Gnuplot/C

Reference Manual

Version 3.10

15 January, 2014

GNUPLOT/C INTRODUCTION 3

GNUPLOT/C INSTALLATION 4

Rebuilding the Library 4

Modifying Gnuplot/C - Debug And Development 4

Gnuplot Usability Suggestions 5

GNUPLOT/C FUNCTION DESCRIPTIONS 6

gpc_init_2d 6
gpc_plot_2d 7
gpc_init_xy 8
gpc_plot_xy 9
gpc_init_pz 10
gpc_plot_pz 11
gpc_init_spectrogram 12
gpc_plot_spectrogram 13
gpc_init_image 14
gpc_plot_image 15
gpc_close 16

COPYRIGHT 17

LICENSE 17

Gnuplot/C Introduction

Gnuplot/C is an open source C/C++ interface library for the Gnuplot application : http://www.gnuplot.info. Gnupolot/C has been developed and tested under UNIX (Linux) and Windows. It is available from http://www.numerix-dsp.com/files or .

The API is based on the original Numerix Host Library (NHL) that was written in the early 1990s for Microsoft MS/DOS using the Microsoft and Borland C compilers. The API has been updated to change the underlying API to more closely match that of Gnuplot for example the 2D graph types point, line and stem are now “point”, “line” and “impulse” respectively. The original NHL colour #defines are now replaced by the Gnuplot colors which can be found by performing the following command in Gnuplot :

```
gnuplot> show colornames
```

Note : please accept our apologies for mixing the spellings of the words colour and color in this library. It is for purely historical reasons that the original library and documentation used the spelling colour.

Gnuplot/C supports multiple plots and multiple functions (datasets) within a plot.

The maximum number of graphs supported on a plot is 50 but this can be changed by modifying the `#define MAX_NUM_GRAPHS` in `gpcPlot.h`. The maximum number of plots is unlimited.

The strategy used for managing Gnuplot is to open a separate pipe to independent Gnuplot instances, for each plot required. Plots containing multiple separate graphs use intermediate files for saving the data for each graph. This has the side effect that it is slightly slower than regular plotting so Gnuplot/C supports two modes : Multiplot and fastplot modes are selected using the `GPC_MULTIPLOT` and `GPC_FASTPLOT` options.

Unlike NHL there is no limit to the maximum number of points in a dataset.

The original NHL graph types of `line`, `stem` and `point` are now replaced by the Gnuplot versions `“lines”` `“impulses”` and `“points”`. It is now possible to use any of the additional Gnuplot plot styles such as `“linespoints”` and `“steps”`. In addition it is also possible to include further Gnuplot style controls for example to specify circular points of size 1.5 use the following function parameter :

```
    "points pt 7 ps 1.5",
```

Gnuplot/C Installation

1/ Download and install Gnuplot from http://www.gnuplot.info.
2/ Ensure that the Gnuplot binary folder is registered in the PATH environment variable so that you can call the Gnuplot executable from any folder.
3/ Extract Gnuplot/C into a folder.
4/ Ensure that the Gnuplot/C folder is registered in the INCLUDE and LIB environment variables so that your compiler can locate the header and library files.

Rebuilding the Library

This library has been developed and tested using Microsoft 64 bit Visual C/C++ Express and GCC under Ubuntu v13.

To rebuild the library under Windows you can use the following batch files :

mbuildlib_64.bat	- Release mode
mbuildlib_64d.bat	- Debug mode, enables Gnuplot debug output

To rebuild the library under Linux you can use the following shell script files :

makefile.lx	- Release mode

The functions are little more than parsers that output text values via pipes so this library can be used under any operating system to which Gnuplot is ported.

IMPORTANT
AFTER INSTALLATION PLEASE ENSURE THAT THE LIBRARY AND INCLUDE FILE DIRECTORIES ARE INCLUDED IN THE COMPILER; LIBRARY AND INCLUDE PATHS.

Modifying Gnuplot/C - Debug And Development

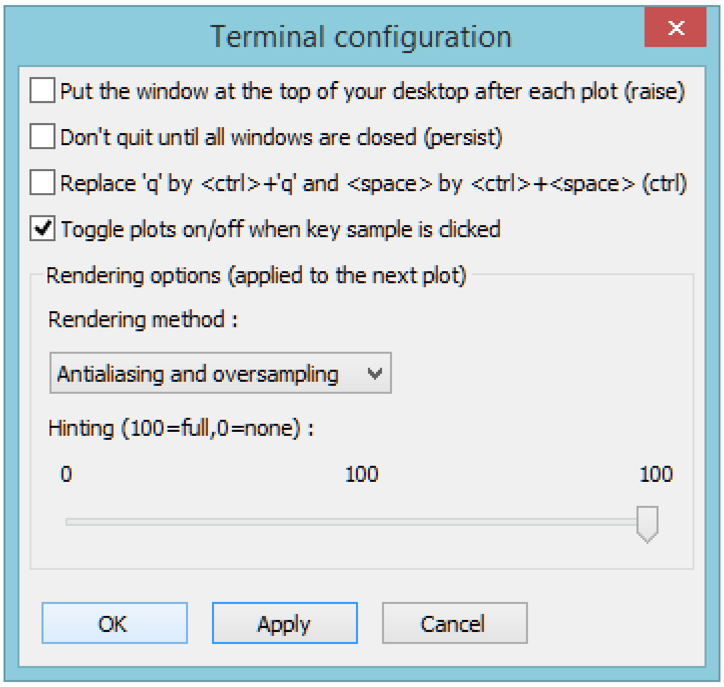
By default the library pipes the text output from Gnuplot to `null` (`nul` in Windows). This improves plotting performance because Gnuplot doesn't then echo the commands received, via the pipe, to the screen. If you wish to modify this library and debug your changes then a really useful tip is to use Gnuplot without output redirection so that the commands can be viewed in Gnuplot.

The `#define GPC_DEBUG` in `gpcPlot.h` can be set to `'1'` to enable command viewing or this can be defined on the compiler command line by using the following compiler option :

```
-D "GPC_DEBUG=1"
```

Gnuplot Usability Suggestions

By default Gnuplot brings the plot window to the front, which takes control away from the application generating the plot. In order to stop Gnuplot from doing this open the Configuration Dialog from any Gnuplot plot window and uncheck the tick box entitled : “Put the window at the top of your desktop after each plot (raise)” :


Terminal configuration

[ ] Put the window at the top of your desktop after each plot (raise)
[ ] Don't quit until all windows are closed (persist)
[ ] Replace 'q' by <ctrl>+'q' and <space> by <ctrl>+<space> (ctrl)
[✓] Toggle plots on/off when key sample is clicked

Rendering options (applied to the next plot)
Rendering method : Antialiasing and oversampling
Hinting (100=full,0=none) : 0 100 100

Click OK to save this configuration.

Gnuplot/C Function Descriptions

gpc_init_2d

FUNCTION NAME

gpc_init_2d

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_2d (char *plotTitle,	Plot title
char *xLabel,	X axis label
char *yLabel,	Y axis label
double scalingMode,	Scaling mode
enum gpcPlotSignMode signMode,	Sign mode
enum gpcMultiFastMode multiFastMode,	Multiplot / fast plot mode
enum gpcKeyMode keyMode);	Legend / key mode

FUNCTION DESCRIPTION

Initialize the 2D plot function and returns a handle to a new plot.

NOTES ON USE

Scaling mode is either the maximum value on the Y axis or `GPC_AUTO_SCALE` which auto scales the Y axis.

signMode should be set to either `GPC_SIGNED`, `GPC_POSITIVE` or `GPC_NEGATIVE` depending on whether the plot should display signed (positive and negative) or only positive or only negative numbers.

multiFastMode should be set to either `GPC_MULTIPLOT` or `GPC_FASTPLOT` depending on which mode is required.

keyMode should be set to either `GPC_KEY_DISABLE` or `GPC_KEY_ENABLE` depending on whether or not the key/legend is required.

FUNCTION NAME

gpc_plot_2d

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int gpc_plot_2d (h_GPC_Plot *plotHandle,	Plot handle
double *pData,	Dataset pointer
int graphLength,	Dataset length
char *pDataName,	Dataset title
double xMin,	Minimum X value
double xMax,	Maximum X value
char *plotType,	Plot type
char *pColour,	Colour
int addMode);	Add / new mode

FUNCTION DESCRIPTION

Plots the dataset onto the 2D graph.

NOTES ON USE

plotHandle is the plot created with the init function.

plotType is one of the standard Gnuplot plot types e.g. `"lines"`, `"points"`, `"impulses"`, `"linespoints"`, `"steps"` etc.

pColour is a standard Gnuplot color string e.g. `“blue”`. Use gnuplot> show colornames to see available colours.

addMode should be set to either `GPC_NEW` or `GPC_ADD` depending on whether or not this is a new graph or the dataset should be added to an existing plot.

FUNCTION NAME

gpc_init_xy

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_xy (char *plotTitle,	Plot title
char *xLabel,	X axis label
char *yLabel,	Y axis label
double dimension,	Dimension - this is square
enum gpcKeyMode keyMode);	Legend / key mode

FUNCTION DESCRIPTION

Initialize the XY plot function and returns a handle to a new plot.

NOTES ON USE

Scaling mode is either the maximum value on the Y axis or `GPC_AUTO_SCALE` which auto scales the Y axis.

keyMode should be set to either `GPC_KEY_DISABLE` or `GPC_KEY_ENABLE` depending on whether or not the key/legend is required.

FUNCTION NAME

gpc_plot_xy

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

```
int gpc_plot_xy (h_GPC_Plot *plotHandle,          Plot handle
        ComplexRect_s *pData,                     Dataset pointer
        int graphLength,                          Dataset length
        char *pDataName,                          Dataset title
        char *plotType,                           Plot type
        char *pColour,                            Colour
        enum gpcNewAddGraphMode addMode);         Add / new mode
```

FUNCTION DESCRIPTION

Plots the dataset onto the XY graph.

NOTES ON USE

plotHandle is the plot created with the init function.

plotType is one of the standard Gnuplot plot types e.g. `"lines"`, `"points"`, `"impulses"`, `"linespoints"`, `"steps"` etc.

pColour is a standard Gnuplot color string e.g. `“blue”`. Use gnuplot> show colornames to see available colours.

addMode should be set to either `GPC_NEW` or `GPC_ADD` depending on whether or not this is a new graph or the dataset should be added to an existing plot.

The complex data type is defined as :

```
typedef struct              // Complex data type
{
  double  real;
  double  imag;
} ComplexRect_s;
```

FUNCTION NAME

gpc_init_pz

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_pz (char *plotTitle,	Plot title
double dimension,	Dimension - this is square
enum gpcKeyMode keyMode);	Legend / key mode

FUNCTION DESCRIPTION

Initialize the pole-zero plot function and returns a handle to a new plot.

NOTES ON USE

keyMode should be set to either `GPC_KEY_DISABLE` or `GPC_KEY_ENABLE` depending on whether or not the key/legend is required.

FUNCTION NAME

gpc_plot_pz

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

```
int gpc_plot_pz (h_GPC_Plot *plotHandle,            Plot handle
        ComplexRect_s *pData,                       Dataset pointer
        int graphLength,                            Dataset length
        char *pDataName,                            Dataset title
        enum gpcPoleZeroMode poleZeroMode,          Pole-zero mode
        enum gpcNewAddGraphMode addMode);           Add / new mode
```

FUNCTION DESCRIPTION

Plots the dataset onto the pole-zero graph.

NOTES ON USE

plotHandle is the plot created with the init function.

poleZeroMode should be set to either is one of the standard Gnuplot plot types e.g. "GPC_COMPLEX_POLE", "GPC_CONJUGATE_POLE", "GPC_COMPLEX_ZERO" or "GPC_CONJUGATE_ZERO " depending on what the data values represent.

addMode should be set to either GPC_NEW or GPC_ADD depending on whether or not this is a new graph or the dataset should be added to an existing plot.

The complex data type is defined as :

```
typedef struct              // Complex data type
{
  double  real;
  double  imag;
} ComplexRect_s;
```

FUNCTION NAME

gpc_init_spectrogram

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot * gpc_init_spectrogram (char *plotTitle,	Plot title
char *xLabel,	X axis label
char *yLabel,	Y axis label
int xAxisLength,	X axis length
int yAxisLength,	Y axis length
double yMin,	Minimum Y value
double yMax,	Maximum Y value
double zMin,	Minimum Z value
double zMax,	Maximum Z value
char *colourPalette,	Colour colourPalette
enum gpcKeyMode keyMode);	Legend / key mode

FUNCTION DESCRIPTION

Initialize the spectrogram plot function and returns a handle to a new plot.

NOTES ON USE

colourPalette can be set to either of the standard palettes L `GPC_MONOCHROME` or `GPC_COLOUR` or you can supply your own palette in the following Gnuplot format :

```
"set palette defined (0 'black', 1 'blue', 2 'red', 3
'yellow', 4 'white')"
```

keyMode should be set to either `GPC_KEY_DISABLE` or `GPC_KEY_ENABLE` depending on whether or not the key/legend is required.

FUNCTION NAME

gpc_plot_spectrogram

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int gpc_plot_spectrogram (h_GPC_Plot *plotHandle,	Plot handle
double *pData,	Dataset pointer
char *pDataName,	Dataset title
double xMin,	Minimum X value
double xMax);	Maximum X value

FUNCTION DESCRIPTION

Plots the dataset onto the spectrogram.

NOTES ON USE

Spectrogram plots plot by column, rather than row as per a standard 2D image.

plotHandle is the plot created with the init function.

This function can support spectrogram datasets that do not fill up the complete X axis range specified in gpc_init_spectrogram but passing the virtual pointer "`GPC_END_PLOT`" to the function as the data array pointer. For example :

```
gpc_plot_spectrogram (hSpectrogram,  // Graph handle
                      GPC_END_PLOT,  // Dataset pointer
                     "Plot Title",   // Dataset title
                      X_MIN,         // Minimum X value
                      X_MAX);        // Maximum X value
```

FUNCTION NAME

gpc_init_image

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_image (char *plotTitle,	Plot title
int xAxisLength,	X axis length
int yAxisLength,	X axis length
unsigned int zMin,	Minimum Z value
unsigned int zMax,	Maximum Z value
char *colourPalette,	Colour colourPalette
enum gpcKeyMode keyMode);	Legend / key mode

FUNCTION DESCRIPTION

Initialize the image plot function and returns a handle to a new plot.

NOTES ON USE

colourPalette can be set to either of the standard palettes L `GPC_MONOCHROME` or `GPC_COLOUR` or you can supply your own palette in the following Gnuplot format :

If zMin and zMax are both set to “`GPC_IMG_AUTO_SCALE`” then the image will autoscale the z axis values.

```
"set palette defined (0 'black', 1 'blue', 2 'red', 3
'yellow', 4 'white')"
```

keyMode should be set to either `GPC_KEY_DISABLE` or `GPC_KEY_ENABLE` depending on whether or not the key/legend is required.

FUNCTION NAME

gpc_plot_image

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

```
int gpc_plot_image (h_GPC_Plot *plotHandle,        Plot handle
        unsigned char *pData,                      Dataset pointer
        char *pDataName);                          Dataset title
```

FUNCTION DESCRIPTION

Plots the dataset onto the image graph.

NOTES ON USE

plotHandle is the plot created with the init function.

FUNCTION NAME

gpc_close

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void gpc_close (h_GPC_Plot *);	Plot handle

FUNCTION DESCRIPTION

Plots closes the plot, frees all associated memory and closes the Gnuplot window.

NOTES ON USE

plotHandle is the plot created with the init function.

Copyright

License