Plotting in Fortran on the fly with GNUPLOT

Anjishnu Sarkar

Version: 9.41

Contents

1 Prerequisites

For the scripts to work the pre-requisites are gcc, fortran and gnuplot. The scripts had been tested with the following softwares in the given operating systems. The scripts might also work in other softwares/operating systems. The version numbers are indicative too. The scripts can probably run on lower versions too (except `Fortran 90/95`).

Softwares	Linux	Windows
GCC	7.3.0	Mingw / Strawberry Perl*
GNUPLOT	5.2	
FORTRAN	GNU fortran (90/95)	

I would love to hear if the script works in other versions or not.

Note that the scripts were developed mostly on linux operating system. Even though effort was made to make it workable even on windows system, mileage may vary.

2 Requirements

The requirements to plot graphs in fortran on the fly via this method are the two scripts `fortplot.f90` and `cgprint.c`.

3 Installation and compilation

3.1 Compilation without installation (local)

To check if the programs work on your system you can run the following commands (without installation)

```
gcc -c cgprint.c                              # Creates cgprint.o
gfortran -c fortplot.f90                      # Creates fortplot.o
gfortran -c example.f90                       # Creates example.o
gfortran cgprint.o fortplot.o example.o       # Creates a.out or a.exe
./a.out                                       # In linux
a.exe                                         # In windows
```

`example.f90` is any of the examples in the examples folder.

3.2 Installation (global)

Once you are sure that the scripts run on your system you can install the files globally with the following commands:

- Compile the scripts.

```
gcc -c cgprint.c            # Creates cgprint.o
gfortran -c fortplot.f90    # Creates fortplot.o
gfortran -c example.f90     # Creates example.o
```

- Create the library.

```
# Creates libfp.a
ar qc libfp.a fortplot.o cgprint.o
```

*Includes both gcc and gfortran from mingw.

- Move the *.a files to the standard library directory.

```
# On linux
sudo mv libfp.a /usr/lib/

# On windows (with mingw version 8.2.0)
move libfp.a C:\mingw64\lib\gcc\x86_64-w64-mingw32\8.2.0\

# On windows (with Strawberry perl version 4.9.2)
move libfp.a C:\Strawberry\c\lib\gcc\x86_64-w64-mingw32\4.9.2
```

- Copy the `mod` file to the finclude directory.

  In linux to find the current version of `gcc` run the command: `gcc -v`. This provides the version (among other lines) `gcc version 7.3.0`. So

```
# On linux
sudo mv fortplot.mod /usr/lib/gcc/x86_64-linux-gnu/7.3.0/finclude/

# On windows, in one line (with mingw)
move fortplot.mod
C:\mingw64\lib\gcc\x86_64-w64-mingw32\8.2.0\finclude

# On windows, in one line (with strawberry perl)
move fortplot.mod
C:\Strawberry\c\lib\gcc\x86_64-w64-mingw32\4.9.2\finclude
```

- After the installation, compile and run any of the example codes with the following command.

```
# On linux
gfortran example.f90 -lfp && ./a.out

# On windows
gfortran example.f90 -lfp && a.exe
```

4 Simple 2D plot

The following sample code shows how to plot a simple graph after generating the data.

```
1  program main
2  use fortplot
3  implicit none
4
5  ! Generate the data array: x and y
6  ...
7
8  ! Plot
9  ! Kill previous gnuplot windows (if any) and
10 ! open a new pipe to gnuplot.
11 call openplot()
12
```

```
13 ! Optional if only one figure is to be plotted.
14 ! Mandatory for multiple plots (with different 'n') to plot
15 ! on multiple plot windows.
16 call figure()
17 ! call figure(n) ! where n is an integer.
18
19 ! Change default terminal from 'qt' to 'wxt'
20 ! call terminal('wxt')
21
22 ! Configure the plot. Has to be BEFORE plot2d() command
23 call xlabel("x")
24 call ylabel("f(x)")
25 call title('Plot of a graph')
26
27 ! Plot the figure
28 ! 'options' and 'key' are optional fields.
29 ! call plot2d(x,y)
30 ! call plot2d(x,y,options="with lines")
31 call plot2d(x,y, options = "with lines", key = 'f(x)')
32
33 ! To save as pdf/eps/tex file (if required).
34 call saveas('filename.pdf')
35 ! call saveas('filename.pdf', driver = 'postscript')
36 ! call saveas('filename.tex', driver = 'epslatex', &
37 !              options = 'standalone')
38 ! call saveas('filename.tex', options = 'eps input')
39 ! call saveas('filename.eps', savefig = 1)
40 ! call saveas('filename.eps', driver = 'ps')
41
42 ! Close gnuplot and clean up temporary files.
43 ! The delay value delays the deletion of temporary files.
44 ! Required so that gnuplot can plot the graphs.
45 ! Increase or decrease the delay value as per requirement.
46 ! Default: 1.0 secs.
47 call closeplot()
48 ! call closeplot(delay = 1.5)
49
50 end program main
```

Compile and run as given in Sec. 3 to view the plot.

5 *file() subroutines

Some the subroutines have a equivalent `*file()` partner. These subroutines operate on the datafile rather than x, y arrays. Other options remain the same.

Array	File
`plot2d(x,y)`	`plotfile(datafile)`
`plot3d(x,y,z)` or `plot3dg(x,y,z)`	`plotfile(datafile, &` `options = 'using 1:2:3')`
`fit(x,y,...)`	`fitfile(datafile,...)`
`hist(x)`	`histfile(datafile)`

6 Functions and subroutines defined

Following are the list of public functions and subroutines defined within the fortplot module.

6.1 closeplot()

Summary:

```
closeplot(delay)
```

Arguments:

- *delay:* Optional, real. Delays the deletion of files by secs. Default: 1.5 secs.

Example:

```
call openplot()
call figure(1)
plot2d(x,y)

! Close gnuplot with default arguments.
! call closeplot()

! Wait for 0.1 secs before closing the open figure files.
! call closeplot(delay = 5.0)
```

6.2 contour()

Summary:

```
contour(x,y,z,clabel,filled,cntr_options,key)
```

Arguments:

- *x:* Mandatory, 1D real array. x-axis of the plot.
- *y:* Mandatory, 1D real array. y-axis of the plot.
- *z:* Mandatory, 1D real array. z-axis of the plot.

• *clabel:* Optional, logical. Whether to put the contour labels on the plot or not. Default: `.true.`.

• *filled:* Optional, logical. Whether the contour should be filled or not. Default: `.false.`.

• *cntr_options:* Optional, character. Pass any valid contour related gnuplot commands.

• *key:* Optional character. Legend of the plot.

Example:

```
call openplot()
call figure()

! Set the contour levels
call gpcmd("set cntrparam levels incremental 0, 10, 100")

! Set the color palette
call gpcmd("set palette rgbformulae 33,13,10")

! Plot contour
call contour(x,y,u,filled = .true.)

call closeplot()
```

6.3 cplot()

Summary:

`cplot(cfunc,plot,infile,options,key,ivar)`

Plot a function passed as a character array. Useful for plotting after fitting a curve.

Arguments:

• *cfunc:* Mandatory, character. Function to be plotted.

• *plot:* Optional, character. Default value is `'plot'`. For 3D plot pass the option as `plot = 'splot'`.

• *infile:* Optional, character. Pass a input parameter file, if required.

• *options:* Optional, character. Pass standard gnuplot plot options.

• *key:* Optional, character. Legend.

• *ivar:* Optional, character. Default independent variable in gnuplot is x. To change the independent variable to `'theta'` (say), pass `ivar = 'theta'`.

Example:

- Plotting a 2D curve

```
character(:), allocatable :: curve, inparam
real :: a, b, c

! Define the curve
curve = 'a*exp(- (x-b)**2 / (2*c**2))'

! The parameters
a = 0.3967 ;  b = 0.0019 ;  c = 1.0039

! Write to input parameter file
inparam = 'parameters.txt'
open(10, file = trim(inparam))
write(10,*) "a = ", a ;     write(10,*) "b = ", b  ;
write(10,*) "c = ", c
close(10)

call openplot()
call figure()

call cplot(curve, infile = inparam)

call closeplot()
```

- Plotting a 3D curve

```
character(:), allocatable :: curve

curve = 'sin(x)*cos(y)'

call openplot()
call figure()
call xrange(-1.0,1.0)
call yrange(-2.0,2.0)

call gpcmd("set isosamples 40")
call gpcmd("set hidden3d")
call gpcmd("set ztics 0.3")

call gpcmd("set view 75, 50")

call cplot(curve,plot = 'splot')

call closeplot()
```

See Also:

fit, fplot2d, fplot3d

6.4 deg2rad()

Summary:

`deg2rad(theta_deg)`

Convert a angle in degrees to radians.

Arguments:

• *theta_ deg:* Mandatory, real. Angle in degrees.

Example:

```
theta_deg = (/ 30.0, 60.0, 90.0 /)
! theta_deg = 180.0
theta_rad = deg2rad(theta_deg)
```

See Also:

rad2deg

6.5 extname()

Summary:

`extname(infile)`

Arguments:

• *infile:* Mandatory, character. Name of the file whose extension is required. This function is internally called by `saveas()` subroutine to determine the extension of the input file. Not usually required by the user.

Example:

```
character(:), allocatable :: ext
ext = extname(savefile)
```

6.6 figure()

Summary:

`figure(figid)`

This command also creates a temporary gnuplot file and saves all the gnuplot commands passed by the script. The name of the files is generated as follows: `gp_000001.tpl`, `gp_000002.tpl` and so on, where the number is the number of the `figid`.

Note: The figure id should be in the range 1 to 399.

Arguments:

• *figid:* Optional, integer. Necessary command to plot multiple graphs in different windows.

Example:

```
! Set figure id to 1. Note that default figure id starts from 1.
call figure(1)
call plot2d(x,y1)

! Plot to a different window
call figure()  ! Automatically set the next figure id to 2.
call plot2d(x,y2)
```

6.7 fit()

Summary:

```
fit(x, y, func, infile, outfile, ivar, options)
```

Arguments:

- *x:* Mandatory, 1D real array. x-axis of the plot.
- *y:* Mandatory, 1D real array. y-axis of the plot.
- *func:* Mandatory, character. Function to be fitted.
- *infile:* Mandatory, character. Input parameter file before fitting.
- *outfile:* Optional, character. Output parameter file after fitting. The fitted parameters are by default displayed on the terminal.
- *ivar:* Optional, character. Re-define the independent variable. Default: 'x'.
- *options:* Optional, character. Options for fitting.

Example:

```
real :: a, b, c
character(:), allocatable ::  curve, inparam, outparam

inparam = 'initial_parameters.txt'
outparam = 'fitted_parameters.txt'

! Default values of parameters
a = 1.3 ; b = 0.2 ; c = 0.7 ;

! Write to input parameter file
open(10, file = trim(inparam))
write(10,*) "a = ", a ;     write(10,*) "b = ", b  ;
write(10,*) "c = ", c
close(10)

curve = 'a * cos( b + c * x )'

! Fit x and y (1D arrays) with curve with input parameters written
! in inparam file. Pass the fitted variables to outparam
```

```
call fit(x,y,curve,inparam,outparam)

! Call on hold to plot on the same window
call hold(.true.)

! Plot the data
call plot2d(x,y,options = 'with points', key = 'Data')

! Plot the curve with fitted parameters
call cplot(curve,outparam,options = 'with lines', key = 'Fitted Curve')

! Call off hold
call hold(.false.)
```

See Also:

cplot, fitfile

6.8 fitfile()

Summary:

`fitfile(datafile, func, infile, outfile, ivar, fitoptions, options)`

Arguments:

• *datafile:* Mandatory, character. Name of the file containing the data to be fitted instead of x and y arrays.

• *fitoptions:* Optional, character. Pass options to `set fit quiet`.

Rest of the options are same as `fit()` (See Sec. 6.7).

Example:

```
call fitfile(datafile,curve,inparam)
```

See Also:

fit

6.9 fplot2d()

Summary:

`fplot2d(func,xmin,xmax,xn,options,key)`

Arguments:

• *func:* Mandatory, character. External / Internal function defined.

• *xmin:* Mandatory, real. Lower range of xrange.

- *xmax:* Mandatory, real. Upper range of xrange.
- *xn:* Optional, integer. Number of samples between *xmin* and *xmax*. Default is 100.
- *options:* Optional, character. Gnuplot lines/points options for the plot.
- *key:* Optional, character. Legend of the plot.

Example:

```
character(:), allocatable :: curve, inparam
real :: a, b, c

curve = 'a*exp(- (x-b)**2 / (2*c**2))'

! Write to input parameter file
inparam = 'parameters.txt'
open(10, file = trim(inparam))
a = 0.3967 ;    write(10,*) "a = ", a ;
b = 0.0019 ;    write(10,*) "b = ", b ;
c = 1.0039 ;    write(10,*) "c = ", c ;
close(10)

call cplot(curve,infile = inparam)
```

See Also:

cplot, fplot3d, plot2d, plot3d, plot3dg, plotfile

6.10 fplot3d()

Summary:

```
fplot3d(func,xmin,xmax,ymin,ymax,xn,yn,options,key)
```

Arguments:

- *func:* Mandatory, character. External / Internal function defined.
- *xmin:* Mandatory, real. Lower range of xrange.
- *xmax:* Mandatory, real. Upper range of xrange.
- *ymin:* Mandatory, real. Lower range of yrange.
- *ymax:* Mandatory, real. Upper range of yrange.
- *xn:* Optional, integer. Number of samples between *xmin* and *xmax*. Default is 100.
- *yn:* Optional, integer. Number of samples between *ymin* and *ymax*. Default is 100.
- *options:* Optional, character. Gnuplot lines/points options for the plot.
- *key:* Optional, character. Legend of the plot.

Example:

See Also:

cplot, fplot2d, plot2d, plot3d, plot3dg, plotfile

6.11 get_fwhm()

Summary:

`get_fwhm(gaussian,infile,outfile,ivar)`

Arguments:

• *gaussian:* Mandatory, character. The gaussian curve with proper parameters in string format.

• *infile:* Optional, character. Name of file which contains the parameter values.

• *outfile:* Optional, character. Name of data file which would contain the Full-Width-Half-Maximum (FWHM) values.

• *ivar:* Optional, character. Set the independent variable of the gaussian curve. Default value of the independent variable is x.

Example:

```
tablefile = 'hist_table.dat'

! Input and fitted parameter filenames
inparam  = 'input_params.tdf'
fitparam = 'fitted_params.tdf'

! Define a gaussian curve
gaussian = 'a*exp(-(x-b)**2/(2*c**2))'

! Initialize the parameters with suitable values.
a = 0.65 ;   b  = 0.1 ;  c = 2.0 ;

! Initialize the parametes and write them to the parameter file
open(10, file = inparam)
write(10,*) "# Input parameters for fitting."
write(10,*) "a = ", a ;     write(10,*) "b = ", b ;
write(10,*) "c = ", c ;
close(10)

call hold(.true.)

call hist(x, nbins = nbins, norm = 'probability', &
    key = 'Histogram', alpha = 0.1, outfile = tablefile)

! Fit the curve
! Use the outfile of hist to fit a gaussian curve
call fitfile(tablefile,gaussian,inparam,outfile = fitparam)
```

```
! Plot the fitted curve
xmin = minval(x)
xmax = maxval(x)

call xrange(xmin,xmax)
call cplot(gaussian,fitparam, options = "with lines linecolor 7", &
            key = 'a*exp(-(x-b)^2/(2*c^2))')

fwhmfile = 'fwhm.dat'
fwhm = get_fwhm(gaussian, infile = fitparam, outfile = fwhmfile)
print*, "FWHM = ", fwhm

! Plot the FWHM using the outfile of get_fwhm()
call plotfile(fwhmfile, options = 'with lines linecolor -1 dashtype 2',
                key = 'FWHM')

call hold(.false.)
```

See Also:

fit, fitfile, cplot

6.12 gpcmd()

Summary:

`gpcmd(text,newline,tofile)`

The most important command in this module which is used frequently by other subroutines.

Arguments:

- *text:* Mandatory, character. Pass any valid gnuplot commands via this command.
- *newline:* Optional, logical. Decides whether the `text` should be terminated by newline or not. Default: `.true.`
- *tofile:* Optional. logical. Valid values are `.true.` or `.false.`. Default value is `.true.`. Decides whether to write to the gnuplot file or not.

Example:

See example of Sec. 6.27.

6.13 gptofort()

Summary:

`gptofort(iofile, nmlist)`

Convert a file created by gnuplot to be ready for upload in fortran via namelist. Also replaces '#' comment symbol by '!' symbol.

Arguments:

- *iofile:* Mandatory, character.
- *nmlist:* Mandatory, character.

Example:

See Also:

6.14 grid()

Summary:

```
grid(stat, options)
```

Arguments:

- *stat:* Mandatory, logical. Valid values are .true. or .false.. Sets the grid.
- *options:* Optional, character. For more details check gnuplot help on grid.

Example:

```
call grid(.true.)
call plot2d(x,y, options = 'with lines', key = 'Plot 1')
```

6.15 hist()

Summary:

```
hist(x,nbins,norm,key,fillcolor,linecolor,alpha)
```

The histogram formula used in this subroutine is given by

```
xmin = STATS_min
xmax = STATS_max
dx = (xmax - xmin) / real(nbins)
hist(x,dx) = dx*( floor( (x-xmin)/dx ) + 0.5 ) + xmin
```

This follows the discussion on stackoverflow [1].

Arguments:

- *x:* Mandatory, 1D array. The array whose histogram is required.
- *nbins:* Optional, integer. Number of bins of the histogram. Default: 50.

• *norm:* Optional, character. Normalization factor of the histogram. The various types of normalization and their corresponding normalization factor is taken from normalization table from [2]. Default: `count`.

• *key:* Optional, character. Legend of the histogram.

• *fillcolor:* Optional, character. Fill color of the histogram Default: Green.

• *linecolor:* Optional, character. Line color of the histogram. Default: Green.

• *alpha:* Optional, real. Transparency of the histogram. Default: 0.6

Example:

```
call hist(x)
! call hist(x, nbins = 30, fillcolor = 'blue', alpha = 0.8)
```

See Also:

histfile

6.16 histfile()

Summary:

Draws a histogram (same as Sec. 6.15). However instead of x arrays it accepts filename which has the relevant data in the 1st column.

```
histfile(datafile,nbins,norm,key,fillcolor,linecolor,alpha,outfile)
```

Arguments:

• *datafile:* Mandatory, character. Name of the file which contains the data in the 1st column.

Rest of the arguments are same as in Sec. 6.15.

Example:

```
call histfile('monte.dat', nbins = 20)
```

See Also:

hist

6.17 histtable()

Summary:

```
histtable(x,outfile,nbins,norm)
```

Get the histogram table. The table is the x (center of the bar) and y (top of the bar) values of the histogram bars. This table can be used to fit a gaussian.

Arguments:

• *x:* Mandatory, 1D array. The array whose histogram is required.

• *outfile:* Mandatory, character. The subroutine outputs the top center of each bar in a table format along with their corresponding x values in the supplied filename.

• *nbins:* Optional, integer. Number of bins of the histogram. Default: 50.

• *norm:* Optional, character. Normalization factor of the histogram. The various types of normalization and their corresponding normalization factor is taken from normalization table from [2]. Default: `count`.

Example:

```
tablefile = 'histtable' // '.' // dext
norm = 'probability'
nbins = 50

call histtable(x, outfile = tablefile, nbins = nbins, norm = norm)\

! Define a gaussian curve
gaussian = 'a*exp(-(x-b)**2/(2*c**2))'
! Input and fitted parameter filenames
inparam  = 'input_params' // '.' // fext
fitparam = 'fitted_params' // '.' // fext

! Initialize the parametes and write them to the parameter file
open(10, file = inparam)
write(10,*) "# Input parameters for fitting."

a = 0.65 ;   b  = 0.1 ;  c = 2.0 ;

write(10,*) "a = ", a ;     write(10,*) "b = ", b ;
write(10,*) "c = ", c ;
close(10)

! Fit the curve
! Use the outfile of hist to fit a gaussian curve
call fitfile(tablefile,gaussian,inparam,outfile = fitparam)
```

6.18 hold()

Summary:

`hold(stat)`

If multiple plots are required off the same size on the same window (not multiplot), then this subroutine has to be called. Must be called before the first `plot` command.

Arguments:

• *stat:* Mandatory, logical. Valid values are `.true.` or `.false.`.

Example:

```
call figure()
call hold(.true.) ! Must be called BEFORE the first plot command.
call plot2d(x,y1,options = 'with lines', key = 'Plot 1')
call plot2d(x,y2,options = 'with lines', key = 'Plot 2')
call hold(.false.)
```

6.19 ignoreu()

Summary:

`ignoreu()`

Useful function to define to ignore the rows with `'u'` values when the data is saved as a table by gnuplot. This is due to a bug in gnuplot [3]

Arguments:

This subroutine doesn't accept any arguments.

Example:

```
call ignoreu()
```

6.20 int2str()

Summary:

`int2str(i, wfmt)`

Arguments:

- *i:* Mandatory, integer. Integer number which needs to be converted to string. Also see Sec. 6.32.
- *wfmt:* Optional, character. Formatting command. Short for write format.

Example:

```
character(:), allocatable :: label_str
integer :: i

i = 10
label_str = 'i = ' // int2str(i)           ! For simple integer
! label_str = 'i = ' // int2str(i,'i0.6') ! Integer with zero padding
call label(1,label_text,0.1,0.8)
```

See Also:

real2str

6.21 key()

Summary:

`key(stat,options)`

Arguments:

- *options:* Mandatory, character. Any valid `key` options passed to gnuplot.

Example:

```
! Set the key.
call key("on")

! Unset the key
call key("off")

! Set the key with a box on the bottom left corner.
! "on" keyword not required.
call key("box bottom left")
```

6.22 label()

Summary:

`label(tag,label_text,x,y,z,overlay,options)`

Arguments:

- *tag:* Mandatory, integer. Tag number of the label.
- *label_text:* Mandatory, character. Set the label text.
- *x:* Mandatory, real. x position of the label.
- *y:* Mandatory, real. y position of the label.
- *z:* Optional, real. z position of the label for 3D plots.
- *overlay:* Optional, character. Set the overlay. Specify the coordinate system. Default is `screen`. For more details check gnuplot help on `coordinates`.
- *options:* Optional, character. Any other gnuplot options to be passed to label. Default: Empty.

Example:

```
character(:), allocatable :: label_text
real :: a, g

a = 2.0 ; g = 20.0 ;
label_text = "a = " // real2str(a,'f5.2') // ",\n" // &
             "g = " // real2str(g,'f5.2')
call label(1,label_text,0.15,0.8)
```

6.23 mkdir()

Summary:

`mkdir(dirname)`

Arguments:

- *dirname:* Create a directory with name `dirname`.

Example:

```
call mkdir('mytempdir')
```

See Also:

rmdir

6.24 multiplot()

Summary:

`multiplot(stat,options)`

Plot multiple figures on the same window with different plot sizes.

Arguments:

- *stat:* Mandatory, logical. Valid values are `.true.` or `.false.`.
- *options:* Optional, character. Pass any valid gnuplot options to multiplot.

Example:

```
character(:), allocatable :: mplot_options

mplot_options = "layout 2,2 rowsfirst title 'Simple multiplot example'"

call multiplot(.true., options = mplot_options)

! Set some configurations
call gpcmd('set sample 300')
call gpcmd("set tics font ',10'")
call key(.true., "nobox noopaque font ',10'")

call plot2d(x,y1, options="with lines", key = 'sin(x)')
call plot2d(x,y2, options="with lines", key = 'sin(x^2)')
call plot2d(x,y3, options="with lines", key = '1/(x+1)')
call plot2d(x,y4, options="with lines", key = 'tan(x)')

call multiplot(.false.)

call saveas('multiplot.pdf')
```

6.25 openplot()

Summary:

```
openplot(closeall, debug, delete)
```

Arguments:

• *closeall:* Optional, logical. Whether to remove any previous gnuplot windows or not. Default: `.true.`.

• *debug:* Optional, logical. Whether to print the gnuplot commands on the terminal or not instead of actual plotting. Default: `.false.`.

• *delete:* Optional, logical. To delete the temporary directory `gnuplot_files` before plotting. Default: `.true.`.

Example:

See example given in Sec. 4.

6.26 plot2d()

Summary:

```
plot2d(x,y,options,key)
```

Arguments:

• *x:* Mandatory, 1D real array. x-axis of the plot.

• *y:* Mandatory, 1D real array. y-axis of the plot.

• *options:* Optional, character. Gnuplot lines/points options for the plot.

• *key:* Optional character. Legend of the plot.

Example:

```
call plot2d(x,y, options = "with lines linetype -1 linewidth 1.5", &
            key = 'f(x)')
```

See Also:

cplot fplot2d, fplot3d, plot3d, plot3dg,

6.27 plot3d()

Summary:

`plot3d(x, y, z, options, key)`

The array size of x, y and z should be same.

Arguments:

- *x:* Mandatory, 1D real array. x-axis of the plot.
- *y:* Mandatory, 1D real array. y-axis of the plot.
- *z:* Mandatory, 1D real array. z-axis of the plot.
- *options:* Optional, character. Gnuplot line/points options for the plot.
- *key:* Optional character. Legend of the plot.

Example:

```
real, dimension(:), allocatable :: t, x, y, z
real :: tmax, tmin, dt
integer:: i, n, ok

tmin = 0.0 ; tmax = 20.0 ; dt = 0.1
n = floor( (tmax - tmin) / dt ) + 1
t = [(tmin + (i-1)*dt, i = 1,n)]

allocate(x(n), y(n), z(n), stat = ok)
if (ok /= 0) stop "Couldn't allocate x, y or z arrays."

x = t * cos(t)
y = t * sin(t)
z = -t**2         ! z is 1D array, just like x and y.

call openplot()

! Change default terminal to 'wxt' for 3D plots, to rotate figure
call terminal('wxt')

call gpcmd('set ztics 0,100,400')
call gpcmd('set view 60,45')

call plot3d(x,y,z,options = 'with lines', key = '3D plot')
call closeplot()
```

See Also:

cplot fplot2d, fplot3d, plot2d, plot3dg

6.28 plot3dg()

Summary:

`plot3dg(x, y, z, options, key)`

The array size of x, y need NOT be same. However, `size(z)= size(x)* size(y)`.

Arguments:

- *x:* Mandatory, 1D real array. x-axis of the plot.
- *y:* Mandatory, 1D real array. y-axis of the plot.
- *z:* Mandatory, 2D real array. z-axis of the plot.
- *options:* Optional, character. Gnuplot line/points options for the plot.
- *key:* Optional character. Legend of the plot.

Example:

```
! Here z is 2D array.
do i = 1,xn
    do j = 1,yn
        z(i,j) = func(x(i),y(j))
    end do
end do

! The data is in a grid format. So call plot3dg().
call plot3dg(x,y,z,options = 'with lines', key = '3D plot2')
```

See Also:

cplot fplot2d, fplot3d, plot2d, plot3d

6.29 plotfile()

Summary:

`plotfile(datafile,plot,options,key)`

Arguments:

- *datafile:* Mandatory, character. Save the data in a file and pass the name of the datafile to plot.
- *plot:* Optional, character. Set the plot command to `plot` for 2D or `splot` for 3D. Default is `plot`.
- *options:* Optional, character. Gnuplot line/points options for the plot
- *key:* Optional character. Legend of the plot.

Example:

```
call plotfile("datafile.txt", options = "with lines", key = 'f(x)')
```

See Also:

cplot fplot2d, fplot3d, plot2d, plot3d, plot3dg

6.30 polarplot()

Summary:

`polarplot(theta,rho,options,key,polar_options)`

Plot in polar coordinates.

Arguments:

- *theta:* Mandatory, real. Theta of polar coordinate (in radians).
- *rho:* Mandatory, real. Rho of polar coordinate.
- *options:* Optional, character. Gnuplot line/points options for the plot.
- *key:* Optional character. Legend of the plot.
- *polar_options:* Optional, character. Specific options for polar plot.

Example:

```
pi = 4.0 * atan(1.0)
dtheta = 0.01
n = floor((2*pi - 0 ) / dtheta) + 1

! theta is in radians
theta = [(0+(i-1)*dtheta, i = 1,n)]
rho = sin(2*theta)*cos(2*theta)

call openplot()
call figure()

call xrange(-0.5,0.5)
call yrange(-0.5,0.5)

call polarplot(theta,rho, options = "with lines", &
                key = "Polar plot" )

call saveas("polarplot1.eps")

call closeplot()
```

6.31 rad2deg()

Summary:

`rad2deg(theta_rad)`

Convert a angle in radians to degrees.

Arguments:

• *theta_ rad:* Mandatory, real. Angle in radians.

Example:

```
pi = 4.0 * atan(1.0)
theta_deg = rad2deg(pi/2.0)
```

See Also:

deg2rad

6.32 real2str()

Summary:

`real2str(x,wfmt)`

Arguments:

• *x:* Mandatory, real. Real number which needs to be converted to string. Also see Sec. 6.20.

• *wfmt:* Optional, character. Formatting command. Short for write format.

Example:

See example of Sec. 6.22.

See Also:

int2str

6.33 rmdir()

Summary:

`rmdir(dirname)`

Arguments:

• *dirname:* Remove the directory `dirname`. Deletes only empty directory.

Example:

```
call rmdir('mytempdir')
```

See Also:

mkdir

6.34 saveas()

Summary:

```
saveas(savefile, savefig, options, driver)
```

Arguments:

• *savefile:* Mandatory, character. Define the output file. The terminal is set based on the file extension. Valid extensions, drivers (if required) and their corresponding default terminal options are given in the table below.

File extension	Driver	Default terminal options
eps	epscairo†	`epscairo enhanced color notransparent`
	postscript	`postscript enhanced color eps`
pdf	pdfcairo†	`pdfcairo enhanced color notransparent`
	postscript‡	`postscript enhanced color eps`
tex	cairolatex†	`cairolatex color pdf notransparent`
	epslatex	`epslatex color background 'white'`
gif	gif	`enhanced`§
jpg	jpeg	`jpeg enhanced`
png	pngcairo	`pngcairo enhanced color notransparent`
ps	postscript	`postscript enhanced color`
svg	svg	`svg enhanced`

• *savefig:* Optional, integer. Figure number to be saved.

• *options:* Optional, character. Pass any other valid gnuplot terminal options.

• *driver:* Optional, character. Only valid for the following file extensions: eps, pdf, tex.

Example:

See example in Sec. 4

†Default driver.
‡Requires epstopdf. However, this doesn't work in windows OS.
§See Sec. 7.1 for animated gif.

6.35 set_polarplot()

Summary:

`set_polarplot(setpolar)`

The subroutine sets the polar plot specific configurations. After the polar plot the configurations are reset.

Arguments:

• *setpolar:* Mandatory, logical. Valid values are `.true.` and `.false.`.

Example:

```
call set_polarplot(.true.)
! call set_polarplot(.false.)
```

See Also:

polarplot

6.36 slumber()

Summary:

`slumber(delay)`

Arguments:

• *delay:* Mandatory, real. Pause the code for `delay` seconds. Can even pause for microseconds.

Example:

```
do i = 1, n
    a = a + 0.1
    y = sin(x+a)  ! Where x is an array.
    call plot2d(x, y, options = 'with lines')
    call slumber(0.1)
end do
```

6.37 standby()

Summary:

`standby(portfolio, delay, maxiter, dir)`

Subroutine to wait for a file/folder to be created before proceeding. Note that this subroutine doesn't create the file/folder. Only waits for it to be created by others.

Arguments:

• *portfolio:* Mandatory, character. Name of the file/folder which should have been created.

• *delay:* Mandatory, real. Sleeps for `delay` secs in each loop.

• *maxiter:* Mandatory, integer. Loop for maxiter number of times, sleeping for `delay` secs in each loop.

• *dir:* Optional, logical. Can be `.true.` or `.false.`. Specifies whether the string passed as portfolio is a directory or not. Default is `.false.`.

Example:

```
! Wait for file "fortfile.txt" to be created in 10 maximum iterations,
! each time waiting for 0.2 secs.
call standby("fortfile.txt", 0.2, 10)

! Same as above for the directory "data".
call standby("data", 0.2, 10, dir = .true.)
```

6.38 terminal()

Summary:

```
terminal(term)
```

Arguments:

• *term:* Mandatory, character. Sets the terminal for the plot. Default terminal is `'qt'`. Should be used before calling the `figure()` (Sec. 6.6) subroutine.

Example:

```
call openplot()
call terminal('wxt')
call figure()
...
call closeplot()
```

6.39 test()

Summary:

```
test()
```

Arguments:
This subroutine doesn't accept any arguments. It shows the test window for the terminal selected.

Example:

```
call openplot()
call test()
call closeplot()
```

6.40 title()

Summary:

```
title(text)
```

Arguments:

- *text:* Mandatory, character. Set the title text of the plot.

Example:

```
call title('Plot of a graph')
```

6.41 [x|y|z]label()

Summary:

```
xlabel(text)
ylabel(text)
zlabel(text)
```

Arguments:

- *text:* Mandatory, character. Set the x|y|z label.

Example:

```
call xlabel('x')
call ylabel('f(x)')
call zlabel('z')  ! Only for 3D plots.
```

6.42 [x|y|z]range()

Summary:

```
xrange(llim,ulim)
yrange(llim,ulim)
zrange(llim,ulim)
```

Arguments:

- *llim:* Optional, real. Sets the lower limit of [x|y|z]range.
- *ulim:* Optional, real. Sets the upper limit of [x|y|z]range.

Example:

```
call xrange(0.0,5.0)
call yrange(llim = -10.0) ! Set the lower limit only.
call zrange(ulim = 8.0)   ! Only for 3D plot. Set the upper limit only.
```

7 Animation

7.1 Animated gif

Recent versions of gnuplot supports gif animation. Relevant parts of the code (to save as animated gif) is given below.

```
! Start the plot
call openplot()
call figure()

! Any gnuplot commands can be passed using the gpcmd command
call gpcmd("set zeroaxis")
call gpcmd("set autoscale fix")
call key(.true., "box opaque bottom left")

do j = 1, an

    ! Data generation
    y1 = func1(x,a(j))
    y2 = func2(x,a(j))

    write(*,*) "j = ", j, ", a = ", a(j)

    call gpcmd('set style textbox opaque')
    write(label_text,'(a,i3,a,f4.2,$)') "j = ",j,"\na = ",a(j)
    call label(1,label_text,0.85,0.82,options = 'boxed')

    ! Special subroutines have been provided to set the labels &
    title.
    call xlabel('x')
    call ylabel('sin(x-a), sin(x+a)')
    call title('My hold on graph plots')

    ! To plot multiple plots on the same graph
    call hold(.true.)
    call plot2d(x,y1, options = "with lines", key = 'sin(x-a)')
    call plot2d(x,y2, options = "with lines dashtype 2", &
                key = 'sin(x+a)')
    ! To switch off plotting multiple plots on the same graph
    call hold(.false.)

    ! Slow the program, so that the plot window can be updated.
    call slumber(0.2)

```

```
39 end do
40
41 ! Save the plot as animated gif
42 call saveas("holdon2.gif", options = 'animate delay 0.2 loop 0')
43
44 ! Clean up
45 call closeplot()
```

7.2 Video file

Gnuplot cannot save figures as mp4 or other video files. So current workaround is to save each frame as png file and use external software like imagemagick to create the video file from the resultant png files. External software can also be used to create avi/mp4 files. Relevant code to create the pngfiles is given below.

Example:

```
character(:), allocatable :: dirname, pngfile

call openplot()

! Remove and then re-create any previous directory
! where the png files will be stored.
dirname = "animate"
call rmdir(dirname)
call mkdir(dirname)

do j = 1, gn

    call figure(1)
    call xrange(0.0,4.0)
    call yrange(-2.0,7.0)

    print*, "j =",j

    ! Get values of y array
    y = func(x,g(j),a)

    label_text = "a = " // real2str(a,'f5.2') // ",\n" // &
                 "g = " // real2str(g(j),'f5.2')
    call label(1,label_text,0.15,0.8)

    call plot2d(x,y, options = "with lines", key = 'f(x)')

    ! Generate the png filename
    pngfile = trim(dirname) // "/" // "frame_" // &
                int2str(j,'i0.6') // '.png'
    call saveas(trim(pngfile))

    ! Slow down so that gnuplot can plot
    call slumber(0.2)
```

```
end do

! Clean up
call closeplot()
```

Now convert the png files to video files using mencoder (part of mplayer) and ffmpeg.

```
# In one line
mencoder mf://animate/*.png -mf fps=5:type=png -ovc lavc \
    -lavcopts vcodec=mpeg4:mbd=2:trell -oac copy -o animate.avi

# Create mp4 file
ffmpeg -i animate.avi animate.mp4
```

8 Bugs

The following bugs exist.

- **qt slow font initialization.**
  The `qt` terminal is at times slow in font initialization.

- **wxt terminal crash**
  The `wxt` terminal also crashes at times with the following message.

  ```
  The program 'gnuplot' received an X Window System error.
  This probably reflects a bug in the program.
  The error was 'BadMatch (invalid parameter attributes)'.
    (Details: serial 1005 error_code 8 request_code 130 minor_code 3)
    (Note to programmers: normally, X errors are reported
       asynchronously;
     that is, you will receive the error a while after causing it.
     To debug your program, run it with the --sync command line
     option to change this behavior. You can then get a meaningful
     backtrace from your debugger if you break on the gdk_x_error()
     function.)
  Gnuplot exiting abnormally. Trying to execute exit handlers anyway.
  Segmentation fault (core dumped)
  ```

- **pdf output doesn't match 'qt' terminal output**
  The default pdf output is via `pdfcairo` which uses the cairo graphics libraries (same as the `wxt` terminal.) Hence the output of `qt` terminal doesn't match the pdf output.

  However if one uses the following command:

  ```
  ! Requires 'epstopdf' to be in the path
  call saveas('filename.pdf', driver = 'ps')
  ```

  then the pdf output is closer to the `qt` terminal (though not exact). Unfortunately there is no `pdfqt` terminal [4].

Note that all the above bugs are gnuplot related.

References

[1] Histogram in gnuplot. https://stackoverflow.com/questions/2471884/histogram-using-gnuplot.

[2] Histogram appearance and behaviour. https://in.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram-properties.html.

[3] Invalid points in data that is made monotonic. https://sourceforge.net/p/gnuplot/bugs/1274/.

[4] Qt terminal: Render to a file. https://sourceforge.net/p/gnuplot/patches/665/.