,20,185,277);

   // plot several frames (r*r cm) and a curve in it

   curve(snail ,20,10  ,"snail-line",            "(x^2^+y^2^-ax)^2^=b^2^(x^2^+y^2^)");
   curve(cassin,20,19  ,"cassini's ovals",       "(x^2^+y^2^+a^2^)^2^=4a^2^x^2^+b^4^");
   curve(cassin,20,20  ,"bernoulli's lemniscate","(x^2^+y^2^)^2^=a^2^(x^2^-y^2^)");
   curve(folium,10, 0  ,"descartes' folium",     "x^3^+y^3^=3axy");
   curve(astrid,30, 0  ,"astroid",               "x^2/3^+y^2/3^=a^2/3^");
   curve(concho,10,20  ,"nicomedes' conchoid",   "(y-a)^2^(x^2^+y^2^)=b^2^y^2^");
   exit(0);
}
```



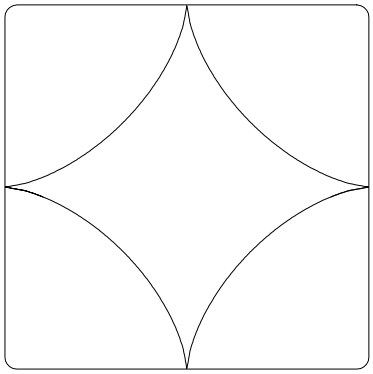
snail–line
$(x^2+y^2-ax)^2=b^2(x^2+y^2)$
a=20 b=10



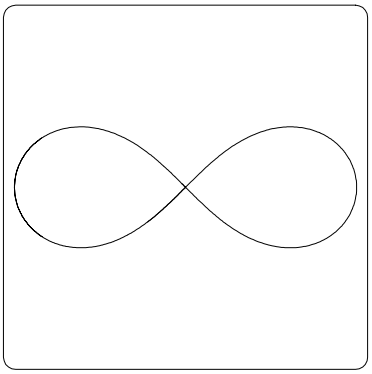
descartes' folium
$x^3+y^3=3axy$
a=10 b=0



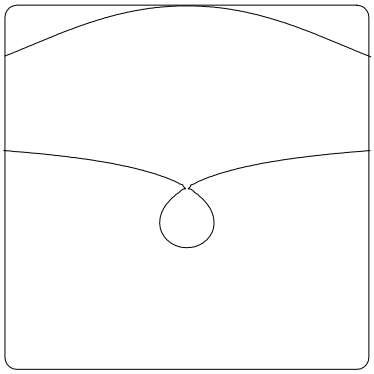
cassini's ovals
$(x^2+y^2+a^2)^2=4a^2x^2+b^4$
a=20 b=19



astroid
$x^{2/3}+y^{2/3}=a^{2/3}$
a=30 b=0



bernoulli's lemniscate
$(x^2+y^2)^2=a^2(x^2-y^2)$
a=20 b=20



nicomedes' conchoid
$(y-a)^2(x^2+y^2)=b^2y^2$
a=10 b=20

plu - move to a position without plotting

parameter	type	units	description
x,y	float	uu	position to which the pen is moved
returns:	void		

Description

plu moves the pen to (**x,y**) without plotting anything. It is equivalent to plot(**x,y,UP**).

See also

plot, pld

Examples

See also

plfill plclip plarcr plbox plformat plframe pltrace plfont plreserv plloop simplot program plpolar

plum - move to a position in mm without plotting

parameter	type	units	description
x,y	float	mm	position to which the pen is moved
returns:	void		

Description

plum moves the pen to (**x,y**) in millimeter units, without drawing anything. It is equivalent to plotm(**x,y,UP**).

See also

plotm, plu

plunclip - remove current clipping rectangle

parameter	type	units	description
—	void	—	—
returns:	void		

Description

plunclip removes the current clipping rectangle and replaces it with the default one, restricting drawing to the paper size set with plinit.

See also

plclip

plunsave - restore (unsave) graphics context saved with plsave

parameter	type	units	description
—	void	—	—
returns:	void		

Description

plunsave Return previous position, origin and scaling parameters

Examples

See also

plreserv

plxbar - plot a horizontal error bar centered around the current point

parameter	type	units	description
sd	float	uu	distance of the crosses to the center
range	float	uu	total length of the bar
returns:	void		

Description

plxbar plots a horizontal error bar centered around the current position. The total length of the bar is **range** and is meant to indicate the range of the data represented by the bar. Two crosses with a width of 2 mm are drawn at equal distances **sd** left and right of the starting point. The distance between the crosses is thus two times **sd**. They are supposed to indicate the standard deviation in the data represented by the error bar.

plybar - plot a vertical error bar centered around the current point

parameter	type	units	description
sd	float	uu	distance of the crosses to the center
range	float	uu	total length of the bar
returns:	void		

Description

plybar plots a vertical error bar centered around the current position The total length of the bar is **range** and is meant to indicate the range of the data represented by the bar. Two crosses with a width of 2 mm are drawn at equal distances **sd** above and below the starting point. The distance between the crosses is thus two times **sd**. They are supposed to indicate the standard deviation in the data represented by the error bar.