gnuplot-cpp

0.9

Generated by Doxygen 1.5.8

Wed Sep 16 12:55:07 2009

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Gnuplot Class Reference

```
#include <gnuplot_i.hpp>
```

Public Member Functions

- **Gnuplot** (const std::string &style="points")

  set a style during construction

- **Gnuplot** (const std::vector< double > &x, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")

  plot a single std::vector at one go

- **Gnuplot** (const std::vector< double > &x, const std::vector< double > &y, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y")

  plot pairs std::vector at one go

- **Gnuplot** (const std::vector< double > &x, const std::vector< double > &y, const std::vector< double > &z, const std::string &title="", const std::string &style="points", const std::string &labelx="x", const std::string &labely="y", const std::string &labelz="z")

  plot triples std::vector at one go

- ~**Gnuplot** ()

  destructor: needed to delete temporary files

- **Gnuplot** & **cmd** (const std::string &cmdstr)

  send a command to gnuplot

- **Gnuplot** & **operator**<< (const std::string &cmdstr)

  *Sends a command to an active gnuplot session, identical to **cmd()** (p. 10) send a command to gnuplot using the << operator.*

- **Gnuplot** & **showonscreen** ()

sets terminal type to terminal_std

- **Gnuplot** & **savetops** (const std::string &filename="gnuplot_output")

  saves a gnuplot session to a postscript file, filename without extension

- **Gnuplot** & **set_style** (const std::string &stylestr="points")
- **Gnuplot** & **set_smooth** (const std::string &stylestr="csplines")
- **Gnuplot** & **unset_smooth** ()

  unset smooth attention: smooth is not set by default

- **Gnuplot** & **set_pointsize** (const double pointsize=1.0)

  scales the size of the points used in plots

- **Gnuplot** & **set_grid** ()

  turns grid on/off

- **Gnuplot** & **unset_grid** ()

  grid is not set by default

- **Gnuplot** & **set_multiplot** ()
- **Gnuplot** & **unset_multiplot** ()
- **Gnuplot** & **set_samples** (const int samples=100)

  set sampling rate of functions, or for interpolating data

- **Gnuplot** & **set_isosamples** (const int isolines=10)

  set isoline density (grid) for plotting functions as surfaces (for 3d plots)

- **Gnuplot** & **set_hidden3d** ()
- **Gnuplot** & **unset_hidden3d** ()
- **Gnuplot** & **set_contour** (const std::string &position="base")
- **Gnuplot** & **unset_contour** ()
- **Gnuplot** & **set_surface** ()
- **Gnuplot** & **unset_surface** ()
- **Gnuplot** & **set_legend** (const std::string &position="default")
- **Gnuplot** & **unset_legend** ()

  Switches legend off attention:legend is set by default.

- **Gnuplot** & **set_title** (const std::string &title="")

  sets and clears the title of a gnuplot session

- **Gnuplot** & **unset_title** ()

  Clears the title of a gnuplot session The title is not set by default.

- **Gnuplot** & **set_ylabel** (const std::string &label="x")

  set x axis label

- **Gnuplot** & **set_xlabel** (const std::string &label="y")

  set y axis label

- **Gnuplot** & **set_zlabel** (const std::string &label="z")

set z axis label

- **Gnuplot** & **set_xrange** (const double iFrom, const double iTo)

  set axis - ranges

- **Gnuplot** & **set_yrange** (const double iFrom, const double iTo)

  set y-axis - ranges

- **Gnuplot** & **set_zrange** (const double iFrom, const double iTo)

  set z-axis - ranges

- **Gnuplot** & **set_xautoscale** ()
- **Gnuplot** & **set_yautoscale** ()
- **Gnuplot** & **set_zautoscale** ()
- **Gnuplot** & **set_xlogscale** (const double base=10)

  turns on/off log scaling for the specified xaxis (logscale is not set by default)

- **Gnuplot** & **set_ylogscale** (const double base=10)

  turns on/off log scaling for the specified yaxis (logscale is not set by default)

- **Gnuplot** & **set_zlogscale** (const double base=10)

  turns on/off log scaling for the specified zaxis (logscale is not set by default)

- **Gnuplot** & **unset_xlogscale** ()
- **Gnuplot** & **unset_ylogscale** ()
- **Gnuplot** & **unset_zlogscale** ()
- **Gnuplot** & **set_cbrange** (const double iFrom, const double iTo)

  set palette range (autoscale by default)

- **Gnuplot** & **plotfile_x** (const std::string &filename, const unsigned int column=1, const std::string &title="")
- template<typename X >
  Gnuplot & **plot_x** (const X &x, const std::string &title="")

  from std::vector

- **Gnuplot** & **plotfile_xy** (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const std::string &title="")
- template<typename X , typename Y >
  Gnuplot & **plot_xy** (const X &x, const Y &y, const std::string &title="")

  from data

- **Gnuplot** & **plotfile_xy_err** (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const unsigned int column_dy=3, const std::string &title="")
- template<typename X , typename Y , typename E >
  Gnuplot & **plot_xy_err** (const X &x, const Y &y, const E &dy, const std::string &title="")

  from data

- **Gnuplot** & **plotfile_xyz** (const std::string &filename, const unsigned int column_x=1, const unsigned int column_y=2, const unsigned int column_z=3, const std::string &title="")
- template<typename X , typename Y , typename Z >
  Gnuplot & **plot_xyz** (const X &x, const Y &y, const Z &z, const std::string &title="")

from std::vector

- **Gnuplot** & **plot_slope** (const double a, const double b, const std::string &title="")

  plot an equation of the form: y = ax + b, you supply a and b

- **Gnuplot** & **plot_equation** (const std::string &equation, const std::string &title="")
- **Gnuplot** & **plot_equation3d** (const std::string &equation, const std::string &title="")
- **Gnuplot** & **plot_image** (const unsigned char ∗ucPicBuf, const unsigned int iWidth, const unsigned int iHeight, const std::string &title="")

  plot image

- **Gnuplot** & **replot** (void)

  replot repeats the last plot or splot command. this can be useful for viewing a plot with different set options, or when generating the same plot for several devices (showonscreen, savetops)

- **Gnuplot** & **reset_plot** ()

  resets a gnuplot session (next plot will erase previous ones)

- **Gnuplot** & **reset_all** ()

  resets a gnuplot session and sets all variables to default

- void **remove_tmpfiles** ()

  deletes temporary files

- bool **is_valid** ()

  Is the gnuplot session valid ??

Static Public Member Functions

- static bool **set_GNUPlotPath** (const std::string &path)

  *optional function: set **Gnuplot** (p. 5) path manual attention: for windows: path with slash '/' not backslash '\'*

- static void **set_terminal_std** (const std::string &type)

3.1.1 Detailed Description

Definition at line 68 of file gnuplot_i.hpp.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Gnuplot::Gnuplot (const std::string & *style* = `"points"`) `[inline]`

set a style during construction

Definition at line 612 of file gnuplot_i.hpp.

References set_style().

```
                               :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)

{
    init();
    set_style(style);
}
```

3.1.2.2 Gnuplot::Gnuplot (const std::vector< double > & *x*, const std::string & *title* = "", const std::string & *style* = "points", const std::string & *labelx* = "x", const std::string & *labely* = "y") [inline]

plot a single std::vector at one go

Definition at line 624 of file gnuplot_i.hpp.

References plot_x(), set_style(), set_xlabel(), and set_ylabel().

```
                               :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)
{
    init();

    set_style(style);
    set_xlabel(labelx);
    set_ylabel(labely);

    plot_x(x,title);
}
```

3.1.2.3 Gnuplot::Gnuplot (const std::vector< double > & *x*, const std::vector< double > & *y*, const std::string & *title* = "", const std::string & *style* = "points", const std::string & *labelx* = "x", const std::string & *labely* = "y") [inline]

plot pairs std::vector at one go

Definition at line 645 of file gnuplot_i.hpp.

References plot_xy(), set_style(), set_xlabel(), and set_ylabel().

```
                               :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)
{
    init();

    set_style(style);
    set_xlabel(labelx);
    set_ylabel(labely);

    plot_xy(x,y,title);
}
```

3.1.2.4 Gnuplot::Gnuplot (const std::vector< double > & *x*, const std::vector< double > & *y*, const std::vector< double > & *z*, const std::string & *title* = "", const std::string & *style* = "points", const std::string & *labelx* = "x", const std::string & *labely* = "y", const std::string & *labelz* = "z") [inline]

plot triples std::vector at one go

Definition at line 667 of file gnuplot_i.hpp.

References plot_xyz(), set_style(), set_xlabel(), set_ylabel(), and set_zlabel().

```
                           :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)
{
    init();

    set_style(style);
    set_xlabel(labelx);
    set_ylabel(labely);
    set_zlabel(labelz);

    plot_xyz(x,y,z,title);
}
```

3.1.2.5 Gnuplot::∼Gnuplot ()

destructor: needed to delete temporary files

Definition at line 944 of file gnuplot_i.hpp.

```
{
//  remove_tmpfiles();

    // A stream opened by popen() should be closed by pclose()
#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__TOS_WIN__)
    if (_pclose(gnucmd) == -1)
#elif defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
    if (pclose(gnucmd) == -1)
#endif
        throw GnuplotException("Problem closing communication to gnuplot");
}
```

3.1.3 Member Function Documentation

3.1.3.1 Gnuplot & Gnuplot::cmd (const std::string & *cmdstr*)

send a command to gnuplot

Definition at line 1641 of file gnuplot_i.hpp.

Referenced by main(), operator<<(), plot_equation(), plot_equation3d(), plot_image(), plot_slope(), plotfile_x(), plotfile_xy(), plotfile_xy_err(), plotfile_xyz(), replot(), reset_all(), savetops(), set_cbrange(), set_contour(), set_grid(), set_hidden3d(), set_isosamples(), set_legend(), set_multiplot(), set_pointsize(), set_samples(), set_surface(), set_xautoscale(), set_xlabel(), set_xlogscale(), set_xrange(), set_yautoscale(), set_ylabel(), set_ylogscale(), set_yrange(), set_zautoscale(), set_zlabel(), set_zlogscale(), set_zrange(), showonscreen(), unset_contour(), unset_grid(), unset_hidden3d(), unset_legend(), unset_multiplot(), unset_surface(), unset_xlogscale(), unset_ylogscale(), and unset_zlogscale().

```
{
    if( !(valid) )
    {
        return *this;
    }


    // int fputs ( const char * str, FILE * stream );
    // writes the string str to the stream.
    // The function begins copying from the address specified (str) until it
    // reaches the terminating null character ('\0'). This final
    // null-character is not copied to the stream.
    fputs( (cmdstr+"\n").c_str(), gnucmd );

    // int fflush ( FILE * stream );
    // If the given stream was open for writing and the last i/o operation was
    // an output operation, any unwritten data in the output buffer is written
    // to the file.  If the argument is a null pointer, all open files are
    // flushed.  The stream remains open after this call.
    fflush(gnucmd);


    if( cmdstr.find("replot") != std::string::npos )
    {
        return *this;
    }
    else if( cmdstr.find("splot") != std::string::npos )
    {
        two_dim = false;
        nplots++;
    }
    else if( cmdstr.find("plot") != std::string::npos )
    {
        two_dim = true;
        nplots++;
    }

    return *this;
}
```

3.1.3.2 bool Gnuplot::is_valid () `[inline]`

Is the gnuplot session valid ??

Parameters:

—

Returns:

true if valid, false if not

Definition at line 582 of file gnuplot_i.hpp.

```
582 {return(valid);};
```

3.1.3.3 Gnuplot& Gnuplot::operator<< (const std::string & *cmdstr*) `[inline]`

Sends a command to an active gnuplot session, identical to **cmd()** (p. 10) send a command to gnuplot using the << operator.

Parameters:

cmdstr –> the command string

Returns:

<– a reference to the gnuplot object

Definition at line 220 of file gnuplot_i.hpp.

References cmd().

```
                                                        {
        cmd(cmdstr);
        return(*this);
    }
```

3.1.3.4 Gnuplot & Gnuplot::plot_equation (const std::string & *equation*, const std::string & *title* = "")

plot an equation supplied as a std::string y=f(x), write only the function f(x) not y= the independent variable has to be x binary operators: ** exponentiation, * multiply, / divide, + add, - substract, % modulo unary operators: - minus, ! factorial elementary functions: rand(x), abs(x), sgn(x), ceil(x), floor(x), int(x), imag(x), real(x), arg(x), sqrt(x), exp(x), log(x), log10(x), sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(y,x), sinh(x), cosh(x), tanh(x), asinh(x), acosh(x), atanh(x) special functions: erf(x), erfc(x), inverf(x), gamma(x), igamma(a,x), lgamma(x), ibeta(p,q,x), besj0(x), besj1(x), besy0(x), besy1(x), lambertw(x) statistical fuctions: norm(x), invnorm(x)

Definition at line 1345 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == true)
        cmdstr << "replot ";
    else
        cmdstr << "plot ";

    cmdstr << equation << " title \"";

    if (title == "")
        cmdstr << "f(x) = " << equation;
    else
        cmdstr << title;

    cmdstr << "\" with " << pstyle;

    //
    // Do the actual plot
    //
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.5 Gnuplot & Gnuplot::plot_equation3d (const std::string & *equation*, const std::string & *title* = "")

plot an equation supplied as a std::string z=f(x,y), write only the function f(x,y) not z= the independent variables have to be x and y

Definition at line 1378 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == false)
        cmdstr << "replot ";
    else
        cmdstr << "splot ";

    cmdstr << equation << " title \"";

    if (title == "")
        cmdstr << "f(x,y) = " << equation;
    else
        cmdstr << title;

    cmdstr << "\" with " << pstyle;

    //
    // Do the actual plot
    //
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.6 Gnuplot & Gnuplot::plot_image (const unsigned char ∗ *ucPicBuf*, const unsigned int *iWidth*, const unsigned int *iHeight*, const std::string & *title* = "")

plot image

∗ note that this function is not valid for versions of GNUPlot below 4.2

Definition at line 1586 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ofstream tmp;
    std::string name = create_tmpfile(tmp);
    if (name == "")
        return *this;

    //
    // write the data to file
    //
    int iIndex = 0;
    for(int iRow = 0; iRow < iHeight; iRow++)
    {
        for(int iColumn = 0; iColumn < iWidth; iColumn++)
        {
            tmp << iColumn << " " << iRow  << " "
                << static_cast<float>(ucPicBuf[iIndex++]) << std::endl;
        }
    }

    tmp.flush();
    tmp.close();


    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == true)
        cmdstr << "replot ";
    else
        cmdstr << "plot ";

    if (title == "")
        cmdstr << "\"" << name << "\" with image";
    else
        cmdstr << "\"" << name << "\" title \"" << title << "\" with image";

    //
    // Do the actual plot
    //
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.7 Gnuplot & Gnuplot::plot_slope (const double *a*, const double *b*, const std::string & *title* = "")

plot an equation of the form: y = ax + b, you supply a and b

Definition at line 1311 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == true)
        cmdstr << "replot ";
    else
        cmdstr << "plot ";

    cmdstr << a << " * x + " << b << " title \"";

    if (title == "")
        cmdstr << "f(x) = " << a << " * x + " << b;
    else
        cmdstr << title;

    cmdstr << "\" with " << pstyle;

    //
    // Do the actual plot
    //
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.8 template<typename X > Gnuplot & Gnuplot::plot_x (const X & *x*, const std::string & *title* = "") `[inline]`

from std::vector

Plots a 2d graph from a list of doubles: x.

Definition at line 693 of file gnuplot_i.hpp.

References plotfile_x().

Referenced by Gnuplot(), and main().

```
{
    if (x.size() == 0)
    {
        throw GnuplotException("std::vector too small");
        return *this;
    }

    std::ofstream tmp;
    std::string name = create_tmpfile(tmp);
    if (name == "")
        return *this;

    //
    // write the data to file
    //
    for (unsigned int i = 0; i < x.size(); i++)
        tmp << x[i] << std::endl;

    tmp.flush();
    tmp.close();


    plotfile_x(name, 1, title);

    return *this;
}
```

3.1.3.9 template<typename X , typename Y > Gnuplot & Gnuplot::plot_xy (const X & *x*, const Y & *y*, const std::string & *title* = "") `[inline]`

from data

Plots a 2d graph from a list of doubles: x y.

Definition at line 727 of file gnuplot_i.hpp.

References plotfile_xy().

Referenced by Gnuplot(), and main().

```
{
    if (x.size() == 0 || y.size() == 0)
    {
        throw GnuplotException("std::vectors too small");
        return *this;
    }

    if (x.size() != y.size())
    {
        throw GnuplotException("Length of the std::vectors differs");
        return *this;
    }


    std::ofstream tmp;
    std::string name = create_tmpfile(tmp);
    if (name == "")
        return *this;

    //
    // write the data to file
    //
    for (unsigned int i = 0; i < x.size(); i++)
        tmp << x[i] << " " << y[i] << std::endl;

    tmp.flush();
    tmp.close();


    plotfile_xy(name, 1, 2, title);

    return *this;
}
```

3.1.3.10 template<typename X , typename Y , typename E > Gnuplot & Gnuplot::plot_xy_err (const X & *x*, const Y & *y*, const E & *dy*, const std::string & *title* = "") `[inline]`

from data

---

plot x,y pairs with dy errorbars

Definition at line 767 of file gnuplot_i.hpp.

References plotfile_xy_err().

Referenced by main().

```
{
    if (x.size() == 0 || y.size() == 0 || dy.size() == 0)
    {
        throw GnuplotException("std::vectors too small");
        return *this;
    }

    if (x.size() != y.size() || y.size() != dy.size())
    {
        throw GnuplotException("Length of the std::vectors differs");
        return *this;
    }


    std::ofstream tmp;
    std::string name = create_tmpfile(tmp);
    if (name == "")
        return *this;

    //
    // write the data to file
    //
    for (unsigned int i = 0; i < x.size(); i++)
        tmp << x[i] << " " << y[i] << " " << dy[i] << std::endl;

    tmp.flush();
    tmp.close();


    // Do the actual plot
    plotfile_xy_err(name, 1, 2, 3, title);

    return *this;
}
```

3.1.3.11 template<typename X , typename Y , typename Z > Gnuplot & Gnuplot::plot_xyz (const X & *x*, const Y & *y*, const Z & *z*, const std::string & *title* = "") [inline]

from std::vector

Definition at line 812 of file gnuplot_i.hpp.

References plotfile_xyz().

Referenced by Gnuplot(), and main().

```
{
    if (x.size() == 0 || y.size() == 0 || z.size() == 0)
    {
        throw GnuplotException("std::vectors too small");
        return *this;
    }

    if (x.size() != y.size() || x.size() != z.size())
    {
        throw GnuplotException("Length of the std::vectors differs");
        return *this;
    }


    std::ofstream tmp;
    std::string name = create_tmpfile(tmp);
    if (name == "")
        return *this;

    //
    // write the data to file
    //
    for (unsigned int i = 0; i < x.size(); i++)
        tmp << x[i] << " " << y[i] << " " << z[i] <<std::endl;

    tmp.flush();
    tmp.close();


    plotfile_xyz(name, 1, 2, 3, title);

    return *this;
}
```

3.1.3.12 Gnuplot & Gnuplot::plotfile_x (const std::string & *filename*, const unsigned int *column* = 1, const std::string & *title* = "")

plot a single std::vector: x from file

Definition at line 1412 of file gnuplot_i.hpp.

References cmd().

Referenced by plot_x().

```
{
    //
    // check if file exists
    //
    file_available(filename);


    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == true)
        cmdstr << "replot ";
    else
        cmdstr << "plot ";

    cmdstr << "\"" << filename << "\" using " << column;

    if (title == "")
        cmdstr << " notitle ";
    else
        cmdstr << " title \"" << title << "\" ";

    if(smooth == "")
        cmdstr << "with " << pstyle;
    else
        cmdstr << "smooth " << smooth;

    //
    // Do the actual plot
    //
    cmd(cmdstr.str()); //nplots++; two_dim = true;  already in cmd();

    return *this;
}
```

3.1.3.13 Gnuplot & Gnuplot::plotfile_xy (const std::string & *filename*, const unsigned int *column_x* = `1`, const unsigned int *column_y* = `2`, const std::string & *title* = `""`)

plot x,y pairs: x y from file

Definition at line 1457 of file gnuplot_i.hpp.

References cmd().

Referenced by plot_xy().

```
{
    //
    // check if file exists
    //
    file_available(filename);


    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == true)
        cmdstr << "replot ";
    else
        cmdstr << "plot ";

    cmdstr << "\"" << filename << "\" using " << column_x << ":" << column_y;

    if (title == "")
        cmdstr << " notitle ";
    else
        cmdstr << " title \"" << title << "\" ";

    if(smooth == "")
        cmdstr << "with " << pstyle;
    else
        cmdstr << "smooth " << smooth;

    //
    // Do the actual plot
    //
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.14 Gnuplot & Gnuplot::plotfile_xy_err (const std::string & *filename*, const unsigned int *column_x* = `1`, const unsigned int *column_y* = `2`, const unsigned int *column_dy* = `3`, const std::string & *title* = `""`)

plot x,y pairs with dy errorbars: x y dy from file

Definition at line 1502 of file gnuplot_i.hpp.

References cmd().

Referenced by plot_xy_err().

```
{
    //
    // check if file exists
    //
    file_available(filename);

    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == true)
        cmdstr << "replot ";
    else
        cmdstr << "plot ";

    cmdstr << "\"" << filename << "\" using "
           << column_x << ":" << column_y << ":" << column_dy
           << " with errorbars ";

    if (title == "")
        cmdstr << " notitle ";
    else
        cmdstr << " title \"" << title << "\" ";

    //
    // Do the actual plot
    //
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.15 Gnuplot & Gnuplot::plotfile_xyz (const std::string & *filename*, const unsigned int *column_x* = 1, const unsigned int *column_y* = 2, const unsigned int *column_z* = 3, const std::string & *title* = "")

plot x,y,z triples: x y z from file

Definition at line 1544 of file gnuplot_i.hpp.

References cmd().

Referenced by plot_xyz().

```
{
    //
    // check if file exists
    //
    file_available(filename);

    std::ostringstream cmdstr;
    //
    // command to be sent to gnuplot
    //
    if (nplots > 0  &&  two_dim == false)
        cmdstr << "replot ";
    else
        cmdstr << "splot ";

    cmdstr << "\"" << filename << "\" using " << column_x << ":" << column_y
           << ":" << column_z;

    if (title == "")
        cmdstr << " notitle with " << pstyle;
    else
        cmdstr << " title \"" << title << "\" with " << pstyle;

    //
    // Do the actual plot
    //
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.16 void Gnuplot::remove_tmpfiles ()

deletes temporary files

Definition at line 1948 of file gnuplot_i.hpp.

```
                              {
    if ((tmpfile_list).size() > 0)
    {
        for (unsigned int i = 0; i < tmpfile_list.size(); i++)
            remove( tmpfile_list[i].c_str() );

        Gnuplot::tmpfile_num -= tmpfile_list.size();
    }
}
```

3.1.3.17 Gnuplot& Gnuplot::replot (void) `[inline]`

replot repeats the last plot or splot command. this can be useful for viewing a plot with different set options, or when generating the same plot for several devices (showonscreen, savetops)

Parameters:

—

Returns:

—

Definition at line 563 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
563 {if (nplots > 0) cmd("replot");return *this;};
```

3.1.3.18 Gnuplot & Gnuplot::reset_all ()

resets a gnuplot session and sets all variables to default

Definition at line 976 of file gnuplot_i.hpp.

References cmd(), and showonscreen().

Referenced by main().

```
977 {
978 //  remove_tmpfiles();
979
980     nplots = 0;
981     cmd("reset");
982     cmd("clear");
983     pstyle = "points";
984     smooth = "";
985     showonscreen();
986
987     return *this;
988 }
```

3.1.3.19 Gnuplot & Gnuplot::reset_plot ()

resets a gnuplot session (next plot will erase previous ones)

Definition at line 962 of file gnuplot_i.hpp.

Referenced by main().

```
963 {
964 //  remove_tmpfiles();
965
966     nplots = 0;
967
968     return *this;
969 }
```

3.1.3.20 Gnuplot & Gnuplot::savetops (const std::string & *filename* = "gnuplot_output")

saves a gnuplot session to a postscript file, filename without extension

Definition at line 1077 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    cmd("set terminal postscript color");

    std::ostringstream cmdstr;
    cmdstr << "set output \"" << filename << ".ps\"";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.21 Gnuplot & Gnuplot::set_cbrange (const double *iFrom*, const double *iTo*)

set palette range (autoscale by default)

Definition at line 1295 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set cbrange[" << iFrom << ":" << iTo << "]";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.22 Gnuplot & Gnuplot::set_contour (const std::string & *position* = "base")

enables/disables contour drawing for surfaces (for 3d plot) base, surface, both

Definition at line 1190 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    if (position.find("base")    == std::string::npos  &&
        position.find("surface") == std::string::npos  &&
        position.find("both")    == std::string::npos  )
    {
        cmd("set contour base");
    }
    else
    {
        cmd("set contour " + position);
    }

    return *this;
}
```

3.1.3.23 bool Gnuplot::set_GNUPlotPath (const std::string & *path*) [static]

optional function: set **Gnuplot** (p. 5) path manual attention: for windows: path with slash '/' not backslash '\'

Parameters:

path –> the gnuplot path

Returns:

true on success, false otherwise

Definition at line 856 of file gnuplot_i.hpp.

```
{

    std::string tmp = path + "/" + Gnuplot::m_sGNUPlotFileName;


#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__TOS_WIN__)
    if ( Gnuplot::file_exists(tmp,0) ) // check existence
#elif defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
    if ( Gnuplot::file_exists(tmp,1) ) // check existence and execution permission
#endif
    {
        Gnuplot::m_sGNUPlotPath = path;
        return true;
    }
    else
    {
        Gnuplot::m_sGNUPlotPath.clear();
        return false;
    }
}
```

3.1.3.24 Gnuplot& Gnuplot::set_grid () [inline]

turns grid on/off

Definition at line 266 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{cmd("set grid");return *this;};
```

3.1.3.25 Gnuplot& Gnuplot::set_hidden3d () [inline]

enables/disables hidden line removal for surface plotting (for 3d plot)

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 302 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{cmd("set hidden3d");return *this;};
```

3.1.3.26 Gnuplot & Gnuplot::set_isosamples (const int *isolines* = `10`)

set isoline density (grid) for plotting functions as surfaces (for 3d plots)

Definition at line 1175 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;
    cmdstr << "set isosamples " << isolines;
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.27 Gnuplot & Gnuplot::set_legend (const std::string & *position* = `"default"`)

switches legend on/off position: inside/outside, left/center/right, top/center/bottom, nobox/box

Definition at line 1092 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;
    cmdstr << "set key " << position;

    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.28 Gnuplot& Gnuplot::set_multiplot () `[inline]`

set the mulitplot mode

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 277 of file gnuplot_i.hpp.

References cmd().

```
{cmd("set multiplot") ;return *this;};
```

3.1.3.29 Gnuplot & Gnuplot::set_pointsize (const double *pointsize* = `1.0`)

scales the size of the points used in plots

Definition at line 1148 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;
    cmdstr << "set pointsize " << pointsize;
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.30 Gnuplot & Gnuplot::set_samples (const int *samples* = `100`)

set sampling rate of functions, or for interpolating data

Definition at line 1161 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;
    cmdstr << "set samples " << samples;
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.31 Gnuplot & Gnuplot::set_smooth (const std::string & *stylestr* = `"csplines"`)

interpolation and approximation of data, arguments: csplines, bezier, acsplines (for data values > 0), sbezier, unique, frequency (works only with plot_x, plot_xy, plotfile_x, plotfile_xy (if smooth is set, set_-style has no effekt on data plotting)

Definition at line 1041 of file gnuplot_i.hpp.

Referenced by main().

```
{
    if (stylestr.find("unique")    == std::string::npos  &&
        stylestr.find("frequency") == std::string::npos  &&
        stylestr.find("csplines")  == std::string::npos  &&
        stylestr.find("acsplines") == std::string::npos  &&
        stylestr.find("bezier")    == std::string::npos  &&
        stylestr.find("sbezier")   == std::string::npos  )
    {
        smooth = "";
    }
    else
    {
        smooth = stylestr;
    }

    return *this;
}
```

3.1.3.32 Gnuplot & Gnuplot::set_style (const std::string & *stylestr* = `"points"`)

set line style (some of these styles require additional information): lines, points, linespoints, impulses, dots, steps, fsteps, histeps, boxes, histograms, filledcurves

Definition at line 995 of file gnuplot_i.hpp.

Referenced by Gnuplot(), and main().

```
{
    if (stylestr.find("lines")          == std::string::npos  &&
        stylestr.find("points")         == std::string::npos  &&
        stylestr.find("linespoints")    == std::string::npos  &&
        stylestr.find("impulses")       == std::string::npos  &&
        stylestr.find("dots")           == std::string::npos  &&
        stylestr.find("steps")          == std::string::npos  &&
        stylestr.find("fsteps")         == std::string::npos  &&
        stylestr.find("histeps")        == std::string::npos  &&
        stylestr.find("boxes")          == std::string::npos  &&  // 1-4 columns of data are requirec
        stylestr.find("filledcurves")   == std::string::npos  &&
        stylestr.find("histograms")     == std::string::npos  )   //only for one data column
//      stylestr.find("labels")         == std::string::npos  &&  // 3 columns of data are requirec
//      stylestr.find("xerrorbars")     == std::string::npos  &&  // 3-4 columns of data are requir
//      stylestr.find("xerrorlines")    == std::string::npos  &&  // 3-4 columns of data are requir
//      stylestr.find("errorbars")      == std::string::npos  &&  // 3-4 columns of data are requir
//      stylestr.find("errorlines")     == std::string::npos  &&  // 3-4 columns of data are requir
//      stylestr.find("yerrorbars")     == std::string::npos  &&  // 3-4 columns of data are requir
//      stylestr.find("yerrorlines")    == std::string::npos  &&  // 3-4 columns of data are requir
//      stylestr.find("boxerrorbars")   == std::string::npos  &&  // 3-5 columns of data are requir
//      stylestr.find("xyerrorbars")    == std::string::npos  &&  // 4,6,7 columns of data are requ
//      stylestr.find("xyerrorlines")   == std::string::npos  &&  // 4,6,7 columns of data are requ
//      stylestr.find("boxxyerrorbars") == std::string::npos  &&  // 4,6,7 columns of data are requ
//      stylestr.find("financebars")    == std::string::npos  &&  // 5 columns of data are requirec
//      stylestr.find("candlesticks")   == std::string::npos  &&  // 5 columns of data are requirec
//      stylestr.find("vectors")        == std::string::npos  &&
//      stylestr.find("image")          == std::string::npos  &&
//      stylestr.find("rgbimage")       == std::string::npos  &&
//      stylestr.find("pm3d")           == std::string::npos  )
    {
        pstyle = std::string("points");
    }
    else
    {
        pstyle = stylestr;
    }

    return *this;
}
```

3.1.3.33 Gnuplot& Gnuplot::set_surface () `[inline]`

enables/disables the display of surfaces (for 3d plot)

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 332 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{cmd("set surface");return *this;};
```

3.1.3.34 void Gnuplot::set_terminal_std (const std::string & *type*) `[static]`

optional: set standart terminal, used by showonscreen defaults: Windows - win, Linux - x11, Mac - aqua

Parameters:

type –> the terminal type

Returns:

—

Definition at line 884 of file gnuplot_i.hpp.

```
{
#if defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
    if (type.find("x11") != std::string::npos && getenv("DISPLAY") == NULL)
    {
        throw GnuplotException("Can't find DISPLAY variable");
    }
#endif


    Gnuplot::terminal_std = type;
    return;
}
```

3.1.3.35 Gnuplot& Gnuplot::set_title (const std::string & *title* = "") `[inline]`

sets and clears the title of a gnuplot session

Parameters:

title –> the title of the plot [optional, default == ""]

Returns:

<– reference to the gnuplot object

Definition at line 366 of file gnuplot_i.hpp.

Referenced by main(), and unset_title().

```
{
    std::string cmdstr;
    cmdstr = "set title \"";
    cmdstr+=title;
    cmdstr+="\"";
    *this<<cmdstr;
    return *this;
}
```

3.1.3.36 Gnuplot& Gnuplot::set_xautoscale () `[inline]`

autoscale axis (set by default) of xaxis

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 409 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{cmd("set xrange restore");cmd("set autoscale x");return *this;};
```

3.1.3.37 Gnuplot & Gnuplot::set_xlabel (const std::string & *label* = `"y"`)

set y axis label

Definition at line 1211 of file gnuplot_i.hpp.

References cmd().

Referenced by Gnuplot(), and main().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set xlabel \"" << label << "\"";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.38 Gnuplot & Gnuplot::set_xlogscale (const double *base* = `10`)

turns on/off log scaling for the specified xaxis (logscale is not set by default)

Definition at line 1106 of file gnuplot_i.hpp.

References cmd().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set logscale x " << base;
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.39 Gnuplot & Gnuplot::set_xrange (const double *iFrom*, const double *iTo*)

set axis - ranges

Definition at line 1252 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set xrange[" << iFrom << ":" << iTo << "]";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.40 Gnuplot& Gnuplot::set_yautoscale () `[inline]`

autoscale axis (set by default) of yaxis

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 418 of file gnuplot_i.hpp.

References cmd().

```
{cmd("set yrange restore");cmd("set autoscale y");return *this;};
```

3.1.3.41 Gnuplot & Gnuplot::set_ylabel (const std::string & *label* = "x")

set x axis label

Definition at line 1224 of file gnuplot_i.hpp.

References cmd().

Referenced by Gnuplot(), and main().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set ylabel \"" << label << "\"";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.42 Gnuplot & Gnuplot::set_ylogscale (const double *base* = 10)

turns on/off log scaling for the specified yaxis (logscale is not set by default)

Definition at line 1120 of file gnuplot_i.hpp.

References cmd().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set logscale y " << base;
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.43 Gnuplot & Gnuplot::set_yrange (const double *iFrom*, const double *iTo*)

set y-axis - ranges

Definition at line 1266 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set yrange[" << iFrom << ":" << iTo << "]";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.44 Gnuplot& Gnuplot::set_zautoscale () [inline]

autoscale axis (set by default) of zaxis

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 427 of file gnuplot_i.hpp.

References cmd().

```
{cmd("set zrange restore");cmd("set autoscale z");return *this;};
```

3.1.3.45 Gnuplot & Gnuplot::set_zlabel (const std::string & *label* = "z")

set z axis label

Definition at line 1237 of file gnuplot_i.hpp.

References cmd().

Referenced by Gnuplot(), and main().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set zlabel \"" << label << "\"";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.46 Gnuplot & Gnuplot::set_zlogscale (const double *base* = 10)

turns on/off log scaling for the specified zaxis (logscale is not set by default)

Definition at line 1134 of file gnuplot_i.hpp.

References cmd().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set logscale z " << base;
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.47 Gnuplot & Gnuplot::set_zrange (const double *iFrom*, const double *iTo*)

set z-axis - ranges

Definition at line 1280 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{
    std::ostringstream cmdstr;

    cmdstr << "set zrange[" << iFrom << ":" << iTo << "]";
    cmd(cmdstr.str());

    return *this;
}
```

3.1.3.48 Gnuplot & Gnuplot::showonscreen ()

sets terminal type to terminal_std

Definition at line 1065 of file gnuplot_i.hpp.

References cmd().

Referenced by main(), and reset_all().

```
{
    cmd("set output");
    cmd("set terminal " + Gnuplot::terminal_std);

    return *this;
}
```

3.1.3.49 Gnuplot& Gnuplot::unset_contour () `[inline]`

contour is not set by default, it disables contour drawing for surfaces

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 323 of file gnuplot_i.hpp.

References cmd().

```
{cmd("unset contour");return *this;};
```

3.1.3.50 Gnuplot& Gnuplot::unset_grid () `[inline]`

grid is not set by default

Definition at line 268 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
{cmd("unset grid");return *this;};
```

3.1.3.51 Gnuplot& Gnuplot::unset_hidden3d () `[inline]`

hidden3d is not set by default

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 311 of file gnuplot_i.hpp.

References cmd().

```
311 {cmd("unset hidden3d"); return *this;};
```

3.1.3.52 Gnuplot& Gnuplot::unset_legend () `[inline]`

Switches legend off attention:legend is set by default.

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 357 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
357 {cmd("unset key"); return *this;}
```

3.1.3.53 Gnuplot& Gnuplot::unset_multiplot () `[inline]`

unsets the mulitplot mode

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 286 of file gnuplot_i.hpp.

References cmd().

```
286 {cmd("unset multiplot");return *this;};
```

3.1.3.54 Gnuplot& Gnuplot::unset_smooth () `[inline]`

unset smooth attention: smooth is not set by default

Parameters:

—

Returns:

<– a reference to a gnuplot object

Definition at line 259 of file gnuplot_i.hpp.

Referenced by main().

```
259 { smooth = ""; return *this;};
```

3.1.3.55 Gnuplot& Gnuplot::unset_surface () `[inline]`

surface is set by default, it disables the display of surfaces (for 3d plot)

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 342 of file gnuplot_i.hpp.

References cmd().

Referenced by main().

```
342 {cmd("unset surface"); return *this;}
```

3.1.3.56 Gnuplot& Gnuplot::unset_title () `[inline]`

Clears the title of a gnuplot session The title is not set by default.

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 384 of file gnuplot_i.hpp.

References set_title().

Referenced by main().

```
384 {this->set_title();return *this;}
```

3.1.3.57 Gnuplot& Gnuplot::unset_xlogscale () `[inline]`

turns off log scaling for the x axis

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 444 of file gnuplot_i.hpp.

References cmd().

```
444 {cmd("unset logscale x"); return *this;};
```

3.1.3.58 Gnuplot& Gnuplot::unset_ylogscale () `[inline]`

turns off log scaling for the y axis

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 453 of file gnuplot_i.hpp.

References cmd().

```
453 {cmd("unset logscale y"); return *this;};
```

3.1.3.59 Gnuplot& Gnuplot::unset_zlogscale () `[inline]`

turns off log scaling for the z axis

Parameters:

—

Returns:

<– reference to the gnuplot object

Definition at line 462 of file gnuplot_i.hpp.

References cmd().

```
462 {cmd("unset logscale z"); return *this;};
```

The documentation for this class was generated from the following file:

- **gnuplot_i.hpp**

3.2 GnuplotException Class Reference

A C++ interface to gnuplot.

`#include <gnuplot_i.hpp>`

Public Member Functions

- **GnuplotException** (const std::string &msg)

3.2.1 Detailed Description

A C++ interface to gnuplot.

The interface uses pipes and so won't run on a system that doesn't have POSIX pipe support Tested on Windows (MinGW and Visual C++) and Linux (GCC)

Version history: 0. C interface by N. Devillard (27/01/03) 1. C++ interface: direct translation from the C interface by Rajarshi Guha (07/03/03) 2. corrections for Win32 compatibility by V. Chyzhdzenka (20/05/03) 3. some member functions added, corrections for Win32 and Linux compatibility by M. Burgis (10/03/08)

Requirements: gnuplot has to be installed (`http://www.gnuplot.info/download.html`) for Windows: set Path-Variable for **Gnuplot** (p. 5) path (e.g. C:/program files/gnuplot/bin) or set **Gnuplot** (p. 5) path with: **Gnuplot::set_GNUPlotPath(const std::string &path)** (p. 23);

Definition at line 60 of file gnuplot_i.hpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 GnuplotException::GnuplotException (const std::string & *msg*) `[inline]`

Definition at line 63 of file gnuplot_i.hpp.

```
63 : std::runtime_error(msg){}
```

The documentation for this class was generated from the following file:

- **gnuplot_i.hpp**

Chapter 4

File Documentation

4.1 example.cc File Reference

```
#include <iostream>
#include "gnuplot_i.hpp"
```

Defines

- #define **SLEEP_LGTH** 2
- #define **NPOINTS** 50

Functions

- void **wait_for_key** ()
- int **main** (int argc, char ∗argv[ ])

4.1.1 Define Documentation

4.1.1.1 #define NPOINTS 50

Definition at line 20 of file example.cc.

Referenced by main().

4.1.1.2 #define SLEEP_LGTH 2

Definition at line 19 of file example.cc.

4.1.2 Function Documentation

4.1.2.1 int main (int *argc*, char ∗ *argv*[ ])

Definition at line 27 of file example.cc.

References Gnuplot::cmd(), NPOINTS, Gnuplot::plot_equation(), Gnuplot::plot_equation3d(), Gnuplot::plot_image(), Gnuplot::plot_slope(), Gnuplot::plot_x(), Gnuplot::plot_xy(), Gnuplot::plot_-xy_err(), Gnuplot::plot_xyz(), Gnuplot::replot(), Gnuplot::reset_all(), Gnuplot::reset_plot(), Gnu-plot::savetops(), Gnuplot::set_cbrange(), Gnuplot::set_contour(), Gnuplot::set_grid(), Gnuplot::set_-hidden3d(), Gnuplot::set_isosamples(), Gnuplot::set_legend(), Gnuplot::set_pointsize(), Gnuplot::set_-samples(), Gnuplot::set_smooth(), Gnuplot::set_style(), Gnuplot::set_surface(), Gnuplot::set_title(), Gnuplot::set_xautoscale(), Gnuplot::set_xlabel(), Gnuplot::set_xrange(), Gnuplot::set_ylabel(), Gnuplot::set_yrange(), Gnuplot::set_zlabel(), Gnuplot::set_zrange(), Gnuplot::showonscreen(), Gnuplot::unset_grid(), Gnuplot::unset_legend(), Gnuplot::unset_smooth(), Gnuplot::unset_surface(), Gnuplot::unset_title(), and wait_for_key().

```
{
    // if path-variable for gnuplot is not set, do it with:
    // Gnuplot::set_GNUPlotPath("C:/program files/gnuplot/bin/");

    // set a special standard terminal for showonscreen (normally not needed),
    //   e.g. Mac users who want to use x11 instead of aqua terminal:
    // Gnuplot::set_terminal_std("x11");

    cout << "*** example of gnuplot control through C++ ***" << endl << endl;

    //
    // Using the GnuplotException class
    //
    try
    {
        Gnuplot g1("lines");

        //
        // Slopes
        //
        cout << "*** plotting slopes" << endl;
        g1.set_title("Slopes\\nNew Line");

        cout << "y = x" << endl;
        g1.plot_slope(1.0,0.0,"y=x");

        cout << "y = 2*x" << endl;
        g1.plot_slope(2.0,0.0,"y=2x");

        cout << "y = -x" << endl;
        g1.plot_slope(-1.0,0.0,"y=-x");
        g1.unset_title();

        //
        // Equations
        //
        g1.reset_plot();
        cout << endl << endl << "*** various equations" << endl;

        cout << "y = sin(x)" << endl;
        g1.plot_equation("sin(x)","sine");

        cout << "y = log(x)" << endl;
        g1.plot_equation("log(x)","logarithm");

        cout << "y = sin(x) * cos(2*x)" << endl;
        g1.plot_equation("sin(x)*cos(2*x)","sine product");

        //
        // Styles
        //
        g1.reset_plot();
        cout << endl << endl << "*** showing styles" << endl;

        cout << "sine in points" << endl;
        g1.set_pointsize(0.8).set_style("points");
        g1.plot_equation("sin(x)","points");

        cout << "sine in impulses" << endl;
        g1.set_style("impulses");
        g1.plot_equation("sin(x)","impulses");

        cout << "sine in steps" << endl;
        g1.set_style("steps");
        g1.plot_equation("sin(x)","steps");

        //
        // Save to ps
        //
        g1.reset_all();
        cout << endl << endl << "*** save to ps " << endl;

         cout << "y = sin(x) saved to test_output.ps in working directory" << endl;
         g1.savetops("test_output");
         g1.set_style("lines").set_samples(300).set_xrange(0,5);
         g1.plot_equation("sin(12*x)*exp(-x)").plot_equation("exp(-x)");

         g1.showonscreen(); // window output


         //
         // User defined 1d, 2d and 3d point sets
         //
         std::vector<double> x, y, y2, dy, z;

         for (int i = 0; i < NPOINTS; i++)  // fill double arrays x, y, z
         {
             x.push_back((double)i);             // x[i] = i
             y.push_back((double)i * (double)i); // y[i] = i^2
             z.push_back( x[i]*y[i] );           // z[i] = x[i]*y[i] = i^3
             dy.push_back((double)i * (double)i / (double) 10); // dy[i] = i^2 / 10
         }
         y2.push_back(0.00); y2.push_back(0.78); y2.push_back(0.97); y2.push_back(0.43);
         y2.push_back(-0.44); y2.push_back(-0.98); y2.push_back(-0.77); y2.push_back(0.02);


         g1.reset_all();
         cout << endl << endl << "*** user-defined lists of doubles" << endl;
         g1.set_style("impulses").plot_x(y,"user-defined doubles");

         g1.reset_plot();
         cout << endl << endl << "*** user-defined lists of points (x,y)" << endl;
         g1.set_grid();
         g1.set_style("points").plot_xy(x,y,"user-defined points 2d");

         g1.reset_plot();
         cout << endl << endl << "*** user-defined lists of points (x,y,z)" << endl;
         g1.unset_grid();
         g1.plot_xyz(x,y,z,"user-defined points 3d");

         g1.reset_plot();
         cout << endl << endl << "*** user-defined lists of points (x,y,dy)" << endl;
         g1.plot_xy_err(x,y,dy,"user-defined points 2d with errorbars");


         //
         // Multiple output screens
         //
         cout << endl << endl;
         cout << "*** multiple output windows" << endl;

         g1.reset_plot();
         g1.set_style("lines");
         cout << "window 1: sin(x)" << endl;
         g1.set_grid().set_samples(600).set_xrange(0,300);
         g1.plot_equation("sin(x)+sin(x*1.1)");

         g1.set_xautoscale().replot();

         Gnuplot g2;
         cout << "window 2: user defined points" << endl;
         g2.plot_x(y2,"points");
         g2.set_smooth().plot_x(y2,"cspline");
         g2.set_smooth("bezier").plot_x(y2,"bezier");
         g2.unset_smooth();

         Gnuplot g3("lines");
         cout << "window 3: log(x)/x" << endl;
         g3.set_grid();
         g3.plot_equation("log(x)/x","log(x)/x");

         Gnuplot g4("lines");
         cout << "window 4: splot x*x+y*y" << endl;
         g4.set_zrange(0,100);
         g4.set_xlabel("x-axis").set_ylabel("y-axis").set_zlabel("z-axis");
         g4.plot_equation3d("x*x+y*y");

         Gnuplot g5("lines");
         cout << "window 5: splot with hidden3d" << endl;
         g5.set_isosamples(25).set_hidden3d();
         g5.plot_equation3d("x*y*y");

         Gnuplot g6("lines");
         cout << "window 6: splot with contour" << endl;
         g6.set_isosamples(60).set_contour();
         g6.unset_surface().plot_equation3d("sin(x)*sin(y)+4");

         g6.set_surface().replot();

         Gnuplot g7("lines");
         cout << "window 7: set_samples" << endl;
         g7.set_xrange(-30,20).set_samples(40);
         g7.plot_equation("besj0(x)*0.12e1").plot_equation("(x**besj0(x))-2.5");

         g7.set_samples(400).replot();

         Gnuplot g8("filledcurves");
         cout << "window 8: filledcurves" << endl;
         g8.set_legend("outside right top").set_xrange(-5,5);
         g8.plot_equation("x*x").plot_equation("-x*x+4");

         //
         // Plot an image
         //
         Gnuplot g9;
         cout << "window 9: plot_image" << endl;
         const int iWidth  = 255;
         const int iHeight = 255;
         g9.set_xrange(0,iWidth).set_yrange(0,iHeight).set_cbrange(0,255);
         g9.cmd("set palette gray");
         unsigned char ucPicBuf[iWidth*iHeight];
         // generate a greyscale image
         for(int iIndex = 0; iIndex < iHeight*iWidth; iIndex++)
         {
             ucPicBuf[iIndex] = iIndex%255;
         }
         g9.plot_image(ucPicBuf,iWidth,iHeight,"greyscale");

         g9.set_pointsize(0.6).unset_legend().plot_slope(0.8,20);

         //
         // manual control
         //
         Gnuplot g10;
         cout << "window 10: manual control" << endl;
         g10.cmd("set samples 400").cmd("plot abs(x)/2"); // either with cmd()
         g10 << "replot sqrt(x)" << "replot sqrt(-x)";    // or with <<

         wait_for_key();

     }
     catch (GnuplotException ge)
     {
         cout << ge.what() << endl;
     }


     cout << endl << "*** end of gnuplot example" << endl;

     return 0;
 }
```

4.1.2.2 void wait_for_key ()

Definition at line 242 of file example.cc.

Referenced by main().

```
{
#if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__TOS_WIN__)  // every keypress
    cout << endl << "Press any key to continue..." << endl;

    FlushConsoleInputBuffer(GetStdHandle(STD_INPUT_HANDLE));
    _getch();
#elif defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
    cout << endl << "Press ENTER to continue..." << endl;

    std::cin.clear();
    std::cin.ignore(std::cin.rdbuf()->in_avail());
    std::cin.get();
#endif
    return;
}
```

4.2 gnuplot_i.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <fstream>
#include <sstream>
#include <stdexcept>
#include <cstdio>
#include <cstdlib>
#include <list>
```

Classes

- class **GnuplotException**

  A C++ interface to gnuplot.

- class **Gnuplot**

Functions

- template<typename Container >
  void **stringtok** (Container &container, std::string const &in, const char ∗const delimiters=" \t\n")

4.2.1 Function Documentation

4.2.1.1 template<typename Container > void stringtok (Container & *container*, std::string const & *in*, const char ∗const *delimiters* = " \t\n") [inline]

Definition at line 905 of file gnuplot_i.hpp.

```
{
    const std::string::size_type len = in.length();
          std::string::size_type i = 0;

    while ( i < len )
    {
        // eat leading whitespace
        i = in.find_first_not_of (delimiters, i);

        if (i == std::string::npos)
            return;   // nothing left but white space

        // find the end of the token
        std::string::size_type j = in.find_first_of (delimiters, i);

        // push token
        if (j == std::string::npos)
        {
            container.push_back (in.substr(i));
            return;
        }
        else
            container.push_back (in.substr(i, j-i));

        // set up for next loop
        i = j + 1;
    }

    return;
}
```

Index