Patrick ROBERT

RoGraLib

a **Gra**phic **Lib**rary
to produce high quality plots
from C or Fortran code

ScientiDev
scientific software development
www.scientidev.fr

ACKNOWLEDGMENTS

The author is deeply indebted to F. Mottez who spent a lot of time to check and use this library. Being an intensive user, many thanks for his discovery of various bugs or unsatisfy process, as well as his usefull suggestions of improvement.

FOREWORD

This document content the Rogralib manual, and allow the user to understand how Rogralib does works. The knowledge of Fortran 77 or Fortran 90/95 language is required.

After a short introduction, Chapter 2 show how make simple plots, and content many examples to understand how Rogralib works.

Chapter 3 is dedicated to colors and images, while chapter 4 shows more advanced plots. In all example, the source code is given, as the corresponding plot deduced from the generated Postscript files. This can be very useful to help the user to write new program, and show how Rogralib can be used, and showing various possibilities of Rogralib.

Contents

Chapter 1 : INTRODUCTION

1.1 Short history

The firs version of **Rogralib** has been created in February 1986, from different tools already developed to get the possibility of making plots which be not dependent of the Operating System of the computer, and not dependent of any private graphical libraries.

Afterwards, the library has been completed and updated by many others subroutines and programs, to reach the today version which offer to the developer a skill tools to produce high quality plots, where the user can be at each time the master of what he want and what he do. If a special routine is requested, he can write it himself from existing modules.

The table 1.1 given hereafter briefly summarize the various states of evolution of **Rogralib**.

The **Rogralib** library is a living product. It has been extensively developed during the last 20 years, and continue to evolve. This means mainly that new subroutines will be regularly added, providing new facilities. In all case, ascendant compatibility is guaranteed.

The following books was used for the Postscript driver development: [1], [2], [3].

1.2 Introduction to Rogralib

This library is written in Fortran 77, portable on any machine and O.S. where a F77 compiler is available, as **gfortran**. To use this library, the user must write a program in Fortran (f77, f90, f95...) or C. This program include a suit of CALL to successive subroutines library (see chapter 2). The result is a Postscript file, which can be send to a printing device or used in any other software which accept Postscript files in input. The Postscript file can be also converted in pdf, png etc. by GoshtScript (see section 7.3) This library can also be compiled in Fortran 90, and used from F90 programs. C code is also available.

The **Rogralib** library is a complete tools to make high quality black and white or color plots (from 256 to 16 Millions of colors). Default paper type is A4, but other type are available (A3-A0,letter, etc).

The various available subroutines allows to do everything that you want. The programmer can use the default parameters to save time and produce quickly a plot, but he can also keep control of any parameters, and produce a plot having all the desired object (thickness of the line, size of font type, plot resolution, image importation, length and position of figure graduation sticks, and so on).

If desired subroutines does not exists, the user can write it, corresponding to his needs, and can use to write it facilities provided by others existing utility subroutines. In this way, suggestion and new subroutines of common interest can be sent to the author, for check and integration in a complementary package.

In this part, we will see first how use quickly **Rogralib**, and then how use more sophisticated subroutines do to “what we want”. In this way, the full list of available subroutines are given in chapter 6.

Table 1.1: *Successive evolution of* **Rogralib** *versions*

	History of **Rogralib** software Copyright (c) 1986-2022, Patrick ROBERT, Scien-tiDev	
V1.0	Initial version from archives:	Fev 1986
V2.0	Hierarchy and Structure of call :	Fev 1986
V3.0	Creation of the Postscript driver:	Mar 1990
V4.0	General revision and codified names of modules:	Mai 1993
V5.0	Introduction of standard Postscript fonts PS :	Mar 1995
V6.0	New modules of calculation and drawing :	Dec 1995
V7.0	Redefinition of dedicated categories and stucturation of the tree in modules :	Dec 1996
V7.1	- V7.4 improvements and various connections	
V7.5	Extension 8 bits of the standard PS fonts :	Dec 1997
V7.6	New modules of drawing :	Mar 1998
V7.7	Optimization for text writing :	Nov 1998
V8.0	Management of the proportional PS fonts :	Oct 1999
V8.1	General revision of common and simplifications, Suppression of the interface in pixels Fully vectorial drawing, management of the images :	Nov 1999
V8.2	Introduction of the paper size A3,A2,A1 et A0 :	Avr 2000
V8.3	Corrections of some bugs:	Dec 2001
V8.4	Correction of some bold fonts problems:	Dec 2003
V8.5	Corrections of some bugs:	Avr 2005
V8.6	New adjustment for the calculation of graduations, and for the high close values :	Aou 2005
V8.7	Compatibility for Ghostview drivers :	Nov 2005
V8.8	Verification FORESYS and introduction of some color maps	Jan 2006
V8.9	Compatibility for G77 PC Windows :	Feb 2006
V9.0	25 new color maps, possibility to define a new color map :	Jun 2006
V9.1	Deletion of unfinished drivers other than Postscript, possibility of dashed lines, Insensitive Format case, new times routines Revision of categories :	Oct 2007
V9.2	Correction calculus Page Bounding Box	Nov 2007
V9.3	Dedication power function (x**n) for portability	Nov 2007
V9.4	Bug corrections on cfiggra_ and PS driver	May 2008
V9.5	New calendar functions, same code for SUN/WIN/Linux, activation of dfigout_, check limit values in PS driver, optimize distances graduation to axes, etc	June 2008
V9.6	Full gfortran compatible	Jun 2010
V9.7	Introduction of logarithmic graduations	Sep 2011
V9.8	Interpolation of figure overtaking, errors code update, add of ppagpar_, uchkNaN_,cclhsb_,ppagsun_ remove fgetc function in reading bmp files,	Jan 2016
V9.9	best management of figure overtaking, adding rou-tines cfigout_, uppagcha_, gfiglim_, introduction of new character fonts, possibility to mixt latin/grec with ppagstr_ and pfigstr_, adding gchapos_, drecver_, recto-verso option,	Jul 2020
V10.0	'Introduction paper size A0-A9, B, C etc.	Nov 2022

Chapter 2 : MAKE A BASIC PLOT

2.1 Generalities

2.1.1 Open and close postscript file

The goal of a program using **Rogralib** is to produce a PostScript file corresponding to what the user wants to draw. A such program is a suit of call to various **Rogralib** subroutines. Of course, any other Fortran instructions can be inserted at any time, but for more visibility of the source code, it is recommended to separate what is data plot or all what is necessary to prepare the object to be plotted, or other computations.

In any case, all **Rogralib** calls must be placed into the two following instructions:

```
call dopegra_(7,'toto.ps')
....
Rogralib commands
...
call dclogra
```

The call to **dopegra_** open the graphical file, here on file unit 7 (it can be another), and create the toto.ps Postscript file (or another name), with header and default parameters. The call to **dclogra_** close this file. All **Rogralib** calls must be placed into this two basic calls.

2.1.2 Coordinates and objects definition

All plots are done on a paper sheet of a given size. This size is defined from the media used (A4, A3,...A0 etc.) On this sheet, we define a page, with position and size. Inside this page, we can define a figure, with position and size. At the top and the bottom of this page, we can define a header and a foot, with position and size.

Then various objects can be plotted on the page, or on the figure. All plot on page are defined with cm as units. All plots on figure are defined with the units of axis. Figure 2.1 summarize these definitions.

2.1.3 Plotting objects

As we will see on section 2.4, plotting objects are done by calling appropriate subroutines. All parameters used by these subroutine have a default value,as, for instance, the page size, the page header font, size and position, the size and position of the figure label and graduations, etc. The complete list of default values is given in section 5.1. Nevertheless, these values can be changed at any time by the 'define' subroutines (section 5.2) .

2.2 Example of graphical program

The simple program given hereafter in listing 2.1 show the principles of a rogralib program. This is the program used to produce figure 2.1. It show how define a figure and plot a curve on it.

First instructions defines the curve to be plotted. Then we open the Postscript file and plot page frame and header &foot. Then a block of instruction define the figure attributes, while a second block plots all the figure objects on the page. Plot is terminated by closing the Postscript file.

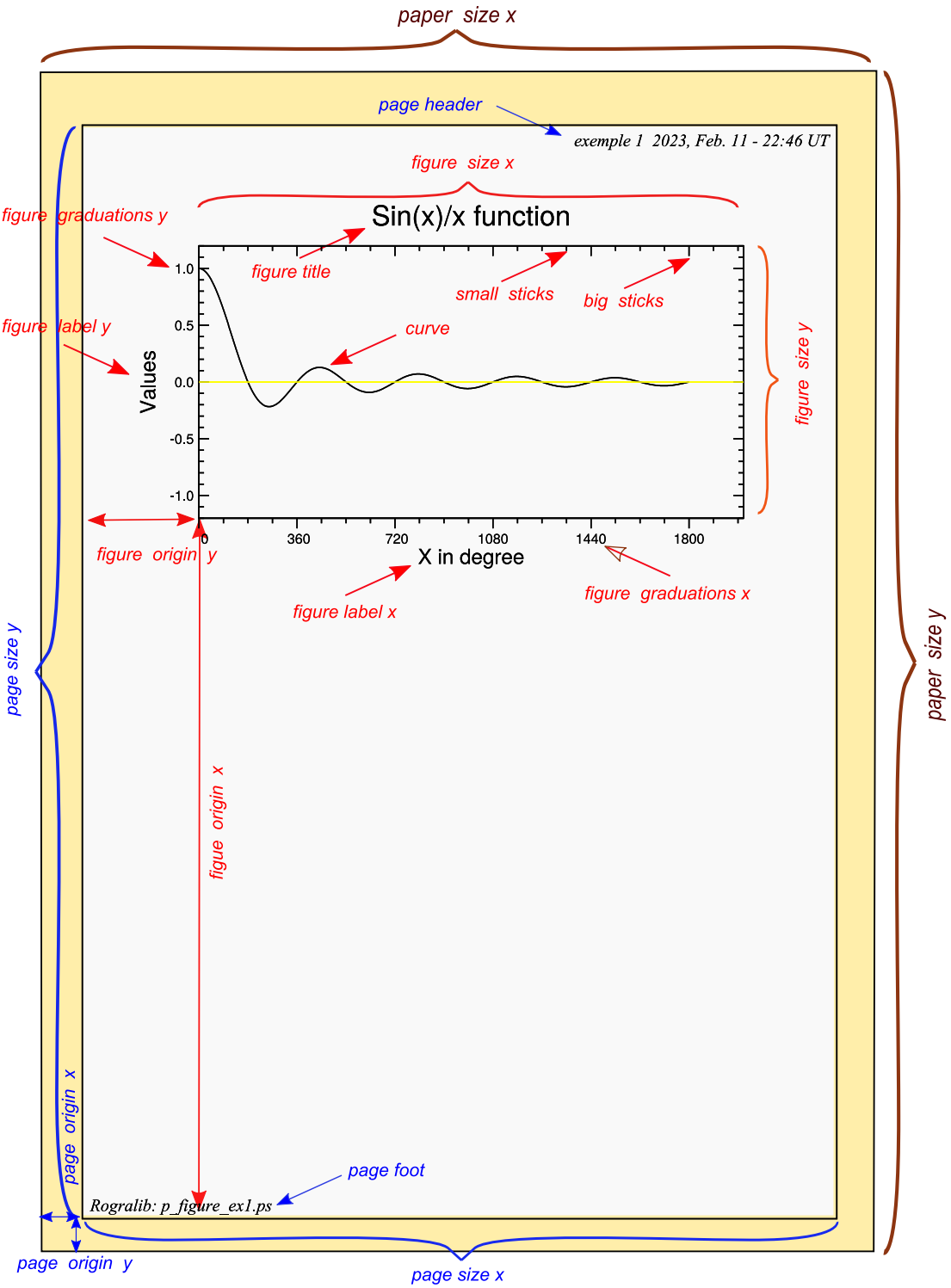


Fig. 2.1: *Coordinates and objects definition of a Rogralib plot*

```
      program p_figure_ex1
c
c     ------------------------------------------------------------------0--
c *   plot on page a simple figure
c     ------------------------------------------------------------------0--
c
      real x(1800), y(1800)
c
c *** define a curve to be plotted
c
      do i=1,360*5
         x(i)=float(i)
         xrad=x(i)*3.14159/180.
         y(i)=sin(xrad)/xrad
      enddo
c
c *** open graphical file, and plot page frame and  header/foot
c
      call dopegra_(1,'p_figure_ex1.ps')
      call ppagfra_
      call ppaghea_('exemple 1')
      call ppagfoo_('Rogralib:  ')
c
c *** define size and position on the figure on the current page
c
      call dfigsiz_(14.,7.)
      call dfigori_(3.,18.)
c
c *** define limits of the figure and graduations (Xmin,Xmax,Ymin,Ymax)
c
      x1=0.
      x2=2000.
      y1=-1.2
      y2= 1.2

      call dfiglim_(x1,x2,y1,y2)
c
c *** define font type as 'Helvetica'
c
      call dfontyp_('Helvetica')
c
c *** plot of figure frame, figure title, figure graduations and labels
c
      call pfigfra_
      call pfigtit_('Sin(x)/x function')
      call pfiggrax(0.,360.,90.,'(f5.0)')
      call pfiggray(0., 0.5,0.1,'(f4.1)')
      call pfiglabx('X in degree')
      call pfiglaby('Values')
c
c *** plot of the curve (x,y)
c
      call pfigcur_(x,y,1800)
c
c *** plot on the figure an horizontal and yellow line
c
      call dlincol_('y')
      call pfighli_(0.)
c
c *** close graphical file
c
      call dclogra_
c
      stop 'OK p_figure_ex1'
      end
```

Listing 2.1: *Simple program showing how plot a figure*

2.3 To go fast

If you are in a hurry to plot a simple function f(x), you can use pre-defined subroutine which plot the corresponding figure. You have jut to write a simple program as below (listing 2.2) which produce the plot of figure 2.2. The subroutine `ppagfig1`, `ppagfig2`, `ppagfig3` are included in **Rogralib**, so no need to link it. The source are given hereafter, thus the user can rename them and modify their contents at will to improve the plot.

```
      program p_xy
c     ------------------------------------------------------------------0--
c *   plot on page a curve y=f(x)
c     ------------------------------------------------------------------0--
      real x(900), y1(900), y2(900), y3(900)
      character*64 labx,laby1,laby2,laby3,tit,psfil
c
c *** define  curves to be plotted
      n=900
      do i=1,n
         x(i)=float(i)
         xrad=x(i)*3.14159/180.
         y1(i)=sin(xrad)/(xrad)
         y2(i)=sin(2.*xrad)/(xrad)
         y3(i)=sin(3.*xrad)/(xrad)
      enddo

c *** plot the curves
      sx = 14.
      sy = 7.
      ox = 3.
      oy  = 19.
      labx='Angle, degrees'
      laby1='sin(X)/X'
      laby2='sin(2X)/X'
      laby3='sin(3X)/X'
      tit ='Plot y=f(x)'
      psfil='plot_xy.ps'

      call dopegra_(1,psfil)
      call ppagfig1(ox,oy,    sx,sy,' ' ,laby1,tit,x,y1,n)
      call ppagfig2(ox,oy-8.,sx,sy,' ' ,laby2,' ',x,y1,y2,n)
      call ppagfig3(ox,oy-16.,sx,sy,labx,laby3,' ',x,y1,y2,y3,n)
      call dclogra_

      stop 'OK p_xy'
      end
```

Listing 2.2: *Simple program to plot simple curves*

```
      subroutine ppagfig1(ox,oy,sx,sy,labx,laby,tit,x,y,n)
c
      real x(n),y(n)
      character*(*) labx,laby,tit
      character*8 fx,fy
c
      call dfigori_(ox,oy)
      call dfigsiz_(sx,sy)

      call cminmax_(x,n,xmin,xmax)
      call cminmax_(y,n,ymin,ymax)
      call cfiggra_(xmin,xmax,xmi,xma,bgx,sgx,fx)
      call cfiggra_(ymin,ymax,ymi,yma,bgy,sgy,fy)
      call dfiglim_(xmi,xma,ymi,yma)

      call pfiggrax(0.,bgx,sgx,fx)
      call pfiggray(0.,bgy,sgy,fy)
      call pfiglabx(labx)
      call pfiglaby(laby)
      call pfigtit_(tit)
      call pfigcur_(x,y,n)
c
      return
      end
```

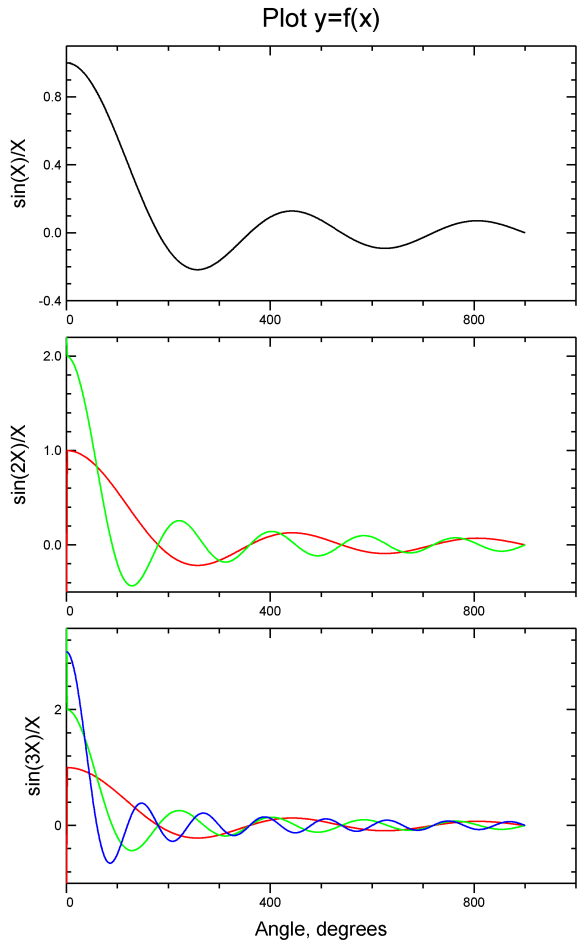


Fig. 2.2: *Plot of simple figures with program 2.2.*

```
      subroutine ppagfig2(ox,oy,sx,sy,labx,laby,tit,x,y1,y2,n)
      real x(n),y1(n),y2(n)
. . . . . same as ppagfig1 . . .
      call cminmax_(x ,n,xmin,xmax)
      call cminmax_(y1,n,ymin1,ymax1)
      call cminmax_(y2,n,ymin2,ymax2)
      ymin=min(ymin1,ymin2)
      ymax=max(ymax1,ymax2)
. . . . . same as ppagfig1 . . .
      call pfigtit_(tit)
      call dlincol_('r')
      call pfigcur_(x,y1,n)
      call dlincol_('g')
      call pfigcur_(x,y2,n)
      call dlincol_('n')
. . . . . .
      subroutine ppagfig3(ox,oy,sx,sy,labx,laby,tit,x,y1,y2,y3,n)
      real x(n),y1(n),y2(n)
. . . . . same as ppagfig1 . . .
      call cminmax_(x ,n,xmin,xmax)
      call cminmax_(y1,n,ymin1,ymax1)
      call cminmax_(y2,n,ymin2,ymax2)
      call cminmax_(y3,n,ymin3,ymax3)
      ymin=min(ymin1,ymin2,ymin3)
      ymax=max(ymax1,ymax2,ymax3)
. . . . . same as ppagfig1 . . .
      call pfigtit_(tit)
      call dlincol_('r')
      call pfigcur_(x,y1,n)
      call dlincol_('g')
      call pfigcur_(x,y2,n)
      call dlincol_('b')
      call pfigcur_(x,y3,n)
      call dlincol_('n')
. . . . . .
```

2.4 Categories of subroutine

From the two preceding examples, we can remarks that there is various categories of subroutines. For example, the subroutines having a name beginning by "**d**" are subroutines which define parameter for page or figure, for instance "**d**linwid_", "**d**fontyp_", "**d**lincol_", "**d**figsiz_", and so on. The subroutines whitch plot something have a name beginning by "**p**", as "**p**pagfra_", "**p**pagcha_", "**p**figfra", "**p**figtit", "**p**figcur_", ands so on.

There is others categories of subroutines described below.

- **`Define parameters`**

The arguments of these subroutines set parameter for the plot; page parameter, figure parameter, color and width of line, character font, etc.
Some examples are given below. Full list is given in section 6.2

dlinwid_`(iwidth)`	in pixel, for next drawn lines
dfontyp_`(fontyp)`	as 'c','h','hbobl',...
dlincol_`(col)`	as 'r' or nrgbcmypwi
dfigori_`(figorx,figory)`	in cm of the lower left corner
dfigsiz_`(figsix,figsiy)`	in cm.
dpagsiz_`(pagsix,pagsiy,rmargex,rmargey)`	in cm. erase default size

These subroutines write corresponding instructions in the PostScript file by the way of the driver subroutines (see 6.9).

- **`Give parameters`**

This is the inverse function of "define": the arguments returns the current values of the parameters. The parameters can have been setting by the users from a "define" subroutine, or set by default parameter at the time of the dopegra_ instruction.
Some examples are given below. Full list is given in section 6.3

glinwid_`(iwidth)`	in pixel, for current drawn lines
gfontyp_`(fontyp)`	as 'c','h','hbold',...
glincol_`(col)`	as 'n', or rgbcmypw
gfigori_`(figorx,figory)`	in cm. from lower left corner
gfigsiz_`(figsix,figsiy)`	in cm.
gpagsiz_`(pagsix,pagsiy)`	in cm.

These subroutines have no effect on the content of the PostScript file.

- **`Give current date and time`**

All these routines gives the current date, time, or date and time, in various forms. All return arguments are of character type. The parameters can have been setting by the users from a "define" subroutine, or set by default parameter at the time of the dopegra_ instruction.
Some examples are given below. Full list is given in section 5.3.1

gclopar_`(imonth,iday,iyear,ih,im,is)`	at the time call
gdidate_`(date10)`	as '2007-09-27'
gditime_`(time8)`	as '13:03:25'
gfrdati_`(dati28)`	as '23 Octobre 1992 - 23:50:10'
gusdati_`(dati29)`	as 'October 23, 1992 - 23:50:10'
gfrdate_`(date17)`	as '21 Septembre 1988' (17c.)
gusdate_`(date18)`	as 'September 21, 1988' (18c.)

These subroutines have no effect on the content of the PostScript file.

- **Compute and Convert**

These subroutines does some computations or conversion, for instance:

cminmax_(ax,n,axmin,axmax)	compute minimum and maximum of ax(n)
cplarot_(xcen,ycen,angle,xi,yi,xc,yc)	compute the result of a plane rotation on xi,yi
cmanexp_(x,rmant,iexp)	compute mantissa and exponent of a real x
cfiggra_(xmin,xmax,x1,x2,bgx,sgx,fx)	compute figure graduations and graduation format from xmin and xmax
cfigpag_(fx,fy,px,py)	convert figure units to page units
ccarpol_(x,y,r,theta)	convert Cartesian to polar coordinates
ccarsph_(x,y,z,r,theta,phi)	convert Cartesian to spherical components
crgbhsb_(r,g,b,hue,sat,bri)	convert R,G,B colour to Hue, Saturation, Brightness colours

These subroutines have no effect on the content of the Postscript file.

- **Plot objects on page or figure**

Those are the basic subroutines, for instance:

ppagfra_	plot page frame
ppagcha_(ox,oy,ipos,tcx,tcy,angle,textel)	plot on page a character string
ppaglin_(ox,oy,tox,toy)	plot on page a line
pfigtit_(figtit)	plot figure title on an existing figure
pfiglabx(labelx)	plot figure label X
pfigcurv_(ax,ay,n)	plot on figure a curve defined by arrays

These subroutines write corresponding instructions in the PostScript file by the way of the driver subroutines (see 6.9).

- **Read bmp file**

Used mainly to import bmp image; ifc is the unit file code used to read the file.

rdimbmp_(ifc,fichbmp,nx,ny)	read dimension of a bmp image file.
rimabmp_(ifc,fichbmp,image,nx,ny)	read content of a bmp image file.
roctfil_(ifc,io)	read an octet on the ifc unit.

- **Library utility**

These subroutines are used from other categories. It can by used directly, but it is not a normal mode. Example :

uboundi_(ix, ix1,ix2)	u bounds integer is as ix1 <ix <ix2
udecfor_(format,ifg,n,nd)	u decode format
uencfor_(format,ifg,n,nd)	u encode format
urinbel_	u ring bell by printing '007' character

- **Postscript Driver**

This routines are internal subroutines used by **Rogralib**. It is not recommended to use it. Nevertheless full list is given in section 6.9
medskip

2.5 Nomenclature of the subroutines names

We can wonder why most of subroutine names ends by a "_" character. The main reason is that the name of the subroutine have always 8 characters length, for the following reasons:

1. The first character indicate the category name : "**d**" for "define", "**g**" for "give", "**c**" for "compute or convert", "**p**" for "plot".
2. The 2 following groups or 3 characters indicate the function : for example, ppagfra_ means "plot page frame".
3. The 8^{th} character may be a "_", or "x" or "y" for two action on x and y, for instance pfiglabx means plot figure label of X axis, pfiglaby means plot figure label of Y axis, while pfiglab_ means plot figure label of X and Y axis.

In any cases, do not forget the "_" character which terminate most of subroutine names.

2.6 Hierarchie of calls

The first call must be of course **dopegra_** which open the postscript file and set default parameters. Then, as the page is defined, all of "plot-page" routine can be used, as well as "define", "compute", "convert" subroutine. For making a figure, the "call" sequence must be as following:

• Figure position and figure size must the first defined by : **call dfigpos_** and **call dfigsiz_**

• Then, limits of the x and y axes must be defined for further axe's graduation by : **call dfiglim_** values of graduation can be computed automatically by **cminmax_** and **cfiggra_**

• Then, the figure is fully defined, and all the plot_figure subroutine are available, such as : **pfigfra_, pfigtit_, pfiggrax_, pfiggray_, pfiglabx_, pfiglaby_, pfigcur_** and this in any order.

Be careful with color change : the color change such as **dlincol_ or dlinrgb_, dlinhsb_** remains active until a new call to change or restore the previous color. The last graphical call must be **dclogra_**, which close the Postscript file.

2.7 Superimposed plots

The principle of drawing lines, text, images or others objects is simple: each graphical object, such a line, an image, etc. which is plotted by a specific call is plotted over the existing plots. This means that previous plot must be hidden by the new one. So the order of the various calls is both fundamental and useful : for instance, if you want a colored background in your figure (see 2.8.2), you must first fill the figure frame with the desired color, and then all curves or others objects will be plotted over this background. A more detailed example is given section 4.5. There is no known limit to this over-plotting.

Figure 2.3 show the importance of order in the call: The yellow rectangle must be plot first, then the gray zone and only after the curves and legends, and at last the figure frame and graduation.

```
<< The source code is available in the software distribution >>
```

Listing 2.3: *Program showing superimposed plots*

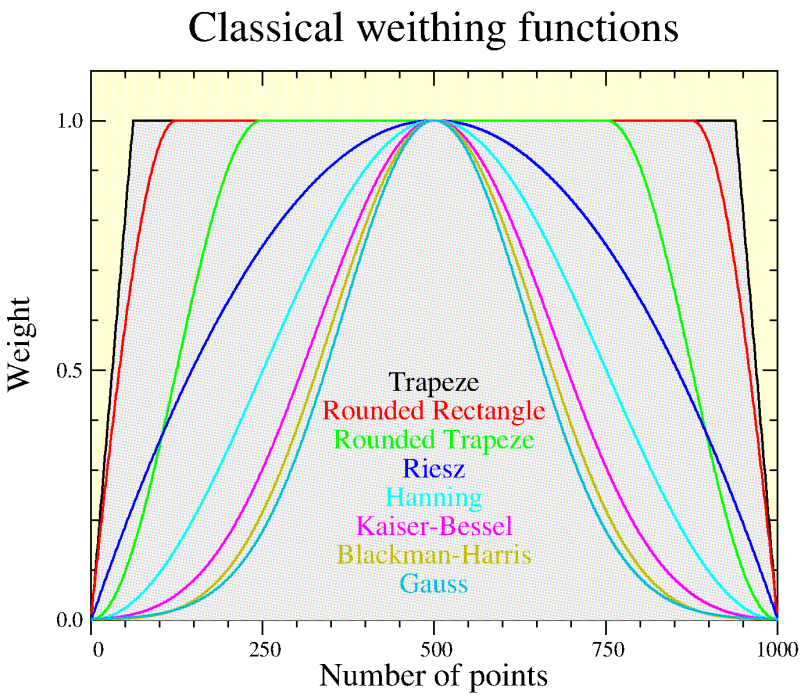


Fig. 2.3: *Importance of the order in which calls are made.*

Afterwards, if the Postscript file is open with a PC software such as Micrograph Designer™, CorelDraw™, Inkscape™or Adobe Illustrator™, each visible or hidden object can be moved or transformed separately, even put in background or foreground. This can be useful to modify an existing figure, for a publication for instance.

2.8 Zone filling

Rogralib allow to fill the inside of a given path. A path can be defined by a pen motion between a start and a end point. The subroutines **dfilzon_** allow to define the beginning of the path to fill while **pfilzon_** fill inside the path. The path must be open or closed. Examples are given below. A more detailed example is given section 4.5.

2.8.1 Example of zone filling in a page

The example above use **ppagcir_** which draw a circular arc in the page, and **ppagpmo_**, which complete the zone to fill. Result is shown on figure 2.4.

```
      program p_figure_ex2
c
c     ------------------------------------------------------------------0--
c *   plot on page a pie chart
c     ------------------------------------------------------------------0--
c
      data xc,yc,r /5.,5.,6./
      data teta1,teta2,dteta /0.,60.,1./
c
c *** open graphical file and define a zone to fill
c
      call dopegra_(1,'p_figure_ex2.ps')
      call dfilzon_
c
c *** plot on page a part of a circle and fill the defined zone
c
      call ppagcir_(xc,yc,r,teta1,teta2,dteta)
      call dlincol_('y')
      call pfilzon_
c
c *** redraw the contour line with a given width
c
      call dlinwid_(5)
      call dlincol_('n')
      call ppagcir_(xc,yc,r,teta1,teta2,dteta)
c
c *** same but fill until the center of the circle
c
      call dlincol_('y')
      call ppagcir_(xc+7.,yc,r,teta1,teta2,dteta)call ppagpmo_(xc,yc)
      call ppagpmo_(xc,yc)
      call pfilzon_
c
c *** redraw the contour line with a new color
c
      call dlincol_('r')
      call ppagcir_(xc,yc,r,teta1,teta2,dteta)
      call ppagpmo_(xc,yc)
      call ppagpmo_(xc+r,yc)
c
c *** close graphical file
c
      call dclogra_
c
      stop 'OK p_figure_ex2'
      end
```

Listing 2.4: *Example using filling zone in a page*

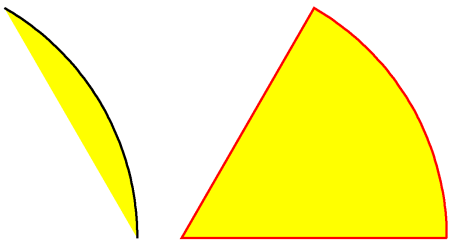


Fig. 2.4: *Example of plot using filling zone in a page*

2.8.2 Example of zone filling in a figure

Following example is the the same that the one in section 2.2 but after defining the figure attribute (until line 36) we put the following instructions. Fist, we define a path, or a contour line, by using **pfigcur_** and **pfigpmo_**, which define the zone to fill, and second we fill this zone by using **pfilzon_**. Result is shown on figure 2.5

```
      call dfiglim_(x1,x2,y1,y2)
c
c *** fill zone above  the curve
      call dlincol_('y')
      call dfilzon_
      call pfigcur_(x,y,1800)
      call pfigpmo_(x2,y(1800))
      call pfigpmo_(x2,y2)
      call pfigpmo_(x1,y2)
      call pfigpmo_(x1,y(1))
      call pfilzon_
c
c *** fill zone below  the curve
      call dlincol_('c')
      call dfilzon_
      call pfigcur_(x,y,1800)
      call pfigpmo_(x2,y(1800))
      call pfigpmo_(x2,y1)
      call pfigpmo_(x1,y1)
      call pfigpmo_(x1,y(1))
      call pfilzon_
c
c *** define font type as 'Helvetica' and return to black color
c
      call dfontyp_('Helvetica')
      call dlincol_('n')
c
c *** plot of figure frame, figure title, figure graduations and labels
c
      call pfigfra_
      call pfigtit_('Sin(x)/x function')
      call pfiggrax(0.,360.,90.,'(f5.0)')
      call pfiggray(0., 0.5,0.1,'(f4.1)')
      call pfiglabx('X in degree')
      call pfiglaby('Values')
c
c *** plot of the curve (x,y)
c
      call pfigcur_(x,y,1800)
```

Listing 2.5: *Example using filling zone in a figure*

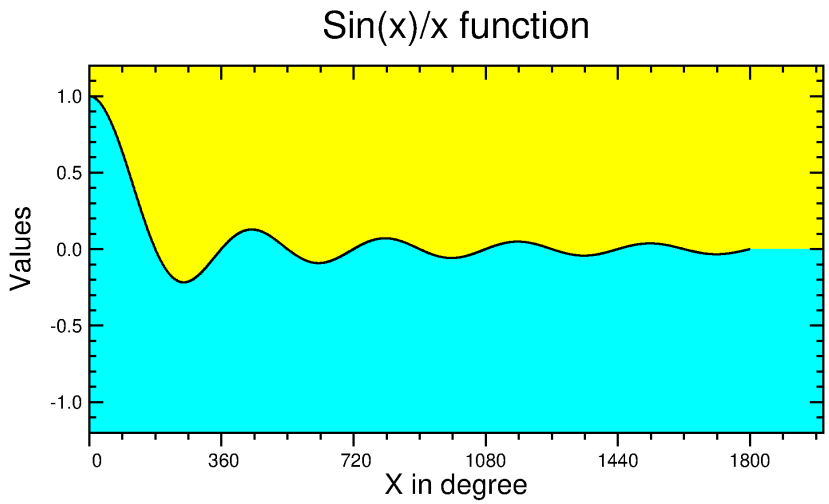


Fig. 2.5: *Example of plot using filling zone in a figure*

2.8.3 Check trespassing limits in a figure

When a curve is plotted on a figure, it can append that the values are beyond the limits of the figure. **Rogralib** can manage this by the subroutine **dfigout_(arg)**. If arg='y', drawing outside the figure is allowed. If arg='n', only the part of the curve inside the figure is drawn.

Program 2.6 and the corresponding plot on figure 2.6 illustrates how this command works.

```
      program p_check_overframe

c     ------------------------------------------------------------------0--
c *   check trepasiing of figure and page
c     ------------------------------------------------------------------0--

      real x(11), y(11)
      character*14 vercha

c *** define a curve to be plotted

      x(1)= 0.2
      y(1)= 0.8
      x(2)=-0.2
      y(2)= 0.6
      x(3)=-0.2
      y(3)= 0.4
      x(4)= 0.2
      y(4)= 0.2
      x(5)= 0.4
      y(5)=-0.2
      x(6)= 0.8
      y(6)= 0.2
      x(7)= 1.6
      y(7)= 0.4
      x(8)= 0.8
      y(8)= 0.8
      x(9)= 1.2
      y(9)= 1.1
      x(10)= 0.3
      y(10)= 1.1
      x(11)= 0.2
      y(11)= 0.8
      n=11

      call dopegra_(1,'p_check_overframe.ps')
      call gvercha_(vercha)
      call ppaghea_(vercha//' - ')
      call ppagfoo_('P. Robert')
      call ppagfra_

c *** define size and position on the figure on the current page

      call dfigsiz_(12.,12.)
      call dfigori_(4.,12.)

c *** define limits of the figure and graduations

      call dfiglim_(0.,1.,0.,1.)
      call dfontyp_('Arial-B')
      call dgrasiz_(0.4,0.4)

c *** plot of figure frame, figure graduations

      call pfigfra_
      call pfigtit_("Test of dfigout_")
      call pfiggrax(0.,0.5,0.1,'(f4.1)')
      call pfiggray(0.,0.5,0.1,'(f4.1)')

c *** plot of the curve (x,y) with overplot allowed

      call dfigout_('y')
      call dlincol_('r')
      call pfigcur_(x,y,n)
```

```
c *** re-plot, but WITHOUT overplot

      call dfigout_('n')

c * first,  fill zone and re-draw graduations

      call dfilzon_
      call dlincol_('y')
      call pfigcur_(x,y,n)
      call pfilzon_

      call dlincol_('n')
      call pfiggrax(0.,0.5,0.1,'(f4.1)')
      call pfiggray(0.,0.5,0.1,'(f4.1)')

c *   re-plot without overplot

      call dlincol_('b')
      call dlinwid_(10)
      call pfigcur_(x,y,n)

      call dclogra_

      stop 'OK p_check_overframe'
      end
```

Listing 2.6: *Program to check curve trespassing figure limits*

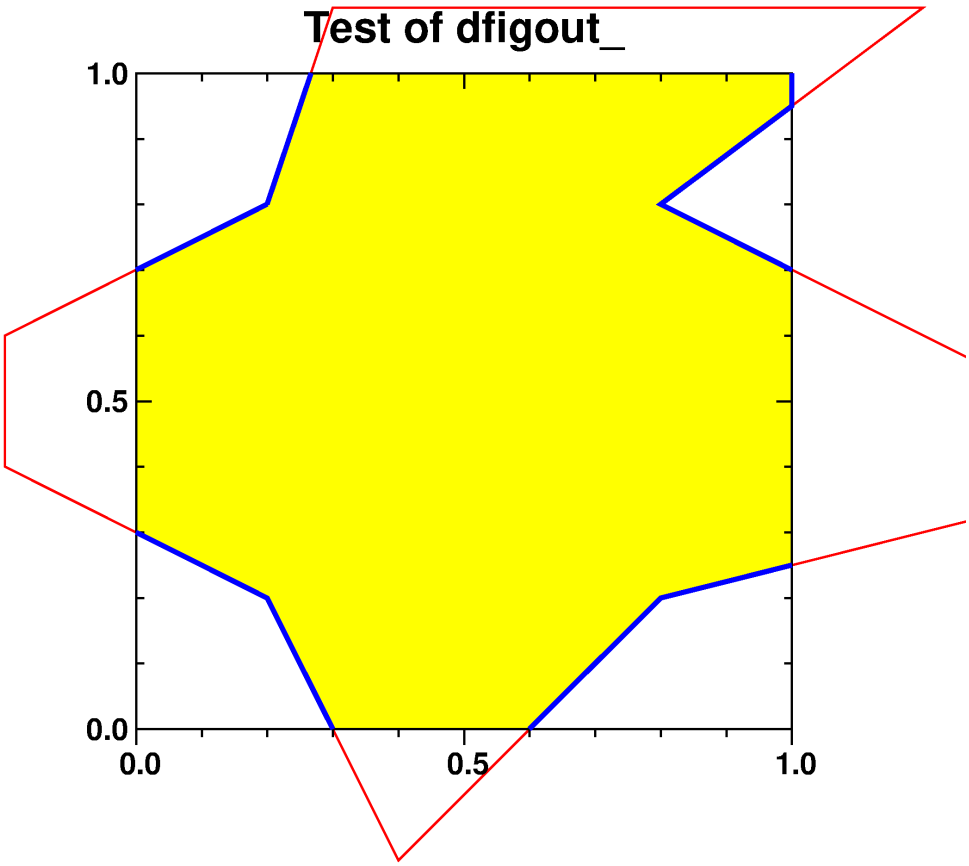


Fig. 2.6: *Plot to check curve trespassing figure limits (from program 2.6)*

2.9 Character fonts and line width

2.9.1 Available character fonts

Rogralib manage mains characters fonts. The list, and example of plot, can be produced by the program given in listing 2.7, while result is given in figure 2.7.

```
      program p_char_fonts_2
c     ------------------------------------------------------------------+--
c     plot all available fonts; P. Robert, SD, Mars 2022
c     ------------------------------------------------------------------+--
      character*32 fontyp

      call dopegra_(1,'p_cha_fonts_2.ps')

      ifon=0
      do i=1,12
         oy=25.-float(i)*0.8
         do j=1,4
            if(ifon.gt.43) exit
            if(i.eq.10 .and. j.gt.2) exit
            if(i.eq.11 .and. j.eq.2) exit
            ox=0.6+float(j-1)*4.5
            ifon=ifon+1
            call dfonnum_(ifon)
            call gfontyp_(fontyp)
            call ppagcha_(ox,oy,-1,0.3,0.3,0.,fontyp)
         enddo
      enddo

      call dclogra_

      stop 'OK p_char_fonts_2'
      end
```

Listing 2.7: *Program to plot all available character fonts.*

Courier	**Courier-B**	*Courier-I*	***Courier-BI***
Times	**Times-B**	*Times-I*	***Times-BI***
Helvetica	**Helvetica-B**	*Helvetica-I*	***Helvetica-BI***
AvantGarde	**AvantGarde-B**	*AvantGarde-I*	***AvantGarde-BI***
Bookman	**Bookman-B**	*Bookman-I*	***Bookman-BI***
HelveNarr	**HelvetNarr-B**	*HelvetNarr-I*	***HelvetNarr-BI***
LetterGhotic	**LetterGhotic-B**	*LetterGhotic-I*	***LetterGhotic-BI***
NewCentury	**NewCentury-B**	*NewCentury-I*	***NewCentury-BI***
Palatino	**Palatino-B**	*Palatino-I*	***Palatino-BI***
ZapfChancery	✺❁❒❄✤❇■✳✪❁▼▲		
Σψμβολ			
Arial	**Arial-B**	*Arial-I*	***Arial-BI***

Fig. 2.7: *Available character fonts (from program 2.7)*

2.9.2 Character table for main fonts

Characters of the ASCII table, reduced to [33-248], are plotted here, for the main fonts Courier, Times, Helvetian and Symbols.

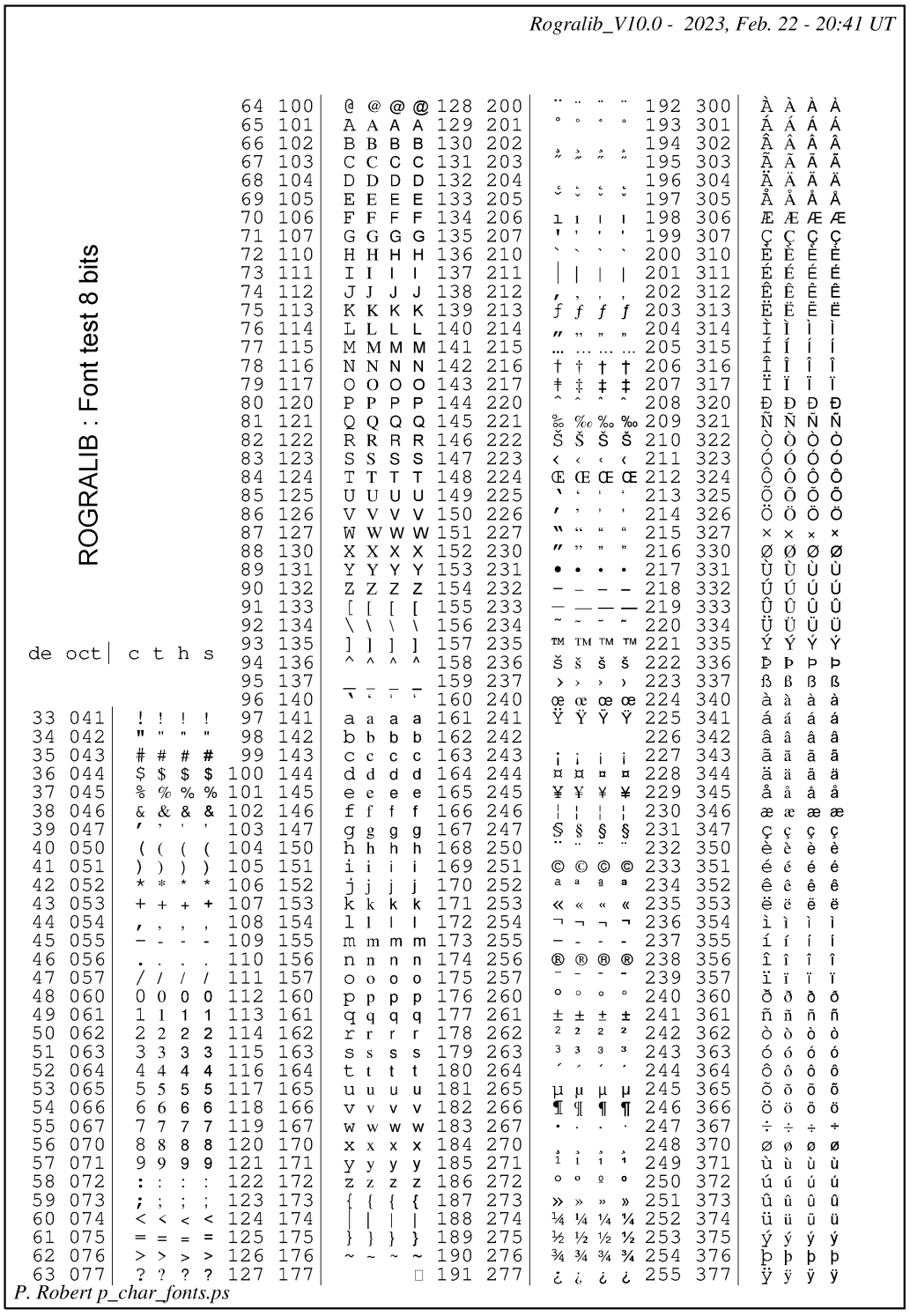


Fig. 2.8: *Character table for main fonts (program 2.8)*

```
      program p_char_fonts

c     ------------------------------------------------------------------0--
c *   plot all availables p_char_fonts and characters
c     P. Robert, SD, August 1999
c     ------------------------------------------------------------------0--

      character*14 vercha

      call dopegra_(1,'p_char_fonts.ps')

      call ppagfra_
      call gvercha_(vercha)
      call ppaghea_(vercha//' - ')
      call ppagfoo_('P. Robert')
      call ppagcha_(2.,16.,-1,0.4,0.4,90.,'ROGRALIB : Font test 8 bits')

      call plot_table(1,'c')
      call plot_table(2,'t')
      call plot_table(3,'h')
      call plot_table(4,'s')

      call dfontyp_('c')
      call ppagcha_(0.5,14.,-1,0.25,0.25,0.,'de oct| c t h s')

      call dclogra_
      stop 'OK p_char_fonts'
      end

c     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

      subroutine plot_table(it,font)

c ---------------------------------------------------------------------------
c     plot character table for main fonts (Cour, Times, Helve, Symb)
c ---------------------------------------------------------------------------

      character*(*) font
      character*8 text
      character*1  cara

      call dfontyp_(font)

      idec=-1
      tc=0.25

c **  3 loops for octal computation of idec value [33-248]

      do i=0,3
         do j=0,7
            do k=0,7
               idec=idec+1
               if(idec.lt.33) cycle
               icol= idec/64 +1
               x=0.3 + float(icol-1)*4.5
               y= 25.8 -float(idec-(icol-1)*64)*tc*1.6
               x2=x+8.*tc +float(it)*2.*tc

               write(text,100) idec,i,j,k
               write(cara,200) char(idec)
               if(it.eq.1) call ppagcha_(x,y,-1,tc,tc,0.,text)
               call ppagcha_(x2,y,-1,tc,tc,0.,cara)
            enddo
         enddo
      enddo

  100 format(i3,1x,3i1,'|')
  200 format(a)
      return
      end
```

Listing 2.8: *Program to produce main font table.*

2.9.3 Line width

There is two ways to define the line width:

• by choosing the line width in pixel by `dlinwid_(iwidth)` where iwidth is the number of pixels used to draw the line,

• by `dlinwidr(width)` where width is a floating value of the pixel number.
Example is done by the program given in listing 2.9, while result is given on figure 2.9.

```
      program p_width_line
c     ------------------------------------------------------------------+--
c     plot lines with various width; P. Robert, SD, Mars 2022
c     ------------------------------------------------------------------+--
      data ox,oy,tc,nw /0.,15.,0.4, 80/

      call dopegra_(1,'p_width_line.ps')

      do i=0,nw,4
         ox=ox+0.8
         call ppagval_(ox,oy+0.5,-1,tc,tc,0.,float(i),'(i2)')
         call dlinwid_(i)
         call dlinhsb_(float(i)/float(nw),1.,1.)
         call ppaglin_(ox,oy,ox+0.8,oy)
      enddo

      ox=0.6
      do i=0,nw+3
         ox=ox+0.2
         call dlinwidr(float(i))
         call dlinhsb_(float(i)/float(nw),1.,1.)
         call ppaglin_(ox,oy-1.,ox+0.2,oy-1.)
      enddo

      call dclogra_

      stop 'OK p_width_line'
      end
```

Listing 2.9: *Program to plot lines with various width.*

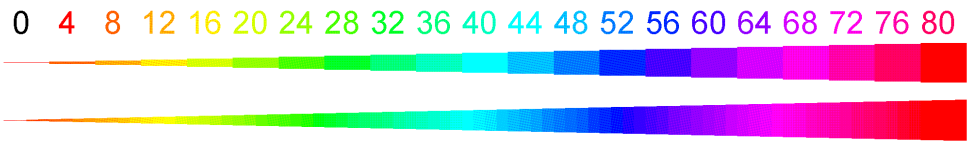


Fig. 2.9: *Example of lines with various width (from program 2.9)*

2.10 Plot objects on a page

2.10.1 Plot geometric shapes

The below program shows how to plot some geometric shapes on a page, and use option of filling zone seen previously.

```
      program p_ex_geom

      data ox,oy,sx,sy,r_rec,r_cir,a,b /1.,10.,4.,3.,0.4,2.,2.5,1.5/

      call dopegra_(1,'p_ex_geom.ps')
      call dfontyp_('t')
```

```
      call ppagrec_(ox,oy,sx,sy)
      call ppagrro_(ox+5.,oy,sx,sy,r_rec)

      call ppagcir_(ox+12.,oy+r_cir,r_cir,0.,360.,1.)
      call ppagcir_(ox+16.,oy+r_cir,r_cir,0.,135.,1.)

      call ppagell_(ox+a,oy-2.,    a,b, 0.,0.,360.,1.)
      call ppagell_(ox+10.,oy-2.,a,b,30.,0.,360.,1.)

      call dlincol_('y')
      call dfilzon_
      call ppagrro_(ox,oy-7.,sx*2.,sy*0.8,r_rec)
      call pfilzon_
      call dlincol_('b')
      call ppagcha_(ox+0.5 ,oy-6.,-1,0.6,0.6,0.,'Rounded rectangle')
      call dlinwid_(6)
      call ppagrro_(ox,oy-7.,sx*2.,sy*0.8,r_rec)

      call dclogra_
      stop 'OK p_ex_geom'
      end
```

Listing 2.10: *Plot example of geometric shape*

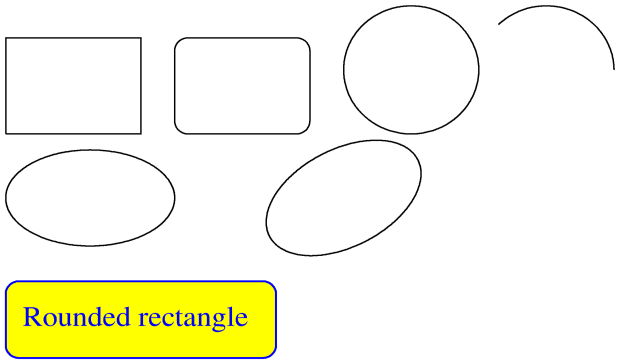


Fig. 2.10: *Result of the above program*

2.10.2 Plot Character strings with Greek font

There is two subroutines to plot a character string. The second one allow to plot some parts of the string in Greek font. This is shown on the example below.

```
      program p_greek_string

      character*80 string

      call dopegra_(1,'p_greek_string.ps')
      call dfontyp_('t')

      ox=5.
      oy=10.
      sx=0.4
      sy=0.4
      string=' :  {a} = 30  {b} = 0  {d} = 4 etc.'

      call ppagcha_(ox,oy,-1,sx,sy,10.,'with ppagcha_'//string)
      call ppagstr_(ox,oy-2.,sx,sy,10.,'with ppagstr_'//string)
      call dlincol_('b')
      call ppagstr_(ox,oy-4.,sx,sy, 0.,'with ppagstr_'//string)

      call dclogra_
      stop 'OK p_greek_string'
      end
```

Listing 2.11: *Program to plot character strings including Greek fonts*

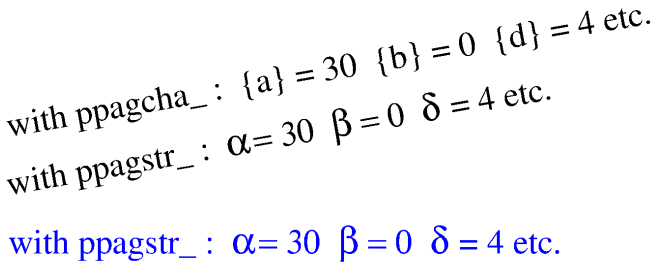


Fig. 2.11: *Example of plot character strings including Greek fonts*

2.10.3 Plot formatted text

This program use the subroutine **ppagtxt_**`(ox,oy,cheight,twidth,spac,txt,nl)` : **plot_page_text**. Arguments are position of the first word (bottom left corner),character height, width of the text, line spacing, and a array txt(nl) containing the text without special format. Program source is given in listing 2.12, and result is shown on figure 2.12.

The text will be reformatted, and written according to the desired width. Its position is defined by the first word, and the number of lines necessary for its justification is calculated. The result is shown below.

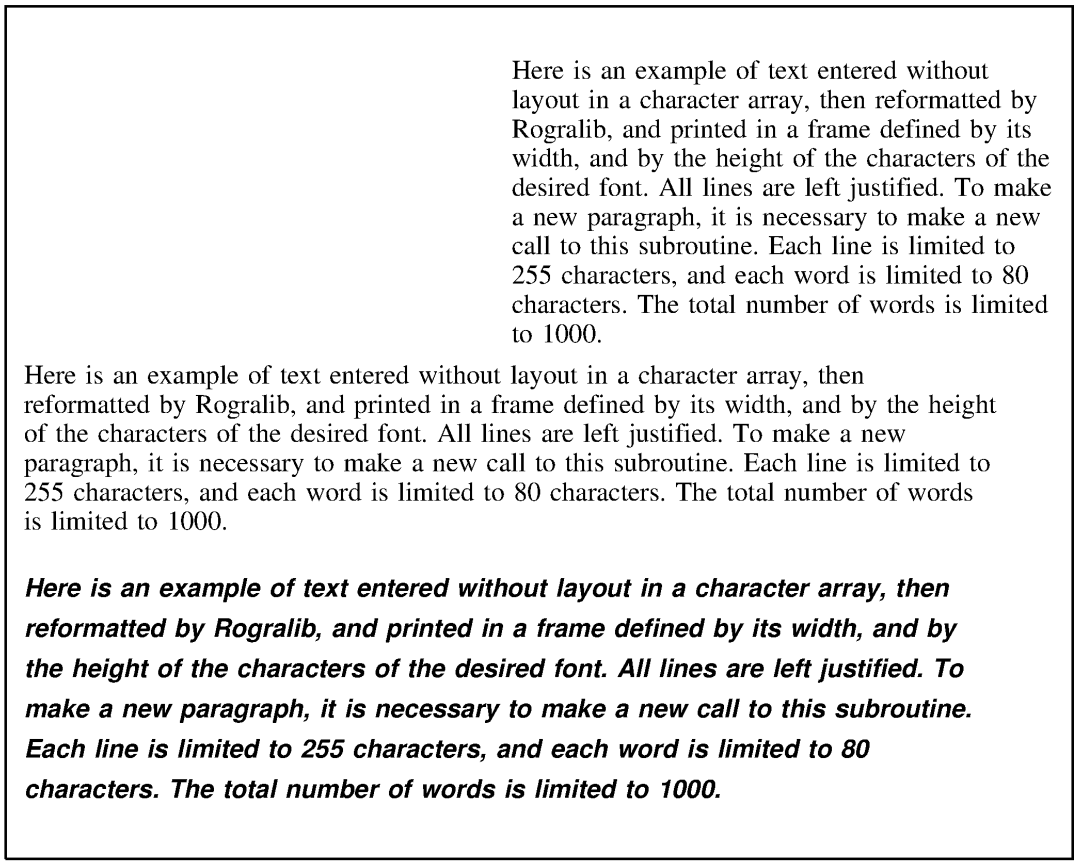


Fig. 2.12: *Plot on page a formatted text (from program 2.12)*

```
      program p_for_txt
c
c     ------------------------------------------------------------0--
c     plot formatted text
c     ------------------------------------------------------------0--
c
      character*255 txt(100)
c
      txt( 1)='Here is an example of text entered without layout in a '
      txt( 2)='character array, then reformatted by Rogralib, '
      txt( 3)='and printed in a frame defined by its width, and by '
      txt( 4)='the height of the characters of the desired font. '
      txt( 5)='All lines are left justified. To make a new paragraph, '
      txt( 6)='it is necessary to make a new call to this subroutine.'
      txt( 7)='Each line is limited to 255 characters, and'
      txt( 8)=' '
      txt( 9)='each word is limited to 80 characters.'
      txt(10)='The total number of words is limited to 1000.'

      nl=10
c
      call dopegra_(1,'p_for_txt.ps')
      call dfontyp_('t')

      ox=10.
      oy=20.
      cheight=0.3
      twidth=9.
      spac=cheight*0.8

      call ppagtxt_(ox,oy,cheight,twidth,spac,txt,nl)

c *** The same thing, with a larger width

      ox=2.
      oy=15.
      twidth=16.

      call ppagtxt_(ox,oy,cheight,twidth,spac,txt,nl)

c *** The same thing, with another font and a large spacing

      ox=2.
      oy=11.5
      twidth=16.
      spac=cheight*1.1

      call dfontyp_('Arial-BI')
      call ppagtxt_(ox,oy,cheight,twidth,spac,txt,nl)
      call ppagrec_(ox-0.3,oy-1.,17.5,14.)

      call dclogra_

      stop 'OK p_for_txt'
      end
```

Listing 2.12: *Program to show how plot a formatted text*

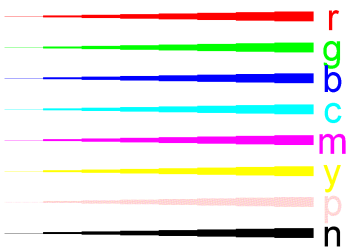
Chapter 3 : MANAGE COLORS & IMAGES

Rogralib can manage colors in various ways:

- basic colors Red, Green, Blue, Cyan, Magenta, Yellow, Pink, Black, White,
- R, G, B components, to get 16M of colors or more,
- H, S, B components.

Following sections show many examples to use and manage colors.

3.1 Basic colors




```
      program p_basic_colors

      character*1 col(9)
      data col /'r','g','b','c','m','y','p','n','w'/
c
      call dopegra_(1,'p_basic_colors.ps')
      call dfontyp_('Arial')

      do icol=1,8
         ox=1.
         oy=25. -icol*0.8
         call dlincol_(col(icol))
         do iwid=1,32,4
            call dlinwid_(iwid)
            ox=ox+1.
            call ppaglin_(ox,oy,ox+1.,oy)
         enddo
         call ppagcha_(ox+1.5,oy,0,0.7,0.7,0.,col(icol))
      enddo

      call dclogra_
      stop 'OK p_basic_colors'
      end
```

3.2 Cube RGB

A fist view of color management is to use RGB components. Values are between [0. - 1.] Figure 3.1 below show the corresponding colors for various values of R,G,B components. Listing 3.1 show the program used to plot this cube.

```
      program p_cube_rgb
c
      character*14 vercha
c
c     ------------------------------------------------------------------0--
c     plot RGB cube; P. Robert, SD, Fevrier 2012
c     ------------------------------------------------------------------0--
c
      call dopegra_(1,'p_cube_rgb.ps')
c
      call gvercha_(vercha)
      call ppaghea_(vercha//' - ')
      call ppagfoo_('P. Robert')
      call ppagfra_
      call dfontyp_('Arial-B')
c
      deca=0.8
      rn=10.*deca
      angle=45.
      ca=cos(angle*3.14159/180.)
      sa=sin(angle*3.14159/180.)
      ox= 8.8
      oy=12.
c
c *** Pannel G,B
c
      do ig=1,11
      do ib=1,11
         g=float(ig-1)*deca
         b=float(ib-1)*deca
         xgb= ox+g
         ygb= oy+b
         g=g/rn
         b=b/rn
         call p_gb(xgb,ygb,deca,deca,0.,g,b)
         if(ib.eq.1) call ppagval_(xgb+0.1,ygb-0.5,-1,0.25,0.3,0.,
     &                   float(ig-1)/10.,'(f3.1)')
      enddo
      enddo

      call dlincol_('g')
      call ppagcha_(ox+11.3*deca,oy-0.8,-1,0.8,0.8,0.,'G')
c
c *** Pannel B,R

      oy=oy+deca
      ox=ox-deca/2.

      do ib=1,11
      do ir=1,11
         b=float(ib-1)*deca
         r=float(ir-1)*deca
         xbr= ox -r*ca -deca*ca
         ybr= oy +b -r*sa -deca*sa
         r=r/rn
         b=b/rn
         call p_br(xbr,ybr,deca,deca,r,0.,b,ca,sa)
         if(ir.eq.1) call ppagval_(xbr+0.7,ybr+deca,-1,0.25,0.3,0.,
     &                   float(ib-1)/10.,'(f3.1)')
      enddo
      enddo

      call dlincol_('b')
      call ppagcha_(ox,oy+11.3*deca,-1,0.8,0.8,0.,'B')

      call dclogra_
c
      stop 'OK p_cube_rgb'
      end
```

```
      subroutine p_gb(xgb,ygb,dg,db,r,g,b)
c
      call dlinrgb_(r,g,b)
      call ppagrec_(xgb,ygb,dg,db)
      call pfilzon_
c
      return
      end
c     ------------------------------------------------------------------0--
      subroutine p_rg(xrg,yrg,dr,dg,r,g,b,ca,sa)
c
      call dlinrgb_(r,g,b)
      call dfilzon_
      call dpagppo_(xrg,yrg)
      call ppagpmo_(xrg+dg,yrg)
      call ppagpmo_(xrg+dg+dg*ca,yrg+dr*sa)
      call ppagpmo_(xrg+dg*ca,yrg+dr*sa)
      call dpagppo_(xrg,yrg)
      call pfilzon_
c
      return
      end
c     ------------------------------------------------------------------0--
      subroutine p_br(xbr,ybr,db,dr,r,g,b,ca,sa)
c
      call dlinrgb_(r,g,b)
      call dfilzon_
      call dpagppo_(xbr,ybr)
      call ppagpmo_(xbr+dr*ca,ybr+dr*sa)
      call ppagpmo_(xbr+dr*ca,ybr+dr*sa+db)
      call ppagpmo_(xbr,ybr+db)
      call dpagppo_(xbr,ybr)
      call pfilzon_
c
      return
      end
```

Listing 3.1: *Program to plot a RGB cube*

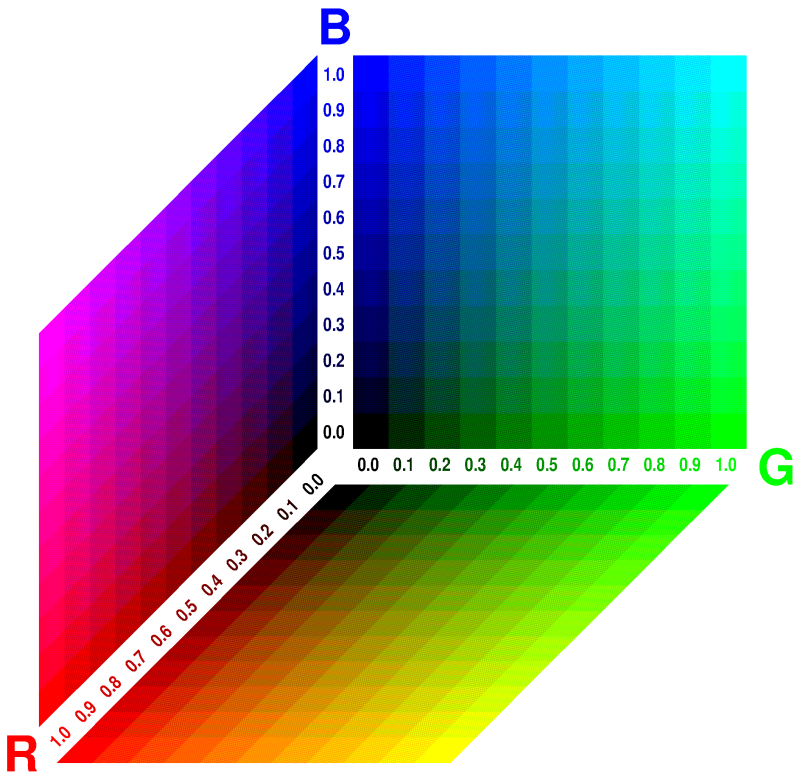


Fig. 3.1: *Example of RGB color cube (from program 3.1)*

3.3 Maxwell's color triangle

In Maxwell's triangle (1831-1879), the three primary colors (red, green and blue) are located in the angles of the equilateral triangle; mixing them in equal amounts gives the center a white light. Complementary colors (blue-green, magenta, and yellow) appear near the middle of the sides and are produced by mixing green and blue, blue and red, and red and green, respectively. This is called the trichromatic specification of colors (from Wikipedia).
The code given in listing 3.2 allow to plot this triangle, as shown in figure 3.2.

```
      program p_color_triangle
c     ------------------------------------------------------------------0--
c     P. Robert, ScientiDev, 2008
c     ------------------------------------------------------------------0--
      call p_col_tri(0.3,15.,72,0)
      call p_col_tri(0.3,2.,10,1)
      call dclogra_
      stop 'OK p_color_triangle'
      end
c     ------------------------------------------------------------------0--
      subroutine p_col_tri(orix,oriy,ncel,icom)

c *** plot Maxwell color triangle
c *   side size etc.
      a= 14.
      h= a*sqrt(3.)/2.
      da=a/float(ncel)
      dh=da*sqrt(3.)/2.
      tc=a/30.
      cc=dh/4.
      cc2=1.6*cc
c
c *** draw line contour of the main triangle
      call dlincol_('n')
      call ppagtrie(orix,oriy,a,1)
      ox=orix+a*0.77
      oy=oriy+h/2.
      call ppagcha_(ox,oy,-1,tc,tc,0.,'Maxwell color triangle')

c *** plot and fill  of all sub-triangles
c *   loop on the height
      do ih=1,ncel+1
         ncelh= ncel -ih+1
c *      x loop for sub-triangles up
         do ix=1,ncelh
            x=float(ix-1)*da +float(ih-1)*da/2.
            y=float(ih-1)*dh
            call calrgb(x,y,a,h,da, r,g,b)
            call dlinrgb_(r,g,b)
            call dfilzon_
            call ppagtrie(orix+x,oriy+y,da,1)
            call pfilzon_
            if(icom.eq.1) then
               call dlincol_('n')
               xx=orix+x+1.5*cc
               yy=oriy+y-cc
               call ppagval_(xx,yy+cc2,-1,cc,cc,0.,r,'(f3.1)')
               call ppagval_(xx,yy    ,-1,cc,cc,0.,g,'(f3.1)')
               call ppagval_(xx,yy-cc2,-1,cc,cc,0.,b,'(f3.1)')
            endif
         enddo
c *   x loop for sub-triangles down
         do ix=1,ncelh-1
            x=float(ix-1)*da +float(ih-1)*da/2. +da/2.
            y=float(ih-1)*dh +dh
            call calrgb(x,y,a,h,da, r,g,b)
            call dlinrgb_(r,g,b)
            call dfilzon_
            call ppagtrie(orix+x,oriy+y,da,-1)
            call pfilzon_
         enddo
      enddo
c *** redraw line contour of the main triangle
      call dlincol_('n')
      call ppagtrie(orix,oriy,a,1)
c
      return
      end
```

Listing 3.2: *Program to plot Maxwell's color triangle*

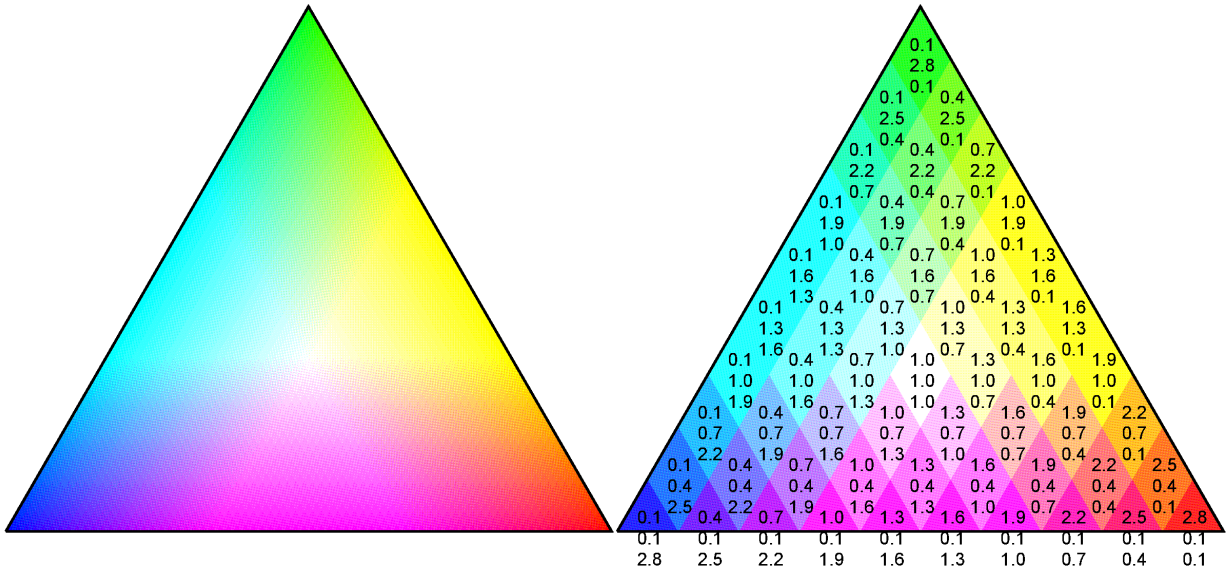


Fig. 3.2: *Maxwell's color triangle (from program 3.2)*

```
      subroutine ppagtrie(xori,yori,a,idir)

c *   plot_figure_triangle_equilateral ; idir=1 -> up, idir=-1 -> down

      real x(4), y(4)

      h= a*sqrt(3.)/2.
      x(1)=xori
      y(1)=yori
      x(2)=xori +a
      y(2)=yori
      x(3)=xori +a/2.
      y(3)=yori +float(idir)*h
      x(4)=x(1)
      y(4)=y(1)

      call dpagppo_(x(1),y(1))

      do i=2,4
         call ppagpmo_(x(i),y(i))
      enddo

      return
      end

c     ----------------------------------------------------------------0--

      subroutine calrgb(x,y,a,h,da, r,g,b)

c *   compute color of the sub-triangle from its position

      xc= x +da/2.
      yc= y+da*sqrt(3.)/6.
      g=2.*yc/(h*sqrt(3.))
      g=yc/h
      r=abs(xc/a -g/2.)
      b=1.-r-g
      r=r*3.
      g=g*3.
      b=b*3.

      return
      end
```

3.4 Chromatic wheel

The color wheel is a way to represent colors using HSB components. The figure 3.3, produced by the program given in the listing 3.3, gives an idea of these colors. In this drawing, gloss B has always been taken to its maximum 1.

```
      program p_color_wheel
c     ------------------------------------------------------------------0--
c     P. Robert, ScientiDev, 2008
c     ------------------------------------------------------------------0--
      call dopegra_(1,'p_color_wheel.ps')
c
      radius=6.
      iarc=10
      pisd=acos(-1.)/180.
      tc=radius/20.

      orix=9.
      oriy=10.
c
c *** draw line contour of the main wheel
      call dlincol_('n')
      call ppagcir_(orix,oriy,radius,0.,360.,1.)
c
c *** plot and fill  of all sub-wheels
c
c *   loop on the angle
c
      do itet=1,360,iarc
         tet=float(itet-1)
         hue=float(itet-1)/359
         aa=tet
         rr=radius+tc
         tt=tet+3.
         if(tet.gt.90. .and. tet.lt.270.) then
            aa=tet+180.
            rr=rr+3.*tc
            tt=tet+6.
         endif
         xx=orix+rr*cos(tt*pisd)
         yy=oriy+rr*sin(tt*pisd)
         call dlincol_('n')
         call ppagval_(xx,yy,-1,tc,tc,aa,hue,'(f5.2)')
c
c *      loop on the radius
c
         do ir=int(radius),1,-1
            rad=float(ir)
            bri=1.
            sat=float(ir)/radius
            call dlinhsb_(hue,sat,bri)
            call dfilzon_
            call p_sector_part(orix,oriy,rad,rad-1.,tet,tet+10.,1.)
            call pfilzon_
         enddo
      enddo
c
c* plot sat values
c
      call dlincol_('n')
      do ir=int(radius),1,-1
         rad=float(ir)
         sat=float(ir)/radius
         xx=orix+rad-0.8
         call ppagval_(xx,oriy,-1,tc*0.7,tc,0.,sat,'(f5.2)')
      enddo

      call curved_title(orix,oriy,radius+3.)
      call ppagarr_(orix+radius+2.,oriy,-1,2.,0.2,0.6,0.5,90.)
      call dfontyp_('Arial-B')
      tc=tc*1.4
      call ppagcha_(orix+radius+2.5,oriy,-1,tc,tc,0.,'Hue')
      call ppagcha_(orix+radius/2.,oriy-0.7,-1,tc,tc,0.,'Sat')
c
      call dclogra_
c
      return
      end
```

Listing 3.3: *Program to plot a chromatic wheel*

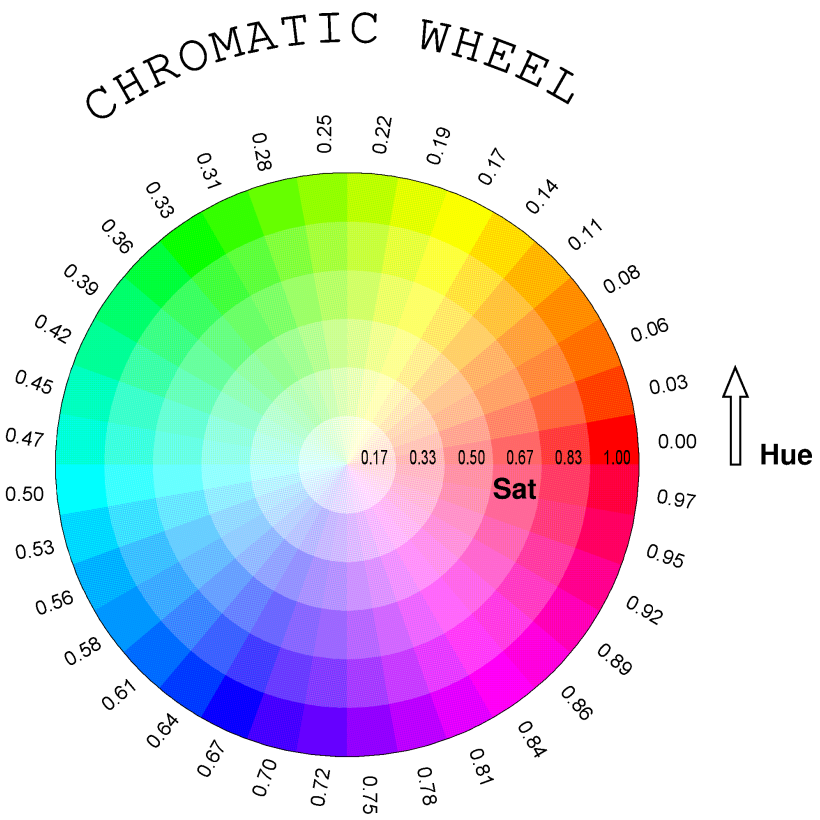


Fig. 3.3: *Chromatic wheel with HSB components (Bri=1.) (from program 3.3)*

```
c     ------------------------------------------------------------------0--
      subroutine p_sector_part(orix,oriy,rad1,rad2,tet1,tet2,dtet)

      call ppagcir_(orix,oriy,rad1,tet1,tet2,dtet)
      call gpagppo_(ox,oy)
      call ppagcir_(orix,oriy,rad2,tet2,tet1,-dtet)

      return
      end
c     ------------------------------------------------------------------0--
      subroutine curved_title(orix,oriy,rr)

      character*80 title

      pisd=acos(-1.)/180.

      title='CHROMATIC WHEEL'

      call cchalen_(title,n)
      call dlincol_('n')
      call dfontyp_('Courier')

      tct=rr/15.
      teta=360.*tct*float(n)*1.2/(6.283*rr)
      dteta=teta/float(n)
      teta1=90.+teta/2.

      do i=1,n
         tet=teta1-float(i-1)*dteta
         xx=orix +rr*cos(tet*pisd)
         yy=oriy +rr*sin(tet*pisd)
         call ppagcha_(xx,yy,0,tct,tct,tet-90.,title(i:i))
      enddo

      return
      end
```

3.5 Color maps

Rogralib provides various color maps. Figure 3.4 show all pre-defined color maps. The user can also build himself his own color map with the subroutine **dcolmapu** (see 5.2.6). Source program used to produce Fig. 3.4 is given in listing 3.4.

```
      program p_color_maps
c     ------------------------------------------------------------------0--
c *   check available color maps
c     P. Robert, November 1997, revised Juin 2006
c     ------------------------------------------------------------------0--
      character*10 paldef(25),colmap
      character*14 vercha
      data orix,oriy /2.5,25.6/
      data sizx,sizy /16.,0.8/
      data isens,decay /1,0.2 /
c
      call dopegra_(1,'p_color_maps.ps')
      call gvercha_(vercha)
      call ppaghea_(vercha//' - ')
      call ppagfoo_('P. Robert')
      call gcolmapa(paldef,n)
c
c *   classic022 and spectro color maps
      do i=1,7
         call dcolmap_(paldef(i))
         call ppagcmah(orix,oriy,sizx,sizy,isens)
         call ppagrec_(orix,oriy,sizx,sizy)
         call ppagcha_(orix/2.,oriy+sizy/2.,0,0.25,0.25,0.,paldef(i))
         if(i.eq.1) then
                 call gcolmap_(colmap,nlevel)
                 do n=1,nlevel
                     rlevel=float(nlevel)
                     xx=orix +sizx/(2.*rlevel) +float(n-1)*sizx/(rlevel)
                     yy=oriy +sizy +0.5
                     anum=float(nlevel-n+1)
                     call ppagval_(xx,yy,0,0.2,0.2,0.,anum,'(i2)')
                 enddo
         endif
         oriy=oriy -sizy -decay
      enddo

c *   rainbow color maps
      oriy=oriy -decay
      do i=8,13
         call dcolmap_(paldef(i))
         call ppagcmah(orix,oriy,sizx,sizy,isens)
         call ppagrec_(orix,oriy,sizx,sizy)
         call ppagcha_(orix/2.,oriy+sizy/2.,0,0.25,0.25,0.,paldef(i))
         oriy=oriy -sizy -decay
      enddo

c *   full hue color maps
      oriy=oriy -decay
      do i=14,19
         call dcolmap_(paldef(i))
         call ppagcmah(orix,oriy,sizx,sizy,isens)
         call ppagrec_(orix,oriy,sizx,sizy)
         call ppagcha_(orix/2.,oriy+sizy/2.,0,0.25,0.25,0.,paldef(i))
         oriy=oriy -sizy -decay
      enddo

c *   grey levels color maps
      oriy=oriy -decay -0.3
      do i=20,25
         call dcolmap_(paldef(i))
         call ppagcmah(orix,oriy,sizx,sizy,isens)
         call ppagrec_(orix,oriy,sizx,sizy)
         call ppagcha_(orix/2.,oriy+sizy/2.,0,0.25,0.25,0.,paldef(i))
         if(i.eq.20) then
                 call gcolmap_(colmap,nlevel)
                 do n=1,nlevel
                     rlevel=float(nlevel)
                     xx=orix +sizx/(2.*rlevel) +float(n-1)*sizx/(rlevel)
                     yy=oriy +sizy +0.12
                     anum=float(nlevel-n+1)
                     call ppagval_(xx,yy,0,0.2,0.2,0.,anum,'(i2)')
                 enddo
         endif
         oriy=oriy -sizy -decay
      enddo
```

```
      call dclogra_
      stop 'OK p_color_maps'
      end
```

Listing 3.4: *Program to plot all pre-defined color maps*

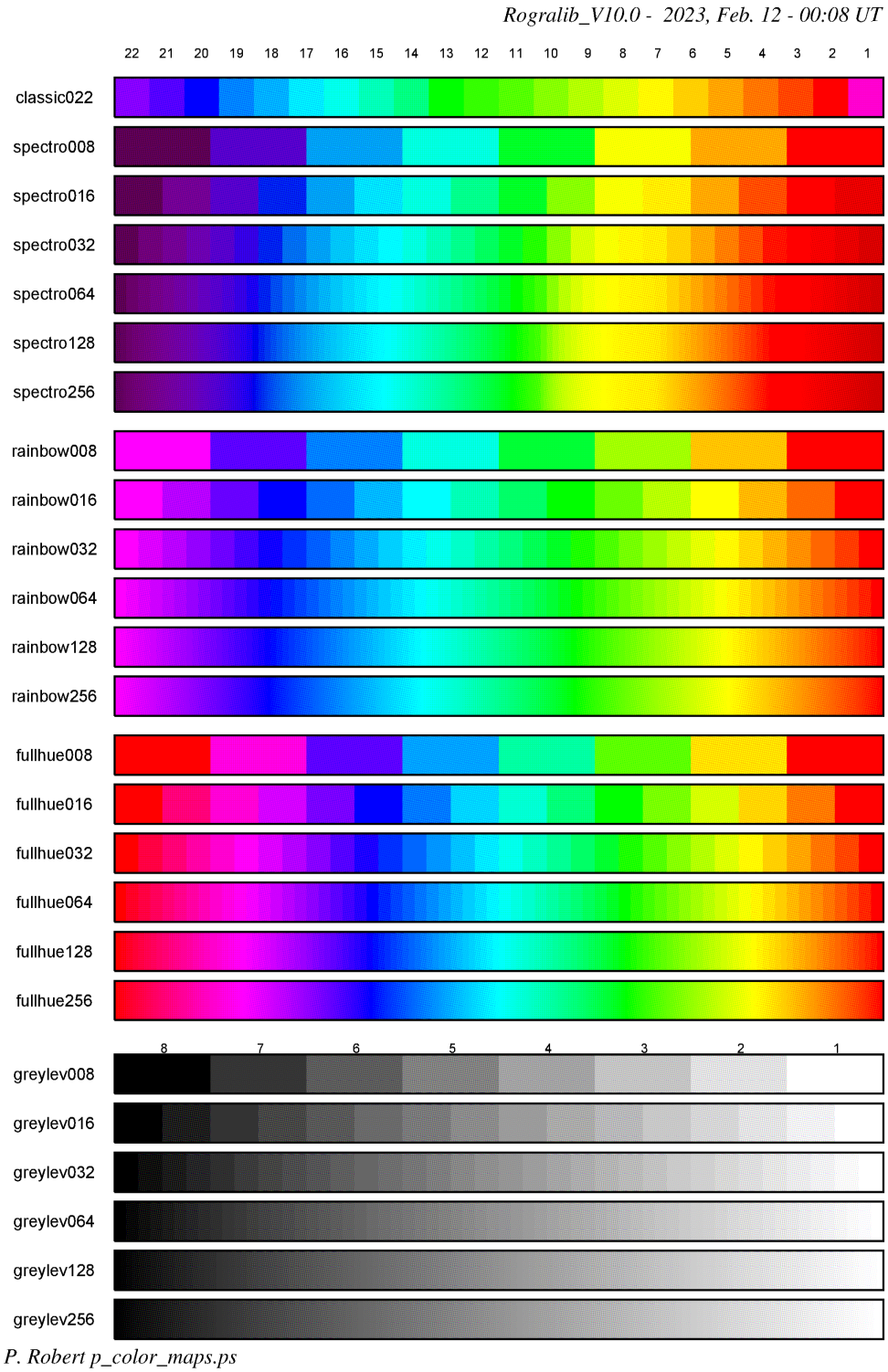


Fig. 3.4: *Predefined color maps available in Rogralib*

3.6 Work on images

3.6.1 Image definition

The color of each pixel of an image can be define by 3 values: Red, Green, Blue definition, or Hue, Saturation, Brightness definition. But for image definition, it is more useful to define the pixel color by a single and integer value: this is what we call a **ico** unit.

ico unit is defined as: `ico= 65536*ir + 256*ig + ib`

with ir, ig, ib are Red, Green, Blue definition, in [0-255] range. **ico** defines in a single value a color among 16 M (256**3) of colors. Example: for (ir,ig,ib) = (127,36,25) then ico = 8332313. Image is a two-dimensional integer array, as ima(nx,ny), whose the value is a color in ico value. The program in listing 3.5 show how define and plot a simple image.

3.6.2 Image tools

- **cscamul**`_(sca,nx,ny,rmul)` : **convert_scalar_image_by_multiplication**

  Input : `sca,nx,ny,rmul` → sca(nx,ny), multiplicative factor
  Output: `sca` → sca(nx,ny)
- **cimahsy**`_(ima,nx,ny)` : **convert_image_to_horizontal_symmetry**

  Input : `ima,nx,ny` → image in ico unit
  Output: `ima` → horizontal symmetric image
- **cscaima**`_(sca,nx,ny,scamin,scamax,ima,sbla,swhi)` : **convert_scalar_to_image**

  Input : `sca,nx,ny,scamin,scamax,sbla,swhi` → sca(nx,ny)
  Output: `ima` → integer ima(nx,ny) in ico unit

  We spread the image between the scamin and scamax values of sca(nx,ny).
  We put in black what is lower than sbla (threshold of black) while we put in white what is greater than swhi (threshold of white). No black and white if sbla=swhi
- **cscalog**`_(sca,nx,ny,valzer,valneg)` : **convert_scalar_image_to_log_values**

  Input : `sca,nx,ny,valzer,valneg` → sca(nx,ny), valzer, valneg
  Output: `sca` → sca(nx,ny) in log values, valzer if sca < 1.e-20, valneg if sca <0.
- **cimasizc**`(cima,nxmax,nx,ny)` : **convert_image_size_character**

  Input : `cima,nxmax,nx,ny` → character type cima(N) with N > nxmax*ny
  Output: `cima` → character type cima(nx,ny)

  allows to use an array dimensioned as (nx,ny) in a subroutine while it is dimensioned as (N) in the main program.It can then no longer be used normally in the main program.
  Useful in F77 only, when no dynamic memory available (not used is F90).
- **cimasizi**`(iima,nxmax,nx,ny)` : **convert_image_size_integer**

  Input : `iima(nxmax*ny)` → integer type iima(N) with N > nxmax*ny
  Output: `iima(nx,ny)` → integer type iima(nx,ny)

  same remark as previous
- **cimasizr**`(rima,nxmax,nx,ny)` : **convert_image_size_real**

  Input : → real type rima(N) with N > nxmax*ny
  Output: → real type rima(nx,ny)

  same remark as previous
- **rdimbmp**`_(ifc,bmpname,nx,ny)` : **read_dimension_bmp**

  Input : → file code, bmp file name
  Output: → image dimension nx,ny

- **rimabmp_(ifc,bmpname,image,nx,ny) : read_image_bmp**
  Input : → file code, bmp file name, empty array image(nx,ny)
  Output: → image(nx,ny) in ico values
- **wimabmp_(ifc,bmpname,image,nx,ny) : write_image_bmp**
  Input : → file code, bmp file name, image(nx,ny)
  Output: → a file named bmpname will be created.

3.6.3 Image plot

Image plot is done by the subroutine **ppagima_**. Program given in listing 3.5 show how define and plot a simple image. Result is shown on figure 3.5

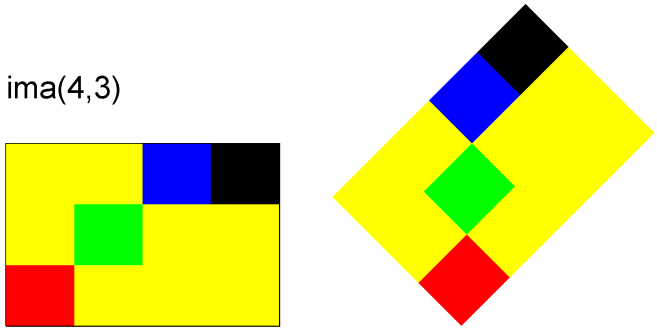


Fig. 3.5: *Example of plot of a simple image*

```
      program p_image
      integer image(4,3)
c     ------------------------------------------------------------------0--
c     Check subroutine ppagima_ de rogralib
c     P. Robert, January 1996
c     ------------------------------------------------------------------0--
      call dopegra_(1,'p_image.ps')
c
   ........... image definition, size, position,   etc... ........
c
c *** yellow everywhere (ico unit)
      do i=1,nx
         do j=1,ny
            call crgbico_(255,255,0,image(i,j))
         enddo
      enddo
c
c *** red, green, blue, black for some pixels
      call crgbico_(255,  0,  0,image(1,1))
      call crgbico_(  0,255,  0,image(2,2))
      call crgbico_(  0,  0,255,image(3,3))
      call crgbico_(  0,  0,  0,image(4,3))
c
c *** Plot image and title
      call ppagrec_(orix,oriy,sizx,sizy)
      call ppagima_(orix,oriy,ipos,sizx,sizy,0.,image,nx,ny)
      call ppagcha_(orix,oriy+sizy+1.,-1,0.5,0.5,0.,'ima(4,3)')
c
c *** same with rotated image
      call ppagima_(12.,oriy,ipos,sizx,sizy,45.,image,nx,ny)
c
      call dclogra_
      stop 'OK p_image'
      end
```

Listing 3.5: *Program to define and plot a simple image*

3.6.4 Plot a 2D function as an image

In this example we plot the amplitude of a 2D function via a color map. The function is defined in listing 3.6 below:

```
      subroutine compute_fonc2D(fonc,nx,ny, x1,x2,y1,y2)

      real fonc(nx,ny)

      dx= (x2-x1)/float(nx)
      dy= (y2-y1)/float(ny)

      do ix=1,nx
      do iy=1,ny
         x= x1+float(ix-1)*dx
         y= y1+float(iy-1)*dy
         fonc(ix,iy)= 10.*exp(-x**2)*exp(-y**2)
      enddo
      enddo
c
      return
      end
```

Listing 3.6: *subroutine defining a simple 2D function*

The program to compute the corresponding image and to plot it is given in listing 3.6. Result is shown on figure 3.6.

```
      program p_fonc_2D
c
c     ------------------------------------------------------------------0--
c     Plot a 2D function as an image; P. Robert, July 2010
c     ------------------------------------------------------------------0--
c
      parameter (nx=200,ny=100)

      real fonc(nx,ny)
      integer image(nx,ny)
      character*8 fx,fy
      character*32 labx,laby,title

      data ox,oy /3.,10./
      data sx,sy /14.,14./
      data x1,x2 /-6.,6./
      data y1,y2 /-6.,6./
      data labx,laby,title /'X','Y','f(x,y)=10*exp(-x**2)*exp(-y**2)'/

c *** compute 2D function

      call compute_fonc2D(fonc,nx,ny, x1,x2,y1,y2)

c *** open graphical file

      call dopegra_(7,'p_fonc_2D.ps')
c
      print*
      print*, 'function converted in log values'
c
      valzer=1.e-20
      valneg=1.e-30
c
      print*, 'zero      values set to ',valzer
      print*, 'negatives values set to ',valneg
c
      call cscalog_(fonc,nx,ny,valzer,valneg)

c *** compute min and max of the function in the given interval
c
      call cmatlim_(fonc,nx,ny,smin,smax)
c
c *** select color map
c
```

```
      call dcolmap_('spectro256')
c
c *** image computation; one force smin to be > log(1.e-15)
c
      if(smin.lt. -15.) smin=-15.
c
      sbla=-15.
      swhi= 10.
c
      call cscaima_(fonc,nx,ny,smin,smax,image,sbla,swhi)
      call ppagima_(ox,oy,-1,sx,sy,0.,image,nx,ny)
c
c *   plotting color map above the figure
c
      call dfontyp_('h')
      call ppagcmah(12.0, oy+sy+0.5, 5.,1.,1)
      call ppagval_(11.9, oy+sy+1.9, 0,0.4,0.4,0.,smin,'(f5.1)')
      call ppagval_(16.9, oy+sy+1.9, 0,0.4,0.4,0.,smax,'(f5.1)')
c
c *** computation and plot of the axis graduations
c
      print*, 'x1,x2=',x1,x2
      print*, 'y1,y2=',y1,y2
c
      call cfiggra_(x1,x2,x1a,x2a,bgx,sgx,fx)
      call cfiggra_(y1,y2,y1a,y2a,bgy,sgy,fy)
c
      call dfigori_(ox,oy)
      call dfigsiz_(sx,sy)
      call dfiglim_(x1,x2,y1,y2)
      call dstipos_('oo','oo')
      call dgrasiz_(0.5,0.5)
c
      call pfigfra_
      call pfiggrax(0.,bgx,sgx,fx)
      call pfiggray(0.,bgy,sgy,fy)
      call pfiglabx(labx)
      call pfiglaby(laby)

      call dclogra_
c
      stop 'OK p_fonc_2D'
      end
```

Listing 3.7: *Program to compute and plot an image from the the function 3.6*

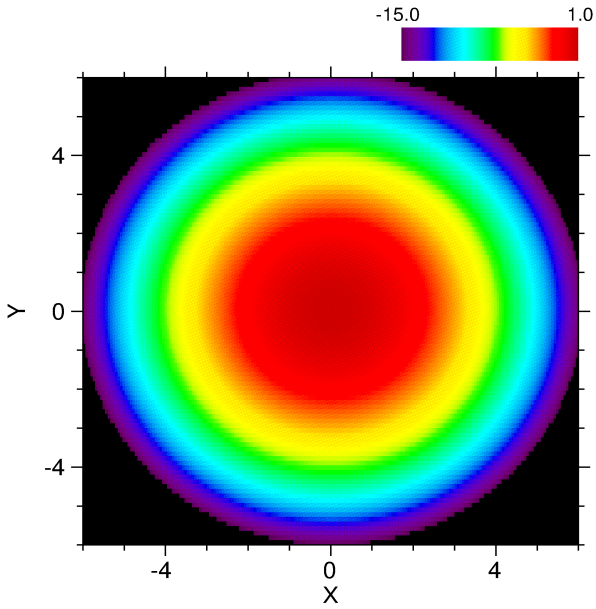


Fig. 3.6: *Plot of an image corresponding to a 2D function (from program 3.7).*

3.6.5 Read and plot bitmap image

This program use the following subroutine:

rdimbmp_(1,'./data/golden_roc.bmp',nx,ny) : **read_dimension_bmp**
rimabmp_(1,'./data/golden_roc.bmp',imaico,nx,ny) : **read_image_bmp**
ppagima_(ox,oy,-1,sx,sy,0.,imaico,nx,ny) : **plot_page_image_bmp**

Program source is given in listing 3.8, and result is shown on figure 3.7.

```
      program p_image_bmp
c
      integer imaico(2048,1024)
c
c     ------------------------------------------------------------------0--
c     check subroutine rimabmp_ by reading and plot the image
c     P. Robert, October 1997
c     ------------------------------------------------------------------0--
c
c *** open PS file and page attribute
      call dopegra_(7,'p_image_bmp.ps')
c
c *** reading bmp file and convert image to ico format
      call rdimbmp_(1,'./data/golden_roc.bmp',nx,ny)
      call rimabmp_(1,'./data/golden_roc.bmp',imaico,nx,ny)
c
c *** plot image
      sx=float(nx)/100.
      sy=float(ny)/100.
c
      call ppagima_(2.,15.,-1,sx,sy,0.,imaico,nx,ny)
      call dclogra_
c
      stop 'OK p_image_bmp'
      end
```

Listing 3.8: *Program to read and plot a bmp image.*

Fig. 3.7: *Read and plot on page a bitmap image (from program 3.8).*

3.6.6 Save an ico image in bmp format

When you have computed an image and plot it, as for instance the image plotted in figure 3.6 you can save this image in bitmap format (.bmp) as it is shown in listing 3.9

```
c
c *** select color map
c
      call dcolmap_('spectro256')
c
      call cscaima_(fonc,nx,ny,smin,smax,image,sbla,swhi)
      call ppagima_(ox,oy,-1,sx,sy,0.,image,nx,ny)
c
c *** sauvegarde de l'image
c
      call wimabmp_(3,'p_fonc_2D.bmp',image,nx,ny)
c
```

Listing 3.9: *Program to save a bmp image*

In another program, you can read again this image and plot it as it is explained in the previous section 3.6.5 and you will obtain the image of figure 3.8. Note that only the image is saved and plotted.

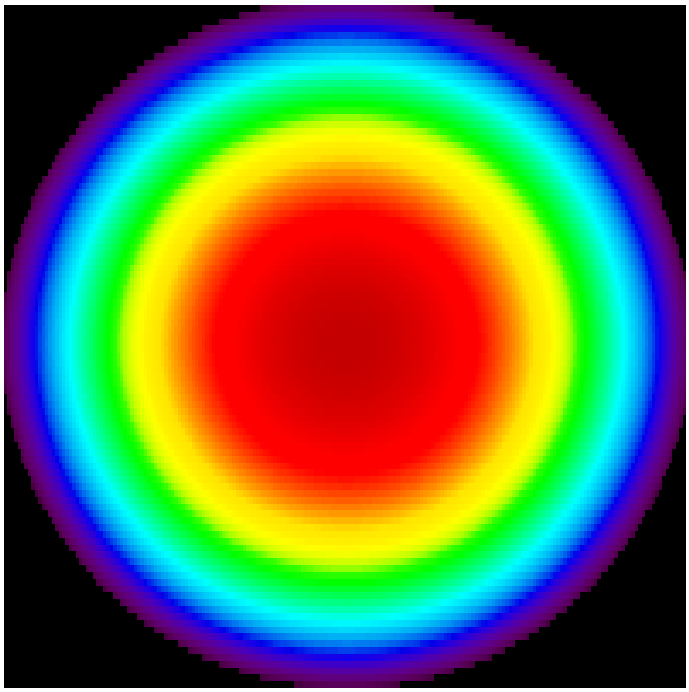


Fig. 3.8: *Read and plot a previously saved bitmap image*

Chapter 4 : MORE ADVANCED PLOTS

4.1 Figure with curves in dashed lines

This program use the subroutine **dlindas_**(size1,blan1,size2,blan2) : **define_line_dash** Arguments are size of dashes and blanks in the line. Note that back to normal line can be done by dlincon_.

```
      program p_dash_line

........ define figure parameters ..................
c
c *** plot of the curves (x,y) with different dashed lines
c
      call dlinwid_(5)
      call pfigcur_(x,y1,nbp)
      call pfigcha_(10.,y1(nbp),-1,tc,tc,0.,' a=1')

      call dlindas_(0.2,0.5,0.2,0.5)
      call pfigcur_(x,y2,nbp)
      call pfigcha_(10.,y2(nbp),-1,tc,tc,0.,' a=2')

      call dlindas_(0.2,0.2,0.4,0.2)
      call pfigcur_(x,y3,nbp)
      call pfigcha_(10.,yf2*0.95,-1,tc,tc,0.,' a=4')

      call dlindas_(0.1,0.2,0.5,0.2)
      call pfigcur_(x,y4,nbp)
      call pfigcha_(7.,yf2*0.95,-1,tc,tc,0.,' a=8')
c
........... end ..............
```

Listing 4.1: *Program to show how use dash line option in a figure*

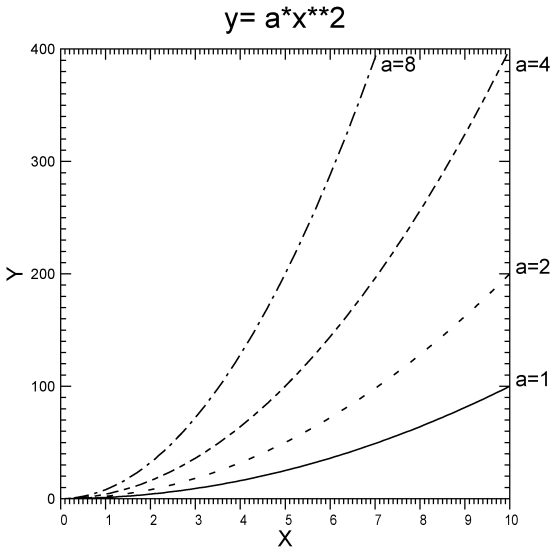


Fig. 4.1: *Figure using dash line option (from program 4.1)*

4.2 Figure with logarithmic graduations

Program given on listing 4.2 show how to use logarithmic axis on a figure. Note that the order of the call is important: you must define the logarithmic style of the Y axis (by `dgrasty_(0,1)`) **before** calling the computation of the logarithmic graduations (by `cfiggral`), and define after the limits of the Y axis (by `dfiglimy`).

```
      program p_figure_log
c
      real x(100), y(100)
      character*8 fx,fy
c
      call dopegra_(1,'p_figure_log.ps')
      call dlinwid_(3)
c
c *** define a curve to be plotted
c
      do i=1,100
         x(i)=float(i)/25.
         y(i)=10.**x(i)
      enddo
c
c *** define size and position on the figure on the current page
c
      call dfigsiz_(14.,5.5)
      call dfigori_(4.,20.5)
      call dtitsiz_(0.45)
      call dgrasiz_(0.4,0.35)
c
      print*, 'Plot figure 1, linear axis'
c
      call cfiggra_(x(1),x(100),x1,x2,bgx,sgx,fx)
      call cfiggra_(y(1),y(100),y1,y2,bgy,sgy,fy)
      call dfiglimx(x1,x2)
      call dfiglimy(y1,y2)
c
      call pfigfra_
      call pfigtit_('Figure 1: Lin-Lin scale')
      call pfiggrax(0.,bgx,sgx,fx)
      call pfiggray(0.,bgy,sgy,fy)
      call pfiglabx('X (linear)')
      call pfiglaby('Y (linear)')
      call pfigcur_(x,y,100)
c
      print*, 'Plot figure 2, y axis in log scale'
c
      call dgrasty_(0,1)
      call cfiggral(y(1),y(100),y1,y2)
      call dfiglimy(y1,y2)
      call dfigoriy(11.5)
      call dgrasiz_(0.4,0.6)
c
      call pfigfra_
      call pfigtit_('Figure 2: Lin-Log scale')
      call pfiggrax(0.,bgx,sgx,fx)
      call pfiggrly(1)
      call pfiglabx('X (linear)')
      call pfiglaby('Y (logarithmic)')
      call pfigcur_(x,y,100)
c
      print*, 'Plot figure 3, x and y axis in log scale'
c
      call dgrasty_(1,1)
      call cfiggral(x(1),x(100),x1,x2)
      call dfiglimx(x1,x2)
      call dfigoriy(2.5)
      call dgrasiz_(0.6,0.6)
c
      call pfigfra_
      call pfigtit_('Figure 3: Log-Log scale')
      call pfiggrlx(1)
      call pfiggrly(1)
      call pfiglabx('X (logarithmic)')
      call pfiglaby('Y (logarithmic)')
      call pfigcur_(x,y,100)
c
      call dclogra_
      stop 'OK p_figure_log'
      end
```

Listing 4.2: *Program to plot curves in log scale*

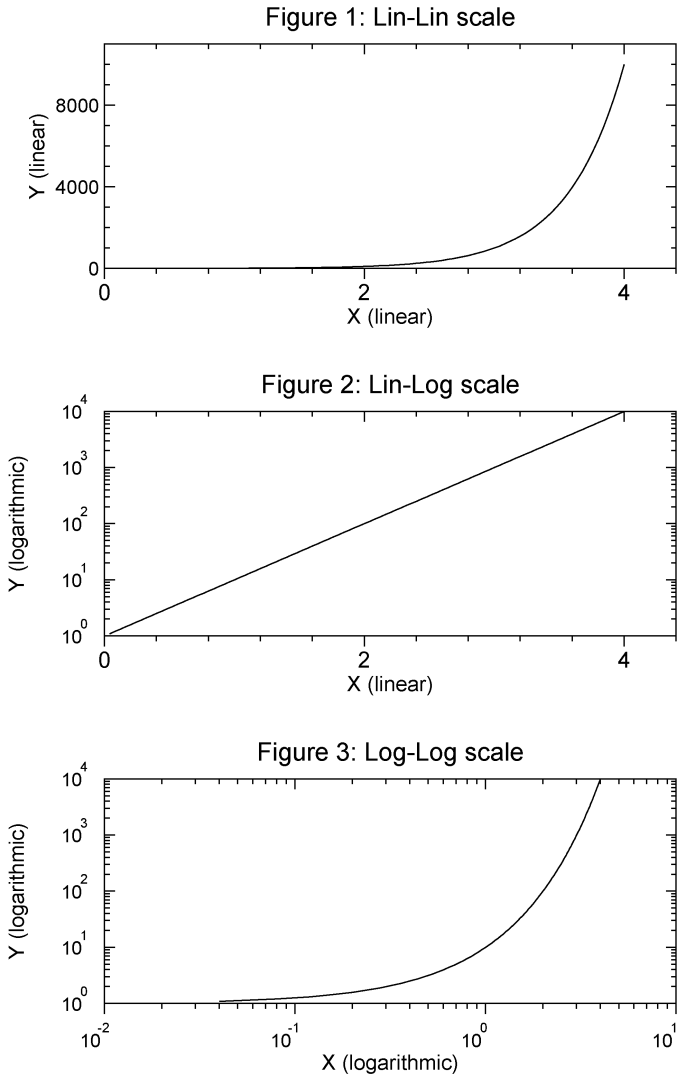


Fig. 4.2: *Figure using logarithmic scale (from program 4.2)*

4.3 Spectrum and dynamic spectra

4.3.1 Spectrum

The follow program show how use the complex fast Fourier transform `ccpxfft_` to compute and plot the spectrum of a waveform. Result is gven on figure 4.3

```
      program p_spectrum
c
c     ------------------------------------------------------------------0--
c     Compute and plot input time signal and  corresponding spectra
c     P. Robert, 2015
c     ------------------------------------------------------------------0--
c
      parameter (nbp=512)
      real bx(nbp),tx(nbp),fx(nbp)
      complex sx(nbp)
c
      character*16 forx,fory
      data orix,oriy,sizx,sizy /3.,20.,15.,6./
```

```
c *** compute test signal
c
      pi=acos(-1.)
      amp=10.
      phi=45*pi/180.
      fe=25.
      df=fe/float(nbp)
      fre= 0.5
      m=int(f/df)
c
      do i=1,nbp
         tx(i)=float(i-1)/fe
         bx(i)=amp*sin(2.*pi*fre*tx(i) +phi)
         sx(i)=cmplx(bx(i),0.)
      enddo
c
c *** plot of test signal
c
      call dopegra_(7,'p_spectrum.ps')
c
      call gvercha_(vercha)
      call ppaghea_(vercha//' - ')
      call ppagfoo_('P. Robert')
      call ppagfra_
c
      call cminmax_(tx,nbp,x1,x2)
      call cminmax_(bx,nbp,y1,y2)
      call cfiggra_(x1,x2,x1a,x2a,bgx,sgx,forx)
      call cfiggra_(y1,y2,y1a,y2a,bgy,sgy,fory)
c
      call dfigsiz_(sizx,sizy)
      call dfiglim_(x1a,x2a,y1a,y2a)
      call dlabsiz_(0.4,0.4)
      call dfigori_(orix,oriy)
c
      call pfigfra_
      call pfigtit_('Pure sine waveform')
      call pfiggrax(0.,bgx,sgx,forx)
      call pfiggray(0.,bgy,sgy,fory)
      call pfiglabx('Temps (sec.)')
      call pfiglaby('Amplitude (nT)')
      call pfigcur_(tx,bx,nbp)
c
c *** plot of spectrum with y linear
c
      call ccpxfft_(sx,nbp, 1)
c
      do k=1,nbp/2
         fx(k)=float(k-1)*df
         bx(k)=cabs(sx(k))
         kk=nbp-k+1
         fx(kk)=-float(k)*df
         bx(kk)=cabs(sx(kk))
      enddo
c
c *   reorder frequency for plot
c
      do k=1,nbp/2
         fs=fx(k)
         bs=bx(k)
         kk=nbp/2+k
         fx(k)=fx(kk)
         bx(k)=bx(kk)
         fx(kk)=fs
         bx(kk)=bs
      enddo
c
      call cminmax_(fx,nbp,x1,x2)
      call cminmax_(bx,nbp,y1,y2)
      call cfiggra_(x1,x2,x1a,x2a,bgx,sgx,forx)
      call cfiggra_(y1,y2,y1a,y2a,bgy,sgy,fory)

      call dfigoriy(11.1)
      call dfiglim_(x1a,x2a,y1a,y2a)
c
      call pfigfra_
      call pfigtit_('Spectrum')
      call pfiggrax(0.,bgx,sgx,forx)
      call pfiggray(0.,bgy,sgy,fory)
      call pfiglabx('Frequence (Hz)')
      call pfiglaby('Amplitude (nT)')
      call dfilzon_
      call dlincol_('c')
      call pfigcur_(fx,bx,nbp)
      call pfilzon_
      call dlincol_('n')
      call pfigcur_(fx,bx,nbp)
```

```
c
c *** plot of spectrum with y logaritmic
c
      call dgrasty_(0,1)
      call dfiglimy(0.001,10.)
      call dfigoriy(2.3)
c
      call pfigtit_('Spectrum, log amplitude')
      call pfiggrax(0.,bgx,sgx,forx)
      call pfiggrly(1)
      call pfiglabx('Frequence (Hz)')
      call pfiglaby('Amplitude (nT)')
      call dfilzon_
      call dlincol_('c')
      call pfigcur_(fx,bx,nbp)
      call pfigpmo_(fx(nbp),0.001)
      call pfigpmo_(fx(1),0.001)
      call pfilzon_
      call dlincol_('n')
      call pfigcur_(fx,bx,nbp)
      call pfigfra_
c
      call dclogra_
c
      stop 'OK p_spectrum'
      end
```

Listing 4.3: *Program to plot wave form and its spectrum*

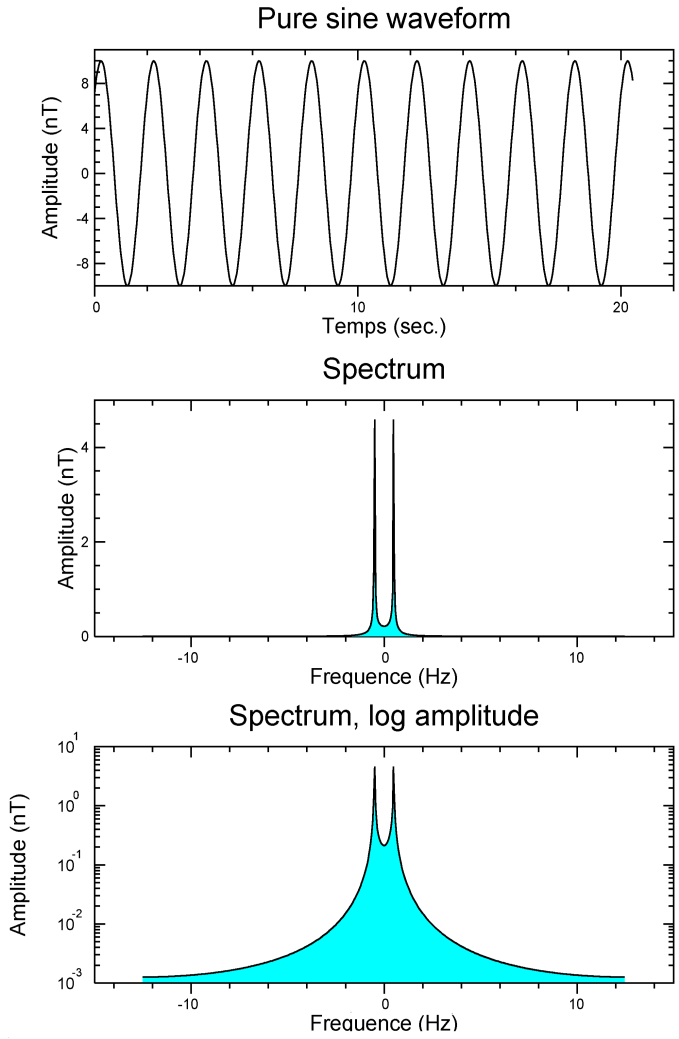


Fig. 4.3: *Plot a wave form and its spectrum (from program 4.3)*

4.3.2 Dynamic spectra

The follow program show how use the complex fast Fourier transform `ccpxfft_` subroutine to compute and plot the dynamic spectra of a long time series. Result is given on figure 4.4

This program also show how to use the **cimasizr** subroutine, used in F77 for array rearrangement because there is no dynamic memory in f77. After the call **cimasizr**`(speb,nxmax,nx,ny)`, initial array `speb(nxmax,ny)` can be view as `speb(nx,ny)` (see `plot_spectro` subroutine).

In f90, it would be better to use allocated arrays, and replace `speb(nxmax,nymax)` by `speb(nbp,ny)`, where `nbp` and `ny` are computed values (see beginning of program).

Full listing can be viewed in the delivery package, but as it is long, only the main program and `plot_spectro` subroutines are given.

```
      program p_spectro
c
c     ------------------------------------------------------------------0--
c     Compute and plot dynamic spectra of a time serie
c     which is a summ of 2 monochromatic waves
c     P. Robert, 2007, revised October 2021 for inclusion in test
c     ------------------------------------------------------------------0--
c
      parameter (nbp=51200)
      parameter (nxmax=200, nymax=1024)
c
      real sb1(nbp),sb2(nbp),sbt(nbp)
      real speb(nxmax,nymax),smob(nymax)
      complex cwork(2*nymax)
      integer ima(nxmax,nymax)
c
c *** Input signal computation
c
      call co_signal(sb1,sb2,sbt,nbp,fe)
c
c *** window width
c
      nbpwin=512
      nx=nbp/nbpwin
      ny=nbpwin/2
c
      if(nx.gt.nxmax) stop '*** spectro ABORTED ! nx > nxmax'
      if(ny.gt.nymax) stop '*** spectro ABORTED ! ny > nymax'
c
c *** spectrogram computation
c
      call co_spectrog(sbt,nbp,fe,cwork,nbpwin,durwin,speb,nxmax,nx,ny)
c
c *** calcul du spectre moyen
c
c *   ta and tb time interval for averaged spectrum computation
c
      ta= 100.
      tb= 400.
c
      call co_spectrum(speb,nxmax,nx,ny,ta,tb,durwin,smob)
c
c *** plot of waveform, spectrogram and averaged spectrum
c
      call dopegra_(7,'p_spectro.ps')
      call gvercha_(vercha)
      call ppaghea_(vercha//' - ')
      call ppagfoo_('P. Robert')
      call ppagfra_
      call dfontyp_('h')
c
      call plot_signal(sbt,nbp,fe,100)
      call plot_spectro(speb,ima,nx,ny,nxmax,nbp,nbpwin,fe)
      call plot_spectrum(smob,ny,fe,ta,tb)
c
      call dclogra_
      stop 'OK p_spectro'
      end
```

```
      subroutine plot_signal(bx,nbp,fe,nvis)
c     ---------------------------------------------------------------0--
c     Plot input signal versus time.
c     P. Robert, 1999,revised October 2021 for inclusion in test package
c     ---------------------------------------------------------------0--
      real bx(nbp)
      character*8 fx,fy
      data orix, oriy / 3.0, 21.5/
      data sx,sy /15.0,4.0/
c
c *** min max time
      dx=1./fe
      x1=0.
      x2=float(nvis)*dx
      call cminmax_(bx,nvis,y1x,y2x)
c
      y1=y1x
      y2=y2x
c
c *   force min and max big sticks number
      call dstinum_(3,9)
      call cfiggra_(x1,x2,x1a,x2a,bgx,sgx,fx)
      call cfiggra_(y1,y2,y1a,y2a,bgy,sgy,fy)
      call dfigsiz_(sx,sy)
      call dfiglim_(x1,x2,y1,y2)
      call dlabsiz_(0.3,0.3)
c
      print*,'Plot of the figure for standard computed period'
c
      call dfigori_(orix,oriy)
      call pfigfra_
      call pfiggrax(0.,bgx,sgx,fx)
      call pfiggray(0.,bgy,sgy,fy)
      call pfiglabx('Time zoom from the starting point (s)')
      call pfiglaby('Amplitude (nT)')
      call pfigtit_('Wave form')
      call pfigcurd(x1,dx,bx,nvis)
c
      return
      end
c
c     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX0XX
c
      subroutine plot_spectrum(smob,ny,fe,ta,tb)
c     ---------------------------------------------------------------0--
c     Plot averaged spectrum smob
c     P. Robert, 1999,revised October 2021 for inclusion in test package
c     ---------------------------------------------------------------0--
      real smob(ny)
      character*8 fx,fy
c
      data orix /3.0/
      data oriy /2.3/
      data dy   / 0.6/
      data sx,sy /15.0,6.8/
c
      fmax=fe/2.
      deltaf=fmax/float(ny)
c
      call cminmax_(smob,ny,smin,smax)
c
      y1=smin
      y2=smax*1.05
c
      x1=0.
      x2=fmax
c
c *** Plot of the spectrum
      call cfiggra_(x1,x2,x1a,x2a,bgx,sgx,fx)
      call cfiggra_(y1,y2,y1a,y2a,bgy,sgy,fy)
      call dfigsiz_(sx,sy)
      call dfiglim_(x1,x2,y1,y2)
      call dfigori_(orix,oriy)
      call dlabsiz_(0.4,0.3)
      call dfontyp_('h')

      call pfigfra_
      call pfiggrax(0.,bgx,sgx,fx)
      call pfiggray(0.,bgy,sgy,fy)
      call pfiglabx('Frequency (Hz)')
      call pfiglaby('Power spectral density (nT2/Hz)')
      call pfigcurd(x1,deltaf,smob,ny)
      call pfigtit_('Average spectrum ')
      call ppagtav_(orix,oriy+sy+0.8,-1,0.3,0.3,0.,'t1=',ta,'(f5.1)')
      call ppagtav_(orix,oriy+sy+0.3,-1,0.3,0.3,0.,'t2=',tb,'(f5.1)')
c
      return
      end
```

```
      subroutine plot_spectro(speb,ima,nx,ny,nxmax,nbp,nwin,fe)
c     ------------------------------------------------------------------0--
c     Plot the 3 spectrograms x, y, z
c     P. Robert, 1999,revised October 2021 for inclusion in test package
c     ------------------------------------------------------------------0--
      real speb(nxmax,ny)
      integer ima(nxmax,ny)
      character*8 fx,fy
      character*160 sptit
      data orix / 3.0/
      data oriy /12./
      data dy   / 0.6/
      data sx,sy /15.0,6.5/
c
c *** set color map
      call dcolmap_('spectro256')
c
c *** set title/subtitle and plot
      fmax=fe/2.
      dt=1./fe
      deltaf=fmax/float(ny)
      deltat=1./deltaf
      write(sptit,100) nbp,fe,dt,nwin,nx,ny,deltat,deltaf
  100 format('nbp=',i6,' fe=',f8.3,' dt=',f8.3,' nwin=',i4,
     &       '  nx=',i5,'  ny=',i5,'  deltat=',f8.3,'  deltaf=',f9.5)
      call gpagsiz_(psix,psiy)
      yy=psiy-1.2
      call ppagcha_(orix,yy,-1,0.25,0.25,0.,sptit)
c
c *** array rearrangement
c     after this call, speb(nxmax,ny) cab be view as speb(nx,ny)
c     (no dynamic memory en f77)
      call cimasizr(speb,nxmax,nx,ny)
c
c *** conversion of spectra in log scale
      valzer=1.e-20
      valneg=1.e-30
      call cscalog_(speb,nx,ny,valzer,valneg)
c
c ***  min and max computation
      call cmatlim_(speb,nx,ny,smin,smax)
c
c *   % of blueshift and redshift for better spectrogram
      pcmin=  6.0
      pcmax=  0.5
c
c *   compute_scalar_significant_min_max
c
      call cscasli_(speb,nx,ny,smin,smax,pcmin,pcmax)
c
c *   plotting color map
c
      call ppagcmah(13.0, oriy+sy+0.3, 5.,0.5,1)
      call ppagval_(13.0, oriy+sy+0.9,-1,0.25,0.25,0.,smin,'(f4.1)')
      call ppagval_(17.2, oriy+sy+0.9,-1,0.25,0.25,0.,smax,'(f4.1)')
c
c *** computation of the image and plotting
c
      sbla=-20.
      swhi= 30.
c
      call cscaima_(speb,nx,ny,smin,smax,ima,sbla,swhi)
      call ppagima_(orix,oriy,-1,sx,sy,0.,ima,nx,ny)
c
c *** computation and plot of the axis graduations
c
      gt= 2.*float(ny)/fe
      df= 1./gt
      x1=0.
      x2= float(nx)*gt
      y1=0.
      y2= float(ny)*df
c
      call cfiggra_(x1,x2,x1a,x2a,bgx,sgx,fx)
      call cfiggra_(y1,y2,y1a,y2a,bgy,sgy,fy)
c
      call dfigsiz_(sx,sy)
      call dfiglim_(x1,x2,y1,y2)
      call dfigori_(orix,oriy)
      call dlabsiz_(0.4,0.3)
```

```
      call pfigfra_
      call pfiggrax(0.,bgx,sgx,fx)
      call pfiggray(0.,bgy,sgy,fy)
      call pfiglabx('Time (s)')
      call pfiglaby('Frequency (Hz)')
      call pfigtit_('Spectrogram')
c
      return
      end
```

Listing 4.4: *Program to compute and plot the dynamic spectra of a time series*

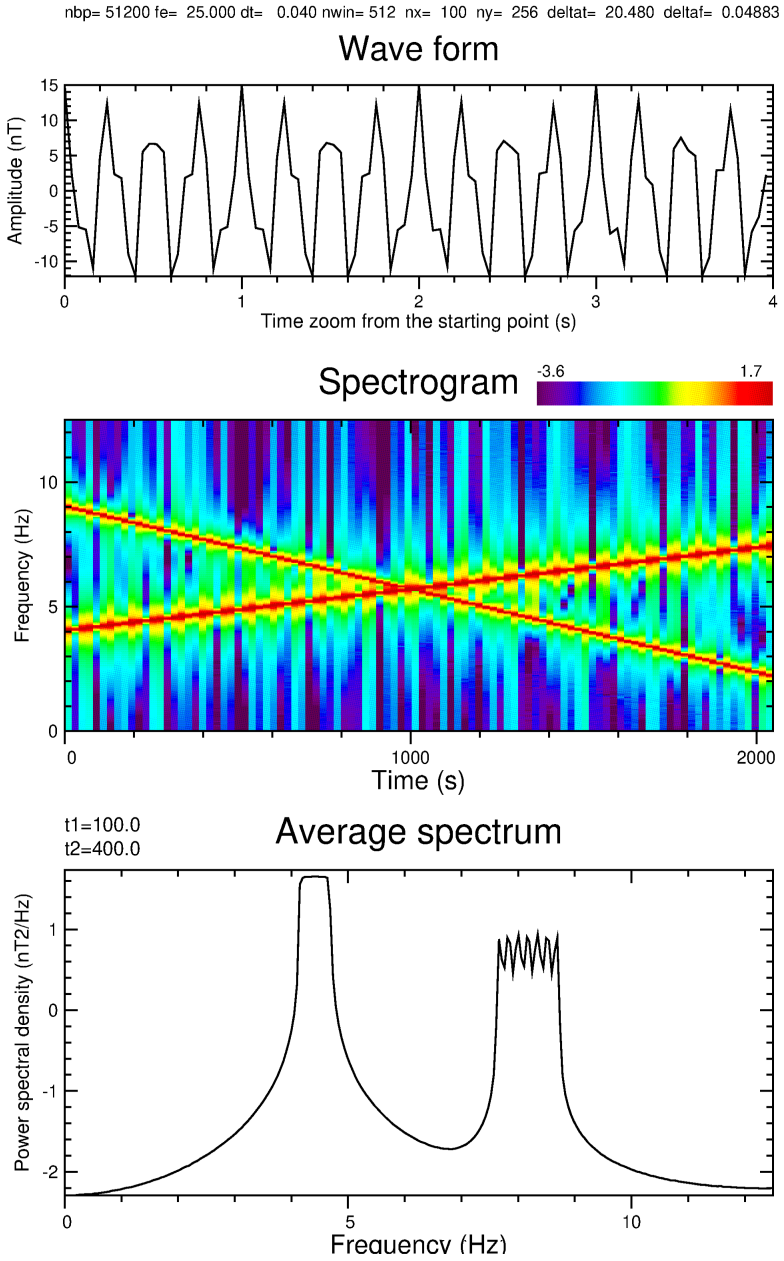


Fig. 4.4: *Plot a dynamic spectra (from program 4.4)*

4.4 Random bubbles

This program show an example to use color maps, HSB colors, sorted arrays and filled paths. All is done in the subroutine `pbubbles` , which is given in listing 4.5. Result is given on figure 4.5.

```
      program p_bubbles
c
c     ----------------------------------------------------------------0--
c *   plot figure with coloured random bubbles
c     P. Robert, August 1999
c     ----------------------------------------------------------------0--

      call dopegra_(7,'p_bubbles.ps')
      call pbubbles
      call dclogra_
c
      stop 'OK p_bubbles'
      end
```

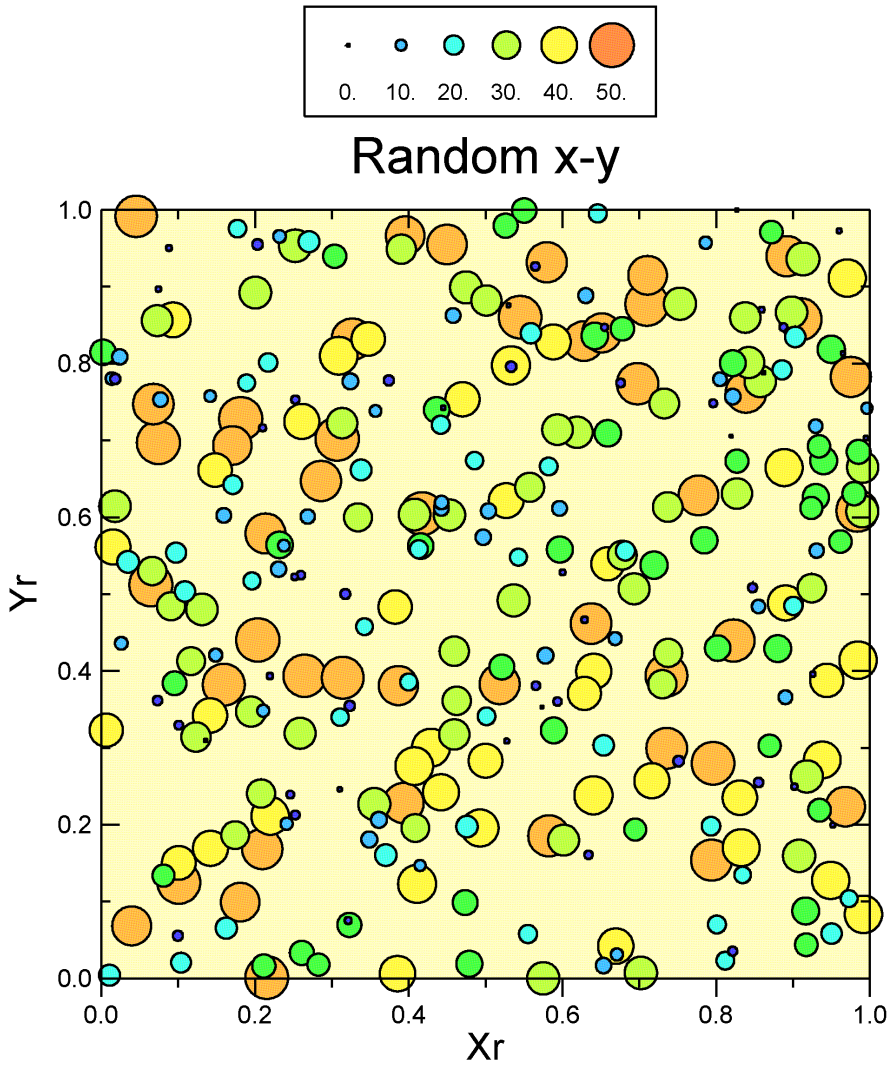


Fig. 4.5: *Plot a cloud of random bubbles (from program 4.5)*

```
      subroutine pbubbles
c
      real x(300),y(300),val(300)
c
      character*90 title
      character*16 labx,laby
c
c *   plot figure with coloured random bubbles
c
      data nt /300/
      data title/'Random x-y'/
      data labx,laby/'Xr','Yr'/
c
      data ofx,ofy/ 4., 8./
      data sfx,sfy/14.,14./
c
      data vmin,vmax /0.,50./
      data smin,smax /0.05,0.80/

      data satur /0.8/
c
c *** loading random bubbles positions
c
      do  i=1,nt
         call cranval_(ran)
         val(i)=ran*50.
         call cranval_(x(i))
         call cranval_(y(i))
      enddo
c
c *** Figure plot with default color map classic022
c
      call dfigori_(ofx,ofy)
      call dfigsiz_(sfx,sfy)
      call pfigtit_(title)

      call ccollev_(7,hue,sat,bri)
      call dlinhsb_(hue,0.3,1.)
      call pfigfra_
      call pfilzon_
c
      call dfiglimx(0.,1.)
      call dfiglimy(0.,1.)
c
      call csor3ra_(val,x,y,nt,1,nt,-1)
c
c *** bubbles plot
c
      do i=1,nt
c
         ts=(val(i)-vmin)*(smax-smin)/(vmax-vmin) +smin
         if(ts.gt.smax) ts=smax
         if(ts.lt.smin) ts=smin
c
         call colevel(ts,level)
         call ccollev_(level,hue,sat,bri)
         call dlinhsb_(hue,satur,1.)
         call pfigsym_(x(i),y(i),0,ts,ts,'circ')
         call pfilzon_
c
         call dlincol_('n')
         call pfigsym_(x(i),y(i),0,ts,ts,'circ')
c
      enddo
c
      call pfiggrax(0.,0.2,0.1,'(f3.1)')
      call pfiggray(0.,0.2,0.1,'(f3.1)')
      call pfiglabx(labx)
      call pfiglaby(laby)
c
      call plotindic(ofx+3.7,ofy+sfy+3.,smin,smax,vmin,vmax,5,satur)
c
      return
      end
```

Listing 4.5: *Subroutine to plot a cloud of various bubbles in a figure*

```
      subroutine plotindic(ox,oy,smin,smax,vmin,vmax,nbniv,satur)

      character*8 for

      tc=0.3
      rlmir=smax+4.*tc

      do i=1,nbniv+1
         val=float(i-1)*(vmax-vmin)/float(nbniv) + vmin
         ts=(val-vmin)*(smax-smin)/(vmax-vmin) +smin
         if(ts.gt.smax) ts=smax
         if(ts.lt.smin) ts=smin
         x=ox+smax+float(i-1)*smax*1.2
         y=oy

         call colevel(ts,level)
         call ccollev_(level,hue,sat,bri)
         call dlinhsb_(hue,satur,1.)
         call ppagsym_(x,y,0,ts,ts,'circ')
         call pfilzon_
         call dlincol_('n')
         call ppagsym_(x,y,0,ts,ts,'circ')
         call cbesfor_(val,for)
         call ppagval_(x,y-smax/2.-tc*1.5,0,tc,tc,0.,val,for)
      enddo

      call ppagrec_(ox,y-smax/2.-tc*3.,x-ox+smax,rlmir)
      return
      end
```

4.5 Plot a 3D cube

This is an example of how manage more deeply superimposed fill objects and text. The figure 4.6 below has been produced with program given in listing 4.6.

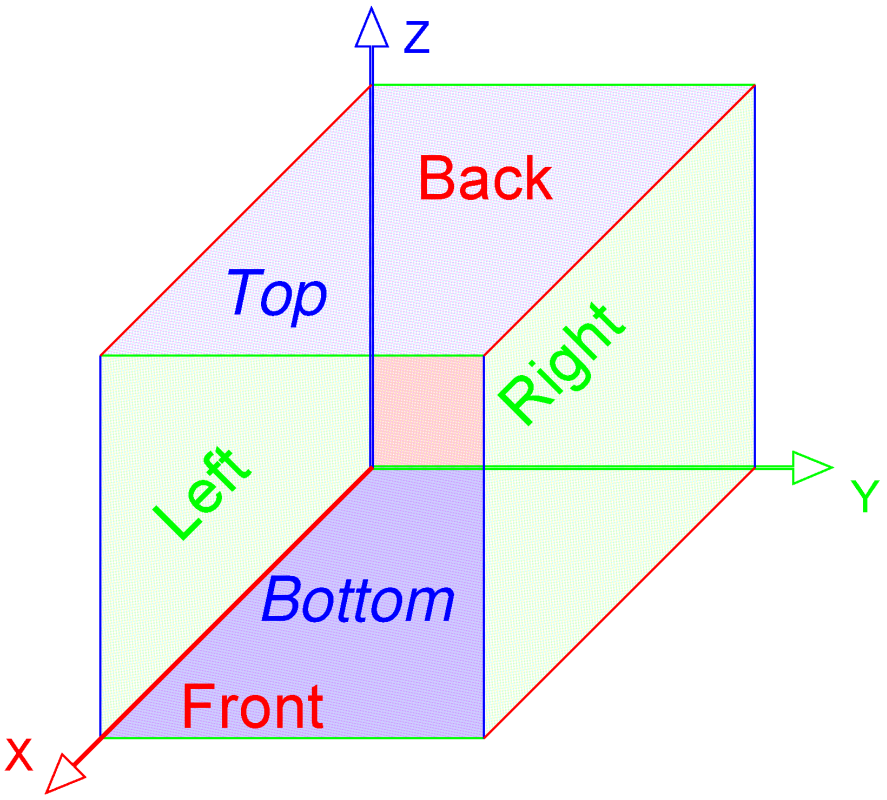


Fig. 4.6: *Plot of a 3D cube (from program 4.6)*

```
      program p_cube_3D
c     ------------------------------------------------------------------0--
c *   plot a 3D cube - P. Robert, SD, 2022
c     ------------------------------------------------------------------0--

      real xs(8),ys(8)
      data ox,oy,a,tc /8.,10.,6.,0.5/
      data bodyl,bodyw,headl,headw /7.2,0.04,0.6,0.5/
      data xs /-0.707, 0.293,0.293,-0.707,0.,1.,1.,0./
      data ys /-0.707,-0.707,0.293, 0.293,0.,0.,1.,1./

      do i=1,8
         xs(i)=ox+xs(i)*a
         ys(i)=oy+ys(i)*a
      enddo

      call dopegra_(7,'p_cube_3D.ps')
      call dlinwid_(3)
      call dfontyp_('Arial')

c *** fill face of the cube
      call pfilpoly(xs,ys,1,5,8,4,0.3,0.1,1.)
      call pfilpoly(xs,ys,5,6,7,8,0.0,0.2,1.)
      call pfilpoly(xs,ys,1,2,6,5,0.7,0.3,1.)
      call pfilpoly(xs,ys,2,6,7,3,0.3,0.1,1.)
      call pfilpoly(xs,ys,4,3,7,8,0.7,0.1,1.)

c *** draw edges of the cube
      call dlincol_('g')
      call ppagcha_(xs(6)+3.*tc,ys(6)-1.4*tc,-1,tc,tc,0.,'Y')
      call ppagarr_(ox,oy,-1,bodyl,bodyw,headl,headw,  0.)
      call ppaglin_(xs(1),ys(1),xs(2),ys(2))
      call ppaglin_(xs(7),ys(7),xs(8),ys(8))
      call ppaglin_(xs(3),ys(3),xs(4),ys(4))

      call dlincol_('b')
      call ppagcha_(xs(8)+tc,ys(8)+tc, -1,tc,tc,0.,'Z')
      call ppagarr_(ox,oy,-1,bodyl,bodyw,headl,headw, 90.)
      call ppaglin_(xs(2),ys(2),xs(3),ys(3))
      call ppaglin_(xs(4),ys(4),xs(1),ys(1))
      call ppaglin_(xs(6),ys(6),xs(7),ys(7))

      call dlincol_('r')
      call ppagcha_(xs(1)-3.*tc,ys(1)-3.*tc ,-1,tc,tc,0.,'X')
      call ppagarr_(ox,oy,-1,bodyl,bodyw,headl,headw,-135.)
      call ppaglin_(xs(4),ys(4),xs(8),ys(8))
      call ppaglin_(xs(3),ys(3),xs(7),ys(7))
      call ppaglin_(xs(2),ys(2),xs(6),ys(6))

c *   plot legend of faces
      tc=0.7
      call dlincol_('r')
      call ppagcha_(ox-0.5*a,oy-0.68*a,-1,tc,tc,0.,'Front')
      call ppagcha_(ox+tc,oy+0.7*a,-1,tc,tc,0.,'Back')
      call dlincol_('g')
      call ppagcha_(ox+0.4*a,oy+0.1*a,-1,tc,tc,45.,'Right')
      call ppagcha_(ox-0.5*a,oy-0.2*a,-1,tc,tc,45.,'Left')
      call dfontyp_('Arial-I')
      call dlincol_('b')
      call ppagcha_(ox-0.4*a,oy+0.4*a,-1,tc,tc,0.,'Top')
      call ppagcha_(ox-0.3*a,oy-0.4*a,-1,tc,tc,0.,'Bottom')

      call dclogra_
      stop 'OK p_cube_3D'
      end
c     ------------------------------------------------------------------0--
      subroutine pfilpoly(xs,ys,k1,k2,k3,k4,h,s,b)

c *** fill polygone with desired hsb color

      real xs(8),ys(8)

      call dlinhsb_(h,s,b)
      call dfilzon_
      call dpagppo_(xs(k1),ys(k1))
      call ppagpmo_(xs(k2),ys(k2))
      call ppagpmo_(xs(k3),ys(k3))
      call ppagpmo_(xs(k4),ys(k4))
      call pfilzon_

      return
      end
```

Listing 4.6: *Program to plot a 3D cube.*

4.6 Plot a tetrahedron from 6 points of view

This is an example of advanced plot, where some new features are introduced:

• The possibility to plot a figure with and reverse axis: The directions of the axis, and therefore of the graduations, no longer go from left to right as normal, but in the opposite direction, from right to left.
This is done by the subroutine `dfigrad_` which means "define_figure_reverse_axes_direction".

• The four faces of the tetrahedron must be plotted in a good order: the face more far that the point of view must be plotted first, and the faces closed to the p.o.v. plotted in last, in order to erase the faces below.

The source code is too long to be given here, but some important parts are given hereafter. Result can be shown on figure 4.7.

```
      call p_cube_3D(11.,23.,3.,0.4)

      ox=ox +a +1. +tc/2
      call p_pov(ox,oy,s,is,a,tc, 2, 3, 1,'Front view')

      ox=ox +a +1. +tc/2
      call p_pov(ox,oy,s,is,a,tc,-1, 3, 2,'Right view')

      ox=orix
      oy=oriy-a-3.5
      call p_pov(ox,oy,s,is,a,tc, 2,-1, 3,'Top view')
```

Listing 4.7: *Part of the code to plot some points of view of the tetrahedron defined by its 4 submits on 3 components s(3,4). Faces are defined by 3 submits, the first one having a sign to define the sens of the point of view.*

```
      call dfigori_(orix,oriy)
      call dfigsiz_(a,a)
      call dfiglim_(0.,a,0.,a)
      call dfigzat_(2.5)
      call dlincol_('n')
      call dstipos_('oi','oo')

      u1=sign(1.,float(k1))
      u2=sign(1.,float(k2))

      irevx=-int((u1-1.)/2.)
      irevy=-int((u2-1.)/2.)

      call dfigrad_(irevx,irevy)
      call pfiggrax(0.,1.,0.5,'(i2)')
      call pfiggray(0.,1.,0.5,'(i2)')
```

Listing 4.8: *Part of the code do draw a reverse axis.*

```
c *   order to plot the faces

      call sort_submit(s,k3,is)

      do j=1,4
         iface=is(abs(k3),j)
         call co_face(iface,is1,is2,is3)
         call p_facet(orix,oriy,a,k1,k2,s,is1,is2,is3,colfa(iface))
      enddo
```

Listing 4.9: *Part of the code do plot the 4 faces of the tetrahedron. Submits are sorted from the point of view, then corresponding face are computed and plotted.*

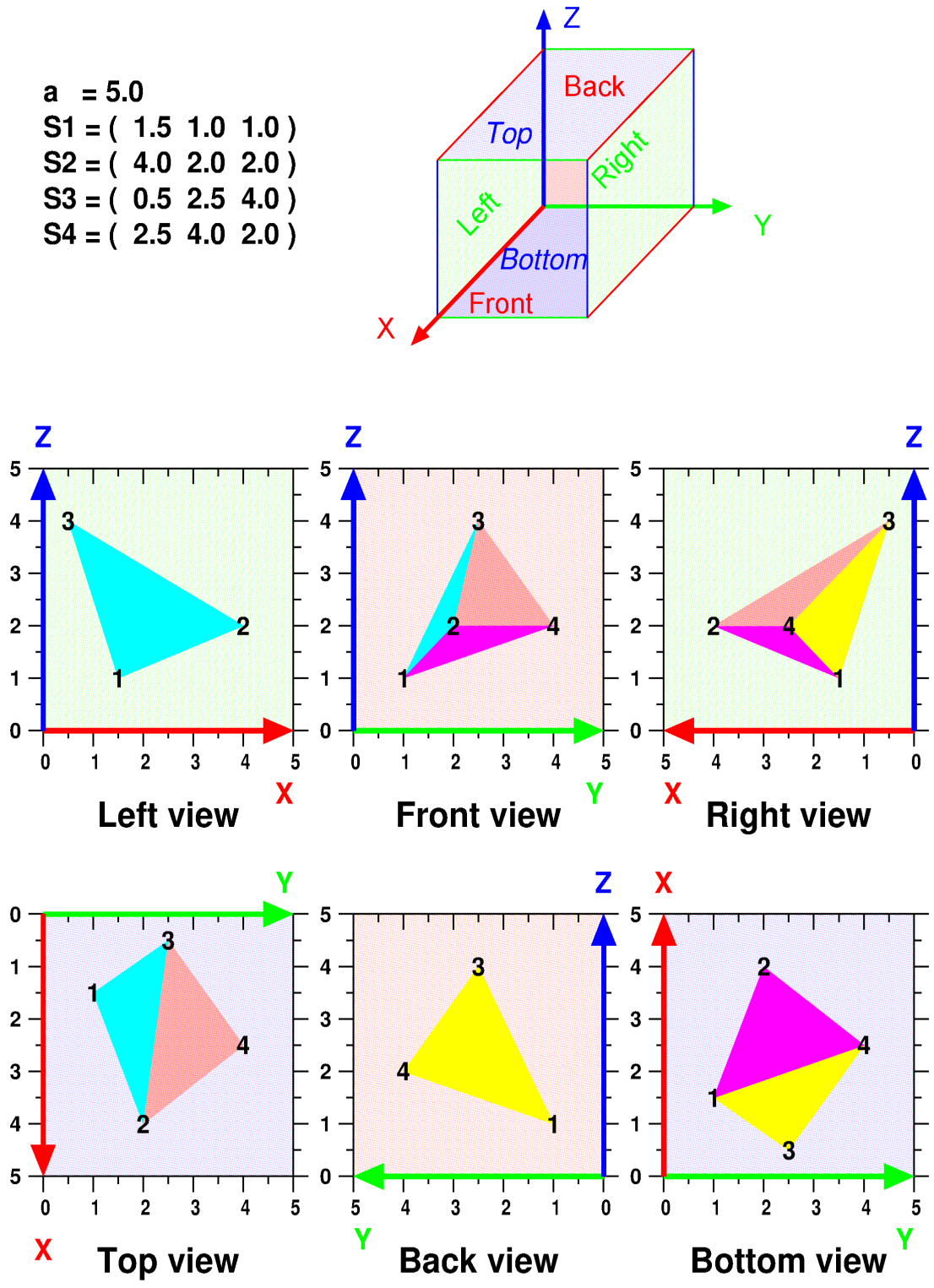


Fig. 4.7: *Plot of a tetrahedron from 6 orthogonal points of view.*

4.7 Plot the field lines of the Earth dipole

```
      subroutine tlindip

..... figure management...........
c
c *** Plot fig. 1
c
      call dfigsiz_(sixf1,siyf1)
      call dfigori_(poxf1,poyf1)
      call dlinwid_(iwid)
      call dfiglimx(x1f1,x2f1)
      call dfiglimy(y1f1,y2f1)
c
      call ppagfra_
      call pfiggrax(0.,5.,1.,'(i2)')
      call pfiggray(0.,2.,1.,'(i2)')
      call pfiglaby('Z /Earth Radius')
      call pfigtit_(tit)
c
c *** Plot Earth
c
      do i=1,46
         teta=90.-float(i-1)*4.
         x(i)=cos(teta*pisd)
         y(i)=sin(teta*pisd)
      enddo
      call pfigcur_(x,y,46)
c
c *** Plot dipolar field lines
c
      call pfighli_(0.)
      if=int(x2f1) -1
      do ir0=2,if,2
         r0=float(ir0)
         c2lat=1./r0
         clat=sqrt(c2lat)
         alat=acos(clat)/pisd
         npi=180
         do jlat=1,npi
            rlat=alat-float(jlat+1)
            ctet2=cos(rlat*pisd)**2
            r=r0*ctet2
            if(r.lt.0.98) then
                    npl=jlat-1
                    exit
            endif
         call cpolcar_(r,rlat,x(jlat),y(jlat))
         enddo
      call pfigcur_(x,y,npl)
      enddo
c
c *** Plot fig. 2
c
      call dlinwid_(iwid)
      call dfigori_(poxf2,poyf2)
      call dfigsiz_(sixf2,siyf2)
      call dfigout_('n')
      call dfiglimx(x1f2,x2f2)
      call dfiglimy(y1f2,y2f2)
c
      call pfiggrax(0.,5.,1.,'(i2)')
      call pfiggray(0.,500.,100.,'(i4)')
      call pfiglabx('X /Earth Radius')
      call pfiglaby('Field Intensity /nT')
c
c *** Plot magnitude
c
      nx=(int(x2f2)-1)*5+1
      do iz=0,5
      az=float(iz)
         do ix=1,nx
            x(ix)=float(ix-1)/5.
            r=sqrt(x(ix)**2 + az**2)
            if(r.lt.1.) r=1.
            st=az/r
            rm=8.1e6/(6.370**3)
            r3=r*r*r
            b=rm*sqrt(1.+3.*st*st)/r3
            y(ix)=b
```

```
            if(ix.eq.35) then
               if(iz.eq.0) then
                  call cfigpag_(x(ix),y(ix),xx,yy)
                  call ppagcha_(xx+tcx,yy+tcy,-1,tcx,tcy,0.,'Z=0')
               endif
               if(iz.eq.5) then
                  call cfigpag_(x(ix),y(ix),xx,yy)
                  call ppagcha_(xx-tcx,yy-tcy, 1,tcx,tcy,0.,'Z=5')
               endif
            endif
         enddo
      call pfigcur_(x,y,nx)
      enddo
c
      return
      end
```

Listing 4.10: *Program to plot the field lines of the Earth dipole (see fig.4.8).*

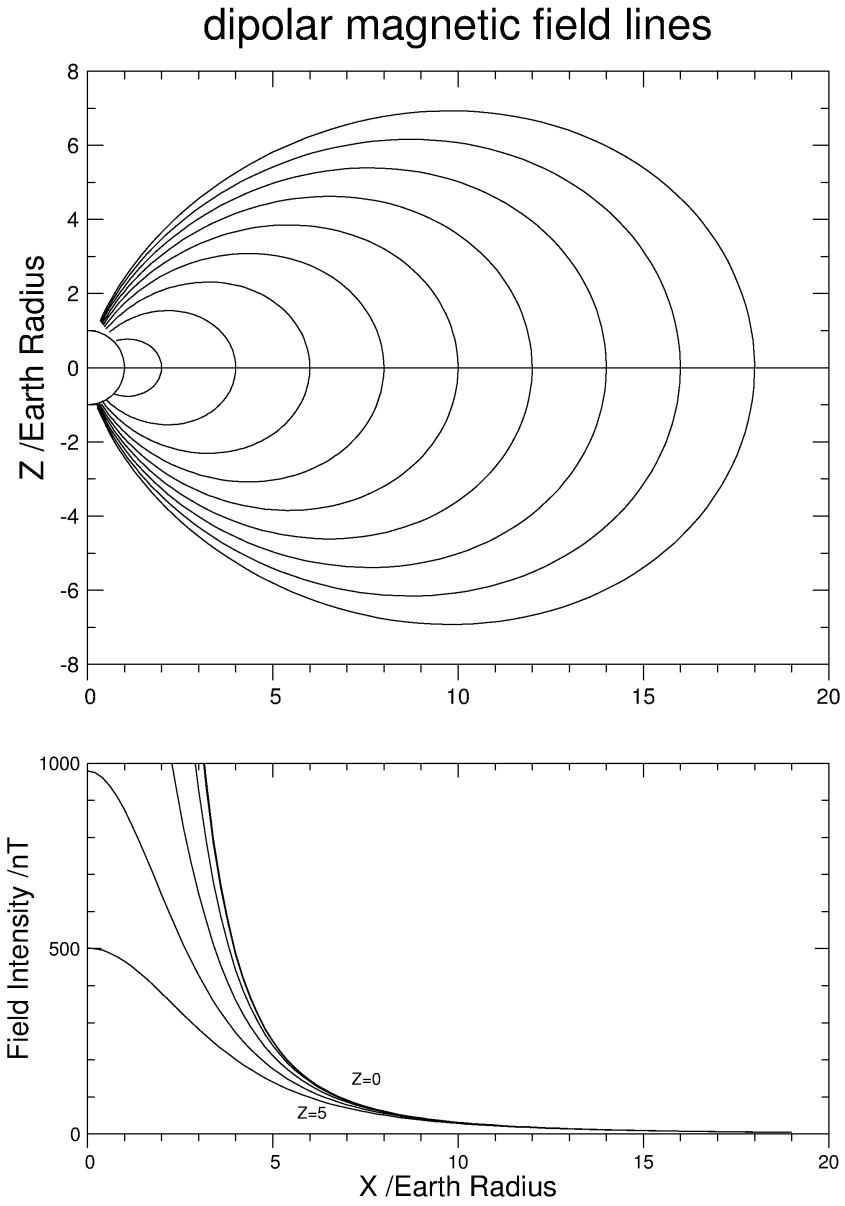


Fig. 4.8: *Field lines of the Earth dipole (from program 4.10)*

4.8 Plot a protractor

A protractor is a half-circle shaped instrument for measuring angles. Code is available on listing 4.11 while plot is given on figure 4.9.

```
      program p_protractor
c     ------------------------------------------------------------------+--
c     plot of a protractor
c     this is a half-circle shaped instrument for measuring angles
c     P. Robert, May 1998
c     ------------------------------------------------------------------+--
      character*8 for

c *** characteristic dimensions (cm)
      high=28.
      rx=high/2.
      tc=0.4
      pisd=acos(-1.)/180.
c     basic unit: length halph degrees
      un=0.6

c *** compure characteristics radius and other quantities
      call radius(rx,un,tc,r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12)

c *** basic options
      call dopegra_(1,'p_protractor.ps')
      call dpagori_(1.,rx+0.8)
      call dlincol_('n')
c
c *** loop on the angles
      do i=-90,90,15
      do j=0,29
         alpha=float(i)+float(j)/2.
         beta=alpha-90.
         gama=alpha+90.
         teta=180.-gama
         x=rx*cos(alpha*pisd)
         y=rx*sin(alpha*pisd)

c *   plot grag of angles, every 15 degrees
         if(j.eq.0) then
                  call ppaglin_(x*r3,y*r3,x,y)
                  call cbesfori(int(alpha),for)
                  call ppagval_(x*r4,y*r4,0,tc,tc,beta,alpha,for)

                  call ppaglin_(x*r5,y*r5,x*r6,y*r6)
                  call cbesfori(int(gama),for)
                  call ppagval_(x*r7,y*r7,0,tc,tc,beta,gama,for)

                  call ppaglin_(x*r8,y*r8,x*r9,y*r9)
                  call cbesfori(int(teta),for)
                  call ppagval_(x*r10,y*r10,0,tc,tc,beta,teta,for)

                  call ppaglin_(x*r11,y*r11,x*r12,y*r12)
         endif
         if(i.eq.90 .and. j.gt.0) exit

c *      plot of halph-degrees
         call ppaglin_(x*r1,y*r1,x,y)

c *      plot of degrees
         if((j/2)*2.eq.j) call ppaglin_(x*r2,y*r2,x,y)

c *      plot of 5 degrees
         if((j/10)*10.eq.j) call ppaglin_(x*r3,y*r3,x,y)
         enddo
      enddo

c *** plot of 1/2 circles
      call ppagcir_(0.,0.,rx ,-90.,90.,0.5)
      call ppagcir_(0.,0.,r11*rx,-90.,90.,0.5)
      call ppagcir_(0.,0.,r12*rx,-90.,90.,5.0)
      call ppagcir_(0.,0.,0.05,-90.,270.,10.0)

      call dclogra_
      stop 'OK p_protractor'
      end
```

Listing 4.11: *Program to plot a protractor*

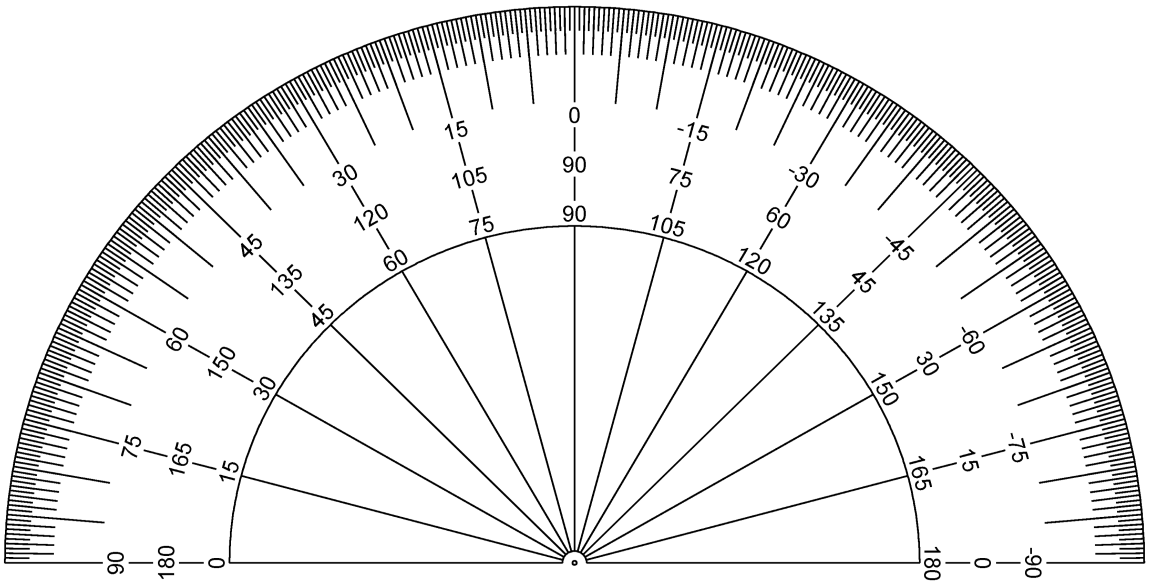


Fig. 4.9: *Protractor drawn from program 4.11*

```
      subroutine radius(rx,un,tc,r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12)

c *** define characteristic lengths for graduation

c     demi degre
      d1 = un
c     degre
      d2 = 2.*un
c     5 degres
      d3 = 4.*un
c     first grad from zero to 90 deg.
      d4 = d3 +0.8*tc
c     stick under the first graduation
      d5 = d4 +0.8*tc
      d6 = d5 +un
c     2nd graduation from zero on the right
      d7 = d6 +0.8*tc
c     stick under the second graduation
      d8 = d7 +0.8*tc
      d9 = d8 +un
c     3th graduation in reverse order
      d10= d9 +0.8*tc
c     big stick to zerp zero
      d11= d10+0.8*tc
      d12= rx -un/2.

c *** normalisation
      r1 = 1. -d1/rx
      r2 = 1. -d2/rx
      r3 = 1. -d3/rx
      r4 = 1. -d4/rx
      r5 = 1. -d5/rx
      r6 = 1. -d6/rx
      r7 = 1. -d7/rx
      r8 = 1. -d8/rx
      r9 = 1. -d9/rx
      r10= 1. -d10/rx
      r11= 1. -d11/rx
      r12= 1. -d12/rx

      return
      end
```

4.9 Plot a draftsman's square

A draftsman's square is a rectangular shaped instrument for measuring angles. Code is available on listing 4.12 while plot is given on figure 4.10. Note that we use here the subroutine `ppagrgl_` to plot the **Rogralib** logo.

```
      program p_square
c     ------------------------------------------------------------------+--
c     plot of a square 20.5 x 29 cm; P. Robert, SD, Mars 2022
c     ------------------------------------------------------------------+--
      character*3 grad
      data rx,ry /20.5, 29./
      data tsti,tgra /0.6,0.7/

      call dopegra_(1,'p_square.ps')
      call dpagsiz_(20.5,29.)
      call dpagori_(0.5,0.5)
      call dfontyp_('Arial')
c
c *** horizontal ruler
c *   centimetres
      do j=0,19
         xx=float(j)
         call dlinwid_(5)
         call dpagppo_(xx,0.)
         call ppagpmo_(xx,tsti)
         write(grad,'(i2)') j
         if((j/5)*5.eq.j.and.j.ne.0) then
             call ppagpmo_(xx,tsti*1.5)
             call ppagcha_(xx,tsti*1.5+0.8*tgra,0,tgra,tgra,180.,grad)
         endif
c *      millimetres
         call dlinwid_(3)
         do k=0,9
            xx=float(j) +float(k)/10.
            call dpagppo_(xx,0.)
            call ppagpmo_(xx,tsti/2.)
         enddo
      enddo

c *** vertical ruler
c *   centimetres
      do j=0,27
         call dlinwid_(5)
         yy=float(j)
         call dpagppo_(0.,yy)
         call ppagpmo_(tsti,yy)
         write(grad,'(i2)') j
         if((j/5)*5.eq.j.and.j.ne.0) then
             call ppagpmo_(tsti*1.5,yy)
             call ppagcha_(tsti*1.5+0.8*tgra,yy,0,tgra,tgra,90.,grad)
         endif
c *      millimetres
         call dlinwid_(3)
         do k=0,9
            yy=float(j) +float(k)/10.
            call dpagppo_(0.,yy)
            call ppagpmo_(tsti/2.,yy)
         enddo
      enddo

c *   external triangle
      call dpagppo_(0.,0.)
      call ppagpmo_(rx,0.)
      call ppagpmo_(0.,ry)
      call ppagpmo_(0.,0.)
c *   internal triangle interieur
      dl=3.
      call dpagppo_(dl,dl)
      call ppagpmo_(rx-(1.+0.707)*dl,dl)
      call ppagpmo_(dl,ry-2.707*dl)
      call ppagpmo_(dl,dl)
c
c *   logo rogralib
      call  ppagrgl_(12.3,10.,-1,3.,0.8,125.,0.6,0.8)
      call dclogra_
      stop
      end
```

Listing 4.12: *Program to plot draftsman's square.*

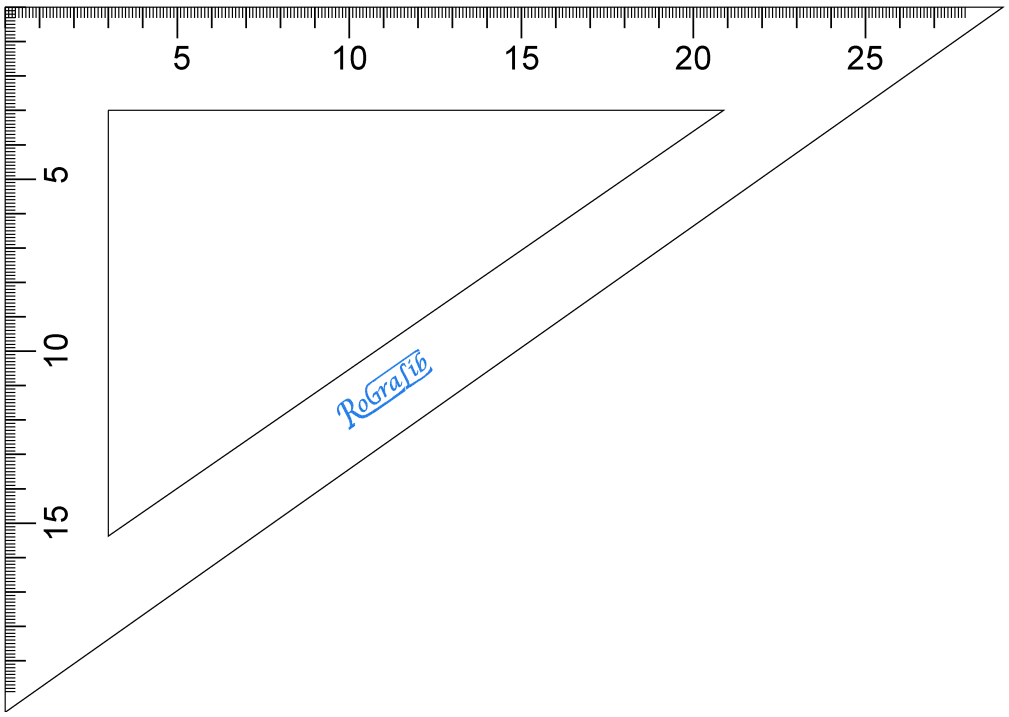


Fig. 4.10: *Draftsman's square drawn from program 4.12.*

4.10 Plot a square with protractor

This program is a mix of the two previous ones. The subroutine radius is the same that one used in program 4.11.

```
      program p_square_protrac
c     ------------------------------------------------------------1-
c     plot of a square + protractor; P. Robert, SD, 2022
c     ------------------------------------------------------------1-
      character*3 grad
      character*8 for
      data rx,ry /20., 29./
      data tsti,tgra /0.6,0.7/
      pisd=acos(-1.)/180.

      call dopegra_(1,'p_square_protac.ps')
      call dpagsiz_(20.5,29.)
      call dpagori_(0.5,0.5)
      call dfontyp_('Arial')

c *** horizontal ruler
c *   centimetres
      do j=0,19
         xx=float(j)
         call dlinwid_(5)
         call dpagppo_(xx,0.)
         call ppagpmo_(xx,tsti)
         write(grad,'(i2)') j
         if((j/5)*5.eq.j.and.j.ne.0) then
             call ppagpmo_(xx,tsti*1.5)
             call ppagcha_(xx,tsti*1.5+0.8*tgra,0,tgra,tgra,180.,grad)
         endif
c *      millimetres
         call dlinwid_(3)
         do k=0,9
            xx=float(j) +float(k)/10.
            call dpagppo_(xx,0.)
            call ppagpmo_(xx,tsti/2.)
         enddo
      enddo
```

```
c
c *** vertical ruler
c *   centimetres
      do j=0,27
         call dlinwid_(5)
         yy=float(j)
         call dpagppo_(0.,yy)
         call ppagpmo_(tsti,yy)
         write(grad,'(i2)') j
         if((j/5)*5.eq.j.and.j.ne.0) then
             call ppagpmo_(tsti*1.5,yy)
             call ppagcha_(tsti*1.5+0.8*tgra,yy,0,tgra,tgra,90.,grad)
         endif
c *      millimetres
         call dlinwid_(3)
         do k=0,9
            yy=float(j) +float(k)/10.
            call dpagppo_(0.,yy)
            call ppagpmo_(tsti/2.,yy)
         enddo
      enddo
      call ppaglin_(rx,yy1,rx,yy2)

c *** plot protractor

c *   characteristic dimensions (cm)
      un=0.2
      tc=0.4
      rxr=14.
      dl=3.

      call dfigori_(dl,dl)
      call dfigsiz_(rx,rx)
      call dfiglim_(0.,rx,0.,rx)

c *   compure characteristics radius and other quantities

      call radius(rxr,un,tc,r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12)
c
c *** loop on the angles

      do i=0,90,15
         do j=0,29
            alpha=float(i)+float(j)/2.
            beta=alpha-90.
            gama=alpha+90.
            teta=180.-gama
            x=rxr*cos(alpha*pisd)
            y=rxr*sin(alpha*pisd)

c *   plot grag of angles, every 15 degrees

            if(j.eq.0) then
                 call pfiglin_(x*r3,y*r3,x,y)
                 call cbesfori(int(alpha),for)
                 call pfigval_(x*r4,y*r4,0,tc,tc,beta,alpha,for)
                 call pfiglin_(x*r5,y*r5,x*r6,y*r6)
                 call cbesfori(int(gama),for)
                 call pfigval_(x*r7,y*r7,0,tc,tc,beta,gama,for)
                 call pfiglin_(x*r8,y*r8,x*r9,y*r9)
                 call cbesfori(int(teta),for)
                 call pfigval_(x*r10,y*r10,0,tc,tc,beta,teta,for)
c                call pfiglin_(x*r11,y*r11,x*r12,y*r12)
            endif

            if(i.eq.90 .and. j.gt.0) exit
c *         plot of halph-degrees
            call pfiglin_(x*r1,y*r1,x,y)
c *         plot of degrees
            if((j/2)*2.eq.j) call pfiglin_(x*r2,y*r2,x,y)
c *         plot of 5 degrees
            if((j/10)*10.eq.j) call pfiglin_(x*r3,y*r3,x,y)
         enddo
      enddo
c
c *** plot of 1/2 circles

      call pfigcir_(0.,0.,rxr ,0.,90.,0.5)
      call pfigcir_(0.,0.,r11*rxr,0.,90.,0.5)
      call pfigcir_(0.,0.,r12*rxr,0.,90.,5.0)
      call pfigcir_(0.,0.,0.05,0.,270.,10.0)
```

```
c *   terminate the rulers

      call ppaglin_(0.,0.,rx,0.)
      call ppaglin_(rx,0.,rx,dl)
      call ppaglin_(rx,dl,rxr+dl,dl)
      call ppaglin_(r11*rxr+dl,dl,dl,dl)

      call ppaglin_(dl,dl,dl,r11*rxr+dl)
      call ppaglin_(dl,rxr+dl,dl,ry)
      call ppaglin_(dl,ry,0.,ry)
      call ppaglin_(0.,ry,0.,0.)

      call dclogra_

      stop
      end
```

Listing 4.13: *Program to plot a Square with protractor.*

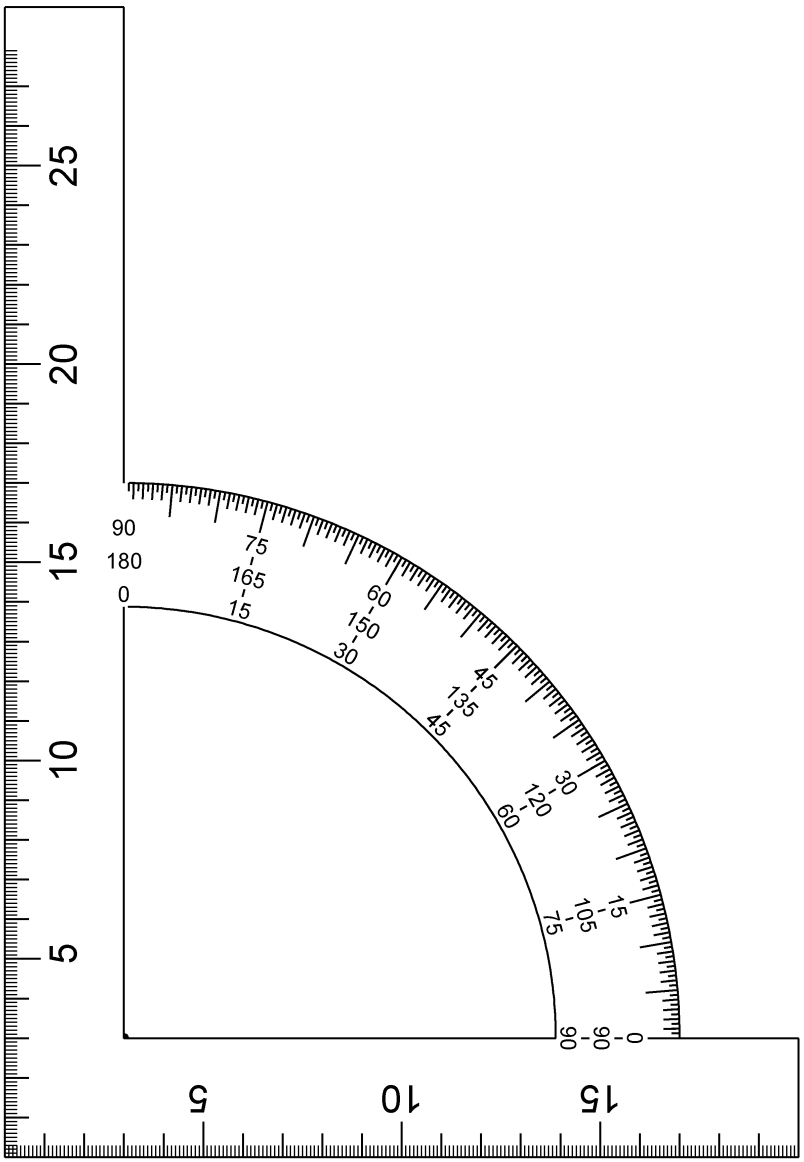


Fig. 4.11: *Square with protractor from program 4.13*

4.11 Plot of a graduated dial and a clock

```
      program p_graduated_dial
c     ------------------------------------------------------------------0-
c     plot of a graduated dial and a clock, P. Robert, SD, 2022
c     ------------------------------------------------------------------0-
      character*8 for
      pisd=acos(-1.)/180.
c
c *** characteristic dimensions (cm)
      tc=0.4
      r1=7.
      call rcara(tc,r1,r2,r3,r4,r5)
c
      call dopegra_(1,'p_graduated_dial.ps')
      call dfigori_(0.1,2.*r1)
      call dfigsiz_(2.*r1,2.*r1)
      call dfiglim_(-r1,r1,-r1,r1)
      call dlincol_('n')
      call dfontyp_('Arial-B')
c
c *** plot of a graduated dial
      do i=0,359,15
c *      plot grag of angles, every 15 degrees
         alpha=float(i)
         apos=90.-alpha
         beta=apos-90.
         x=cos(apos*pisd)
         y=sin(apos*pisd)
         call pfiglin_(x*r3,y*r3,x*r1,y*r1)
         call cbesfori(int(alpha),for)
         call pfigval_(x*r4,y*r4,0,tc,tc,beta,alpha,for)
         do j=0,15
            apos=90.-float(i)-float(j)
            x=cos(apos*pisd)
            y=sin(apos*pisd)
c *         plot of degrees
            call dlinwid_(2)
            call pfiglin_(x*r2*1.03,y*r2*1.03,x*r1,y*r1)
            call dlinwid_(6)
c *         plot of  5 degrees
          if((j/5)*5.eq.j)   call pfiglin_(x*r2,y*r2,x*r1,y*r1)
         enddo
      enddo
c *   plot of circles
      call pfigcir_(0.,0.,r1,0.,361.,0.5)
      call pfigcir_(0.,0.,r5,0.,361.,0.5)
      call pfigcir_(0.,0.,0.05,0.,361.,10.0)
c
c *** plot of a clock
      r1=5.
      tc=0.7
      call rcara(tc,r1,r2,r3,r4,r5)
      call dfigori_(0.1,0.1)
      call dfigsiz_(2.*r1,2.*r1)
      call dfilzon_
      call dlinrgb_(1.,1.,0.8)
      call pfigcir_(0.,0.,r1 ,0.,361.,0.5)
      call pfilzon_
      call dlincol_('n')

      do i=30,360,30
c *      plot hours, every 30 degrees
         ahour= 90. -float(i)
         x=cos(ahour*pisd)
         y=sin(ahour*pisd)
         call dlinwid_(12)
         call pfiglin_(x*r3,y*r3,x*r1,y*r1)
         call cbesfori(int(float(i)/30.),for)
         call pfigval_(x*r4,y*r4,0,tc,tc,0.,float(i)/30.,for)
         do j=0,5
            apos=90.-float(i-30)+float(j)*6.
            x=cos(apos*pisd)
            y=sin(apos*pisd)
            call dlinwid_(6)
            call pfiglin_(x*r2,y*r2,x*r1,y*r1)
         enddo
      enddo
c *** plot of circles
      call pfigcir_(0.,0.,r1 ,0.,361.,0.5)
      call pfigcir_(0.,0.,0.05,0.,361.,10.0)

      call dclogra_
      stop 'OK p_graduated_dial'
      end
```

```
      subroutine rcara(tc,r1,r2,r3,r4,r5)

      r2=r1-0.5
      r3=r2-0.4
      r4=r3-tc
      r5=r4-1.5*tc

      return
      end
```

Listing 4.14: *Program to plot a graduated dial and a clock.*

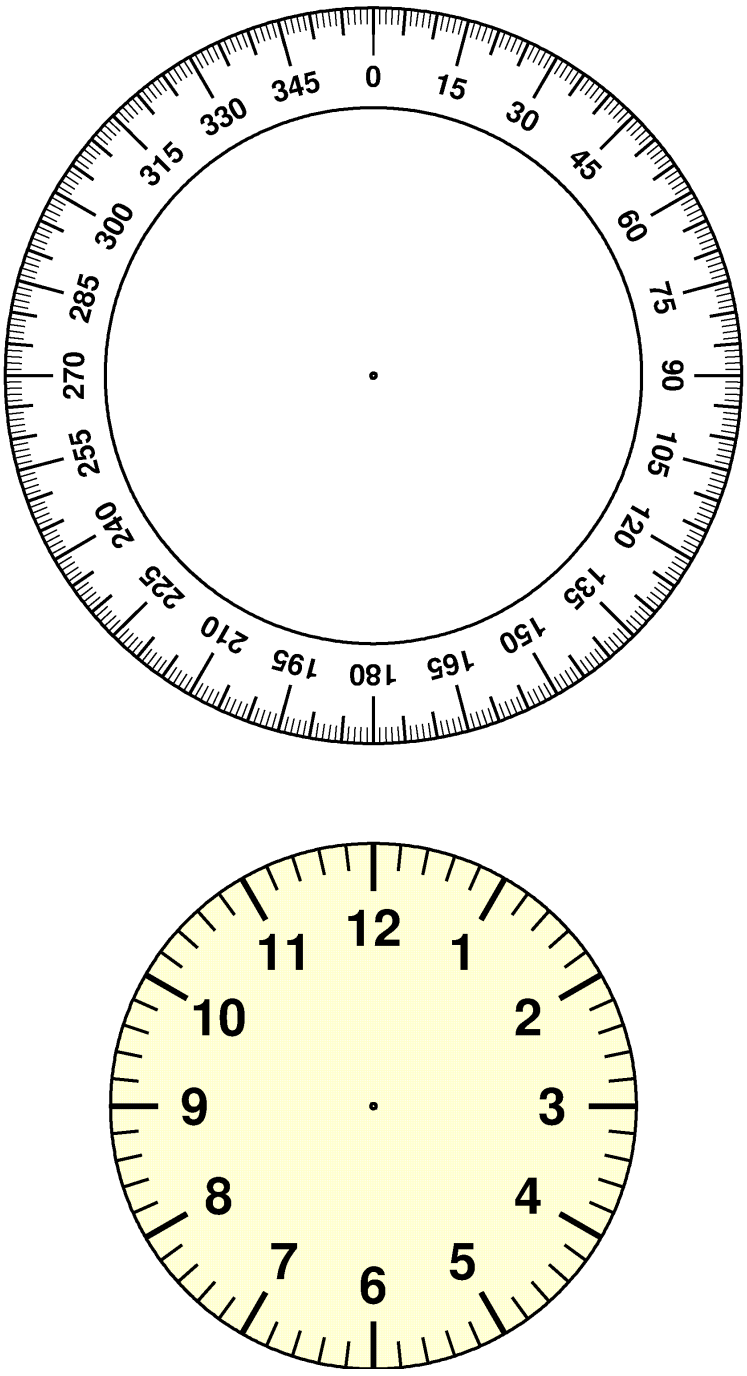


Fig. 4.12: *Graduated dial and clock from program 4.14*

4.12 Make a poster

```
      program p_poster
c
      integer imaico(2048,1024)
      character*14 vercha
      character*255 txt(100)
c
c     ------------------------------------------------------------------0--
c     plot poster example
c     P. Robert, 2022
c     ------------------------------------------------------------------0--
c
      txt( 1)='Here is an example of text entered without layout in a '
      txt( 2)='character array, then reformatted by Rogralib, '
      txt( 3)='and printed in a frame defined by its width, and by '
      txt( 4)='the height of the characters of the desired font. '
      txt( 5)='All lines are left justified. To make a new paragraph, '
      txt( 6)='it is necessary to make a new call to this subroutine.'
      txt( 7)='Each line is limited to 255 characters, and'
      txt( 8)=' '
      txt( 9)='each word is limited to 80 characters.'
      txt(10)='The total number of words is limited to 1000.'
      nl=10

c *** open PS file and page attribute
      call dopegra_(7,'p_poster.ps')
      call dpaptyp('A0')
      call dfontyp_('h')
      call gvercha_(vercha)
      call ppaghea_(vercha//' - ')
      call ppagfoo_('P. Robert')
      call ppagfra_
c
c *** title
      call dlincol_('y')
      call dfilzon_
      call ppagrro_(5.,24.8,10.,2.,0.5)
      call pfilzon_
      call dlincol_('b')
      call ppagcha_(7.,25.5,-1,0.6,0.6,0.,'Poster example')
      call dlinwid_(6)
      call ppagrro_(5.,24.8,10.,2.,0.5)
c
c *** reading bmp files, convert image to ico format and plot
      call rdimbmp_(1,'./BMP/p_spectrum.bmp',nx,ny)
      call rimabmp_(1,'./BMP/p_spectrum.bmp',imaico,nx,ny)
      call ppagima_(1.,13.,-1,9.,11.,0.,imaico,nx,ny)
c
c *** first text on the right
      call ppagtxt_(12.,23.,0.25,7.,0.25,txt,nl)
c
      call rdimbmp_(1,'./BMP/p_fonc_2D.bmp',nx,ny)
      call rimabmp_(1,'./BMP/p_fonc_2D.bmp',imaico,nx,ny)
      call ppagima_(12.,13.5,-1,6.5,3.5,0.,imaico,nx,ny)
c
c *** second text on the meddle and dynamic spectra
      call ppagtxt_(1.,12.5,0.25,18.,0.25,txt,nl)

      call rdimbmp_(1,'./BMP/p_spectro.bmp',nx,ny)
      call rimabmp_(1,'./BMP/p_spectro.bmp',imaico,nx,ny)
      call ppagima_(1.5,4.2,-1,16.,6.,0.,imaico,nx,ny)
c
c *** text, arrow and advertisement
      call dlincol_('b')
      call ppagcha_(1.,2.,-1,1.,1.,0.,'More details')
      call dfilzon_
      call ppagarr_(11.,2.4,-1.,3.5,0.5,1.5,1.,0.)
      call pfilzon_
      call dlincol_('n')
      call ppagarr_(11.,2.4,-1.,3.5,0.5,1.5,1.,0.)
c
      call rdimbmp_(1,'./data/logo_1.bmp',nx,ny)
      call rimabmp_(1,'./data/logo_1.bmp',imaico,nx,ny)
      call ppagima_(13.8,1.6,-1,5.,1.5,0.,imaico,nx,ny)

      call dclogra_
      stop 'OK p_poster'
      end
```

Listing 4.15: *Program to plot a poster on a A0 paper sheet*

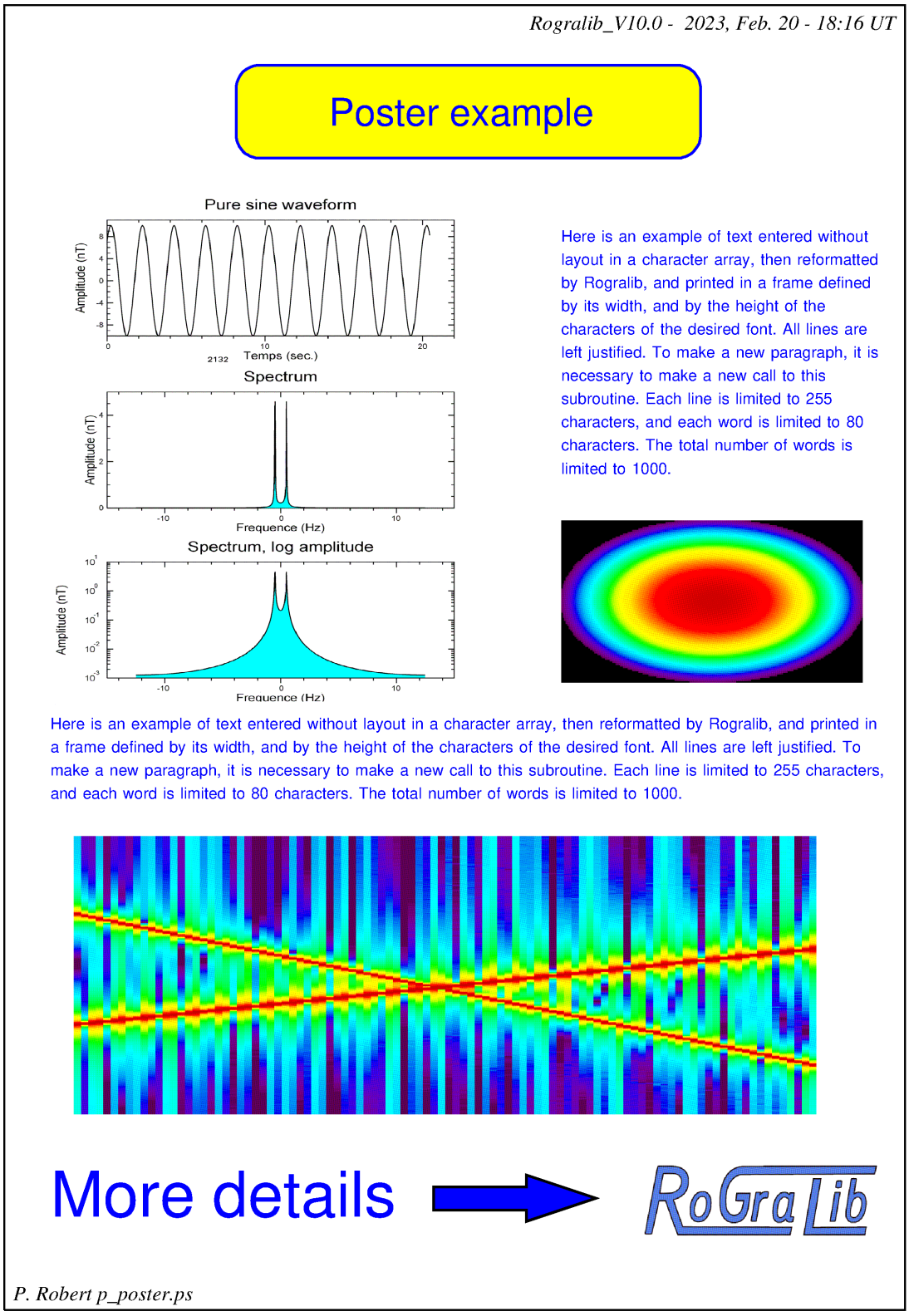


Fig. 4.13: *Example of poster on a A0 paper sheet*

Chapter 5 : PARAMETERS

5.1 Parameters default values

5.1.1 Line properties

Line width = 3 pixels
Line style = continue
Line color = black

5.1.2 Character fonts

Font type = Arial
Font number = 9

5.1.3 Color maps

Color map = classic022

5.1.4 Page parameters

Page format = A4
Page origin x = 0.8 cm
Page origin y = 0.8 cm
Page size x = 19.4 cm
Page size y = 28.1 cm

Header font = Times-I
Header size = 0.3 cm
Header position = right (ipos=1)

Foot font = Times-I
Foot size = 0.3 cm
Foot position = left (ipos=-1)

5.1.5 Figure parameters

Figure origin x = 3. cm
Figure origin y = 18. cm
Figure size x = 14. cm
Figure size y = 7. cm
Figure zoom attribute = 1.
Figure out = no (draw not trespassing figure frame)
Figure graduation style = Linear (ilogx=ilogy=0)

Title position = top

Label position x = center
Label position y = center

Label sens x = horizontal
Label sens y = vertical

Graduation position x = bottom
Graduation position y = left

Sticks position x = oi (output for bottom, input for top)
Sticks position y = ii (input for left and right)

5.2 Define parameters

All call to these subroutines erases the current values and replace it by the new ones. It remains active until a new call. In there is no call to these subroutines, default values are applied. Default values are listed in section 5.1.

5.2.1 Media type and paper size

dpaptyp_`(paptyp)` : **define_paper_type**

Input : `paptyp` → paper type as 'A4', 'A3', 'Letter', etc. (call just after dopegra_)

Complete list is given in section 7.1.

5.2.2 Forcing recto-verso

drecver_ : **define_recto_verso**

Input : `None` → Force recto-verso for PS printing (call before dopegra_)

5.2.3 Open and close graphical file

dopegra_`(ifc,postscript_file)` : **define_open_graphical_file**

Input : `ifc,postscript_file` → unit and name of postscript_file

dclogra : **define_close_graphical_file**

Input : `None`

5.2.4 Line properties

- **`Line width`**
  dlinwid_`(iwidth)` : **define_line_width**

  Input : `iwidth` → line width in pixel (ex: 3)
  dlinwidr`(width)` : **define_line_width_real**

  Input : `width` → line width in pixel, real value (ex: 3.5).

- **`Line style`**
  dlindas_`(size1,blan1,size2,blan2)` : **define_line_dash**

  Input : `size1,blan1,size2,blan2` → size of dashes and blanks in the line

  dlincon_ : **define_line_continue**

  Input : `None` (remove dash line previous parameters).

- **Line color**

  dlincol_(col) : **define_line_color**

  Input : col → as 'r' or any of nrgbcmypwi

  Meaning : n(o), r(ed), g(reen), b(lue), c(yan), m(agenta), y(ellow), p(purple),w(hite),i(gnore)

  dlincoln : **define_line_color_level**

  Input : level → level of current color map

  limits of level depends of the used color map

  dlinhsb_ : **define_line_color_hsb**

  Input : h,s,b → h,s,b values [0.-1.]

  dlinrgb_ : **define_line_color_rgb**

  Input : r,g,b → r,g,b values [0.-1.]

5.2.5 Character fonts

- **Set font by type**

  dfontyp_(fontyp) : **define_font_type**

  Input : fontyp → Any of the following list:

```
couri          cbold          cobli          cbobl
times          tbold          tital          tbita
helve          hbold          hobli          hbobl

Courier        Courier-B      Courier-I      Courier-BI
Times          Times-B        Times-I        Times-BI
Helvetica      Helvetica-B    Helvetica-I    Helvetica-BI
AvantGarde     AvantGarde-B   AvantGarde-I   AvantGarde-BI
Bookman        Bookman-B      Bookman-I      Bookman-BI
HelveNarr      HelvetNarr-B   HelvetNarr-I   HelvetNarr-BI
LetterGhotic   LetterGhotic-B LetterGhotic-I LetterGhotic-BI
NewCentury     NewCentury-B   NewCentury-I   NewCentury-BI
Palatino       Palatino-B     Palatino-I     Palatino-BI
Arial          Arial-B        Arial-I        Arial-BI
ZapfChancery   ZapfDingbats   Symbol
```

- **Set font by number**

  dfonnum_(numfon) : **define_font_number**

  Input : numfon → Correspondence is given below.

```
 1: Courier        2: Courier-B      3: Courier-I      4: Courier-BI
 5: Times          6: Times-B        7: Times-I        8: Times-BI
 9: Helvetica     10: Helvetica-B   11: Helvetica-I   12: Helvetica-BI
13: Symbol
14: AvantGarde    15: AvantGarde-B  16: AvantGarde-I  17: AvantGarde-BI
18: Bookman       19: Bookman-B     20: Bookman-I     21: Bookman-BI
22: HelveNarr     23: HelvetNarr-B  24: HelvetNarr-I  25: HelvetNarr-BI
26: LetterGhotic  27: LetterGhotic-B28: LetterGhotic-I29: LetterGhotic-BI
30: NewCentury    31: NewCentury-B  32: NewCentury-I  33: NewCentury-BI
34: Palatino      35: Palatino-B    36: Palatino-I    37: Palatino-BI
38: ZapfChancery  39: ZapfDingbats
40: Arial         41: Arial-B       42: Arial-I       43: Arial-BI
```

5.2.6 Colors map

See 'Line color' subsection 5.2.4, also valid for fonts

- **Pre-defined color maps**
  dcolmap_(colmap) : **define_color_map**
  Input : colmap → One of the following values

```
classic022
spectro008  spectro016  spectro032  spectro064  spectro128  spectro256
rainbow008  rainbow016  rainbow032  rainbow064  rainbow128  rainbow256
fullhue008  fullhue016  fullhue032  fullhue064  fullhue128  fullhue256
greylev008  greylev016  greylev032  greylev064  greylev128  greylev256
```

- **Build a new color maps**
  dcolmapu(colmapu,nlevel,hue,sat,bri) : **define_color_map_user**
  Input : colmapu,nlevel,hue,sat,bri → colmapu in character type
  real hue(nlevel),sat(nlevel),bri(nlevel)
  level must be less than 256.

5.2.7 Page parameters

- **Page size**
  dpagsiz_(pagsix,pagsiy,rmarx,rmary) : **define_page_size**
  Input : pagsix,pagsiy,rmarx,rmary → page sizes and margins in the sheet (cm)

- **Page origine**
  dpagori_(pagorx,pagory : **define_page_origin**
  Input : pagorx,pagory → origin of the left bottom corner of the page in the sheet
  dpagsiz_ and dpagori_ allow to put many mini-pages in a single sheet.

- **Page format**
  dpagfor_(pl) : **define_page_format**
  Input : pl → "p" as portrait or "l" as landscape

- **Page skip**
  dpagnew_ : **define_page_new**
  Input : none

- **Page header**
  dheapos_(ipos,dist) : **define_header_position**
  Input : iposh,dist → -1,0,1 for left, middle or right ; dist in cm.
  dheafon_(fontyp,fonsiz) : **define_header_font**
  Input : fontyp,fonsiz → see dfontyp_ for fontyp; fonsiz in cm.

- **Page foot**
  dfoofon_(fontyp,fonsiz) : **define_foot_font**
  Input : fontyp,fonsiz → see dfontyp_ for font; fonsize in cm.

- **Page scale factor**
  dpagsca_(scax,scay) : **define_page_scale_factor**
  Input : scax,scay → define a scale factor on all the units on the page.

- **Pen position on the page**
  dpagppo_(pox,poy) : **define_page_pen_position**
  Input : pox,poy → move the pen at the given position, without drawing.

- **Restore default attribute for next page**
  dpagdef_ : **define_page_default** ; set standard page default attributes
  Input : none

5.2.8 Figure parameters

- **Figure size**
  dfigsiz_(figsix,figsiy) : **define_figure_size**
  Input : figsix,figsiy → figure size in cm.

- **Figure origine**
  dfigori_(figorx,figory) : **define_figure_**
  Input : figorx,figory → figure origin in cm of the lower left corner.
  dfigorix(figorx) : **define_figure_origine_x**
  Input : figorx) → only for x
  dfigoriy(figory) : **define_figure_origine_y**
  Input : figory) → only for y

- **Figure limits**
  dfiglim_(xmin,xmax,ymin,ymax) : **define_figure_limit**
  Input : xmin,xmax,ymin,ymax → min and max values of the axes's graduations
  dfiglimx(xmin,xmax) : **define_figure_limit_x**
  Input : xmin,xmax → min and max values of the X axes.
  dfiglimy(ymin,ymax) : **define_figure_limit_y**
  Input : ymin,ymax → min and max values of the Y axes.

- **Figure zoom attribute**
  dfigzat_(zoatt) : **define_figure_zoom_attribute**
  Input : zoatt → allows to modify in a single call all sizes for grad.,lab.,marks,title.

- **Plot beyond figure limits**
  dfigout_(yeno) : **define_figure_out**
  Input : yeno → 'no'= no trepassing fig. limits

- **Pen position on the figure**
  dfigppo_(pox,poy) : **define_figure_pen_position**
  Input : pox,poy → move the pen at the given position, without drawing (figure unit).

- **Restore default attribute for next figure**
  dfigdef_ : **define_figure_default** for position of tit,lab,sti,grad
  Input : none

- **Title size**
  dtitsiz_(titsi) : **define_title_size**
  Input : titsi → title size of the figure (cm).

- **Title position**
  dtitpos_(titpo) : **define_title_position**

  Input : titpo → 't' or 'b' (top or bottom)

- **Label size**
  dlabsiz_(rlasix,rlasiy) : **define_labels_size**

  Input : rlasix,rlasiy → label size (height in cm) for axis of the figure.

- **Label position**
  dlabpos_(labpox,labpoy) : **define_labels_position**

  Input : labpox,labpoy → label position (l,c,r and b,c,t) for axis of the figure
  dlabdis_(dilabax,dilabay) : **define_labels_distance**

  Input : dilabax,dilabay → label distance from axis of the figure

- **Y label sens**
  dlabseny(labsey) : **define_labels_sens_y**

  Input : labsey → label sens (h or v) for y label.

- **Graduation size**
  dgrasiz_(grasix,grasiy) : **define_graduations_size**

  Input : grasix,grasiy → x and y graduation size in cm.

  dgrasizx(grasix) : **define_graduations_size**

  Input : dgrasizx → x graduation size in cm.

  dgrasizy(grasiy) : **define_graduations_size**

  Input : dgrasizy → y graduation size in cm.

- **Graduation position**
  dgrapos_(grapox,grapoy) : **define_graduations_position**

  Input : grapox,grapoy → x and y graduation position
  't' (top), 'b' (bottom) for x, 'r' (right), 'l' (left) for y.

- **Graduation style**
  dgrasty_(ilogx,ilogy) : **define_graduations_style**

  Input : ilogx,ilogy → 0=linear, 1=log

  dgrastyx(ilogx) : **define_graduations_style_x**

  Input : ilogx → 0=linear, 1=log

  dgrastyy(,ilogy) : **define_graduations_style_y**

  Input : ilogy → 0=linear, 1=log

- **Sticks size**
  dstisiz_(stisil,stisis) : **define_sticks_size**

  Input : stisil,stisis → long and short sticks size in cm.

- **Sticks position**
  dstipos_(stipox,stipoy) : **define_sticks_position**

  Input : stipox,stipoy → X and Y sticks potion as "ii,io,oo,oi,"
  'i' for input, 'o' for output, ' ' for no sticks.

- **Maximum & minimum sticks number**
  dstinum_(nmin,nmax) : **define_sticks_number**
  Input : nmin,nmax → min and max big sticks number

- **Define a zone to fill**
  dfilzon_: **define_fill_zone**
  Input : None

- **Define reverse axes direction**
  dfigrad_: **define_reverse_axes_direction**
  Input : None

5.3 Get current parameters

5.3.1 Current date & time

- **Clock parameters**
  gclopar_(imonth,iday,iyear,ih,im,is) : **give_clock_parameters**
  Output: imonth,iday,iyear,ih,im,is → integer values

- **Separate date and time fields**
  gdidate_(date10) : **give_digital_date**
  Output: date10 → character*10 value as '2007-09-27'

  gditime_(time8) : **give_digital_time**
  Output: time8 → character*8 value as '13:03:25'

- **US date-time format**
  gusdate_(date18) : **give_us_date**
  Output: date18 → character*18 value as 'September 21, 1988'

  gusdati_(dati29) : **give_us_date_time**
  Output: dati29 → character*29 value as 'October 23, 1992 - 23:50:10'

- **French date-time format**
  gfrdate_(date17) : **give_french_date**
  Output: date17 → character*17 value as '21 Septembre 1988'

  gfrdati_(dati28) : **give_french_date_and_time**
  Output: dati28 → character*28 value as '23 Octobre 1992 - 23:50:10'

- **Usual date-time format**
  gstdati_(dati25) : **give_standart_date_time**
  Output: dati25 → character*25 value as 'October 23, 18:12, 1992'

- **Astronomic date-time format**
  gasdati_(dati26) : **give_astro_date_time**
  Output: dati26 → character*26 value as 'Oct. 23, 18:12, 1992'

5.3.2 Line properties, character fonts & color maps

- **`Line width`**

  glinwid_`(iwidth)` : **give_lines_width**

  Output: `iwidth` → width in pixel (ex: 3)

  glinwidr`(width)` : **give_lines_width_real**

  Output: `width` → width in pixel (real value, ex: 3.1)

- **`Line color`**

  glincol_`(col)` : **give_line_color**

  Output: `col` → as 'r' or any of nrgbcmypwi

  glinhsb_`(h,s,b)` : **give_line_hsb**

  Output: `h,s,b` → real in [0. - 1.]

  glinrgb_`(r,g,b)` : **give_line_rgb**

  Output: `r,g,b` → real in [0. - 1.]

- **`Character fonts`**

  gchapos_`(string,cha,ipos)` : **give_character_position**

  Input: `string,cha` → string and searched character

  Output: `ipos` → character position in the string

  gfonnum_`(numfon)` : **give_font_number**

  Output: `numfon` → number of the font, see 5.2.5

  gfontyp_`(fontyp)` : **give_font_type**

  Output: `fontyp` → with character*36 fontyp, as "c","h","hbold",...

- **`Color maps`**

  gcolmap_`(colmap,nlevel)` : **give_color_map**

  Output: `colmap,nlevel` → name of color map and nb. of colors

  gcolmapa`(colmap,nbmap)` : **give_color_map_array**

  Output: `colmap,nbmap` → names of all color maps with character*(*) colmap(25)

  gcolmapv`(colmap,nlevel,hue,sat,bri)` : **give_color_map_values** n,hue(n),sat(n),bri(n)

  Output: `colmap,nlevel,hue,sat,bri` → name and nb. of level of current color map with hue(nlevel) ,sat(nlevel) ,bri(nlevel)

5.3.3 Page parameters

- **`Page size`**

  gpagsiz_`(pagsix,pagsiy)` : **give_page_size**

  Output: `pagsix,pagsiy` → in cm

- **`Page origine`**

  gpagori_`(pagorx,pagory)` : **give_page_origine**

  Output: `pagorx,pagory` → page origine in the paper sheet (left bottom corner)

- **Page format**
  gpagfor_(pl) : **give_page_format**
  Output: pl → 'p' as portrait or 'l' as landscape

- **Page number**
  gpagnum_(nupag) : **give_page_number**
  Output: nupag → of the current page

- **Page scale factor**
  gpagsca_(scax,scay) : **give_page_scale_factor**
  Output: scax,scay → default is 1., 1.

- **Give page pen position**
  gpagppo_(pox,poy) : **give_page_pen_position**
  Output: pox,poy → position in cm in the page frame

5.3.4 Figure parameters

- **Figure size**
  gfigsiz_(figsix,figsiy) : **give_figure_size**
  Output: figsix,figsiy → figure size in cm.

- **Figure origine**
  gfigori_(figorx,figory) : **give_figure_origin**
  Output: figorx,figory → figure origin in the page (in cm. from lower left corner)

- **Figure limits**
  gfiglim_(xmin,xmax,ymin,ymax) : **give_figure_limit**
  Output: xmin,xmax,ymin,ymax → in figure unit for x and y axis

  gfiglimx(xmin,xmax) : **give_figure_limit_x**
  Output: xmin,xmax → in figure unit for x axis

  gfiglimy(ymin,ymax) : **give_figure_limit_y**
  Output: ymin,ymax → in figure unit for y axis

- **Figure zoom attribute**
  gfigzat_(zoatt) : **give_figure_zoom_attribute**
  Output: zoatt → for grad.,labels,marks,title (default is 1.)

- **Figure scale conversion factor**
  gfigsca_(figscx,figscy) : **give_figure_scale**
  Output: figscx,figscy figure/page scale, in axis's unit/cm

- **Title size**
  gtitsiz_(titsi) : **give_title_size**
  Output: titsi title size in cm

- **Title position**
  gtitpos_(titpo) : **give_title_position**
  Output: titpo → title position as 't' (top) or 'b' (bottom)

- **Label size**
  glabsiz_(rlasix,rlasiy) : **give_labels_size**
  Output: rlasix,rlasiy → labels size, in cm

- **Label position**
  glabpos_(labpox,labpoy) : **give_labels_position**
  Output: labpox,labpoy → as 't' (top), 'b' (bottom), 'r' (right), 'l' (left)

- **Graduation size**
  ggrasiz_(grasix,grasiy) : **give_graduations_size**
  Output: grasix,grasiy → graduation size, in cm

- **Graduation position**
  ggrapos_(grapox,grapoy) : **give_graduations_position**
  Output: grapox,grapoy → grad. position as 't' (top), 'b' (bottom), 'r' (right), 'l' (left)

- **Sticks size**
  gstisiz_(stisil,stisis) : **give_sticks_size**
  Output: stisil,stisis → long and small stick size in cm

- **Sticks position**
  gstipos_(stipox,stipoy) : **give_sticks_position**
  Output: stipox,stipoy → sticks position as 'ii', 'io', 'oo', 'oi'

- **miscellaneous**

  gpaptyp_(paptyp) : **give_paper_type**
  Output: paptyp → as 'A4', 'A3', 'Letter', etc.
  Complete list is given in section 7.1.

  gvernum_(vernum) : **give_library_version_number**
  Output: vernum → real number as 10.0

  gvercha_(vercha) : **give_library_version_character**
  Output: vercha → as '10.0'

  gboubox_(x1ups,x2ups,y1ups,y2ups,pstocm) : **give_bounding_box** , in PS units
  Output: x1ups,x2ups,y1ups,y2ups,pstocm → bounding box limits, in PS units
  pstocm=2.54/72.

  gpripar_ : **give_print_parameter**
  Output: → print the values of all current parameters

5.4 Compute and convert

5.4.1 Computation subroutines

- **`Fonts & character strings`**

cfonbod_(numfon,height,fonbod) : compute_font_body

Input : numfon,height → font number and size (height in cm)
Output: fonbod → font body in cm

cfonnum_(fontyp,numfon) : compute_font_number

Input : fontyp → character*36, as 'tital'
Output: numfon → font number, as 7 for 'tital'

cfontyp_(numfon,fontyp) : compute_font_type

Input : numfon → font number
Output: fontyp → font type, as 'tital' for 7

cchalen_(chast,nbcha) : compute_character_length

Input : chast → string
Output: nbcha → length of the string without end blanks

cchawid_(chast,numfon,height,width) : compute_character_width

Input : chast,numfon,height → string, font number, height in cm
Output: width → width of the string in cm

cbesfor_(x,format) : compute_best_format

Input : x → real number
Output: format → string > 8 characters, as '(f3.1)' for x=2.5

cbesfori(ix,format) : compute_best_format_integer

Input ix: → integer number
Output: format → string > 8 characters, as '(i3)' for ix=124

- **`Axe's graduation & format`**

cfiggra_(xmin,xmax,x1,x2,bgx,sgx,fx) : compute_figure_graduations

Input : xmin,xmax → min and max of the figure
Output: x1,x2,bgx,sgx,fx → rounded figure limits, big and small grad., grad. format

cfiggraa(x,n,x1,x2,bgx,sgx,fx) : compute_figure_graduations_array

Input : x,n → x(n) to plot
Output: x1,x2,bgx,sgx,fx → fig. limits, big and small grad., grad. format

cfiggrah(t1,t2,valgra,titgra,valdt,nbgma,nbpma) : compute_figure_graduations_hourly

Input : t1,t2 → figure limit, in decimal hours (as 12.10, 13.15)
Output: valgra,titgra,valdt,nbgma,nbpma → valgra(nbgma): grad. values,
grad.title, time interval between two grad., number of big and small marks.

cfiggral(xmin,xmax, x1,x2) : compute_figure_graduations_logarithmic

Input : xmin,xmax → figure limits,not rounded
Output: x1,x2 → valgra(nbgma): rounded figure limits

cfigout(x,y,iout) : **compute_figure_out**

Input : x,y → coordinate of a point, in figure unit
Output: iout → iout =1 if point is outside of the figure limit, 0 if not

cgrafor_(x1,x2,bgx,fx) : **compute_graduations_format**

Input : x1,x2,bgx → fig. limits, big grad.
Output: fx → graduation format (string)

cgraste_(x1,x2,bgx,sgx,x1a,x2a) : **compute_graduations_steps** from axis's limits

Input : x1,x2 → limits of axis
Output: bgx,sgx,x1a,x2a → big and small grad., rounded values of x limits

cgrasts_(x1,x2,bgx,sgx,x1a,x2a) : **compute_graduations_steps_simple**

Input : x1,x2 → limits of axes
Output: bgx,sgx,x1a,x2a → big and small grad., rounded values with x2a=-x1a=bgx

- **Floating point accuraccy**

ctenpon_(n,power) : **compute_ten_power_n**

Input : n → integer
Output: power → 10**n without rounded value, avoid 9.9999997 for code portability

ctwopon_(n,ipower) : **compute_two_power_n**

Input : n → integer
Output: iower → 2**n with test n < 2147483647

capowex_(a,x,apowx) : **compute_a_power_x**

Input : a,x → real values
Output: apowx → test |a**x| < 1.e37 for code portability

cmanexp_(x,rma,ie) : **compute_mantissa_and_exponent**

Input : x → real number
Output: rma,ie → mantissa and exponent (real, integer)

csigdig_(x,nosd) : **compute_significant_digits**

Input x: → real number
Output: nosd → number of significant digits

cscasli_(sca,nx,ny,smin,smax,pcmi,pcma) : **compute_scalar_significant_limit**

Input : sca,nx,ny → real sca(nx,ny)
Output: smin,smax,pcmi,pcma → significant min max by removing %min & %max

- **Real & array utility**

cranval_(ranval) : **compute_random_value**

Input : none
Output: ranval random value between 0. and 1.

cmodpha_(cbx,rmx,rpx) : **compute_module_phase**

Input : cbx → complex number
Output: rmx,rpx → module and phase in degrees

cvecmod_(x,y,z,rmod) : **compute_vector_modulus**

Input : x,y,z → 3D vector components
Output: rmod → modulus

cminmax_(ax,n,axmin,axmax) : **compute_mini_maxi**

Input : ax,n → real ax(n)
Output: axmin,axmax → min and max of the array

csor3ra_(ar0,ar1,ar2,n,n1,n2,isens) : **compute_sort_of_3_real_array**

Input : ar0,ar1,ar2,n,n1,n2,isens → 3 arrays to sort, from n1 to n2
Output: ar0,ar1,ar2 → sorted arrays with ar0 as reference

cmatlim_(rma,nx,ny,rmamin,rmamax) : **compute_matrix_limits**

Input : rma,nx,ny → array rma(nx,ny)
Output: rmamin,rmamax → min and max of the array

cmatlimp(rma,nx,ny,rmamin,rmamax,ixmi,iymi,ixma,iyma : **compute_matrix_limits_positions**

Input : rma,nx,ny → array rma(nx,ny)
Output: rmamin,rmamax,ixmi,iymi,ixma,iyma → min and max and location

cscahis_(sca,nx,ny,scamin,scamax,isto,n) : **compute_scalar_histogram**

Input : sca,nx,ny,scamin,scamax,isto,n → sca(nx,ny),min, max, integer array isto(n)
Output: isto,n → histogram integer array isto(n)

ccpxfft_(sio,nbp,ind) : **compute_complex_fft**

Input : sio,nbp,ind → complex array sio(n), n=2**m, direct fft or inverse (ind=+/-1)
Output: sio → spectrum or waveform (ind=+/-1)

- **Geometry**

ccirarc_(x1,y1,x2,y2,r,xc,yc,tet1,tet2) : **compute_circular_arc**

Input : x1,y1,x2,y2,r → two points of the circular arc and radius
Output: xc,yc,tet1,tet2 → center of the circle, and start/end arc in degrees

cellarc_(x1,y1,x2,y2,a,b,xc,yc,tet1,tet2) : **compute_elliptical_arc** from 2 pts & a,b

Input : x1,y1,x2,y2,a,b → two points of the elliptical arc and semi-axis
Output: xc,yc,tet1,tet2 → center of the ellipse, and start/end arc in degrees

cplarot_(xc,yc,angle,x1,y1,x2,y2) : **compute_plane_rotation** from center & angle

Input : xc,yc,angle,x1,y1 → center and rotation angle, point to rotate
Output: x2,y2 → rotated point

- **miscellaneous**

ccollev_(level,hue,sat,bri) : **compute_color_values** for level of current colormap

Input : level → level of current color map
Output: ,hue,sat,bri → h,s,b values

cleayea_(iyear,ileap) : **compute_leap_year**

Input : iyear → integer year
Output: ileap → integer with ileap=1 for leap year, 0 if not

5.4.2 Conversion subroutines

- **`Coordinate systems`**

  ccarpol_`(x,y,r,teta)` : **convert_cartesian_to_polar**

  Input : `(x,y` → plane Cartesian coordinates
  Output: `r,teta` → plane polar coordinates (teta in degrees)

  cpolcar_`(r,teta,x,y)` : **convert_polar_to_cartesian**

  Input : `r,teta` → plane polar coordinates (teta in degrees)
  Output: `x,y` → plane Cartesian coordinates

  ccarsph_`(x,y,z,r,teta,phi)` : **convert_cartesian_to_spherical** degrees

  Input : `x,y,z` → 3D Cartesian coordinates
  Output: `r,teta,phi` → 3D spherical coordinates (teta, phi in degrees)

  csphcar_`(r,teta,phi,x,y,z)` : **convert_spherical_to_cartesian**

  Input : `r,teta,phi` → 3D spherical coordinates (teta, phi in degrees)
  Output: `x,y,z` → 3D Cartesian coordinates

- **`Time`**

  ctimcha_`(ih,im,is,ctime)` : **convert_time_number_to_char**

  Input : `ih,im,is` → integer hour, min, sec
  Output: `ctime` → string as '13:03:05'

  cchatim_`(ctime,ih,im,is)` : **convert_time_character_to_**

  Input : `ctime` → string as '13:03:05'
  Output: `ih,im,is` → hour, min, sec as 13 03 05

  cdatdoy_`(idoy,iyear,imonth,iday)` : **convert_date_from_day_of_year** and year

  Input : `idoy,iyear` → day of the year and year(1) is first day of the year)
  Output: `imonth,iday`

  cdatj70_`(jd70,iyear,imonth,iday)` : **convert_Julian_day_1970**

  Input : `jd70` → julian day (1 is 1970/01/01)
  Output: `iyear,imonth,iday` → integer year, month and day

  cdatsoy_`(imonth,iday,iyear,ih,im,is,isoy)` : **convert_date_and_time**

  Input : `imonth,iday,iyear,ih,im,is` → month, day, year, hour, min, sec
  Output: `isoy` → second of the year

  csoydat_`(isoy,imonth,iday,nbyear,ih,im,is)` : **convert_second_of_year**

  Input : `isoy` → second of the year, assuming no leap year
  Output: `imonth,iday,nbyear,ih,im,is` → month, day, year, hour, min, sec

  cmoncar_`(imonth,cmonth)` : **convert_month_number_to_char**

  Input : `imonth` → month (1 to 12)
  Output: `cmonth` → string as 'January'for imonth=1

  ctimmil_`(milday,ih,im,is,ims)` : **convert_time_milday**

  Input : `milday` → millisecond of the day
  Output: `ih,im,is,ims` → hour, min, sec, milsec

- **Colors**

chsbrgb_(hue,sat,bri,r,g,b) : **convert_hsb_to_rgb**

Input : (hue,sat,bri → HSB components (O. to 1.)
Output: r,g,b → RGB components (O. to 1.)

crgbhsb_(r,g,b,hue,sat,bri) : **convert_rgb_to_hsb**

Input : r,g,b → RGB components (O. to 1.)
Output: ,hue,sat,bri → HSB components (O. to 1.)

crgbico_(ir,ig,ib,ico) : **convert_rgb_to_ico** [0-16777215]

Input : ir,ig,ib → RGB components (0 to 255)
Output: ico → ico value, ex (ir,ig,ib)=(127,36,25) -> ico=8332313

cicorgb_(ico,ir,ig,ib) : **convert_ico_to_rgb**

Input : ico → ico value [0-16777215]
Output: ir,ig,ib → RGB components (0 to 255)

cclehsb_(level,hue,sat,bri) : **convert_color_level_in_hsb**

Input : level → level of the current color map
Output: hue,sat,bri → H,S,B values [0.-1.]

- **Images**

cimahsy_(ima,nx,ny) : **convert_image_to_horizontal_symmetry**

Input : ima,nx,ny → imahe array ima(nx,ny)
Output: ima,nx,ny → ima(nx,ny) with horizontal symmetry

cimasizc(cima,nxmax,nx,ny) : **convert_image_size_character**

Input : cima,nxmax,nx,ny → character type cima(N) with N > nxmax*ny
Output: cima → character type cima(nx,ny)

allows to use an array dimensioned as (nx,ny) in a subroutine while it is dimensioned as (N) in the main program.It can then no longer be used normally in the main program. Useful in F77 only, when no dynamic memory available (not used is F90).

cimasizi(iima,nxmax,nx,ny) : **convert_image_size_integer**

Input : iima(nxmax*ny) → integer type iima(N) with N > nxmax*ny
Output: iima(nx,ny) → integer type iima(nx,ny)

allows to use an array dimensioned as (nx,ny) in a subroutine while it is dimensioned as (N) in the main program.It can then no longer be used normally in the main program. Useful in F77 only, when no dynamic memory available (not used is F90).

cimasizr(rima,nxmax,nx,ny) : **convert_image_size_real**

Input : → real type rima(N) with N > nxmax*ny
Output: → real type rima(nx,ny)

allows to use an array dimensioned as (nx,ny) in a subroutine while it is dimensioned as (N) in the main program.It can then no longer be used normally in the main program. Useful in F77 only, when no dynamic memory available (not used is F90).

cscaima_(sca,nx,ny,scamin,scamax,ima,sbla,swhi) : **convert_scalar_to_image**

Input : sca,nx,ny,scamin,scamax,sbla,swhi → sca(nx,ny)
Output: ima → integer ima(nx,ny) in ico unit

We spread the image between the scamin and scamax values of sca(nx,ny).
We put in black what is lower than sbla (threshold of black) while we put in white what is greater than swhi (threshold of white). No black and white if sbla=swhi

cscalog_(sca,nx,ny,valzer,valneg) : **convert_scalar_image_to_log_values**

Input : sca,nx,ny,valzer,valneg → sca(nx,ny), valzer if sca < 1.e-20, valneg if sca <0.
Output: sca → sca(nx,ny) in log values

cscamul_(sca,nx,ny,rmul) : **convert_scalar_image_to_sca***

Input : sca,nx,ny,rmul → sca(nx,ny), multiplicative factor
Output: sca → sca(nx,ny)

- **Units**

cpagfig_(px,py,fx,fy) : **convert_page_to_figure_units**

Input : px,py → page coordinate in cm
Output: fx,fy → figure coordinates

cfigpag_(fx,fy,px,py) : **convert_figure_to_page_units**

Input : fx,fy → figure coordinates
Output: px,py → page coordinate in cm

cdechex_(inu,hnu) : **convert_decimal_to_hex**

Input : inu → integer value
Output: hnu → character*2 hnu

coctwor_(i4,i3,i2,i1,iword) : **convert_octet_to_word**

Input : i4,i3,i2,i1 → integer value i4,i3,i2,i1 0-255
Output: iword → integer value 0-16777215
ex: for (i4,i3,i2,i1)=(0,127,36,25) -> iword=8332313

cworoct_(iword,i4,i3,i2,i1) : **convert_word_to_octets**

Input : iword → integer value [0-16777215]
Output: i4,i3,i2,i1 → integer value i4,i3,i2,1 [0-255]
ex: for iword=8332313 -> (i4,i3,i2,i1)=(0,127,36,25)

- **Real utility**

cxtolim_(x) : **convert_x_to_low_integer_mantissa**

Input : x → ex: 13.25
Output: x → ex: 10. for input= 13.25 or 18.7 etc.

cxtouim_(x) : **convert_x_to_upper_integer_mantissa**

Input : x → ex: 13.25
Output: x → ex: 20. for input= 13.25 or 18.7 etc.

cxtogpr_(x,nosd) : **convert_x_to_given_precision** with numb. of signi. digits

Input : x,nosd → real and number of desired significant digits
Output: x → real with desired significant digits

5.5 Plot objects

5.5.1 Plot object on a page

- **Area fill**

pfilzon : **plot_fill_zone**

Input : none → fill a zone defined before by dfilzon_

- **Plot lines**

ppagpmo_(tox,toy) : **plot_page_pen_motion**

Input : tox,toy → draw a line from current position to the input point

ppaglin_(orix,oriy,endx,endy) : **plot_page_line**

Input : orix,oriy,endx,endy → draw a line between two points

ppagvli_(x) : **plot_page_vertical_line**

Input : x → plot vertical line at the x position in cm

ppaghli_(y) : **plot_page_horizontal_line**

Input : y → plot horizontal line at the y position in cm

ppagfra_ : **plot_page_frame**

Input : none

ppagcor : **plot_page_corners**

Input : none → corner size are 1/20 of the page size

ppaggri_ : **plot_page_grid**

Input : none → plot a grid on the entire page with 1 cm resolution

ppaggrix(x1,x2,dx,y1,y2) : **plot_page_grid_x**

Input : x1,x2,dx,y1,y2 → vertical bars along x, dx resolution, between y1 & y2

ppaggriy(y1,y2,dy,x1,x2) : **plot_page_grid_y**

Input : y1,y2,dy,x1,x2 horizontal bars along y, dy resolution, between x1 & x2

ppagrec_(orix,oriy,sizx,sizy) : **plot_page_rectangle**

Input : orix,oriy,sizx,sizy → origin and size of the rectangle

ppagrco_(orix,oriy,sizx,sizy,sizcor) : **plot_page_rectangle_corner**

Input : orix,oriy,sizx,sizy,sizcor → origin and size of the rectangle and corner

- **Plot text**

ppagcha_(orix,oriy,ipos,sizx,height,angle,chast) : **plot_page_character_string**

Input : orix,oriy,ipos,sizx,height,angle,chast → origin, position,size x, size y, angle in degrees, string to plot.

ppagchav(x,y,ipos,tax,tay,texte) : **plot_page_character_string_vertically**

Input : x,y,ipos,tax,tay,texte → origin, position, size x, size y, string to plot.

ppagstr_(orix,oriy,ipos,sizx,height,angle,string) : **plot_page_string**

Input : orix,oriy,ipos,sizx,height,angle,string → string could contain grec field into { }

ppagtxt_(ox,oy,cheight,twidth,spac,txt,nl) : **plot_page_text**

Input : ox,oy,cheight,twidth,spac,txt,nl → position of the first word, character height, width of the text, line spacing, txt(nl). Text will be formatted to the desired width

ppagval_(orix,oriy,ipos,sizx,sizy,angle,value,format) : **plot_page_value**

Input : orix,oriy,ipos,sizx,sizy,angle,value,format → origin, position, size x, size y,angle in degrees, value to plot and corresponding format.

ppagtav_(orix,oriy,ipos,sizx,sizy,angle,tex,val,format : **plot_page_text_and_value**

Input : orix,oriy,ipos,sizx,sizy,angle,tex,val,format → origin, position, size x, size y, angle in degrees, value to plot and corresponding format.

ppaghea_(header) : **plot_page_header** add standard date and time

Input : header → character type

ppagfoo_(foot) : **plot_page_foot** add PostScript file name

Input : foot → character type

ppag10pn_(orix,oriy,ipos,sizx,sizy,angle,n) : **plot_page_10**n**

Input : (orix,oriy,ipos,sizx,sizy,angle,n → n is the exponent of 10**n symbol

- **Plot geometric objects and symbols**

ppagcir_(xc,yc,r,teta1,teta2,dteta) : **plot_page_circular_arc**

Input : xc,yc,r,teta1,teta2,dteta → center, radius, arc1, arc2, δarc (degrees)

ppagell_(xc,yc,a,b,rincli,teta1,teta2,dteta) **plot_page_elliptical_arc**

Input : xc,yc,a,b,rincli,teta1,teta2,dteta → center, semi major & minor axis, inclination, arc1, arc2, δarc (degrees)

ppagpar_(orix,oriy,sizx,sizy,angle) **plot_page_parallelogram**

Input : orix,oriy,sizx,sizy,angle → position, size, inclination (degrees)

ppagarr_(orix,oriy,ipos,bodyl,bodyw,headl,headw,angle) : **plot_page_arrow**

Input : orix,oriy,ipos,bodyl,bodyw,headl,headw,angle →

orix,oriy	origin of the arrow
ipos	-1, 0, +1 for origin at the queue, 0 in the middle, 1 at the head
bodyl,bodyw	length and width of the body
headl,headw	length and width of the head
angle	in degree

ppagsym_(orix,oriy,ipos,sizx,sizy,symb) : **plot_page_symbol**

Input : orix,oriy,ipos,sizx,sizy,symb → symbol must be one of the following: 'tria', 'carr', 'plus', 'croi', 'octo', 'dode', 'diam', 'star', 'pota', 'circ', 'hmar', 'vmar'

ppagsun_(ox,oy,rsun,rray,nray,hues,sats,huer,satr) : **plot_page_sun** symbol

Input : ox,oy,rsun,rray,nray,hues,sats,huer,satr → center, radius, length of the rays,number of rays, hue and sat of sun, hue and sat of rays

ppaglea_(x0,y0,sx,sy,ipos) : **plot_pag_leaf**

Input : x0,y0,sx,sy,ipos → origin, size in cm, position

- **Plot figure with curves**

ppagfig1(ox,oy,sx,sy,labx,laby,tit,x,y,n) : **plot_page_figure_1_curve**

Input : ox,oy,sx,sy,labx,laby,tit,x,y,n → origin, figure size, labels, title, x(n),y(n)

ppagfig2(ox,oy,sx,sy,labx,laby,tit,x,y1,y2,n) : **plot_page_figure_2_curves**

Input : ox,oy,sx,sy,labx,laby,tit,x,y1,y2,n → origin, figure size, labels, title, x(n), y1(n), y2(n)

ppagfig3(ox,oy,sx,sy,labx,laby,tit,x,y1,y2,y3,n) : **plot_page_figure_3_curves**

Input : ox,oy,sx,sy,labx,laby,tit,x,y1,y2,y3,n → origin, fig. size, labels, title, x(n),y1(n), y2(n), y3(n)

- **Plot axes**

ppagaxex(ox,oy,rx,x1,x2,xref,bgx,sgx,tgm,tpm,tg,dga,fg) : **plot_page_axe_x**

Input : ox,oy,rx,x1,x2,xref,bgx,sgx,tgm,tpm,tg,dga,fg

ox,oy	coordinates in cm of the origin of the axis
rx	length in cm of the x axis
x1,x2	values in axis unit of the terminals, in ox and ox+rx
xref	reference value for tick marks, multiple of bgx
bgx	big graduation x, interval between 2 big marks
sgx	small graduation x, interval between 2 small marks
tgm	size in cm of major marks
tpm	size in cm of small marks
tg	size in cm of the graduations
dga	distance in cm from the graduations to the axis
fg	format of graduations ex' >(f7.2)>

ppagaxey(ox,oy,ry,y1,y2,yref,bgy,sgy,tgm,tpm,tg,dga,fg) : **plot_page_axe_y**

Input : ox,oy,ry,y1,y2,yref,bgy,sgy,tgm,tpm,tg,dga,fg

ox,oy	coordinates in cm of the origin of the axis
ry	length in cm of the y axis
y1,y2	values in axis unit of the terminals, in ox and ox+rx
yref	reference value for tick marks, multiple of bgx
bgy	big graduation y, interval between 2 big marks
sgy	small graduation y, interval between 2 small marks
tgm	size in cm of major marks
tpm	size in cm of small marks
tg	size in cm of the graduations
dga	distance in cm from the graduations to the axis
fg	format of graduations ex' >(f7.2)>

ppagaxlx(ox,oy,rx,x1,x2,tgm,tpm,tg,dga) : **plot_page_axes_log_x**

Input : ox,oy,rx,x1,x2,tgm,tpm,tg,dga

ox,oy	coordinates in cm of the origin of the axis
rx	length in cm of the x axis
x1,x2	values in axis unit of the terminals, in ox and ox+rx
tgm	size in cm of major marks
tpm	size in cm of small marks
tg	size in cm of the graduations
dga	distance in cm from the graduations to the axis

ppagaxly(ox,oy,ry,y1,y2,tgm,tpm,tg,dga) : **plot_page_axes_log_y**

Input : ox,oy,ry,y1,y2,tgm,tpm,tg,dga

ox,oy	coordinates in cm of the origin of the axis
ry	length in cm of the y axis
y1,y2	values in axis unit of the terminals, in ox and ox+rx
tgm	size in cm of major marks
tpm	size in cm of small marks
tg	size in cm of the graduations
dga	distance in cm from the graduations to the axis

- **Plot images**

ppagima_(orix,oriy,ipos,sizx,sizy,angle,image,nx,ny) : **plot_page_image**

Input : orix,oriy,ipos,sizx,sizy,angle,image,nx,ny → origin, position, size x, size y,angle in degrees, image(nx,ny) in ico values

ppagcmav_(orix,oriy,sizx,sizy,isens) : **plot_page_color_map_vertical**

Input : orix,oriy,sizx,sizy,isens → origin, size x, size y, isens=+1 -> red on top

ppagcmah(orix,oriy,sizx,sizy,isens) : **plot_page_color_map_horizontal**

Input : orix,oriy,sizx,sizy,isens → origin, size x, size y, isens=+1 -> red on right

ppagcmgh(orix,oriy,sizx,sizy,xmin,xmax,sgra) : **plot_page_color_map_grad_horizontal**

Input : orix,oriy,sizx,sizy,xmin,xmax,sgra → origin, size of the color map, min and max of the graduations, graduation size

ppagrgl_(orix,oriy,ipos,sizx,sizy,angle,h,s) : **plot_page_rogralib_logo**

Input : orix,oriy,ipos,sizx,sizy,angle,h,s → origin, position, size, angle, hue and sat [0.-1.]

5.5.2 Plot object on a figure

- **Plot lines**

pfigpmo_(tox,toy) : **plot_figure_pen_motion**

Input : tox,toy → draw a line from current position to the input point in figure unit

pfiglin_(orix,oriy,endx,endy) : **plot_figure_line**

Input : orix,oriy,endx,endy → draw a line between two points in figure unit

pfighli_(y) : **plot_figure_horizontal_line**

Input : y → plot vertical line at the x position in figure unit

pfigvli_(x) : **plot_figure_vertical_line**

Input : x → in figure unit

pfigsta_(y,x1,x2,iwidth,col) : **plot_figure_status** i.e. horizontal limited line

Input : (y,x1,x2,iwidth,col → plot vertical line from x1 to x2, δx, col=color

pfigfra_ : **plot_figure_frame**

Input : none

pfigcor_ : **plot_figure_corners** with scorner size =1/20 figure size

Input : none → corner size are 1/20 of the figure size

pfiggri_(dx,dy) : **plot_figure_grid**

Input : dx,dy → plot a grid on the entire figure with dx & dy resolution

pfiggrix(x1,x2,dx,y1,y2) : **plot_figure_grid_x**

Input : x1,x2,dx,y1,y2 → plot vertical bars along x, dx resolution, between y1 & y2

pfiggriy(y1,y2,dy,x1,x2) : **plot_figure_grid_y**

Input : y1,y2,dy,x1,x2 → plot horizontal bars along y, dy res., between x1 & x2

- **Plot text**

pfigtit_(title) : **plot_figure_title**

Input : title character type

pfigcha_(orix,oriy,ipos,sizx,sizy,angle,chast : **plot_figure_character_string**

Input : orix,oriy,ipos,sizx,sizy,angle,chast → origin, position,size x, size y, angle in degrees, string to plot.

pfigval_(orix,oriy,ipos,sizx,sizy,angle,value,format) : **plot_figure_value**

Input : orix,oriy,ipos,sizx,sizy,angle,value,format → origin, position, size x, size y ,angle in degrees,value to plot and corresponding format.

pfigtav_(orix,oriy,ipos,sizx,sizy,angle,tex,val,format) : **plot_figure_text_and_value**

Input : orix,oriy,ipos,sizx,sizy,angle,tex,val,format → origin, position, size x, size y ,angle in degrees,texte and value to plot and corresponding format.

pfigsym_(orix,oriy,ipos,sizx,sizy,symb) : **plot_figure_symbol**

Input : orix,oriy,ipos,sizx,sizy,symb → symbol must be one of the following: 'tria', 'carr', 'plus', 'croi', 'octo', 'dode', 'diam', 'star', 'pota', 'circ', 'hmar', 'vmar'

- **Plot geometric objects**

pfigcir_(xc,yc,r,teta1,teta2,dteta)) : **plot_figure_circular_arc**

Input : xc,yc,r,teta1,teta2,dteta → center,radius,arc in degrees

pfigell_(xc,yc,a,b,rincli,teta1,teta2,dteta) : **plot_figure_elliptical_arc**

Input : xc,yc,a,b,rincli,teta1,teta2,dteta → center,a,b,arc in degrees

pfigarr_(orix,oriy,ipos,bodyl,bodyw,headl,headw,angle) : **plot_figure_arrow**

Input : orix,oriy,ipos,bodyl,bodyw,headl,headw,angle →

orix,oriy	origin of the arrow
ipos	-1, 0, +1 for origin at the queue, 0 in the middle, 1 at the head
bodyl,bodyw	length and width of the body
headl,headw	length and width of the head
angle	in degree

pfiglea_(x0,y0,sx,sy,ipos) : **plot_figure_leaf**

Input : x0,y0,sx,sy,ipos → origin, size in cm, position

pfigros_(xc,yc,r,angle) : **plot_figure_rotation_symbol**

Input : xc,yc,r,angle → r: radius of the symbol

- **Plot axes, graduations & labels**

pfiggrax(xref,bgx,sgx,forx) : **plot_figure_graduations_x**

Input : xref,bgx,sgx,forx → graduation reference, big graduation, small graduations, graduation format as '(f3.0)',' ','*'

pfiggray(yref,bgy,sgy,fory) : **plot_figure_graduations_y**

Input : yref,bgy,sgy,fory → graduation reference, big graduation, small graduations, graduation format as '(f3.0)',' ','*'

pfiglab_(labelx,labely) : **plot_figure_label**

Input : labelx,labely → axis's labels

pfiglabx(labelx) : **plot_figure_label_x**

Input : labelx → x axis label

pfiglaby(labely) : **plot_figure_label_y**

Input : labely → y axis label

pfigleg_(orix,oriy,ipos,ssx,ssy,symb,col,slx,sly,leg) : **plot_figure_legend**

Input : orix,oriy,ipos,ssx,ssy,symb,col,slx,sly,leg

orix,oriy	: origin of thelegendw
ipos	: -1, 0, +1 for origin at the left, 0 in the middle, 1 at right
ssx,ssy	: size of symbol
col	: color (cha*1)
slx,sly	: size of the legend
leg	: legend

- **Plot curves**

pfigcur_(ax,ay,n) : **plot_figure_curve**

Input : ax,ay,n → ax(n),ay(n) arrays to plot

pfigcurl(ax,ay,n1,n2,nstep) : **plot_figure_curve_loop**

Input : ax,ay,n1,n2,nstep → ax(n),ay(n) arrays to plot, n=n1,n2,nstep

pfigcurs(ax,ay,n,ipos,sizx,sizy,symb,col) : **plot_figure_curve_symbol** size in cm

Input : ax,ay,n,ipos,sizx,sizy,symb,col → ax(n),ay(n) arrays to plot, symbol position, symbol size, symbol, color. Symbol must be one of the following: 'tria', 'carr', 'plus', 'croi', 'octo', 'dode', 'diam', 'star', 'pota', 'circ', 'hmar', 'vmar'

pfigcurd(x1,dx,ay,n) : **plot_figure_curve_dx**

Input : x1,dx,ay,n → x1 for x=x1+(n-1)*dx, ay(n) to plot

pfigculd(x1,dx,ay,n1,n2,nstep) : **plot_figure_curve_loop_dx**

Input : x1,dx,ay,n1,n2,nstep → x=x1+(n-1)*dx, ay(n) with n=n1,n2,nstep

pfighis_(x1,dx,ay,n,col) : **plot_figure_histogram**

Input : x1,dx,ay,n,col → ay(n) for x=x1+(n-1)*dx

- **Plot images**

pfigima_(orix,oriy,ipos,sizx,sizy,angle,image,nx,ny) : **plot_figure_image**

Input : orix,oriy,ipos,sizx,sizy,angle,image,nx,ny → image in ico values

5.5.3 Read image file

- **Read Bitmap file**

rdimbmp_(ifc,fichbmp,nx,ny) : **read_dimension_image_bmp** for 24 bits file

Input : ifc,fichbmp → file unit, bmp file name

Output : nx,ny → dimension of the image

rimabmp_(ifc,fichbmp,imaico,nx,ny) : **read_image_bmp** for 24 bits file

Input : ifc,fichbmp,imaico,nx,ny → file unit, bmp file name, empty imaico(nx,ny)

Output : imaico → imaico(nx,ny) in ico units

- **`Read octet in a file`**
  roctfil_`((ifc,io,numoct)` : **read_octet_on_file** for file opened in direct access
  Input : `ifc,numoct` → file unit, octet number
  Output : `io` → integer value [0-255]

5.5.4 Write image file

- **`Write Bitmap file`**
  wimabmp_`(ifc,fichbmp,ima,nx,ny)` : **write_image_bmp** 24 bits file
  Input : `ifc,ima,nx,ny` → file unit, bmp file name, ima(nx,ny=)
  Output : `none` → file bmp created

5.6 On-line Help command

It is difficult to keep in mind all the ***RoGraLib*** parameters. It is very convenient to use the on-line command: **bin/rgh**. This produce on the screen the following output:

```
==> subroutine category ?  (cp, cv, d, gdt, g, pf, pp, r or all) ->
```

if you answer, for instance, 'd', the following menu is displayed:

```
d :  define modules
subroutine name ?  (choose in the list)
dclogra_   dcolmap_   dcolmapu   dfigdef_   dfiglim_   dfiglimx   dfiglimy   dfigori_
dfigorix   dfigoriy   dfigout_   dfigppo_   dfigsiz_   dfigzat_   dfilzon_   dfonnum_
dfontyp_   dfoofon_   dgrapos_   dgrasiz_   dgrasizx   dgrasizy   dgrasty_   dgrastyx
dgrastyy   dheafon_   dheapos_   dimasha_   dlabdis_   dlabpos_   dlabseny   dlabsiz_
dlincol_   dlincoln   dlincon_   dlindas_   dlinhsb_   dlinrgb_   dlinwid_   dlinwidr
dopegra_   dpagfor_   dpagnew_   dpagori_   dpagppo_   dpagsca_   dpagsiz_   dpaptyp_
drecver_   dstinum_   dstipos_   dstisiz_   dtitpos_   dtitsiz_
==> subroutine name ?  ->
```

if you answer for instance **dfontyp_** you get the result of all possible arguments of this command:

```
==> subroutine name ?  -> dfontyp_
             usage :  call dfontyp_(fontyp)
             object:  define_font_type as 'c','h','hbobl',...
available values for fontyp:
Courier            Courier-B          Courier-I          Courier-BI
Times              Times-B            Times-I            Times-BI
Helvetica          Helvetica-B        Helvetica-I        Helvetica-BI
AvantGarde         AvantGarde-B       AvantGarde-I       AvantGarde-BI
Bookman            Bookman-B          Bookman-I          Bookman-BI
HelveNarr          HelvetNarr-B       HelvetNarr-I       HelvetNarr-BI
LetterGhotic       LetterGhotic-B     LetterGhotic-I     LetterGhotic-BI
NewCentury         NewCentury-B       NewCentury-I       NewCentury-BI
Palatino           Palatino-B         Palatino-I         Palatino-BI
ZapfChancery       ZapfDingbats       Symbol
Arial              Arial-B            Arial-I            Arial-BI
couri              cbold              cobli              cbobl
times              tbold              tital              tbita
helve              hbold              hobli              hbobl
```

If you forget the arguments of the **ppagarr_** subroutine, you run:

```
bin/rgh
==> subroutine category ?  (cp, cv, d, gdt, g, pf, pp, r or all) -> pp
pp :  plot page modules
subroutine name ?  (choose in the list)
pfilzon_   ppag10pn   ppagarr_   ppagaxex   ppagaxey   ppagaxlx   ppagaxly   ppagcha_
ppagchav   ppaginfo   ppagstr_   ppagcir_   ppagcmah   ppagcmgh   ppagcmav   ppagcor_
ppagell_   ppagfoo_   ppagfra_   ppaggri_   ppaggrix   ppaggriy   ppaghea_   ppaghli_
ppagima_   ppaglea_   ppaglin_   ppagpar_   ppagpmo_   ppagrco_   ppagrec_   ppagsun_
ppagsym_   ppagtav_   ppagval_   ppagvli_
==> subroutine name ?  -> ppagarr_
             usage :  call ppagarr_(orix,oriy,ipos,bodyl,bodyw,headl,headw,angle)
             object:  plot_page_arrow
```

Chapter 6 : SUMMARY OF ALL SUBROUTINES

6.1 Compute and convert subroutines

long name	name and arguments	comments
Compute subroutines		
compute_best_format	cbesfor_(x,format)	as '(f3.1)' for x=2.5
compute_best_format_integer	cbesfori(ix,format)	as '(i3)' for ix=124
compute_character_length	cchalen_(chast,nbcha)	without end blanks
compute_character_width	cchawid_(chast,numfon,height,width)	from height in cm
compute_circular_arc	ccirarc_(x1,y1,x2,y2,r,xc,yc,tet1,tet2)	from 2 points and radius
compute_elliptical_arc	cellarc_(x1,y1,x2,y2,a,b,xc,yc,tet1,tet2)	from 2 points and a,b
compute_figure_graduations	cfiggra_(xmin,xmax,x1,x2,bgx,sgx,fx)	from xmin and xmax dat
compute_figure_graduations_array	cfiggraa(x,n,x1,x2,bgx,sgx,fx)	from 1 array x(n)
converts_figure_to_page_units	cfigpag_(fx,fy,px,py)	
compute_font_number	cfonnum_(fontyp,numfon)	from fontyp, as 7 for 'tital'
compute_font_type	cfontyp_(numfon,fontyp)	from numfon as 'tital' for 7
compute_graduations_format	cgrafor_(x1,x2,bgx,fx)	from axe's boundaries+bigstep
compute_graduations_steps	cgraste_(x1,x2,bgx,sgx,x1a,x2a)	from axe's boundaries
compute_graduations_steps_simple	cgrasts_(x1,x2,bgx,sgx,x1a,x2a)	with x2a=-x1a=bgx
compute_mantisse_and_exponent	cmanexp_(x,rma,ie)	
compute_matrice_limits	cmatlim_(rma,nx,ny,rmamin,rmamax)	of rma(nx,ny)
compute_matrice_limits_positions	cmatlimp(rma,nx,ny,rmamin,rmamax, ixmi,iymi,ixma,iyma)	of rma(nx,ny)
compute_mini_maxi	cminmax_(ax,n,axmin,axmax)	of array ax(n)
compute_module_phase	cmodpha_(cbx,rmx,rpx)	of oa complex number
compute_plane_rotation	cplarot_(xc,yc,angle,x1,y1,x2,y2)	from center and angle
compute_random_value	cranval_(ranval)	between 0. and 1.
compute_scalar_istogramme	cscahis_(sca,nx,ny,scamin,scamax,isto,n)	of sca(nx,ny) for n levels
compute_scalar_significant_min_max	cscasli_(sca,nx,ny,scami,scama,pcmi,pcma)	sca(nx,ny) %min, %max
compute_significant_digits	csigdig_(x,nosd)	for a real number
compute_sorting_of_3_real_array	csor3ra_(ar0,ar1,ar2,n,n1,n2,isens)	ar1,ar2 associated to ar0
Convert subroutines		
converts_date_and_time	cdatsoy_(imonth,iday,iyear,ih,im,is,isoy)	to second of the year
converts_decimal_to_hex	cdechex_(inu,hnu)	
converts_cartesian_to_polar	ccarpol_(x,y,r,teta)	degrees
converts_cartesian_to_spherical	ccarsph_(x,y,z,r,teta,phi)	degrees
converts_time_character_to_time	cchatim_(ctime,ih,im,is)	'13:03:05' to 13 03 05
converts_hsb_to_rgb	chsbrgb_(hue,sat,bri,r,g,b)	
converts_ico_to_rgb	cicorgb_(ico,ir,ig,ib)	ico [0-16777215] ir,ig,ib [0-255]
converts_image_to_horizontal_symetrie	cimahsy_(ima,nx,ny)	
converts_month_number_to_char	cmoncar_(imonth,cmonth)	3→ 'March' (cmonth∗9)
converts_page_to_figure_units	cpagfig_(px,py,fx,fy)	
converts_polar_to_cartesian	cpolcar_(r,teta,x,y)	degrees
converts_rgb_to_hsb	crgbhsb_(r,g,b,hue,sat,bri)	all [0.-1.]
converts_rgb_to_ico	crgbico_(ir,ig,ib,ico)	ir,ig,ib [0-255] ico [0-16777215]
converts_scalar_to_image	cscaima_(sca,nx,ny,scamin,scamax, ima„sbla,swhi)	sca(nx,ny), ima(nx,ny),(ico unit)
converts_scalar_image_to_log_values	cscalog_(sca,nx,ny,valzer,valneg)	
converts_scalar_image_to_sca*rmul	cscamul_(sca,nx,ny,rmul)	
converts_second_of_the_year	csoydat_(isoy,imon,iday,nyear,ih,im,is)	to date and time
converts_spherical_to_cartesian	csphcar_(r,teta,phi,x,y,z)	degrees
converts_time_number_to_char	ctimcha_(ih,im,is,ctime)	'13:03:05'
converts_x_to_given_precision	cxtogpr_(x,nosd)	with num. of signi. digits
converts_x_to_low_integer_mantissa	cxtolim_(x)	ex: 13.25 → 10.
converts_x_to_upper_integer_mantissa	cxtouim_(x)	ex: 13.25 → 20.

6.2 Define subroutines

These subroutines changes the current parameters to new ones. The new parameter remains active until a new call. Parameters value can be obtained by the corresponding “give” subroutine (see 5.3 and 6.3).

long name	short name and arguments	comments
File definition		
define_open_graphical_file	dopegra_(ifc,psfile)	and start the plot
define_close_graphical_file	dclogra_	and terminate the plot
Page definition		
define_page_default	dpagdef_	set standard page default attributes
define_page_format	dpagfor_(pl)	’p’ as portrait or ’l’ as landscape
define_page_new	dpagnew_	with preceding page attribute
define_page_origine	dpagori_(pagorx,pagory)	default=(0.,0.)
define_page_pen_position	dpagppo_(pox,poy)	in cm, without plot
define_page_scale_factor	dpagsca_(scax,scay)	default=(1.,1.)
define_page_size	dpagsiz_(pagsix,pagsiy,rmargex,rmargey)	in cm. erase default size
define_page_header_position	dheapos_(iposhe,disthe)	as ipos (-1,0,1) and oy from bottom
Figure definition		
define_figure_default	dfigdef_	for position of tit,lab,sti,grad
define_figure_limit	dfiglim_(xmin,xmax,ymin,ymax)	
define_figure_limit_x	dfiglimx(xmin,xmax)	
define_figure_limit_y	dfiglimy(ymin,ymax)	
define_figure_origine	dfigori_(figorx,figory)	in cm of the lower left corner
define_figure_origine_x	dfigorix(figorx)	in cm of the lower left corner
define_figure_origine_y	dfigoriy(figory)	in cm of the lower left corner
define_figure_out	dfigout_(yeno)	’no’= no trespassing fig. limits
define_figure_pen_position	dfigppo_(pox,poy)	in cm.
define_figure_size	dfigsiz_(figsix,figsiy)	in cm.
define_figure_zoom_attribute	dfigzat_(zoatt)	for grad.,lab.,marks,title
define_figure_reverse_axis	dfigrad_(irevx,irevy)	0=normal, 1=reverse
define_graduations_position	dgrapos_(grapox,grapoy)	on the figure (b,t and l,r)
define_graduations_size	dgrasiz_(grasix,grasiy)	in cm.
define_graduations_size_x	dgrasizx(grasix)	of x axis in cm.
define_graduations_size_y	dgrasizy(grasiy)	of y axis in cm.
define_labels_position	dlabpos_(labpox,labpoy)	on the figure (l,c,r and b,c,t)
define_labels_sens	dlabseny(labsey)	on the figure (h or v, only labely)
define_labels_size	dlabsiz_(rlasix,rlasiy)	in cm.
define_sticks_position	dstipos_(stipox,stipoy)	as ’ii’,io,oo,oi,’ ’ in,out,no
define_sticks_size	dstisiz_(stisil,stisis)	long and short sticks in cm.
define_sticks_number	dstinum_(nmin,nmax)	i.e. min and max big sticks number
define_title_position	dtitpos_(titpo)	on the figure (top or bottom)
define_title_size	dtitsiz_(titsi)	in cm.
Font definition		
define_font_number	dfonnum_(numfon)	
define_font_type	dfontyp_(fontyp)	as ’c’,’h’,’hbobl’,...
define_header_font	dheafon_(fontyp,fonsiz)	as ’tital’, 0.3 cm
define_foot_font	dfoofon_(fontyp,fonsiz)	as ’tital’, 0.3 cm
Line definition		
define_line_color	dlincol_(col)	as ’r’ or nrgbcmypwi
define_line_color_n	dlincoln(level)	from level of current color map
define_line_dash	dlindas_(siz1,blank1,siz2,blank2)	from given parameters
define_line_continue	dlincon_	remove dash line parameter
define_line_hsb	dlinhsb_(h,s,b)	color components as h,s,b values
define_line_rgb	dlinrgb_(r,g,b)	color components as r,g,b values
define_line_width	dlinwid_(iwidth)	in pixel, for next drawn lines
Color map definition		
define_color_map	dcolmap_(colmap)	among existing color map
define_color_map_user	dcolmapu(colmapu)	from hue(256),sat(256),bri(256)
Fill definition		
define_image_shape	dimasha_	must be followed by contour drawing
define_fill_zone	dfilzon_	start of a path to fill

6.3 Give subroutines

long name	name and arguments	comments
Page parameters		
give_page_format	gpagfor_(pl)	'p' as portrait or 'l' as landscape
give_page_number	gpagnum_(nupag)	of the current page
give_page_origine	gpagori_(pagorx,pagory)	in cm
give_page_pen_position	gpagppo_(pox,poy)	in cm
give_page_scale_factor	gpagsca_(scax,scay)	from lower left corner
give_page_size	gpagsiz_(pagsix,pagsiy)	in cm.
Figure parameters		
give_figure_limit_x	gfiglimx(xmin,xmax)	for x axe
give_figure_limit_y	gfiglimy(ymin,ymax)	for y axe
give_figure_origine	gfigori_(figorx,figory)	in cm. from lower left corner
give_figure_scale	gfigsca_(figscx,figscy)	in axe's uni/cm
give_figure_size	gfigsiz_(figsix,figsiy)	in cm.
give_figure_zoom_attribute	gfigzat_(zoatt)	for grad.,labels,marks,title
give_graduations_position	ggrapos_(grapox,grapoy)	in character*1 variable
give_graduations_size	ggrasiz_(grasix,grasiy)	in cm.
give_labels_position	glabpos_(labpox,labpoy)	in character*1 variable
give_labels_size	glabsiz_(rlasix,rlasiy)	in cm.
give_sticks_position	gstipos_(stipox,stipoy)	in character*1 variable
give_sticks_size	gstisiz_(stisil,stisis)	in cm.
give_title_position	gtitpos_(titpo)	in character*1 variable
give_title_size	gtitsiz_(titsi)	in cm.
Font parameters		
give_font_number	gfonnum_(numfon)	
give_font_type	gfontyp_(fontyp)	as 'c','h','hbold',...
Line parameters		
give_line_color	glincol_(col)	as 'n', or rgbcmypw
give_line_hsb	glinhsb_(h,s,b)	color components as h,s,b values
give_line_rgb	glinrgb_(r,g,b)	color components as r,g,b values
give_lines_width	glinwid_(iwidth)	in pixel, for current drawn lines
Color maps parameters		
give_color_map	gcolmap_(colmap,nlevel)	name and nb. of color levels
give_color_map_array	gcolmapa(colmap,nbmap)	names and nb levels for all maps
give_color_map_values	gcolmapv(colmap,nlevel,hue,sat,bri)	of color maps, n,hue(n),sat(n),bri(n)
Current date time parameters		
give_clock_parameters	gclopar_(imonth,iday,iyear,ih,im,is)	at the time call
give_digital_date	gdidate_(date10)	as '2007-09-27'
give_digital_time	gditime_(time8)	as '13:03:25'
give_french_date	gfrdate_(date17)	as '21 Septembre 1988' (17c.)
give_french_date_and_time	gfrdati_(dati28)	as '23 Octobre 1992 - 23:50:10'
give_standart_date_time	gstdati_(dati25)	as 'October 23, 18:12, 1992'
give_us_date	gusdate_(date18)	as 'September 21, 1988' (18c.)
give_us_date_time	gusdati_(dati29)	as 'October 23, 1992 - 23:50:10'
Other parameters		
give_library_version	glibver_(ver)	as 8.0
give_print_parameters	gpripar_	
give_leaf_dimensions	gleadim_(x0,y0,sx,sy,ipos, sxe,sye,x1,x2,y1,y2,yt)	of useful square of the leaf

6.4 Plot subroutines

long name	name and arguments	comments
Page plot		
plot_page_arrow	ppagarr_(orix,oriy,ipos,bodyl,bodyw,headl,headw,angle)	
plot_page_axe_x	ppagaxex(ox,oy,rx,x1,x2,xref,bgx,sgx,tgm,tpm,tg,dga,fg)	with no predefined options
plot_page_axe_y	ppagaxey(ox,oy,ry,y1,y2,yref,bgy,sgy,tgm,tpm,tg,dga,fg)	with no predefined options
plot_page_axe_logarithmic_x	ppagaxlx(ox,oy,rx,x1,x2,tgm,tpm,tg,dga)	with no predefined options
plot_page_axe_logarithmic_y	ppagaxly(ox,oy,rx,y1,y2,tgm,tpm,tg,dga)	with no predefined options
plot_page_character_string	ppagcha_(orix,oriy,ipos,sizx,height,angle,chast)	ipos=-1,0,+1 sizx no used
plot_page_character_string_v	ppagchav(x,y,ipos,tax,tay,texte)	vertically, top to bottom
plot_page_circular_arc	ppagcir_(xc,yc,r,teta1,teta2,dteta)	
plot_page_corners	ppagcor_	
plot_page_elliptical_arc	ppagell_(xc,yc,a,b,rincli,teta1,teta2,dteta)	
plot_page_foot	ppagfoo_(foot)	add ps name
plot_page_frame	ppagfra_	
plot_page_grid	ppaggri_	on the entire page
plot_page_grid_x	ppaggrix(x1,x2,dx,sizy)	as vertical bars along x
plot_page_grid_y	ppaggriy(y1,y2,dy,sizx)	as horizontal bars along y
plot_page_header	ppaghea_(header)	add st.datime
plot_page_horizontal_line	ppaghli_(y)	
plot_page_image	ppagima_(orix,oriy,ipos,sizx,sizy,angle,image,nx,ny)	image in ico values
plot_page_leaf	ppaglea_(x0,y0,sx,sy,ipos)	
plot_page_line	ppaglin_(orix,oriy,endx,endy)	
plot_page_colormap_horizontal	ppagcmah(orix,oriy,sizx,sizy,isens)	isens=+1, red on right
plot_page_colormap_vertical	ppagcmav(orix,oriy,sizx,sizy,isens)	isens=+1, red on right
plot_page_pen_motion	ppagpmo_(tox,toy)	from cur. pos.
plot_page_rectangle_corner	ppagrco_(orix,oriy,sizx,sizy,sizcor)	
plot_page_rectangle	ppagrec_(orix,oriy,sizx,sizy)	
plot_page_symbol	ppagsym_(orix,oriy,ipos,sizx,sizy,symb)	
plot_page_text_and_value	ppagtav_(orix,oriy,ipos,sizx,sizy,angle,tex,val,format)	
plot_page_value	ppagval_(orix,oriy,ipos,sizx,sizy,angle,value,format)	
plot_page_vertical_line	ppagvli_(x)	
plot_page_figure_1_curve	ppagfig1(ox,oy,sx,sy,labx,laby,tit,x,y,n)	standard automatic plot
plot_page_figure_2_curve	ppagfig2(ox,oy,sx,sy,labx,laby,tit,x,y1,y2,n)	standard automatic plot
plot_page_figure_3_curve	ppagfig3(ox,oy,sx,sy,labx,laby,tit,x,y1,y2,y3,n)	standard automatic plot
Figure plot		
plot_figure_arrow	pfigarr_(orix,oriy,ipos,bodyl,bodyw,headl,headw,angle)	
plot_figure_character_string	pfigcha_(orix,oriy,ipos,sizx,sizy,angle,chast)	size in cm
plot_figure_circular_arc	pfigcir_(xc,yc,r,teta1,teta2,dteta)	
plot_figure_corners	pfigcor_	
plot_figure_curve	pfigcur_(ax,ay,n)	from ax(n),ay(n)
plot_figure_curve_dx	pfigcurd(x1,dx,ay,n)	
plot_figure_curve_loop	pfigcurl(ax,ay,n1,n2,nstep)	n=n1,n2,nstep
plot_figure_curve_loop_dx	pfigculd(x1,dx,ay,n1,n2,nstep)	
plot_figure_curve_symbol	pfigcurs(ax,ay,n,ipos,sizx,sizy,symb,col)	size in cm
plot_figure_elliptical_arc	pfigell_(xc,yc,a,b,rincli,teta1,teta2,dteta)	
plot_figure_frame	pfigfra_	
plot_figure_graduations_x	pfiggrax(bgx,sgx,forx)	as '(f3.0)',' ','*'
plot_figure_graduations_y	pfiggray(bgy,sgy,fory)	as '(f3.0)',' ','*'
plot_figure_grid	pfiggri_(dx,dy)	on the entire figure
plot_figure_grid_x	pfiggrix(x1,x2,dx,y1,y2)	as vertical bars along x
plot_figure_grid_y	pfiggriy(y1,y2,dy,x1,x2)	as horizontal bars along y
plot_figure_histogram	pfighis_(x1,dx,ay,n,col)	
plot_figure_horizontal_line	pfighli_(y)	
plot_figure_image	pfigima_(orix,oriy,ipos,sizx,sizy,angle,image,nx,ny)	image in ico values
plot_figure_label	pfiglab_(labelx,labely)	
plot_figure_label_x	pfiglabx(labelx)	
plot_figure_label_y	pfiglaby(labely)	
plot_figure_leaf	pfiglea_(x0,y0,sx,sy,ipos)	
plot_figure_legende	pfigleg_(orix,oriy,ipos,ssx,ssy,symb,col,slx,sly,leg)	size in cm
plot_figure_line	pfiglin_(orix,oriy,endx,endy)	
plot_figure_pen_motion	pfigpmo_(tox,toy)	
plot_figure_rotation_symbol	pfigros_(xc,yc,r,angle)	Earth's rotation symbol
plot_figure_status	pfigsta_(y,x1,x2,iwidth,col)	i.e. horizontal limited line
plot_figure_symbol	pfigsym_(orix,oriy,ipos,sizx,sizy,symb)	size in cm
plot_figure_text_and_value	pfigtav_(orix,oriy,ipos,sizx,sizy,angle,tex,val,format)	
plot_figure_title	pfigtit_(title)	
plot_figure_value	pfigval_(orix,oriy,ipos,sizx,sizy,angle,value,format)	
plot_figure_vertical_line	pfigvli_(x)	
Other plot		
plot_fill_zone	pfilzon_	zone defined before

6.5 Read subroutines

long name	name and arguments	comments
read_dimension_image_bmp	rdimbmp_(ifc,fichbmp,nx,ny)	for 24 bits file
read_image_bmp	rimabmp_(ifc,fichbmp,imaico,nx,ny)	for 24 bits file, ico format, nx-ny defined
read_octet_on_file	roctfil_(ifc,io)	for opened direct file

6.6 Write subroutines

long name	name and arguments	comments
read_image_bmp	wimabmp_(ifc,fichbmp,imaico,nx,ny)	for 24 bits file, ico format, nx-ny defined

6.7 Utility subroutines

This routines are internal subroutines used by ***RoGraLib***. It is generally not recommanded to use it directly, except for instance : ubounds, uboundi, udecfor, uencfor, uperror, upwarni.

long name	name and arguments	comments
u_bounds	ubounds_(x,x1,x2)	x1<x<x2
u_bounds_integer	uboundi_(ix,ix1,ix2)	ix1<ix<ix2
u_compute_graduations_steps	ucgraste(dx,ggx,pgx)	from dx=x2-x1
u_decode_format	udecfor_(format,ifg,n,nd)	ex : F=n.nd
u_encode_format	uencfor_(format,ifg,n,nd)	
u_figure_boundaries	ufigbou_(xcm,ycm)	force xcm,ycm inside the figure
u_for_chsbrgb_	uhuergb_(rm1,rm2,hue,rgb)	
u_plot_character_element	upchael_(cara,x,y,tax,tay,angle)	used by upchast_
u_plot_curve_in_cm.	upcincm_(x,y,ix,iy,in,rmx,rmy,a,*)	from x-y origin, used by upchael_
u_print_error	uperror_(com)	stop the run and print diagnostics
u_print_warni	upwarni_(com)	may be preceding uperror
u_ring_bell	urinbel_	by printing 'OO7' character
u_verify_format	uverfor_(format,com)	stop run and print com if not valid

6.8 Systeme subroutines

The only one and no standard subroutine is sys_gdattim. This subroutine requires the no standard subroutine "CTIME ()" existing in the most of Fortran compiler, which is considered as a Fortran extension, and not as a standard subroutine. Nevertheless, this subroutine runs at least on the following (and cheked) compilor :

• F77, F90, F95 • gFortan on windows and linux • iFort on linux

long name	name and arguments	comments
system_give_date_time	sys_gdattim(imon,id,iy,ih,im,is)	at the call

6.9 Postscript driver subroutines

This routines are internal subroutines used by ***RoGraLib***. It is not recommanded to use it directly.

long name	name and arguments	comments
sysdrv_ps_compute_bounding_box	sdr_cboubox(opsx,opsy)	in ups
sysdrv_ps_close_graphical_file	sps_dclogra	close PostScript file
sysdrv_ps_define_fill_zone	sps_dfilzon	
sysdrv_ps_define_font_number_and_body	sps_dfonnab(numfon,bodfon)	as 5, 10.25
sysdrv_ps_define_image_shape	sps_dimasha	
sysdrv_ps_define_line_color	sps_dlincol(mode,c1,c2,c3)	rgb or hsb components
sysdrv_ps_define_grey_level	sps_dlingle(gle)	between 0. and 1.
sysdrv_ps_define_line_width	sps_dlinwid(iwidth)	in pixels
sysdrv_ps_begin_plot	sps_dopegra(ifcps,psfile)	open PostScript file
sysdrv_ps_define_page_new	sps_dpagnew(nupa,xmin,xmax,ymin,ymax)	and return page number
sysdrv_ps_define_page_pen_position	sps_dpagppo(pox,poy)	in cm
sysdrv_ps_plot_fill_zone	sps_pfilzon	of preceding plot
sysdrv_ps_plot_character_string	sps_ppagcha(chast,pox,poy,angle)	at bottom left corner
sysdrv_ps_plot_page_image	sps_ppagima(pox,poy,sizx,sizy,angle, imag,nx,ny)	
sysdrv_ps_plot_pen_motion	sps_ppagpmo(pox,poy)	in cm
sysdrv_ps_transform_cm_in_ups	sps_tcmiups(pox,poy,opsx,opsy)	
sysdrv_ps_write_file_comment	sps_wfilcom(com)	in .ps file

Chapter 7 : MISCELLANEOUS

7.1 Available media type and paper size

Rogralib allow to manage various media type. Table 7.1 below give all allowed media type with paper size in PS units and in cm. Default value is A4. Setting a media can be done only one time, just after call to **dopegra_**, by calling **dpaptyp_**(paptyp).

Paper type	x ups	y ups	x cm	y cm
A0	2384	3370	84.11	118.89
A1	1684	2384	59.42	84.10
A2	1191	1684	42.03	59.41
A3	842	1191	29.71	42.02
A4	595	842	21.00	29.70
A5	420	595	14.83	20.99
A6	297	420	10.49	14.82
A7	210	297	7.42	10.48
A8	148	210	5.23	7.41
A9	105	148	3.71	5.22
B0	2920	4127	103.02	145.59
B1	2064	2920	72.82	103.01
B2	1460	2064	51.52	72.81
B3	1032	1460	36.42	51.51
B4	729	1032	25.73	36.41
B5	516	729	18.21	25.72
B6	363	516	12.82	18.20
B7	258	363	9.11	12.81
B8	181	258	6.40	9.10
B9	127	181	4.49	6.39
B10	91	127	3.22	4.48
C0	2600	3677	91.73	129.72
C1	1837	2600	64.82	91.72
C2	1298	1837	45.80	64.81
C3	918	1298	32.39	45.79
C4	649	918	22.91	32.38
C5	459	649	16.20	22.90
C6	323	459	11.40	16.19
Folio	595	935	21.00	32.98
Executive	522	756	18.43	26.67
Letter	612	792	21.60	27.94
Legal	612	1008	21.60	35.56
Ledger	1224	792	43.19	27.94
Tabloid	792	1224	27.95	43.18

Table 7.1: *Avalable media type and paper size*

7.2 Postscript output file

Rogralib create a Postscript file Version 3.0. (ASCII file), and can be edited by many applications as Inkscape for instance.
PS file contains a prologue, allowing use of various character fonts, and a setup.

7.3 Conversion of Postscript file

PS files can be easily converted to PNG, JPEG or PDF with GhostScript.

Linux: gs_program is 'gs'
Windows/Msys: gs_program is 'gswin32c.exe

In the following example, we assume that the Postscript file name is PS_name.ps, and the requested image resolution is 300.

• Transform Postscript file in PNG file

gs_program -dBATCH -dNOPAUSE -dSAFER -dQUIET -sPAPERSIZE=a4 -sDEVICE=*png*256
-sOutputFile=*PS_name.png* -r300x300 *PS_name.ps*

• Transform Postscript file in PDF file

gs_program -dBATCH -dNOPAUSE -dSAFER -dQUIET -sPAPERSIZE=a4 -sDEVICE=pdfwrite
-sOutputFile=*PS_name.pdf* -r300 *PS_name.ps*

LIST OF PLOTS

Bibliography

[1] J-M. Rifflet. *La programmation sous UNIX*. MCGRAW-HILL, Paris, 1986-1989.

[2] S. F. Roth. *Real World PostScript*. Addison-Wesley, USA, 1988.

[3] Adobe Systems Incorporated. *Manuel de reference du language PostScript, Deuxieme edition*. Addison-Wesley, France, 1992.

ALPHABETICAL INDEX OF SUBROUTINES

C

D

G

P

R

S

U

W

NOTES

Write here your personal notes.