

# gnuplot-cpp

## 0.9

Generated by Doxygen 1.5.8

Wed Sep 16 12:55:07 2009



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Gnuplot Class Reference . . . . .	5
3.1.1	Detailed Description . . . . .	8
3.1.2	Constructor & Destructor Documentation . . . . .	8
3.1.2.1	Gnuplot . . . . .	8
3.1.2.2	Gnuplot . . . . .	9
3.1.2.3	Gnuplot . . . . .	9
3.1.2.4	Gnuplot . . . . .	9
3.1.2.5	~Gnuplot . . . . .	10
3.1.3	Member Function Documentation . . . . .	10
3.1.3.1	cmd . . . . .	10
3.1.3.2	is_valid . . . . .	11
3.1.3.3	operator<< . . . . .	11
3.1.3.4	plot_equation . . . . .	12
3.1.3.5	plot_equation3d . . . . .	12
3.1.3.6	plot_image . . . . .	13
3.1.3.7	plot_slope . . . . .	14
3.1.3.8	plot_x . . . . .	15
3.1.3.9	plot_xy . . . . .	15
3.1.3.10	plot_xy_err . . . . .	16
3.1.3.11	plot_xyz . . . . .	17
3.1.3.12	plotfile_x . . . . .	17
3.1.3.13	plotfile_xy . . . . .	18

3.1.3.14	plotfile_xy_err . . . . .	19
3.1.3.15	plotfile_xyz . . . . .	20
3.1.3.16	remove_tmpfiles . . . . .	20
3.1.3.17	replot . . . . .	21
3.1.3.18	reset_all . . . . .	21
3.1.3.19	reset_plot . . . . .	21
3.1.3.20	savetops . . . . .	22
3.1.3.21	set_cbrange . . . . .	22
3.1.3.22	set_contour . . . . .	22
3.1.3.23	set_GNUPlotPath . . . . .	23
3.1.3.24	set_grid . . . . .	23
3.1.3.25	set_hidden3d . . . . .	23
3.1.3.26	set_isosamples . . . . .	24
3.1.3.27	set_legend . . . . .	24
3.1.3.28	set_multiplot . . . . .	24
3.1.3.29	set_pointsize . . . . .	25
3.1.3.30	set_samples . . . . .	25
3.1.3.31	set_smooth . . . . .	25
3.1.3.32	set_style . . . . .	26
3.1.3.33	set_surface . . . . .	27
3.1.3.34	set_terminal_std . . . . .	27
3.1.3.35	set_title . . . . .	27
3.1.3.36	set_xautoscale . . . . .	28
3.1.3.37	set_xlabel . . . . .	28
3.1.3.38	set_xlogscale . . . . .	29
3.1.3.39	set_xrange . . . . .	29
3.1.3.40	set_yautoscale . . . . .	29
3.1.3.41	set_ylabel . . . . .	30
3.1.3.42	set_ylogscale . . . . .	30
3.1.3.43	set_yrange . . . . .	30
3.1.3.44	set_zautoscale . . . . .	30
3.1.3.45	set_zlabel . . . . .	31
3.1.3.46	set_zlogscale . . . . .	31
3.1.3.47	set_zrange . . . . .	31
3.1.3.48	showonscreen . . . . .	32
3.1.3.49	unset_contour . . . . .	32

3.1.3.50	unset_grid . . . . .	32
3.1.3.51	unset_hidden3d . . . . .	33
3.1.3.52	unset_legend . . . . .	33
3.1.3.53	unset_multiplot . . . . .	33
3.1.3.54	unset_smooth . . . . .	34
3.1.3.55	unset_surface . . . . .	34
3.1.3.56	unset_title . . . . .	34
3.1.3.57	unset_xlogscale . . . . .	35
3.1.3.58	unset_ylogscale . . . . .	35
3.1.3.59	unset_zlogscale . . . . .	35
3.2	GnuplotException Class Reference . . . . .	36
3.2.1	Detailed Description . . . . .	36
3.2.2	Constructor & Destructor Documentation . . . . .	36
3.2.2.1	GnuplotException . . . . .	36
<b>4</b>	<b>File Documentation</b>	<b>37</b>
4.1	example.cc File Reference . . . . .	37
4.1.1	Define Documentation . . . . .	37
4.1.1.1	NPOINTS . . . . .	37
4.1.1.2	SLEEP_LGTH . . . . .	37
4.1.2	Function Documentation . . . . .	37
4.1.2.1	main . . . . .	37
4.1.2.2	wait_for_key . . . . .	41
4.2	gnuplot_i.hpp File Reference . . . . .	42
4.2.1	Function Documentation . . . . .	42
4.2.1.1	stringtok . . . . .	42



# Chapter 1

## Class Index

### 1.1 Class List

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

<b>Gnuplot</b> . . . . .	5
<b>GnuplotException</b> (A C++ interface to gnuplot ) . . . . .	36





# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<b>example.cc</b>	37
<b>gnuplot_i.hpp</b>	42



# 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="")
- **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="")
- **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="")
- **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="")
- **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\(\)](#).

```

613                                     :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)
614
615 {
616     init();
617     set_style(style);
618 }

```

### 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\(\)](#).

```

629                                     :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)
630 {
631     init();
632
633     set_style(style);
634     set_xlabel(labelx);
635     set_ylabel(labely);
636
637     plot_x(x,title);
638 }

```

### 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\(\)](#).

```

651                                     :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)
652 {
653     init();
654
655     set_style(style);
656     set_xlabel(labelx);
657     set_ylabel(labely);
658
659     plot_xy(x,y,title);
660 }

```

### 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()`.

```

675                                     :gnucmd(NULL) ,valid(false) ,two_dim(false) ,nplots(0)
676 {
677     init();
678
679     set_style(style);
680     set_xlabel(labelx);
681     set_ylabel(labely);
682     set_zlabel(labelz);
683
684     plot_xyz(x,y,z,title);
685 }
```

### 3.1.2.5 Gnuplot::~Gnuplot ()

destructor: needed to delete temporary files

Definition at line 944 of file `gnuplot_i.hpp`.

```

945 {
946 //  remove_tmpfiles();
947
948     // A stream opened by popen() should be closed by pclose()
949 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__TOS_WIN__)
950     if (_pclose(gnucmd) == -1)
951 #elif defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
952     if (pclose(gnucmd) == -1)
953 #endif
954         throw GnuplotException("Problem closing communication to gnuplot");
955 }
```

## 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()`.

```

1642 {
1643     if( !(valid) )
1644     {
1645         return *this;
1646     }
1647
1648
1649     // int fputs ( const char * str, FILE * stream );
1650     // writes the string str to the stream.
1651     // The function begins copying from the address specified (str) until it
1652     // reaches the terminating null character ('\0'). This final
```



```

1653     // null-character is not copied to the stream.
1654     fputs( (cmdstr+"\n").c_str(), gnucmd );
1655
1656     // int fflush ( FILE * stream );
1657     // If the given stream was open for writing and the last i/o operation was
1658     // an output operation, any unwritten data in the output buffer is written
1659     // to the file. If the argument is a null pointer, all open files are
1660     // flushed. The stream remains open after this call.
1661     fflush(gnucmd);
1662
1663
1664     if( cmdstr.find("replot") != std::string::npos )
1665     {
1666         return *this;
1667     }
1668     else if( cmdstr.find("splot") != std::string::npos )
1669     {
1670         two_dim = false;
1671         nplots++;
1672     }
1673     else if( cmdstr.find("plot") != std::string::npos )
1674     {
1675         two_dim = true;
1676         nplots++;
1677     }
1678
1679     return *this;
1680 }

```

### 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()`.

```

220                                     {
221         cmd(cmdstr);
222         return(*this);
223     }
```

### 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, `-` subtract, `%` 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()`.

```

1347 {
1348     std::ostringstream cmdstr;
1349     //
1350     // command to be sent to gnuplot
1351     //
1352     if (nplots > 0 && two_dim == true)
1353         cmdstr << "replot ";
1354     else
1355         cmdstr << "plot ";
1356
1357     cmdstr << equation << " title \"";
1358
1359     if (title == "")
1360         cmdstr << "f(x) = " << equation;
1361     else
1362         cmdstr << title;
1363
1364     cmdstr << "\" with " << pstyle;
1365
1366     //
1367     // Do the actual plot
1368     //
1369     cmd(cmdstr.str());
1370
1371     return *this;
1372 }
```

### 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()`.

```

1380 {
1381     std::ostringstream cmdstr;
1382     //
1383     // command to be sent to gnuplot
1384     //
1385     if (nplots > 0 && two_dim == false)
1386         cmdstr << "replot ";
1387     else
1388         cmdstr << "splot ";
1389
1390     cmdstr << equation << " title \"";
1391
1392     if (title == "")
1393         cmdstr << "f(x,y) = " << equation;
1394     else
1395         cmdstr << title;
1396
1397     cmdstr << "\" with " << pstyle;
1398
1399     //
1400     // Do the actual plot
1401     //
1402     cmd(cmdstr.str());
1403
1404     return *this;
1405 }
```

### 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()`.

```

1590 {
1591     std::ofstream tmp;
1592     std::string name = create_tmpfile(tmp);
1593     if (name == "")
1594         return *this;
1595
1596     //
1597     // write the data to file
1598     //
1599     int iIndex = 0;
1600     for(int iRow = 0; iRow < iHeight; iRow++)
1601     {
1602         for(int iColumn = 0; iColumn < iWidth; iColumn++)
1603         {
1604             tmp << iColumn << " " << iRow << " "
1605                 << static_cast<float>(ucPicBuf[iIndex++]) << std::endl;
1606         }
1607     }
```

```

1608
1609     tmp.flush();
1610     tmp.close();
1611
1612
1613     std::ostringstream cmdstr;
1614     //
1615     // command to be sent to gnuplot
1616     //
1617     if (nplots > 0 && two_dim == true)
1618         cmdstr << "replot ";
1619     else
1620         cmdstr << "plot ";
1621
1622     if (title == "")
1623         cmdstr << "\"" << name << "\" with image";
1624     else
1625         cmdstr << "\"" << name << "\" title \"" << title << "\" with image";
1626
1627     //
1628     // Do the actual plot
1629     //
1630     cmd(cmdstr.str());
1631
1632     return *this;
1633 }

```

### 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()`.

```

1314 {
1315     std::ostringstream cmdstr;
1316     //
1317     // command to be sent to gnuplot
1318     //
1319     if (nplots > 0 && two_dim == true)
1320         cmdstr << "replot ";
1321     else
1322         cmdstr << "plot ";
1323
1324     cmdstr << a << " * x + " << b << " title \"";
1325
1326     if (title == "")
1327         cmdstr << "f(x) = " << a << " * x + " << b;
1328     else
1329         cmdstr << title;
1330
1331     cmdstr << "\"" with " << pstyle;
1332
1333     //
1334     // Do the actual plot
1335     //
1336     cmd(cmdstr.str());
1337
1338     return *this;
1339 }

```

**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().

```
694 {
695     if (x.size() == 0)
696     {
697         throw GnuplotException("std::vector too small");
698         return *this;
699     }
700
701     std::ofstream tmp;
702     std::string name = create_tmpfile(tmp);
703     if (name == "")
704         return *this;
705
706     //
707     // write the data to file
708     //
709     for (unsigned int i = 0; i < x.size(); i++)
710         tmp << x[i] << std::endl;
711
712     tmp.flush();
713     tmp.close();
714
715     plotfile_x(name, 1, title);
716
717     return *this;
718 }
719 }
```

**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().

```
728 {
729     if (x.size() == 0 || y.size() == 0)
730     {
731         throw GnuplotException("std::vectors too small");
732         return *this;
733     }
734
735     if (x.size() != y.size())
736     {
737         throw GnuplotException("Length of the std::vectors differs");
738         return *this;
739     }
740 }
```

```

739     }
740
741
742     std::ofstream tmp;
743     std::string name = create_tmpfile(tmp);
744     if (name == "")
745         return *this;
746
747     //
748     // write the data to file
749     //
750     for (unsigned int i = 0; i < x.size(); i++)
751         tmp << x[i] << " " << y[i] << std::endl;
752
753     tmp.flush();
754     tmp.close();
755
756
757     plotfile_xy(name, 1, 2, title);
758
759     return *this;
760 }

```

### 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()`.

```

771 {
772     if (x.size() == 0 || y.size() == 0 || dy.size() == 0)
773     {
774         throw GnuplotException("std::vectors too small");
775         return *this;
776     }
777
778     if (x.size() != y.size() || y.size() != dy.size())
779     {
780         throw GnuplotException("Length of the std::vectors differs");
781         return *this;
782     }
783
784
785     std::ofstream tmp;
786     std::string name = create_tmpfile(tmp);
787     if (name == "")
788         return *this;
789
790     //
791     // write the data to file
792     //
793     for (unsigned int i = 0; i < x.size(); i++)
794         tmp << x[i] << " " << y[i] << " " << dy[i] << std::endl;
795
796     tmp.flush();
797     tmp.close();

```

```

798
799
800     // Do the actual plot
801     plotfile_xy_err(name, 1, 2, 3, title);
802
803     return *this;
804 }

```

### 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()`.

```

816 {
817     if (x.size() == 0 || y.size() == 0 || z.size() == 0)
818     {
819         throw GnuplotException("std::vectors too small");
820         return *this;
821     }
822
823     if (x.size() != y.size() || x.size() != z.size())
824     {
825         throw GnuplotException("Length of the std::vectors differs");
826         return *this;
827     }
828
829
830     std::ofstream tmp;
831     std::string name = create_tmpfile(tmp);
832     if (name == "")
833         return *this;
834
835     //
836     // write the data to file
837     //
838     for (unsigned int i = 0; i < x.size(); i++)
839         tmp << x[i] << " " << y[i] << " " << z[i] <<std::endl;
840
841     tmp.flush();
842     tmp.close();
843
844
845     plotfile_xyz(name, 1, 2, 3, title);
846
847     return *this;
848 }

```

### 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()`.

```

1415 {
1416     //
1417     // check if file exists
1418     //
1419     file_available(filename);
1420
1421
1422     std::ostringstream cmdstr;
1423     //
1424     // command to be sent to gnuplot
1425     //
1426     if (nplots > 0 && two_dim == true)
1427         cmdstr << "replot ";
1428     else
1429         cmdstr << "plot ";
1430
1431     cmdstr << "\"" << filename << "\" using " << column;
1432
1433     if (title == "")
1434         cmdstr << " notitle ";
1435     else
1436         cmdstr << " title \"" << title << "\" ";
1437
1438     if(smooth == "")
1439         cmdstr << "with " << pstyle;
1440     else
1441         cmdstr << "smooth " << smooth;
1442
1443     //
1444     // Do the actual plot
1445     //
1446     cmd(cmdstr.str()); //nplots++; two_dim = true;  already in cmd();
1447
1448     return *this;
1449 }
```

### 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()`.

```

1461 {
1462     //
1463     // check if file exists
1464     //
1465     file_available(filename);
1466
1467
1468     std::ostringstream cmdstr;
1469     //
1470     // command to be sent to gnuplot
1471     //
1472     if (nplots > 0 && two_dim == true)
1473         cmdstr << "replot ";
1474     else
1475         cmdstr << "plot ";

```



```

1476
1477     cmdstr << "\"" << filename << "\" using " << column_x << ":" << column_y;
1478
1479     if (title == "")
1480         cmdstr << " notitle ";
1481     else
1482         cmdstr << " title \"" << title << "\" ";
1483
1484     if(smooth == "")
1485         cmdstr << "with " << pstyle;
1486     else
1487         cmdstr << "smooth " << smooth;
1488
1489     //
1490     // Do the actual plot
1491     //
1492     cmd(cmdstr.str());
1493
1494     return *this;
1495 }

```

#### 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().

```

1507 {
1508     //
1509     // check if file exists
1510     //
1511     file_available(filename);
1512
1513     std::ostringstream cmdstr;
1514     //
1515     // command to be sent to gnuplot
1516     //
1517     if (nplots > 0 && two_dim == true)
1518         cmdstr << "replot ";
1519     else
1520         cmdstr << "plot ";
1521
1522     cmdstr << "\"" << filename << "\" using "
1523         << column_x << ":" << column_y << ":" << column_dy
1524         << " with errorbars ";
1525
1526     if (title == "")
1527         cmdstr << " notitle ";
1528     else
1529         cmdstr << " title \"" << title << "\" ";
1530
1531     //
1532     // Do the actual plot
1533     //
1534     cmd(cmdstr.str());
1535
1536     return *this;
1537 }

```

### 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().

```

1549 {
1550     //
1551     // check if file exists
1552     //
1553     file_available(filename);
1554
1555     std::ostringstream cmdstr;
1556     //
1557     // command to be sent to gnuplot
1558     //
1559     if (nplots > 0 && two_dim == false)
1560         cmdstr << "replot ";
1561     else
1562         cmdstr << "splot ";
1563
1564     cmdstr << "\"" << filename << "\" using \" << column_x << \":\" << column_y
1565         << \":\" << column_z;
1566
1567     if (title == "")
1568         cmdstr << " notitle with \" << pstyle;
1569     else
1570         cmdstr << " title \"" << title << "\" with \" << pstyle;
1571
1572     //
1573     // Do the actual plot
1574     //
1575     cmd(cmdstr.str());
1576
1577     return *this;
1578 }
```

### 3.1.3.16 void Gnuplot::remove\_tmpfiles ()

deletes temporary files

Definition at line 1948 of file gnuplot\_i.hpp.

```

1948     {
1949     if ((tmpfile_list).size() > 0)
1950     {
1951         for (unsigned int i = 0; i < tmpfile_list.size(); i++)
1952             remove( tmpfile_list[i].c_str() );
1953
1954         Gnuplot::tmpfile_num -= tmpfile_list.size();
1955     }
1956 }
```

### 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().

```

1078 {
1079     cmd("set terminal postscript color");
1080
1081     std::ostringstream cmdstr;
1082     cmdstr << "set output \"" << filename << ".ps\"";
1083     cmd(cmdstr.str());
1084
1085     return *this;
1086 }
```

### 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().

```

1297 {
1298     std::ostringstream cmdstr;
1299
1300     cmdstr << "set cbrange[" << iFrom << ":" << iTo << "]";
1301     cmd(cmdstr.str());
1302
1303     return *this;
1304 }
```

### 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().

```

1191 {
1192     if (position.find("base") == std::string::npos &&
1193         position.find("surface") == std::string::npos &&
1194         position.find("both") == std::string::npos )
1195     {
1196         cmd("set contour base");
1197     }
1198     else
1199     {
1200         cmd("set contour " + position);
1201     }
1202 }
```

```

1203     return *this;
1204 }

```

### 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.

```

857 {
858
859     std::string tmp = path + "/" + Gnuplot::m_sGNUPlotFileName;
860
861
862     #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__TOS_WIN__)
863         if ( Gnuplot::file_exists(tmp,0) ) // check existence
864     #elif defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
865         if ( Gnuplot::file_exists(tmp,1) ) // check existence and execution permission
866     #endif
867     {
868         Gnuplot::m_sGNUPlotPath = path;
869         return true;
870     }
871     else
872     {
873         Gnuplot::m_sGNUPlotPath.clear();
874         return false;
875     }
876 }

```

### 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().

```

266 {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:**

&lt;– reference to the gnuplot object

Definition at line 302 of file gnuplot\_i.hpp.

References cmd().

Referenced by main().

```
302 {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().

```
1176 {
1177     std::ostringstream cmdstr;
1178     cmdstr << "set isosamples " << isolines;
1179     cmd(cmdstr.str());
1180
1181     return *this;
1182 }
```

**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().

```
1093 {
1094     std::ostringstream cmdstr;
1095     cmdstr << "set key " << position;
1096
1097     cmd(cmdstr.str());
1098
1099     return *this;
1100 }
```

**3.1.3.28 Gnuplot & Gnuplot::set\_multiplot () [inline]**

set the mulitplot mode

**Parameters:**

—

**Returns:**

&lt;— reference to the gnuplot object

Definition at line 277 of file gnuplot\_i.hpp.

References cmd().

```
277 {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().

```
1149 {
1150     std::ostringstream cmdstr;
1151     cmdstr << "set pointsize " << pointsize;
1152     cmd(cmdstr.str());
1153
1154     return *this;
1155 }
```

**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().

```
1162 {
1163     std::ostringstream cmdstr;
1164     cmdstr << "set samples " << samples;
1165     cmd(cmdstr.str());
1166
1167     return *this;
1168 }
```

**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 effect on data plotting)

Definition at line 1041 of file gnuplot\_i.hpp.

Referenced by main().

```

1042 {
1043     if (stylestr.find("unique")      == std::string::npos  &&
1044         stylestr.find("frequency")  == std::string::npos  &&
1045         stylestr.find("csplines")    == std::string::npos  &&
1046         stylestr.find("acsplines")   == std::string::npos  &&
1047         stylestr.find("bezier")      == std::string::npos  &&
1048         stylestr.find("sbezier")     == std::string::npos  )
1049     {
1050         smooth = "";
1051     }
1052     else
1053     {
1054         smooth = stylestr;
1055     }
1056
1057     return *this;
1058 }

```

### 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().

```

996 {
997     if (stylestr.find("lines")      == std::string::npos  &&
998         stylestr.find("points")     == std::string::npos  &&
999         stylestr.find("linespoints") == std::string::npos  &&
1000         stylestr.find("impulses")   == std::string::npos  &&
1001         stylestr.find("dots")        == std::string::npos  &&
1002         stylestr.find("steps")       == std::string::npos  &&
1003         stylestr.find("fsteps")      == std::string::npos  &&
1004         stylestr.find("histeps")     == std::string::npos  &&
1005         stylestr.find("boxes")       == std::string::npos  && // 1-4 columns of data are required
1006         stylestr.find("filledcurves") == std::string::npos  &&
1007         stylestr.find("histograms")  == std::string::npos  ) //only for one data column
1008     //     stylestr.find("labels")      == std::string::npos  && // 3 columns of data are required
1009     //     stylestr.find("xerrorbars")  == std::string::npos  && // 3-4 columns of data are required
1010     //     stylestr.find("xerrorlines") == std::string::npos  && // 3-4 columns of data are required
1011     //     stylestr.find("errorbars")   == std::string::npos  && // 3-4 columns of data are required
1012     //     stylestr.find("errorlines")  == std::string::npos  && // 3-4 columns of data are required
1013     //     stylestr.find("yerrorbars")  == std::string::npos  && // 3-4 columns of data are required
1014     //     stylestr.find("yerrorlines") == std::string::npos  && // 3-4 columns of data are required
1015     //     stylestr.find("boxerrorbars") == std::string::npos  && // 3-5 columns of data are required
1016     //     stylestr.find("xyerrorbars") == std::string::npos  && // 4,6,7 columns of data are required
1017     //     stylestr.find("xyerrorlines") == std::string::npos  && // 4,6,7 columns of data are required
1018     //     stylestr.find("boxxyerrorbars") == std::string::npos  && // 4,6,7 columns of data are required
1019     //     stylestr.find("financebars") == std::string::npos  && // 5 columns of data are required
1020     //     stylestr.find("candlesticks") == std::string::npos  && // 5 columns of data are required
1021     //     stylestr.find("vectors")     == std::string::npos  &&
1022     //     stylestr.find("image")       == std::string::npos  &&
1023     //     stylestr.find("rgbimage")    == std::string::npos  &&
1024     //     stylestr.find("pm3d")       == std::string::npos  )
1025     {
1026         pstyle = std::string("points");
1027     }
1028     else
1029     {
1030         pstyle = stylestr;
1031     }
1032 }

```



```
1033     return *this;
1034 }
```

### 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().

```
332 {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.

```
885 {
886 #if defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
887     if (type.find("x11") != std::string::npos && getenv("DISPLAY") == NULL)
888     {
889         throw GnuplotException("Can't find DISPLAY variable");
890     }
891 #endif
892
893
894     Gnuplot::terminal_std = type;
895     return;
896 }
```

### 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().

```

367     {
368         std::string cmdstr;
369         cmdstr = "set title \"";
370         cmdstr+=title;
371         cmdstr+="\"";
372         *this<<cmdstr;
373         return *this;
374     }

```

**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().

```

409 {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().

```

1212 {
1213     std::ostringstream cmdstr;
1214
1215     cmdstr << "set xlabel \"" << label << "\"";
1216     cmd(cmdstr.str());
1217
1218     return *this;
1219 }

```

### 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()`.

```
1107 {  
1108     std::ostringstream cmdstr;  
1109  
1110     cmdstr << "set logscale x " << base;  
1111     cmd(cmdstr.str());  
1112  
1113     return *this;  
1114 }
```

### 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()`.

```
1254 {  
1255     std::ostringstream cmdstr;  
1256  
1257     cmdstr << "set xrange[" << iFrom << ":" << iTo << "];"  
1258     cmd(cmdstr.str());  
1259  
1260     return *this;  
1261 }
```

### 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()`.

```
418 {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().

```
1225 {  
1226     std::ostringstream cmdstr;  
1227  
1228     cmdstr << "set ylabel \"" << label << "\"";  
1229     cmd(cmdstr.str());  
1230  
1231     return *this;  
1232 }
```

### 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().

```
1121 {  
1122     std::ostringstream cmdstr;  
1123  
1124     cmdstr << "set logscale y " << base;  
1125     cmd(cmdstr.str());  
1126  
1127     return *this;  
1128 }
```

### 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().

```
1268 {  
1269     std::ostringstream cmdstr;  
1270  
1271     cmdstr << "set yrange[" << iFrom << ":" << iTo << "];"  
1272     cmd(cmdstr.str());  
1273  
1274     return *this;  
1275 }
```

### 3.1.3.44 Gnuplot & Gnuplot::set\_zautoscale () [inline]

autoscale axis (set by default) of zaxis

**Parameters:**

—

**Returns:**

&lt;— reference to the gnuplot object

Definition at line 427 of file gnuplot\_i.hpp.

References cmd().

```
427 {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().

```
1238 {
1239     std::ostringstream cmdstr;
1240
1241     cmdstr << "set zlabel \"" << label << "\"";
1242     cmd(cmdstr.str());
1243
1244     return *this;
1245 }
```

**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().

```
1135 {
1136     std::ostringstream cmdstr;
1137
1138     cmdstr << "set logscale z " << base;
1139     cmd(cmdstr.str());
1140
1141     return *this;
1142 }
```

**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().

```

1282 {
1283     std::ostringstream cmdstr;
1284
1285     cmdstr << "set zrange[" << iFrom << ":" << iTo << "];";
1286     cmd(cmdstr.str());
1287
1288     return *this;
1289 }

```

### 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().

```

1066 {
1067     cmd("set output");
1068     cmd("set terminal " + Gnuplot::terminal_std);
1069
1070     return *this;
1071 }

```

### 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().

```

323 {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().

```

268 {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(), Gnuplot::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().

```

28 {
29     // if path-variable for gnuplot is not set, do it with:
30     // Gnuplot::set_GNUPlotPath("C:/program files/gnuplot/bin/");
31
32     // set a special standard terminal for showonscreen (normally not needed),
33     // e.g. Mac users who want to use x11 instead of aqua terminal:
34     // Gnuplot::set_terminal_std("x11");
35
36     cout << "*** example of gnuplot control through C++ ***" << endl << endl;
37
38     //
39     // Using the GnuplotException class
40     //
41     try
42     {
43         Gnuplot gl("lines");
44
45         //
46         // Slopes
47         //
48         cout << "*** plotting slopes" << endl;
49         gl.set_title("Slopes\\nNew Line");
50
51         cout << "y = x" << endl;
52         gl.plot_slope(1.0,0.0,"y=x");
53
54         cout << "y = 2*x" << endl;
55         gl.plot_slope(2.0,0.0,"y=2x");
56
57         cout << "y = -x" << endl;
58         gl.plot_slope(-1.0,0.0,"y=-x");
59         gl.unset_title();
60
61         //
62         // Equations
63         //
64         gl.reset_plot();
65         cout << endl << endl << "*** various equations" << endl;
66
67         cout << "y = sin(x)" << endl;
68         gl.plot_equation("sin(x)","sine");
69
70         cout << "y = log(x)" << endl;
71         gl.plot_equation("log(x)","logarithm");
72
73         cout << "y = sin(x) * cos(2*x)" << endl;
74         gl.plot_equation("sin(x)*cos(2*x)","sine product");
75
76         //
77         // Styles
78         //
79         gl.reset_plot();
80         cout << endl << endl << "*** showing styles" << endl;
81

```

```

82     cout << "sine in points" << endl;
83     gl.set_pointsize(0.8).set_style("points");
84     gl.plot_equation("sin(x)", "points");
85
86     cout << "sine in impulses" << endl;
87     gl.set_style("impulses");
88     gl.plot_equation("sin(x)", "impulses");
89
90     cout << "sine in steps" << endl;
91     gl.set_style("steps");
92     gl.plot_equation("sin(x)", "steps");
93
94     //
95     // Save to ps
96     //
97     gl.reset_all();
98     cout << endl << endl << "*** save to ps " << endl;
99
100    cout << "y = sin(x) saved to test_output.ps in working directory" << endl;
101    gl.savetops("test_output");
102    gl.set_style("lines").set_samples(300).set_xrange(0,5);
103    gl.plot_equation("sin(12*x)*exp(-x)").plot_equation("exp(-x)");
104
105    gl.showonscreen(); // window output
106
107
108    //
109    // User defined 1d, 2d and 3d point sets
110    //
111    std::vector<double> x, y, y2, dy, z;
112
113    for (int i = 0; i < NPOINTS; i++) // fill double arrays x, y, z
114    {
115        x.push_back((double)i); // x[i] = i
116        y.push_back((double)i * (double)i); // y[i] = i^2
117        z.push_back( x[i]*y[i] ); // z[i] = x[i]*y[i] = i^3
118        dy.push_back((double)i * (double)i / (double) 10); // dy[i] = i^2 / 10
119    }
120    y2.push_back(0.00); y2.push_back(0.78); y2.push_back(0.97); y2.push_back(0.43);
121    y2.push_back(-0.44); y2.push_back(-0.98); y2.push_back(-0.77); y2.push_back(0.02);
122
123
124    gl.reset_all();
125    cout << endl << endl << "*** user-defined lists of doubles" << endl;
126    gl.set_style("impulses").plot_x(y, "user-defined doubles");
127
128    gl.reset_plot();
129    cout << endl << endl << "*** user-defined lists of points (x,y)" << endl;
130    gl.set_grid();
131    gl.set_style("points").plot_xy(x,y, "user-defined points 2d");
132
133    gl.reset_plot();
134    cout << endl << endl << "*** user-defined lists of points (x,y,z)" << endl;
135    gl.unset_grid();
136    gl.plot_xyz(x,y,z, "user-defined points 3d");
137
138    gl.reset_plot();
139    cout << endl << endl << "*** user-defined lists of points (x,y,dy)" << endl;
140    gl.plot_xy_err(x,y,dy, "user-defined points 2d with errorbars");
141
142
143    //
144    // Multiple output screens
145    //
146    cout << endl << endl;
147    cout << "*** multiple output windows" << endl;
148

```

```

149     g1.reset_plot();
150     g1.set_style("lines");
151     cout << "window 1: sin(x)" << endl;
152     g1.set_grid().set_samples(600).set_xrange(0,300);
153     g1.plot_equation("sin(x)+sin(x*1.1)");
154
155     g1.set_xautoscale().replot();
156
157     Gnuplot g2;
158     cout << "window 2: user defined points" << endl;
159     g2.plot_x(y2, "points");
160     g2.set_smooth().plot_x(y2, "cspline");
161     g2.set_smooth("bezier").plot_x(y2, "bezier");
162     g2.unset_smooth();
163
164     Gnuplot g3("lines");
165     cout << "window 3: log(x)/x" << endl;
166     g3.set_grid();
167     g3.plot_equation("log(x)/x", "log(x)/x");
168
169     Gnuplot g4("lines");
170     cout << "window 4: splot x*x+y*y" << endl;
171     g4.set_zrange(0,100);
172     g4.set_xlabel("x-axis").set_ylabel("y-axis").set_zlabel("z-axis");
173     g4.plot_equation3d("x*x+y*y");
174
175     Gnuplot g5("lines");
176     cout << "window 5: splot with hidden3d" << endl;
177     g5.set_isosamples(25).set_hidden3d();
178     g5.plot_equation3d("x*y*y");
179
180     Gnuplot g6("lines");
181     cout << "window 6: splot with contour" << endl;
182     g6.set_isosamples(60).set_contour();
183     g6.unset_surface().plot_equation3d("sin(x)*sin(y)+4");
184
185     g6.set_surface().replot();
186
187     Gnuplot g7("lines");
188     cout << "window 7: set_samples" << endl;
189     g7.set_xrange(-30,20).set_samples(40);
190     g7.plot_equation("besj0(x)*0.12e1").plot_equation("(x**besj0(x))-2.5");
191
192     g7.set_samples(400).replot();
193
194     Gnuplot g8("filledcurves");
195     cout << "window 8: filledcurves" << endl;
196     g8.set_legend("outside right top").set_xrange(-5,5);
197     g8.plot_equation("x*x").plot_equation("-x*x+4");
198
199     //
200     // Plot an image
201     //
202     Gnuplot g9;
203     cout << "window 9: plot_image" << endl;
204     const int iWidth = 255;
205     const int iHeight = 255;
206     g9.set_xrange(0,iWidth).set_yrange(0,iHeight).set_cbrange(0,255);
207     g9.cmd("set palette gray");
208     unsigned char ucPicBuf[iWidth*iHeight];
209     // generate a greyscale image
210     for(int iIndex = 0; iIndex < iHeight*iWidth; iIndex++)
211     {
212         ucPicBuf[iIndex] = iIndex%255;
213     }
214     g9.plot_image(ucPicBuf,iWidth,iHeight,"greyscale");
215

```

```
216         g9.set_pointsize(0.6).unset_legend().plot_slope(0.8,20);
217
218         //
219         // manual control
220         //
221         Gnuplot g10;
222         cout << "window 10: manual control" << endl;
223         g10.cmd("set samples 400").cmd("plot abs(x)/2"); // either with cmd()
224         g10 << "replot sqrt(x)" << "replot sqrt(-x)"; // or with <<
225
226         wait_for_key();
227
228     }
229     catch (GnuplotException ge)
230     {
231         cout << ge.what() << endl;
232     }
233
234
235     cout << endl << "*** end of gnuplot example" << endl;
236
237     return 0;
238 }
```

#### 4.1.2.2 void wait\_for\_key ()

Definition at line 242 of file example.cc.

Referenced by main().

```
243 {
244 #if defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__TOS_WIN__) // every keypress
245     cout << endl << "Press any key to continue..." << endl;
246
247     FlushConsoleInputBuffer(GetStdHandle(STD_INPUT_HANDLE));
248     _getch();
249 #elif defined(unix) || defined(__unix) || defined(__unix__) || defined(__APPLE__)
250     cout << endl << "Press ENTER to continue..." << endl;
251
252     std::cin.clear();
253     std::cin.ignore(std::cin.rdbuf()->in_avail());
254     std::cin.get();
255 #endif
256     return;
257 }
```

## 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.

```
908 {
909     const std::string::size_type len = in.length();
910     std::string::size_type i = 0;
911
912     while ( i < len )
913     {
914         // eat leading whitespace
915         i = in.find_first_not_of (delimiters, i);
916
917         if (i == std::string::npos)
918             return; // nothing left but white space
919
920         // find the end of the token
921         std::string::size_type j = in.find_first_of (delimiters, i);
922
923         // push token
924         if (j == std::string::npos)
925         {
926             container.push_back (in.substr(i));
927             return;
```



---

```
928         }
929         else
930             container.push_back (in.substr(i, j-i));
931
932         // set up for next loop
933         i = j + 1;
934     }
935
936     return;
937 }
```

# Index

- ~Gnuplot
  - Gnuplot, 10
- cmd
  - Gnuplot, 10
- example.cc, 37
  - main, 37
  - NPOINTS, 37
  - SLEEP\_LGTH, 37
  - wait\_for\_key, 41
- Gnuplot, 5
  - ~Gnuplot, 10
  - cmd, 10
  - Gnuplot, 8, 9
  - is\_valid, 11
  - operator<<, 11
  - plot\_equation, 12
  - plot\_equation3d, 12
  - plot\_image, 13
  - plot\_slope, 14
  - plot\_x, 14
  - plot\_xy, 15
  - plot\_xy\_err, 16
  - plot\_xyz, 17
  - plotfile\_x, 17
  - plotfile\_xy, 18
  - plotfile\_xy\_err, 19
  - plotfile\_xyz, 19
  - remove\_tmpfiles, 20
  - replot, 20
  - reset\_all, 21
  - reset\_plot, 21
  - savetops, 21
  - set\_cbrange, 22
  - set\_contour, 22
  - set\_GNUPlotPath, 23
  - set\_grid, 23
  - set\_hidden3d, 23
  - set\_isosamples, 24
  - set\_legend, 24
  - set\_multiplot, 24
  - set\_pointsize, 25
  - set\_samples, 25
  - set\_smooth, 25
  - set\_style, 26
  - set\_surface, 27
  - set\_terminal\_std, 27
  - set\_title, 27
  - set\_xautoscale, 28
  - set\_xlabel, 28
  - set\_xlogscale, 28
  - set\_xrange, 29
  - set\_yautoscale, 29
  - set\_ylabel, 29
  - set\_ylogscale, 30
  - set\_yrange, 30
  - set\_zautoscale, 30
  - set\_zlabel, 31
  - set\_zlogscale, 31
  - set\_zrange, 31
  - showonscreen, 32
  - unset\_contour, 32
  - unset\_grid, 32
  - unset\_hidden3d, 32
  - unset\_legend, 33
  - unset\_multiplot, 33
  - unset\_smooth, 33
  - unset\_surface, 34
  - unset\_title, 34
  - unset\_xlogscale, 34
  - unset\_ylogscale, 35
  - unset\_zlogscale, 35
- gnuplot\_i.hpp, 42
  - stringtok, 42
- GnuplotException, 36
  - GnuplotException, 36
- is\_valid
  - Gnuplot, 11
- main
  - example.cc, 37
- NPOINTS
  - example.cc, 37
- operator<<
  - Gnuplot, 11

- plot\_equation
  - Gnuplot, 12
- plot\_equation3d
  - Gnuplot, 12
- plot\_image
  - Gnuplot, 13
- plot\_slope
  - Gnuplot, 14
- plot\_x
  - Gnuplot, 14
- plot\_xy
  - Gnuplot, 15
- plot\_xy\_err
  - Gnuplot, 16
- plot\_xyz
  - Gnuplot, 17
- plotfile\_x
  - Gnuplot, 17
- plotfile\_xy
  - Gnuplot, 18
- plotfile\_xy\_err
  - Gnuplot, 19
- plotfile\_xyz
  - Gnuplot, 19
- remove\_tmpfiles
  - Gnuplot, 20
- replot
  - Gnuplot, 20
- reset\_all
  - Gnuplot, 21
- reset\_plot
  - Gnuplot, 21
- savetops
  - Gnuplot, 21
- set\_cbrange
  - Gnuplot, 22
- set\_contour
  - Gnuplot, 22
- set\_GNUPlotPath
  - Gnuplot, 23
- set\_grid
  - Gnuplot, 23
- set\_hidden3d
  - Gnuplot, 23
- set\_isosamples
  - Gnuplot, 24
- set\_legend
  - Gnuplot, 24
- set\_multiplot
  - Gnuplot, 24
- set\_pointsize
  - Gnuplot, 25
- set\_samples
  - Gnuplot, 25
- set\_smooth
  - Gnuplot, 25
- set\_style
  - Gnuplot, 26
- set\_surface
  - Gnuplot, 27
- set\_terminal\_std
  - Gnuplot, 27
- set\_title
  - Gnuplot, 27
- set\_xautoscale
  - Gnuplot, 28
- set\_xlabel
  - Gnuplot, 28
- set\_xlogscale
  - Gnuplot, 28
- set\_xrange
  - Gnuplot, 29
- set\_yautoscale
  - Gnuplot, 29
- set\_ylabel
  - Gnuplot, 29
- set\_ylogscale
  - Gnuplot, 30
- set\_yrange
  - Gnuplot, 30
- set\_zautoscale
  - Gnuplot, 30
- set\_zlabel
  - Gnuplot, 31
- set\_zlogscale
  - Gnuplot, 31
- set\_zrange
  - Gnuplot, 31
- showonscreen
  - Gnuplot, 32
- SLEEP\_LGTH
  - example.cc, 37
- stringtok
  - gnuplot.i.hpp, 42
- unset\_contour
  - Gnuplot, 32
- unset\_grid
  - Gnuplot, 32
- unset\_hidden3d
  - Gnuplot, 32
- unset\_legend
  - Gnuplot, 33
- unset\_multiplot
  - Gnuplot, 33
- unset\_smooth

- Gnuplot, 33
- unset\_surface
  - Gnuplot, 34
- unset\_title
  - Gnuplot, 34
- unset\_xlogscale
  - Gnuplot, 34
- unset\_ylogscale
  - Gnuplot, 35
- unset\_zlogscale
  - Gnuplot, 35
- wait\_for\_key
  - example.cc, 41