Gnuplot/C

Reference Manual

Version 3.70

11 January, 2026

Table of Contents

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 https://sourceforge.net/projects/gnuplotc/.

The API is based on the original Numerix Host Library 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.

Gnuplot/C uses 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 English spelling colour but Gnuplot uses color.

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

The maximum number of graphs supported on a plot is 100 but this can be changed by modifying the `#define MAX_NUM_GRAPHS` in `gpcPlot.h`. The maximum number of plots is unlimited. Unlike NHL there is no limit to the maximum number of points in a dataset.

The general strategy used for managing Gnuplot is to open a separate pipe to independent Gnuplot instances, for each plot required.

Options for creating new graphs and then subsequently adding graphs to a plot are to use the following two modes :

`GPC_NEW` - Used to indicate the first graph in a plot
`GPC_ADD` - Used to indicate subsequent graphs added to a plot

If a graph contains existing plots then using `GPC_NEW` results in the existing graphs to be removed and replaced with the new graph.

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

Ensure Gnuplot is installed on your computer (see below for Gnuplot installation instructions).
Extract Gnuplot/C into a folder.
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.
Use make (or gmake) to build the library.
Move to the examples directory, build one and run it.

Gnuplot Installation

Windows

Download and install Gnuplot from http://www.gnuplot.info/download.html.
Ensure that the Gnuplot binary folder is registered in the PATH environment variable so that you can call the Gnuplot executable from any folder.

MacOS

Source : http://stackoverflow.com/questions/13001847/wxt-terminal-for-gnuplot-on-mac-os-x

Note : After this I had to reboot MacOS to enable Gnuplot to recognize the new terminal types.

Note : Some OSX installations may struggle with this installation procedure. Please refer to this blog entry for further details :
http://blog.numerix-dsp.com/2017/01/gnuplotc-on-mac.html

Linux

Under Linux you need to install both Gnuplot and Gnuplot-X11 :

```
sudo apt-get install gnuplot
sudo apt-get install gnuplot-x11
```

Rebuilding the Library

This library has been developed and tested using GCC under Ubunty and Microsoft 64 bit Visual Studio Community Edition.

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

      makefile.lx   - Release mode

To rebuild the library under Windows, using the Microsoft compiler, you can use the following batch files :

      mbuildlib.bat - Release mode
      mbuildlibd.bat - Debug mode, enables Gnuplot debug output

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.

<div align="center">IMPORTANT</div>

AFTER INSTALLATION PLEASE ENSURE THAT THE LIBRARY SRC DIRECTORY IS INCLUDED IN THE COMPILER; LIBRARY AND INCLUDE PATHS.

Building The Example Programs

The examples are located in the gnuplot_c/examples folder. To rebuild and execute the examples under Windows, using the Microsoft compiler, you can use the following batch files :

      mbr.bat   - Release mode
      mbrd.bat   - Debug mode, enables Gnuplot debug output

To rebuild the LinesAndPoints example under Linux, using the GCC compiler, you can use the following command :

      gcc LinesAndPoints.c -I ../src -L ../src -l gnuplot_c -o LinesAndPoints
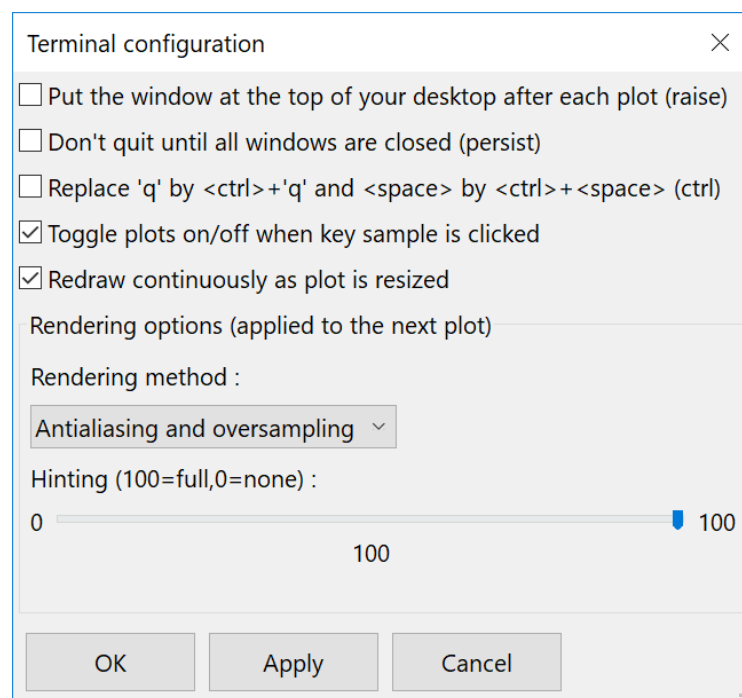
Modifying Gnuplot/C - Debug And Development

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)" :

Click OK to save this configuration.

Gnuplot Terminal Types

When plotting data using Gnuplot/C you may find that Gnuplot reports that the chosen terminal is an unknown terminal type. Gnuplot/C uses the following terminal types :

Runtime/Compiler Environment	Gnuplot Terminal Type
Windows (Microsoft Visual C/C++)	wxt
Linux/OSX	wxt

If you wish to use a different combination then the Gnuplot terminal type can be modified by changing the GPC_TERM #define in the compiler specific section at the top of gnuplot_c.c file.

It is also possible to install the wxWidgets v2.8 library and dependencies using the following commands.

Ubuntu Linux :
```
sudo apt-get install libcairo2-dev libpango1.0-dev
sudo add-apt-repository ppa:nilarimogard/webupd8
sudo apt-get update
sudo apt-get install python-wxgtk2.8
```

sudo add-apt-repository ppa:nilarimogard/webupd8
sudo apt-get update
sudo apt-get install python-wxgtk2.8

Under OSX you should uninstall your current version of Gnuplot and then install the +wxt variant.
`port variants gnuplot` to list available variants.
`sudo port install gnuplot +wxwidgets`

Plotting Colours

When plotting a number of lines or points on a graph it is convenient to have easy to handle colours so Gnuplot/C includes the array gpcPlotColours[] with 10 distinct colours. These can be accessed in a modulo fashion using (index%GPC_NUM_PLOT_COLOURS).

Gnuplot/C Function Descriptions

<div align="right">**gpc_init_2d**</div>

FUNCTION NAME
    gpc_init_2d

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

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

FUNCTION DESCRIPTION

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

NOTES ON USE

scalingMode 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.

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_init_2d_logscalex

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_2d_logscalex (const char *plotTitle,   Plot title
        const char *xLabel,   X axis label
        const char *yLabel,   Y axis label
        const double scalingMode,   Scaling mode
        const enum gpcPlotSignMode signMode,   Sign mode
        const enum gpcKeyMode keyMode);   Legend / key mode

FUNCTION DESCRIPTION

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

NOTES ON USE

scalingMode 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.

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
        const double *pData,   Dataset pointer
        const int graphLength,   Dataset length
        const char *pDataName,   Dataset title
        const double xMin,   Minimum X value
        const double xMax,   Maximum X value
        const char *plotType,   Plot type
        const char *pColour,   Colour
        const enum gpcNewAddGraphMode 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_2d_dual_plot

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_2d_dual_plot (const char *plotTitle,   Plot title
        const enum gpcKeyMode keyMode);   Legend / key mode

FUNCTION DESCRIPTION

Initialize the 2D dual 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_2d_dual_plot

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int gpc_plot_2d_dual_plot (h_GPC_Plot *plotHandle,	Plot handle
const char *xLabel,	X axis label
const double xMin,	Minimum X value
const double xMax,	Maximum X value
const double *pData1,	Dataset #1 pointer
const char *pDataName1,	Dataset #1 title
const char *plotType1,	Dataset #1 Plot type -
"lines", "points", "impulses", "linespoints"	
const char *pColour1,	Dataset #1 Colour - Use
gnuplot> show colornames to see available colours	
const char *yLabel1,	Dataset #1 Y axis label
const double scalingMode1,	Dataset #1 Scaling mode
const enum gpcPlotSignMode signMode1,	Dataset #1 Sign mode -
signed, positive, negative	
const double *pData2,	Dataset #2 pointer
const char *pDataName2,	Dataset #2 title
const char *plotType2,	Dataset #2 Plot type -
"lines", "points", "impulses", "linespoints"	
const char *pColour2,	Dataset #2 Colour - Use
gnuplot> show colornames to see available colours	
const char *yLabel2,	Dataset #2 Y axis label
const double scalingMode2,	Dataset #2 Scaling mode
const enum gpcPlotSignMode signMode2,	Dataset #2 Sign mode -
signed, positive, negative	
const int graphLength);	Dataset lengths

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.

scalingMode 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.

FUNCTION NAME
    gpc_init_3d

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_3d (const char *plotTitle,   Plot title
        const enum gpcKeyMode keyMode);   Legend / key mode

FUNCTION DESCRIPTION

Initialize the 3D 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_3d

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

```
int gpc_plot_3d (h_GPC_Plot *plotHandle,          Plot handle
        const double *pX,                         X dataset pointer
        const double *pY,                         Y dataset pointer
        const double *pZ,                         Z dataset pointer
        const int graphLength,                    Dataset length
        const char *pDataName,                    Dataset title
        const char *plotType,                     Plot type
        const char *pColour,                      Colour
        const enum gpcNewAddGraphMode addMode);   Add / new mode
```


FUNCTION DESCRIPTION

Plots the 3D (x, y and z) datasets onto the 3D graph.

NOTES ON USE

The x, y and z axes are all scaled equally, to match the largest value in the first (new) graph.

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

FUNCTION DESCRIPTION

Initialize the XY 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_xy

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

```
int gpc_plot_xy (h_GPC_Plot *plotHandle,          Plot handle
      const ComplexRect_s *pData,          Dataset pointer
      const int graphLength,          Dataset length
      const char *pDataName,          Dataset title
      const char *plotType,          Plot type
      const char *pColour,          Colour
      const 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 (const char *plotTitle,   Plot title
      const double dimension,   Dimension - this is square
      const 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
const ComplexRect_s *pData,	Dataset pointer
const int graphLength,	Dataset length
const char *pDataName,	Dataset title
const enum gpcPoleZeroMode poleZeroMode,	Pole-zero mode
const 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 (const char *plotTitle,	Plot title
const char *xLabel,	X axis label
const char *yLabel,	Y axis label
const int xAxisLength,	X axis length
const int yAxisLength,	Y axis length
const double yMin,	Minimum Y value
const double yMax,	Maximum Y value
const double zMin,	Minimum Z value
const double zMax,	Maximum Z value
const char *colourPalette,	Colour colourPalette
const 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
          const double *pData,   Dataset pointer
          const char *pDataName,   Dataset title
          const double xMin,   Minimum X value
          const 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
```

gpc_init_image

FUNCTION NAME
        gpc_init_image

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_image (char *plotTitle,   Plot title
        const int xAxisLength,   X axis length
        const int yAxisLength,   X axis length
        const unsigned int zMin,   Minimum Z value
        const unsigned int zMax,   Maximum Z value
        const char *colourPalette,   Colour colourPalette
        const 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 (const h_GPC_Plot *plotHandle,   Plot handle
          const unsigned char *pData,   Dataset pointer
          const 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.

This function can also be used to display a heatmap.

FUNCTION NAME
      gpc_init_polar

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_init_polar (const char *plotTitle,   Plot title
      const double gMin,   Minimum gain value
      const double gMax,   Maximum gain value
      const enum gpcKeyMode keyMode);   Legend / key mode

FUNCTION DESCRIPTION

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

NOTES ON USE

FUNCTION NAME
    gpc_plot_polar

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int gpc_plot_polar (h_GPC_Plot *plotHandle,   Plot handle
    const double *pAngles,   Angles dataset pointer
    const double *pGains,   Gains dataset pointer
    const int graphLength,   Dataset length
    const char *pDataName,   Dataset title
    const char *plotType,   Plot type
    const char *pColour,   Colour
    const enum gpcNewAddGraphMode addMode);   Add / new mode

FUNCTION DESCRIPTION

Plots the dataset onto the polar plot.

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.

FUNCTION NAME

    gpc_plot_confusion_matrix

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_plot_confusion_matrix (const double *, Pointer to source data
    const double,   Maximum value to scale matrix colourization
    const int);   Number of categories

FUNCTION DESCRIPTION

Creates and plots confusion matrix typically used in neural networks.

NOTES ON USE

The maximum scale value is typically either the total number of samples classified or the maximum value of all the categories.

FUNCTION NAME
    gpc_plot_confusion_matrix_percentage

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

h_GPC_Plot *gpc_plot_confusion_matrix_percentage (const double *,   Pointer   to
source data
        const int);   Number of categories

FUNCTION DESCRIPTION

Creates and plots confusion matrix typically used in neural networks, with the results
plotted as percentages.

NOTES ON USE