GraceGTK guide

Software version: `gracegtk-0.4.4`

October 9, 2011

Abstract

This document explains the usage of `GraceGTK` a 2D plotting tool for numerical data.

Contents

List of Tables

Note that most of this document is borrowed to the Grace User Guide (`grace-5.1.22`).

1 Introduction

The name *Grace* is used when the text is valid for `grace-5.1.22` as well as for `GraceGTK` and `GraceGTK` when relevant only for this later.

1.1 What is GraceGTK?

Grace and `GraceGTK` are WYSIWYG tools to make two-dimensional plots of numerical data.
`GraceGTK` is an evolution of `grace-5.1.22` using the GTK Application Programming Interface (API) instead of Motif API. Its goal is to gives a tool that makes easy to grab numerical data, perform some post-processing (Fourier transform, filtering, *etc*) and obtain a drawing able to make a good figure in an article.

`GraceGTK` is derived from `grace-5.1.22` , developed under GPL copyright by a team of volunteers under the coordination of Evgeny Stambulchik. You can get the newest information about Grace and download the latest version[1] at the *Grace home page* `http://plasma-gate.weizmann.ac.il/Grace/`

Grace itself is derived from Xmgr (a.k.a. ACE/gr), originally written by Paul Turner. Paul still maintains and develops a non-public version of Xmgr for internal use.

`GraceGTK` home is `http://gracegtk.sourceforge.net`

The natural operating system for GTK programs is Unix/Linux, but the use of GTK allows to build a standalone distribution for MS Windows.

1.2 Copyright statement

For Grace:

For some libraries required to build `GraceGTK` (which are therefore even included in a suitable version) there may be different Copyright/License statements. Though their License may by chance match the one used for Grace, the Grace Copyright holders can not influence or change them.
N.B. The status of Cephes seems not clear an this library is used by `grace-5.1.22` , but not by `grace-5.99` .

[1] Of course `grace-5.99` is newer than `grace-5.1.22` , but had never gain the status of `grace-6` and remains with a development status. The differences between these two codes are important.

Package	License
cephes	?
T1lib	LGPL
GTK+	GPL
libundo	GPL

Table 1: Licenses

2 Installation guide

2.1 Installing from sources

Requirements.

You need an ANSI C compiler (`gcc` is just fine), and a development version of GTK (`GTK+-2.12` or above[2]. GTK can be found at http://www.gtk.org.
Contour level curves need a FORTRAN compiler: `gfortran` is just fine.
The FORTRAN compiler is detected by `configure` through the environment variable `FC`, e.g. if `gfortran` is the name of the compiler command:

```
export FC=gfortran
```

If you want to compile your own changes to some parts of `GraceGTK` , you will need `bison` (the parser generator) and the GNU `autotools`.

Output drivers and extra libraries.

Since version 0.3.3, `GraceGTK` have output drivers based on the Cairo library. Cairo is integrated in the GTK distribution, thus no extra libraries are needed to use these drivers. Currently, PDF, PNG and SVG Cairo drivers are included and are necessary to render correctly color mapped sets (`xycmap` sets), but these drivers are not at all optimized[3] and in some cases you can prefer original `grace-5.1.22` drivers.
Some features will be available only if additional libraries are installed:

- The JPEG backend needs the IJG's *JPEG library* (`ftp://ftp.uu.net/graphics/jpeg/`), version 6.x.
- The `grace-5.1.22` PNG backend needs the *libpng* library (`http://www.libpng.org/pub/png/libpng.html`version 0.96 or above).
- The `grace-5.1.22` PDFlib driver requires the PDFlib library of Thomas Merz to be installed, which is available *here* `http://www.pdflib.com/`, version 4.0.3 or above.
  An alternative is to convert Postscript files into PDF with `epstopdf` or another conversion tool.
- The `FFTW` library (Fast Fourier Transform) allows to perform Fourier transformation efficiently. If this package is not found, the Fourier transforms are computed by a conventional DFT, i.e. very inefficiently for large sets.
  `GraceGTK` should work with `fftw-2` as well as with `fftw-3`. For more information on this package, see the *FFTW Home page* `http://www.fftw.org`.
  It is highly recommended if you want to compute Fourier transforms. If you want a fine tuning, see details in section 9.5.
- The wavelet transforms are based on GNU Scientific Library (GSL). If the library is not installed, a bundled small part of the library is used.
  The library home is `http://www.gnu.org/software/gsl/`.
- The NetCDF input driver existing in `grace-5.1.22` is not implemented in `GraceGTK` .

Windows XP and Vista.

You must have MinGW, MSYS and GTK+ installed. Each release of `GraceGTK` is checked hastily to run under these OS, but Linux is the natural environment for `GraceGTK` , thus obtaining a working program may need some hacking. You can have to hack autotools files to get a working configure. and some features (e.g. piping) does not works under MS Windows.
No testing has been done for the Cygwin environment.

Compilation.

```
cd gracegtk-x.y.z
./configure
```

[2] It should work also with `GTK+-2.10`, but without the tooltips.
[3] e.g. strings are always written on a pixel basis using the T1lib library and don't define device fonts.

should produce a working `Makefile`.
`make`
should compile the various elements. If something goes wrong, try to see if the problem has been described already in the `FAQ` (in the `doc` directory).
If the `./configure` does not work for you, try to soft link `ac-tools/configure.in` in the main directory and to run `autoconf` to regenerate the configure file.
In a MS Window environment, the `grace_np` directory, is not compiled.

Configuration.
In order to get a working executable of `GraceGTK` *without install*, i.e. to be able to run the program from `gracegtk-`*x.y.z* directory, you need to `export GRACEGTK_HOME=.` environment variable. For other variables, e.g. to launch the help viewer from the `help` menu see section 3.3.1.

Testing.
`make tests`
will run for you various examples. For each example, you can play with the mouse and the menus to change the graphs displayed. If you prefer to run selected examples, just type
`export GRACEGTK_HOME=.`
`./src/ggrace`
and select the examples with `Examples` menus.

Installation.
`make install`
install by default in `/usr/local/gracegtk` (of course you need to have the right permissions). Then you must include in your profile file

```
export GRACEGTK_HOME=/usr/local/gracegtk
export PATH=$PATH:/usr/local/gracegtk/bin
```

Upon start-up, `GraceGTK` loads its init file `gracerc` See section 3.3.2 for details.
You can also customize some environment variables, mainly for GUI size and help viewer configuration: see section 3.3.1.

Removing installation or installing a new version.
Just remove the installation directory (Unix/Linux default: `/usr/local/gracegtk`).
If you want to install a new version and keep the old one, you can simply rename the directory of the old one. Accessing the old one will depend upon local variables setting.

2.2 Binary installation

2.2.1 MS Windows

Windows© binaries (32 bits) are available on `http://gracegtk.sourceforge.net` . Simply unzip the file at the place of your choice and execute `ggrace.exe` in the `bin` directory.
Some features are not yet available under MS Windows.

- Printing: write into a file (e.g. in PDF) and use a reader (e.g. Acrobat Reader) as previewer/printer interface.
- Help: on line help will not work: use a viewer to read `doc/GraceGTK.pdf`
- Piping is not available.

2.2.2 Unix/Linux

No binaries for these OS are distributed by the `GraceGTK` home site.

3 Getting started

For a jump-in start, you can browse the demos (`"Examples"` menu tree). These are ordinary Grace projects, so you can play with them and modify them. Also, read the `grace-5.1.22` *Tutorial* `Tutorial.html`.

O.k. Here's a VERY quick introduction:

1. Start the GUI version: `ggrace` (return).
2. Select/check the output medium and canvas size in `File/Print setup`.

3. Load your data clicking the `Import` toolbar button.
4. Open the Explorer by clicking `TreeView` button.
5. Adjust what you want by selecting items in the drawing tree and using the contextual right pane menus. Default behaviour[4] is that almost all changes in the pane are reflected by redrawing immediately the canvas. In few cases, you have to acknowledge the changes with the "`Apply`" button.
6. Data can be manipulated in `Data/Transformations`.
   As an example, to shift a data set by 20 to the left: open '`Evaluate Expression`', select the source graph and set on the left, the destination graph on the right, and say Formula: $x = x-20$. If no set is selected on the right, a new set is created, if a set is selected, it is overwritten by the new one.
   As you'll probably notice, Grace can do MUCH more than that. Explore at your leisure.
7. When you like your plot, select File/Print. That's it!

Each time it is possible without to much complexity, the original `grace-5` menus are preserved and should work as described in the `grace-5` documentation but the `Menubar/Plot` menus now open the `Explorer` menus instead of old ones.

From the end user point of view, the main novelties are the new toolbar buttons and the Explorer/TreeView window dialog. Try the Examples `GraceGTK` menu entries to discover some of them.

`GraceGTK` evolution is to add features copied from `grace-5.99` , the beta version of grace-6. Thus, you can also have a look on `grace-5.99` and try it.

One of the main novelty in `GraceGTK` is the development of an easy management of "geometric" objects: arcs, boxes, lines, polylines, strings and compounds, inspired by the well known free program `xfig` [5].

3.1 General concepts

3.1.1 `GraceGTK` command files

These files contains instructions for the command interpreter. The language used is in principle upward compatible with `grace-5.1.22` command files.
A file containing all the information necessary to restore a plot is called a *project file*: the default extension of the file is `.agr` and the commands lines begins by a `@`
If it is a "style sheet" containing only the parameters needed to draw similar graphs with other compatible data, it is called a *parameter file*: the default extension of the file is `.par` and for each line the leading `@` is ommitted.
See section 8 for details on `GraceGTK` language.

3.1.2 Input data file

`GraceGTK` reads ASCII text files containing space and comma separated columns of data. The data fields can be either numeric (Fortran 'd' and 'D' exponent markers are also supported) or alphanumeric (with or without quotes).
Several calendar date formats are recognized automatically and you can specify your own reference for numeric date formats.
Blank lines and lines beginning with `#` are ignored.
The end of a dataset is marked by a line with a `&`
You can mix data and commands in a command file: see section 8 for details on `GraceGTK` language.

3.1.3 Datasets and sets

A *dataset* (often called simply a *set* in the following) is a collection of points with x and y coordinates, up to four optional data values (which, depending on the set type, can be displayed as error bars or like) and one optional character string.
More generally, a *set* is a way of representing datasets. It consists of a dataset plus a collection of parameters describing the visual appearance of the data (like color, line dash pattern etc).
Note that each set has is own dataset (e.g. if several curves have the same abscissas, the arrays for x are duplicated).

[4] This "instant update" feature can be disabled in the `Edit/Preferences` menu.
[5] `GraceGTK` is not intended to be a replacement for `xfig` and only some features of `xfig` are included in `GraceGTK` .

3.1.4 Graphs and sets

A *graph* consists of (every element is optional): a graph frame, axes, a title and a subtitle, a number of sets and additional annotative objects (text strings, lines, boxes and arcs of ellipses).

The *graph type* can be any of:

- XY Graph (curves, vector or color maps)
- XY Chart
  The idea of "XY Chart" is to plot bars (or symbols in general) of several sets side by side, assuming the abscissas of all the sets are the same (or subsets of the longest set).
- Polar Graph (see section 4.6.1).
- Fixed Graph (i.e. X/Y ratio = 1)
- Pie chart
- XYHILO draw vertical bars between Hi and Low, with two horizontal ticks at Open and Close.

The *set type* list is in table 2. Not all set types, however, can be plotted on any graph type.

Set type	# of num. cols	Description
XY	2	An x, y scatter and/or line plot, plus (optionally) an annotated value
XYDX	3	Same as XY, but with error bars (either one- or two-sided) along x-axis
XYDY	3	Same as XYDX, but error bars are along y-axis
XYDXDX	4	Same as XYDX, but left and right error bars are defined separately
XYDYDY	4	Same as XYDXDX, but error bars are along y-axis
XYDXDY	4	Same as XY, but with X and Y error bars (either one- or two-sided)
XYDXDXDYDY	6	Same as XYDXDY, but left/right and upper/lower error bars are defined separately
BAR	2	Same as XY, but vertical bars are used instead of symbols
BARDY	3	Same as BAR, but with error bars (either one- or two-sided) along y-axis
BARDYDY	4	Same as BARDY, but lower and upper error bars are defined separately
XYHILO	5	columns are X/Hi/Low/Open/Close
XYZ	3	Same as XY; makes no sense unless the annotated value is z
XYR	3	x, y, Radius. Only allowed in Fixed graphs
XYSIZE	3	Same as XY, but symbol size is variable
XYCOLOR	3	x, y, color index (of the symbol fill)
XYCMAP	3	x, y, *color* interpolated in a dedicated color map.
XYCOLPAT	4	x, y, color index, pattern index (currently used for Pie charts only)
XYVMAP	4	Vector map (x, y, V_x, V_y)
XYBOXPLOT	6	Box plot (x, median, upper/lower limit, upper/lower whisker)

Table 2: Set types

The table 3 summarises the constraints.

3.1.5 World and viewport coordinates

There are two types of coordinates in Grace: the *world coordinates* and the *viewport coordinates*. Points of data sets are defined in the world coordinates and the tickmarks of the axes give the scales. The viewport coordinates correspond to the image of the plot drawn on the canvas. The transformation converting the world coordinates into the viewport ones is determined by both the graph type and the axis scaling.

3.1.6 Regions

Regions are sections of the graph defined by the interior or exterior of a polygon, or a half plane defined by a line. Regions are used to restrict data transformations to a geometric area occupied by region.

Set type	XY Graph	XY Chart	Fixed	Polar	Pie
XY	+	+	+	+	+
XYDX	+	-	+	-	-
XYDY	+	+	+	-	-
XYDXDX	+	-	+	-	-
XYDYDY	+	+	+	-	-
XYDXDY	+	-	+	-	-
XYDXDXDYDY	+	-	+	-	-
BAR	+	+	+	-	-
BARDY	+	+	-	-	-
BARDYDY	+	+	-	-	-
XYHILO	+	-	-	-	-
XYZ	+	-	+	+	-
XYR	-	-	+	-	-
XYSIZE	+	+	+	+	-
XYCOLOR	+	+	+	+	+
XYCMAP	+	-	+	-	-
XYCOLPAT	-	-	-	-	+
XYVMAP	+	-	+	-	-
XYBOXPLOT	+	-	-	-	-

Table 3: Graph/Set type connection

3.1.7 Geometric objects

Boxes, strings, lines and *arcs* can be defined independently of graphs, thus are called *geometric.*

Boxes are simple rectangles

Strings are used to display some text.
It is possible to use new lines (`\n`), Grace escape sequences, *etc* to beautify the output.

Lines are segments linking two points, with possibly arrows at the ends and a label displaying the length.

Polylines are segments connecting points and can be closed or opened, drawn as zigzags or smooth X-spline interpolated curves.

Arcs are arcs of ellipse. For an easy manipulation, it is possible to particularise as a complete ellipse or a circle defined by radius or bounding box.

3.1.8 Identification numbers

A *id-number* is attached to each graph, set or geometric object. The numbering is global for graphs and geometric objects and relative to a graph for the sets. The id-number is attributed by the program when the element is created or the project is red. If one element is killed, its *id-number* is not reused.
Id-numbers are used to build compounds.
Beware when you merge two different command files that the numbers may overlay and in that case, you have to manually change the numbers attached to different objects.

3.1.9 TreeView

The various elements of the drawing are linked by a tree structure that can be visualised in the `Explorer` windows. Each graph is the head of a branch of the tree containing axes, sets, legend box and optionally, geometric objects.

3.1.10 Layers

The drawing is done on one or several *layers*, with depths in the range `[1,100]`, starting to draw the deepest, then the upper ones. The default layer depth is 50 and when more than one layer is used, a column of check boxes appears at the right of the canvas, allowing to choose what layers are to be

displayed.
This allows a fine control of the foreground/background issue.

3.1.11 Compounds

Compounds are used to glue together in a single branch existing geometric objects, allowing to move, copy or delete all of them as a single block.
The geometric objects glued are pointed by the pair (*type-name, id-number*) and must be homogeneous with respect to their attachment and `loctyp` (viewport or world coordinates).

3.1.12 Real Time Input

Real Time Input refers to the ability Grace has to be fed in real time by an external program in a Unix/Linux environment. The Grace process spawned by the driver program is a full featured Grace process: the user can interact using the GUI at the same time the program sends data and commands. The process will adapt itself to the incoming data rate.
A less sophisticated alternative but working also in the MS Windows environment is the use of `Hot links` .

3.1.13 Devices

Grace allows the user to choose between several output devices to produce its graphics. The current list of supported devices is:

- GTK console (the canvas)
- PostScript (level 1 and level 2)
- EPS (encapsulated PostScript)
- MIF (Maker Interchange Format used by FrameMaker)
- SVG (Scalable Vector Graphics, a language for describing two-dimensional vector and mixed vector/raster graphics in XML)
- PDF (see section 2.1: `Extra libraries`),
- PNM (portable anymap file format),
- JPEG (see section 2.1: `Extra libraries`),
- PNG (see section 2.1: `Extra libraries`),

Note that Grace no longer supports GIF due to the copyright policy of Unisys. Grace can also be instructed to launch conversion programs automatically based on file name. As an example you can produce MIF (FrameMaker Interchange Format) or Java applets using pstoedit, or almost any image format using the netpbm suite (see the *FAQ* `FAQ.html`).

3.1.14 Magic path

In many cases, when Grace needs to access a file given with a relative `pathname`, it searches for the file along the following path:
`./pathname:./.grace/pathname:~/.grace/pathname:$GRACE_HOME/pathname`

3.2 Invocation

The name of the executable is `ggrace` (or `ggrace.exe` in a Windows environment). It can be followed by the options below[6].

[6] Note that all the options had not been systematically tested when `grace-5.1.22` had been changed in `GraceGTK` . Please make a report if you encounter problems with these options. A known bug is with the -pipe one: GTK waits the end of the run to display the curves, wiping intermediate results. A workaround is to use the `libgrace_np.a` library to interface your program with `GraceGTK` .

3.2.1 Command line options

-autoscale ***x|y|xy***
Override any parameter file settings

-barebones
Turn off all toolbars

-batch ***batch_file***
Execute batch_file on start up (i.e., after all other options have been processed and the UI initialized)

-block ***block_data***
Assume data file is block data

-bxy ***x:y:etc.***
Form a set from the current block data set using the current set type from columns given in the argument

-datehint ***iso|european|us|days|seconds|nohint***
Set the hint for dates analysis

-dpipe ***descriptor***
Read data from descriptor (anonymous pipe) on startup *(Unix/Linux only)*

-fixed ***width height***
Set canvas size fixed to $width * height$

-graph ***graph_number***
Set the current graph number

-graphtype ***graph_type***
Set the type of the current graph

-hardcopy
No interactive session, just print and quit

-hdevice ***hardcopy_device_name***
Set default hardcopy device

-install
Install private colormap

-legend ***load***
Turn the graph legend on

-log ***x|y|xy***
Set the axis scaling of the current graph to logarithmic

-maxpath ***length***
Set the maximal drawing path length

-noask
Assume the answer is yes to all requests - if the operation would overwrite a file, Grace will do so without prompting

-noinstall
Don't use private colormap

-noprint
In batch mode, do not print

-nosafe
Disable safe mode

-nosigcatch
Don't catch signals

-npipe ***file***
Read data from named pipe on startup *(Unix/Linux only)*

-nxy ***nxy_file***
Assume data file is in X Y1 Y2 Y3 ... format

-param ***parameter_file***
Load parameters from parameter_file to the current graph

-pexec ***parameter_string***
Interpret string as a parameter setting

-pipe
Read data from stdin on startup *(Unix/Linux only)*

-printfile
file Save print output to file

-remove
Remove data file after read

-results ***results_file***
Write results of some data manipulations to results_file

-rvideo
Exchange the color indices for black and white

-safe
Run in the safe mode (default) - no file system modifications are allowed through the batch language

-saveall ***save_file***
Save all graphs to save_file

-seed ***seed_value***
Integer seed for random number generator

-settype ***xy|xydx|...***
Set the type of the next data file

-source ***disk|pipe***
Source type of next data file *(Unix/Linux only)*

-timer ***delay***
Set allowed time slice for real time inputs to delay (in ms) *(Unix/Linux only)*

-timestamp
Add timestamp to plot

-version
Show the program version

-viewport ***xmin ymin xmax ymax***
Set the viewport for the current graph

-world ***xmin ymin xmax ymax***
Set the world coordinates for the current graph

-usage|-help
This message

3.3 Customization

3.3.1 Environment variables

- `GRACEGTK_HOME`
  Set the location of `GraceGTK` . This will be where help files, auxiliary programs, and examples are located. Default installation directory is `/usr/local/gracegtk`. If you are unable to find the location of this directory, contact your system administrator.
  The executables are in `$GRACEGTK_HOME/bin`.
- `GRACEGTK_GUI_SIZE`
  Allow to choose GUI size when `ggrace` is launched, e.g.
  `export GRACEGTK_GUI_SIZE=900x1200`
  allows to launch a window 900 pixels wide and 1200 pixels high.
  Default is set to `900x775`.
- `GRACE_PRINT_CMD`
  Print command. If the variable is defined but is an empty string, "Print to file" will be selected as default.
- `GRACEGTK_HELPVIEWER`
  `GraceGTK` documentation is in PDF format: specify here viewer to use. Default is `firefox %s` for Unix/Linux systems and `acroread %s` for Windows systems.
  The HTML version of the `GraceGTK` guide is not compiled by `make`. You have to follow the instructions in `doc/README` source file to manually compile and install if you want an HTML version.

- `GRACE_HELPVIEWER`
  The shell command to run a help viewer for on-line browsing of the help documents in HTML format.
  Must include at least one instance of "`%s`" which will be replaced with the actual URL by Grace. Default hard coded viewer is presently "`firefox %s`" because it is found in most systems and allows to display many formats.
- `GRACE_EDITOR`
  The editor used for manual editing of dataset values.
- `GRACE_FFTW_WISDOM_FILE` and `GRACE_FFTW_RAM_WISDOM`
  These flags control behavior of the `FFTW` planner (see section 9.5 for detailed info)

3.3.2 Init file

Upon start-up, Grace loads its init file: `gracerc`. The file is searched for in the magic path (see 3.1.14); once found, the rest of the path is ignored. It's recommended that in the `gracerc` file, one doesn't use statements which are part of a project file - such defaults, if needed, should be set in the default template (see 3.3.3).

3.3.3 Default template

Whenever a new project is started, Grace loads the default template, `templates/Default.agr` .
The file is searched for in the magic path (see 3.1.14); once found, the rest of the path is ignored. It's recommended that in the default template, one doesn't use statements which are NOT part of a project file - such defaults, if needed, should be set in the file `gracerc` (see 3.3.2).
`GraceGTK` examples rely on the default template file to run correctly.

4 Guide to the graphical user interface

`GraceGTK` window have below its menubar a status line displaying the *locator* (i.e. the mouse pointer) coordinates on the left and a *status message* on the right.
Below, it display on the left a *toolbar* with various buttons frequently used, and on its right the *canvas*, i.e. the drawing area.

4.1 Using mouse

4.1.1 Using left button (button 1)

- In the left toolbar a click selects an action to be performed[7] (zoom, copy, move,...) . Then, selecting one object in the canvas allows to perform the action on this object[8].
- In the Explorer popup, clicking one row in the left TreeView will select this element and show the corresponding menu in the right pane. If the `instantaneous update` option is on (the default) most of the changes in the right pane are immediately displayed in the canvas. If not, use the `Apply` or `Accept` buttons.
- In the TreeView, the label column is editable. See section 4.4 for details.
- In the TreeView, selecting a `template` in TreeView allows to create a new object.
- It is sometimes difficult to select one object in the canvas when neighbouring objects are also selectable, thus action buttons are also displayed in the bottom of the right panes. The difference with the toolbar buttons is that only one object is selectable if you use these buttons.
- In the canvas, the nature of the action of the mouse is the one selected previously in the left toolbar. In the no "no action" state (see below), a click allows to change the focus from one graph to another.

4.1.2 Using mouse button 3

`Button3` is usually the right button of the mouse.

- In the canvas, mouse `Button3` is used to cancel the action selected in the left toolbar and returns to the "no action" state.

[7] The status message in the upper right corner reflects it.
[8] In this matter, our models are the `xfig` program and Grace.

- In the TreeView, it fires menus depending upon the item(s)[9] selected. The idea is to perform actions on selected elements without further mouse interaction into the canvas.
  The meaning of most entries are obvious, we comment only few.
  - The `Update` entries allows collective but selective updates if the selected items have all the same type. A `Update` window is fired with check buttons to select the parameters used for the update. The update values are red in the right pane (visible or hidden) and the `Apply` button to use is the one in the `Update` popup.
  - The `Use font tool` entries fires the `Font tool` popup and copy the label into (see section 5.5).
  - For creating or changing compounds, see section 5.2.
- In the right panes, some spin buttons are used to enter floating point values (typically, coordinates x_1, y_1, x_2, y_2 for arcs, *etc*). The program try to figure pertinent parameter settings for the `min, max, step` values for these buttons but is not very smart. The menu raised with `Button3` clicked into the entry box offers a `Manage spin button` popup which allows to fix manually the settings and also the value of the variable. Moreover, the entries in this popup are scanned by the command interpreter, allowing to use variables and formulas.
- The `Font tool` can also be used to enter some strings in the right pane.

4.2 Left toolbar

Along the left-hand side of the canvas (if shown) is the `ToolBar`. It is armed with several buttons to provide quick and easy access to the more commonly used Grace functions.
The `Examples/GraceGTK /Explain` menu entry as well as *tooltips* show short explanations about the use of the toolbar buttons. Here is some details.
The upper buttons are inherited from `grace-5.1.22` . The new toolbar buttons are inspired by the well-known program `xfig`.

- `Draw`: This will redraw the canvas and sets. Useful if "Auto Redraw" has been deselected in the `Edit/Preferences` dialog.
- `Lens`: A zoom lens. Click on the lens, then select the area of interest on the active graph with the "rubber box". The region enclosed by the box will fill the entire graph.
- `AS`: AutoScale. Autoscales the active graph to contain all data points of all visible (not hidden) sets.
- `Z/z`: Zoom the active graph in/out by 5%.
  The zoom percentage can be set in the Edit/Preferences dialog.
- `Arrows`: Scroll active graph by 5% in the arrow's direction. The scroll percentage can be set in the Edit/Preferences dialog.
- `AutoT`: AutoTick Axes. This will find the optimum number of major and minor tick marks for both axes.
- `AutoO`: Autoscale On set. Click the `AutoO` button, then click on the graph near the set you wish to use for determining the autoscale boundaries of the graph.
- `ZX,ZY`: Zoom along an axis. These buttons work like the zoom lens above but are restricted to a single axis.
- `AX,AY`: Autoscale one axis only. The following buttons deal with the graph stack and there is a good example under Examples grace-5/General Intro/World Stack.
- `Pu/Po`: Push and pop the current world settings to/from the graph stack. When popping, makes the new stack top current.
- `PZ`: Push before Zooming. Functions as the zoom lens, but first pushes the current world settings to the stack.
- `Cy`: Cycles through the stack settings of the active graph. Each graph may have up to twenty layers on the stack.
- The `Edit` button pops up the explorer with the parameters of the object selected with the mouse in the right pane. It is not true for graphs, which are a special case.
  To select a set in a graph, the graph must be the active graph and you must click on a point of the set, not on the interpolating line.
  In many cases, a double-click on the element produces the same effects without clicking the `Edit` button.

[9] Selecting several elements is made using the GTK standard: `shift` to select all elements between two rows, `control` to add a clicked element to the list.

- The `Move, copy,...` buttons are used to perform the actions with the mouse on the object selected. The check button at the left of the `Edit` button can be used to show/hide the object.
- The frame below contains buttons to delete, move or add a point in a set of the active graph.
- The `Import` button is a direct access to the `Import ASCII` dialog.
- The `TreeView` button pops up the explorer window without changing its state.
- `Exit`: Pretty obvious, eh?

4.3 Menubar

It is used to access the following menus.

4.3.1 File menu

`New`
: Wipe out present drawing to restart a new one.

`Open project`
: As `New`, but also popup a *Open project* window to read a `.agr` command file, usually a project one.

`Save`
: Save the drawing in the present opened project file. If no file is opened, save in the file named `Untitled$`

`Save as...`
: Open a *Save project* window to choose a file to save into.
Be careful that the default format proposed to save data can be not precise enough to fill your needs.
The `Differential` option allows to save only values different from the templates. It is useful if you want to edit manually the file, but beware that it is not self consistent and the result may vary if defaults are changed.

`Revert to saved`
: Abandon all modifications performed on the project since the last save. A confirmation popup is fired to allow the user canceling the operation.

`Load parameters...`
: Open a *Read parameters* window to read a parameters file (see section 3.1.1).

`Save parameters...`
: Open a *Write parameters* window to write a parameters file (see section 3.1.1).

`Import ASCII data`
: Open a *Read data file* window to read ASCII data (see section 3.1.2), i.e. by default a `.dat` file. A graph selector is used to specify the graph where the data should go (except when reading block data, which are copied to graphs later on). The `load type` have following meanings.

  - *Single set* means that if the source contains only one column of numeric data, one set will be created using the indices (from 1 to the total number of points) as abscissas and read values as ordinates and that if the source contains more than one column of data, the first two numeric columns will be used.
  - *NXY* means that the first numeric column will provide the abscissas and all remaining columns will provide the ordinates of several sets.
  - *Block data* means all column will be read and stored and that another popup will allow to select the abscissas and ordinates at will. It should be noted that block data are stored as long as you do not override them by a new read. You can still retrieve data from a block long after having closed all popups, using the set selector.
  - *Select cols.* is an alternative to *Block data* making easy to read several sets at a time by the use of a format string called *Sel.* Each blank(s) separated character in the *Sel.* string stands for one column in the data file and have the following meaning
    - x: the column is for abscissas (should appear only once)
    - y: the column is for ordinates
    - z: the column is for z values (if relevant with respect to set type),
    - `-` : (minus sign) the column is ignored, i.e. no set is created,
    - To ignore the last columns, just erase the corresponding last y

It is possible in the data file to give a name to each data column if the data is headed by a line starting by an exclamation mark (!) followed by the wanted names. These names (one per column) will be used as legend strings for the new sets.
Try `Examples/GraceGTK/Import ASCII`.

The set type can be one of the predefined set presentation types (see tables 2 and 3).

If the source contains date fields, they should be automatically detected. Several formats are recognized (see appendix 12.4 (dates in Grace)). Calendar dates are converted to numerical dates upon reading.
The *Autoscale on read* menu controls whether, upon reading in new sets, which axes of the graph should be autoscaled.

`Export ASCII data`
Save data sets in a file. A set selector is used to specify the set of the active graph to be saved. The format to use for saving data points can be specified: *beware that the default format may be not large enough to fill your needs.*
A warning is displayed if a file with the same name already exists.

`Print setup...`
Set the properties of the printing device.
According to the device, the output can be sent either directly to a printer or directed to a file. Some devices have specific options (see 12.3) and can emit a 'Printout truncated' warning.
Page dimensions units can be pixels/inches/centimetres. The values in pixels depends upon the resolution (dpi) of the device and the predefined formats (A4 or Letter) are automatically detected; thus if in `Custom` mode, you give one of these values, the `Size` combobox is changed.

A TERMINER

You can adjust (*Viewport* in *TreeView/Project/Page*).
In some cases, you can choose between several drivers for the same output format.
See `Output drivers and extra libraries` section for details.

`Print`
Print the project using the current printer settings (either in file or printer).

`Quit`
Exit from Grace. If some work has been done and not saved, a warning popup will be displayed to allow the user to cancel the operation.

4.3.2 Edit menu

`Undo/Redo`
Commands corresponding to changes applied when the `Apply` button is clicked in the explorer are recorded by this subsystem and can be undoed or redoed.This feature works only when two reciprocal commands exists: e.g. data transformations and killings are excluded.

`Explorer`
Popup the Explorer window, analogous to the `TreeView` button in the toolbar.

`Data sets...`
Using the data set popup, you can view the properties of datasets. This include its type, length, associated comment and some statistics (min, max, mean, standard deviation).

`Set operations...`
The set operations popup allows you to interact with sets as a whole. If you want to operate on the data ordering of the sets, you should use the data set operations popup from the `Data` menu. The popup allows you to select a source (one set within one graph) and a destination and perform some action upon them (copy, move, swap). This popup also give you a quick access to several graph and set selectors if you want to perform some other operation like hiding a graph or creating a new set from block data.

`Arrange graphs...`
This entry opens the `Layout` tab of the `Project` row in the Explorer. This allows to dispose several graphs in a regular grid given by M rows and N columns.
The graph list at the bottom allows one to select a number of graphs the arrangement will operate on. If the number of selected graphs isn't equal to $M \times N$, new graphs may be created or extra graphs killed if needed. These options are controlled by the respective check boxes at the top of the pane.

The order in which the matrix is filled in with the graphs can be selected (first horizontally then vertically or vise versa, with either of them inverted). Additionaly, one may choose to fill the matrix in the snake-like manner (adjacent "strokes" are anti-parallel).
The rest of the controls of the dialog window deal with the matrix spacing:
`left/right/top/bottom` page offsets (in the viewport coordinates) and *relative* inter-cell distances, vertical and horizontal. Next to each of the vertical/horizontal spacing spinboxes, a `"Pack"` checkbox is found. Enabling it effectively sets the respective inter-cell distance to zero and alter axis tickmark settings such that only bottom/left-most tickmarks are visible.
If you don't want the regular layout this arrangement gives you, you can change it afterwards using the mouse or the `Main` tab in `Graph` explorer right pane.

`Overlay graphs`
You can overlay a graph on top of another one.
The main use of this feature is to plot several curves using different scales on the "same" graph. Use the TreeView to choose one between overlayed graphs

`Autoscale`
Using this entry, you can autoscale one graph or all graphs according to the specified sets only. This is useful if you need either to have truly comparable graphs despite every one contains data of different ranges, or if you want to focus your attention on one set only while it is displayed with other data in a complex graph.

`Regions menu`

`Status` This small popup only displays the current state (type and whether it is active or not) of the existing regions.

`Define` You can define a new region (or redefine an existing one), the allowed region types are:

- Inside polygon
- Outside polygon
- Above line
- Below line
- Left of line
- Right of line
- In horizontal range
- In vertical range
- Out of horizontal range
- Out of vertical range

A region can be either linked to the current graph only or to all graphs.

`Clear` This kills a region.

`Report on` This popup reports you which sets or points are inside or outside of a region.

`Hot links`
It is a way to update sets when the contents of data files are changed without the need to reopen manually the files.
To establish a link between a file and an existing set, in the `Hot links` window, use the `Files...` button to choose a file, then a set in the `Link to sets` part of the window and push the `Link` button: the link should be displayed in the upper part of the window. Then, pushing the `Update` button will reread all the linked files.
It is also possible to use pipes instead of files[10].
Currently, only simple XY sets can be used for hotlinks.

`Set locator fixed point`
After having selected this menu entry, you can select a point on a graph that will be used as the origin of the locator display (just below the menu bar). The fixed point is taken into account only when the display type of the locator is set to [DX,DY].

`Clear locator fixed point` This entry is provided to remove a fixed point set before and use the default again: point [0, 0].

`Locator props`
The locator props popup allows you to customize the display of the locator, mainly its type and the format and precision of the display. You can use all the formats that are allowed in the graphs scales.

[10] Except in a MS Windows environment.

`Preferences`

The preferences popup allows you to set miscellaneous properties of your Grace session, such as GUI behavior, cursor type, date reading hint and reference date used for calendar conversions.

4.3.3 Data menu

`Data set operations`

This popup gathers all operations that are related to the ordering of data points inside a set or between sets. If you want to operate on the sets as a whole, you should use the `set operations` popup from the `Edit` menu. You can sort according to any coordinate $(X, Y, DX, \ldots)$ in ascending or descending order, reverse the order of the points, join several sets into one, split one set into several others of equal lengths, or drop a range of points from a set. The `set selector` of the popup shows the number of points in each set in square brackets like this: G0.S0[63], the points are numbered from 0 to $n-1$.

`Transformations menu`

The transformations sub-menu gives you access to all data-mining features of Grace.

`Evaluate expression`

Using `evaluate expression` allows you to create a set by applying an explicit formula to another set, or to parts of another set if you use regions restrictions.
All the classical mathematical functions are available (cos, sin, but also lgamma, j1, erf, ...). As usual all trigonometric functions use radians by default but you can specify a unit if you prefer to say cos (x rad) or sin (3 * y deg). For the full list of available numerical functions and operators, see sections 8.2 and 8.3.
In the formula, you can use X, Y, Y1, ..., Y4 to denote any coordinate you like from the source set. An implicit loop will be used around your formula so if you say:

```
x = x - 4966.5
```

you will shift all points of your set 4966.5 units to the left.
You can use more than one set in the same formula, like this:

```
y = y - 0.653 * sin (x deg) + s2.y
```

which means you use both X and Y from the source set but also the Y coordinate of set 2. Beware that the loop is a simple loop over the indices, all the sets you use in such an hybrid expression should therefore have the same number of points and point i of one set should really be related to point i of the other set. If your sets do not follow these requirements, you should first homogenize them using interpolation popup.

`Histograms`

The `histograms` popup allows you to compute either standard or cumulative histograms from the Y coordinates of your data. Optionally, the histograms can be normalized to 1 (hence producing a PDF (Probability Distribution Function).
The bins can be either a linear mesh defined by its min, max, and length values, or a mesh formed by abscissas of another set (in which case abscissas of the set must form a strictly monotonic array).

`Fourier transformation`

The Fourier transform window is now inspired by `grace-5.99` with some changes:
try `Examples GraceGTK /Fourier` and look at section 9 for details.
The main novelty is that you can take into account the translation needed by the FFT to get the true Fourier transform of the function and not the transform of the translated one.
It should work with library `FFTW -2` as well as `FFTW -3`. Legacy code for FFT is not operational thus it is recommended to use FFTW to take the transform of lengthy sets.
X is assumed to be equally spaced.
You can filter the input data window through triangular, Hanning, Welch, Hamming, Blackman and Parzen filters.
If you specify complex input data X is taken as the real part and Y as the imaginary part.

`Wavelet transformations`

Compute Daubechies, Haar and Bsplines 1D transforms for datasets with 2^n points. More details are given in section 10.

`Running averages`

The running average popup allows you to compute some values on a sliding window over your data. You choose both the value you need (average, median, minimum, maximum, standard deviation) and the length of the window and perform the operation. You can restrict the operation to the points belonging to (or outside of) a region.

`Differences/derivation`

The differences popup is used to compute approximations of the first derivative of a function with finite differences.

`Seasonal differences`

The seasonal differences popup is used to subtract data from a period to data of the preceding period (namely y[i] - y[i + period]). Beware that the period is entered in terms of index in the set and not in terms of abscissa!

`Integration`

The integration popup is used to compute the integral of a set using the rectangles rule and optionally to load it. The numerical value of the integral is shown in the text field after computation. Selecting "cumulative sum" in the choice item will create and load a new set with the integral and compute the end value, selecting "sum only" will only compute the end value.

`Interpolation/Splines`

This popup is used to interpolate a given set and if needed create a new one.
Several methods are proposed.

- `linear` is a straight line interpolation,
- `cubic spline` is based the classical resolution of a tridiagonal system, producing often large oscillations,
- `Akima` is a more sophisticated one with less oscillations,
- `X-spline` and `XY-spline` are the sophisticated crossed splines interpolation / approximation methods described in [1]. The last one is already used by program `xfig`.

Except for `XY-spline`, the user choose a sampling array that can be either a linear mesh defined by its min, max, and length values, or a mesh formed by abscissas of another set. Note that if the sampling mesh is not entirely within the source set x bounds, evaluation at the points beyond the bounds will be performed using interpolation parameters from the first (or the last) segment of the source set, which can be considered as a primitive extrapolation. This behaviour can be disabled by checking the "Strict" option on the popup.
Except for `linear` and `XY-spline`, the abscissas of the interpolated set must be *monotonic*.
The `XY-spline` method is a 2D interpolation where the number of points of the resulting set is fixed by the routine itself. The precision parameter p may change the number of interpolating points.
The shape of interleaved splines functions used by `X-spline` and `XY-spline` is controlled by the s parameter ranging in $\in [-1, 1]$. For negative values, the curve is constrained to cross the data points, thus an interpolation is made. Sometimes, this constraint does not allow to obtain curves smooth enough: in that case, it can be removed by using positive values, leading to an approximation of the curve. Details can be found in [1].
The difference with the `spline` option in the `Line/connection` combo box in the `Set` right pane of the `Explorer` is that the latter creates only a temporary set when plotting the curve and display symbols only at the primitive points.

`Regression`

The regression popup can be used to fit a set against polynomials or some specific functions (y=A*x^B, y=A*exp(B*x), y=A+B*ln(x) and y=1/(A+Bx)) for which a simple transformation of input data can be used to apply linear regression formulas.
You can load either the fitted values, the residuals or the function itself. Choosing to load fitted values or residuals leads to a set of the same length and abscissas as the initial set. Choosing to load the function is almost similar to load the fitted values except that you choose yourself the boundaries and the number of points. This can be used for example to draw the curve outside of the data sample range or to produce an evenly spaced set from an irregular one.

`Non-linear fit`

The `Non-linear curve fitting` popup allows to fit a set against your own formula by use of the Levenberg-Marquardt algorithm[11]
With this popup you provide yourself a formula (e.g. `y = a0 * cos(a1 * x + a2)`) where `a0, a1, ..., a9` are up to 10 unknowns to be computed.
You specify a tolerance, starting values and optional bounds and run several steps before ending in the display of results in the console window and the plot of the final curve.

[11] See http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm.
Actually, the program use a C translation of `lmdif subroutine`. See http://www.netlib.org/minpack/

The fit characteristics (number of parameters, formula, ...) can be saved in a file and retrieved as needed using the file menu of the popup.
In the `Advanced` tab, you can additionally apply a restriction to the set(s) to be fitted (thus ignoring points not satisfying the criteria), use one of preset weighting schemes or define your own[12], and choose whether to load the fitted values, the residuals or the function itself. Choosing to load fitted values or residuals leads to a set of the same length and abscissas as the initial set. Choosing to load the function is almost similar to load the fitted values except that you choose yourself the boundaries and the number of points. This can be used for example to draw the curve outside of the data sample range or to produce an evenly spaced set from an irregular one.
N.B. The `Tolerance` *parameter is used to check the residuals between two iteration steps, i.e. the iteration stops when it is stabilised and this does not mean that the distance between the fit and the data has the* `tol` *value.*
The library of functions proposed by Nicola Ferralis is incorporated in `GraceGTK` and accessed via the "`Library`" menu. It gives a set of predefined functions and allows to choose the values of some starting parameters (peak position...) simply with the mouse.
Try `Examples/GraceGTK/NL fit`.
See section 5.6 for a detailed list of the functions available.
N.B. This has been programmed and written in a «blind» manner and needs some checking and testing.

`Correlation/covariance`
: This popup can be used to compute autocorrelation of one set or cross correlation between two sets. You only select the set (or sets) and specify the maximum lag. A check box allows one to evaluate covariance instead of correlation. The result is normalized so that $abs(C(0)) = 1$.

`Digital filter`
: You can use a set as a weight to filter another set. Only the Y part and the length of the weighting set are important, the X part is ignored.

`Linear convolution`
: The convolution popup is used to ... convolve two sets. You only select the sets and apply.

`Geometric transforms`
: You can rotate, scale or translate sets using the geometric transformations popup. You specify the characteristics of each transform and the application order.

`Sample points`
: This popup provides two sampling methods. The first one is to choose a starting point and a step, the second one is to select only the points that satisfy a boolean expression you specify.

`Prune data`
: This popup is devoted to reducing huge sets (and then saving both computation time and disk space).
The interpolation method can be applied only to ordered sets: it is based on the assumption that if a real point and an interpolation based on neighboring points are closer than a specified threshold, then the point is redundant and can be eliminated.
The geometric methods (circle, ellipse, rectangle) can be applied to any set, they test each point in turn and keep only those that are not in the neighborhood of previous points.

`Feature extraction`
: Given a set of curves in a graph, extract a feature from each curve and use the values of the feature to provide the Y values for a new curve (see table 4).

`Import/ASCII`
: Is here for backward compatibility with `grace-5.1.22` . See `Import ASCII` dialog.

`Export/ASCII`
: Is here for backward compatibility with `grace-5.1.22` . See `Export ASCII` dialog.

4.3.4 Plot menu

This menu is present for backward compatibility with `grace-5.1.22` but is not updated when `GraceGTK` is changed, thus the new features are not available and the synchronisation with the `Explorer` can fails. In brief, its usage is discouraged and it is deactivated by default. Use the

[12] Notice that "`dY`" in the preset "`1/dY^2`" one actually refers to the third column of the data set; use the "Custom" function if this doesn't make sense for your data set.

Feature	Description
Y minimum	Minimum Y value of set
Y maximum	Maximum Y value of set
Y average	Average Y value of set
Y std. dev.	Standard deviation of Y values
Y median	Median Y value
X minimum	Minimum X value of set
X maximum	Maximum X value of set
X average	Average X value of set
X std. dev.	Standard deviation of X values
X median	Median X value
Frequency	Perform DFT (or FFT) to find largest frequency component
Period	Inverse of above
Zero crossing	Time of the first zero crossing, + or - going
Rise time	Assume curve starts at the minimum and rises to the maximum, get time to go from 10% to 90% of rise. For single exponential curves, this is 2.2*time constant
Fall time	Assume curve starts at the maximum and drops to the minimum, get time to go from 90% to 10% of fall
Slope	Perform linear regression to obtain slope
Y intercept	Perform linear regression to obtain Y-intercept
Set length	Number of data points in set
Half maximal width	Assume curve starts from the minimum, rises to the maximum and drops to the minimum again. Determine the time for which the curve is elevated more than 50% of the maximum rise.
Barycenter X	Barycenter along X axis
Barycenter Y	Barycenter along Y axis
X (Y max)	X of Maximum Y
Y (X max)	Y of Maximum X
integral	cumulative sum

Table 4: Extractable features

Menubar/View menu to activate it.
Refer to `grace-5.1.22` documentation for a detailed account on the Plot submenus.

4.3.5 View menu

`Page zoom +, Page zoom -,Page zoom reset`
: Change the scaling of the page displayed on the screen. The page size of the printed output must be checked in the `Print setup` window.

`Show locator bar, Show status bar, Show tool bar`
: Toggle the display of the corresponding elements.

`Page setup`
: More general than the `Explorer/Project/Page` tab, this popup is the same as the `Print setup` window.

`Redraw`
: This menu item triggers a redrawing of the canvas.

`White background`
: is the default setting for the screen.

`Black background`
: Sets the screen background to be black as the `-rvideo` command option do, but does not change the output in a file or a printer.

`Update all`
: Usually, everything in the GUI is updated automatically, but if this fails, you can update it manually. *Note: this command is not up-to-date.*

4.3.6 Window menu

`Commands`
: This window allows to execute directly *commands* in `GraceGTK` language. See section 8 for details on the `GraceGTK .agr` language.
The "`Follow me`" mode can be selected with the check button at the bottom of the window: it is a help to write your own scripts. The commands corresponding to changes made in right pane menus are printed in the window when this option is selected[13].
You can edit lines into this window. Clicking the `Apply` or `Accept` buttons execute the lines selected with the mouse.
All the command lines must begin with a `@`.
You can read in files and save the content of the window with the "`File`" menu at the top of the command window.

`Font tool`
: It is a tool to help writing strings with escape sequences and non ASCII characters. See details in section 5.5.

`Console`
: The console window displays errors and results of some numerical operations, e.g. nonlinear fit. The window is popped up automatically whenever an error occurs or new result messages appear. This can be altered by checking the "`Options/Popup only on errors`" option.

4.3.7 Help menu

Try it.
If the menu does not work correctly (a bad installation), you can access directly to the files (manual, FAQ, ...) in the `doc` directory (in a standard installation, `/usr/local/gracegtk/doc`).
Try the `Examples`, it is a very good way to get an idea of `GraceGTK` possibilities. The files are in the `examples` directory.

4.4 Explorer window

It is displayed when the `TreeView` button is clicked in the toolbar or with the `Edit/Explorer` menu in the menubar. It shows a left pane, a right pane and two buttons at the bottom.

[13] Beware that it is not sure that all changes are taken into account.

4.4.1 Left pane

The first column gives a tree view of the drawing components. Visible components are in **bold normal** and hidden in *light italic*.
The second column of the TreeView displays a `label` (see below).
Clicking *one* tree element in the tree with the mouse `Button1` (usually the left button) shows the corresponding menu in the right pane, and the `Apply` button at the bottom allows to apply the values to the selected element[14].
The `Label` column is editable: double-click on the item to edit it and press the *enter key* to record the change.

- For graphs, axis, sets and string this allows to edit graph title, axis label, set legend or the string itself.
- For arcs, lines, boxes and compounds the label is just informative.
- For time stamp, legend boxes and templates, editing the label is not relevant and it will return to its initial state when you press the enter key.

If you need special characters or special formatting in your label, you can use the `font tool` (see section 5.5).

4.4.2 Arcs, boxes, circles, lines, polylines and strings

These geometric objects can be created by using the *template* entries in the `TreeView` and clicking the `"Create & place with mouse"` button. They are always created in the tree as child of the template and in viewport coordinates. By editing them, it is possible to attach to a given graph[15] and in that case to define them in world coordinates.

4.4.3 Right pane menus

Up to now, these menus are copied from `grace-5.1.22` with few changes, but it can be changed in future versions. If the "instant update" feature is on (the default), most of the changes are immediately taken into account; if it not the case, you must click on the bottom `Apply` button to make your changes effective. The `Close` button hide the explorer window with no change; if no item selection is made during the interval, rising again the explorer with the `TreeView` button will show the window in the state it had when you hide it.
Beware that the synchronisation of selections between the TreeView and the canvas is a touchy business and that `GraceGTK` may sometimes behave not as you expected in this matter.
Thus, if it seems that the right pane menu does not work check that in the TreeView

1. the desired item is actually selected,
2. only one item is selected.

`Project` menus allows to choose page format and color background, to scatter several graphs on the page, add sets defined by a given formula in various graphs. The `Apply` button does not behave here like for the other menus.

- The `Page` tab is applied each time the `Apply` button is clicked.
- The `Layout` tab is applied only if the tab is the visible tab. For a detailed description of this notebook page, see `"Arrange graphs..."` section 4.3.2.
- Specialized buttons are placed in other tabs to make an action.

`TimeStamp` menu is used to control the appearance of the time stamp and set the `GraceGTK` configuration for dates

`Graph` main tab includes the properties you will need more often (title for example), and other tabs are used to fine tune some less frequently used options (fonts, sizes, colors, placements). The `sets` tab allows actions on several sets belonging to the same graph.

`Axis` menu make possible to display a second or *alternate axis* in addition to the normal one, using the `choose` combo box at the top of the menu. As long as this axis is marked hidden, it is not displayed in the TreeView, thus you have to uncheck the hide button in the bottom of the right pane menu to make it appearing in the left pane.
The start and stop fields depict the displayed range.
Five types of scales are available: `linear, logarithmic, reciprocal, logit and degrees`

[14] Note that if more than one element is selected, the `apply` button is inactived.
[15] Note that polylines are not intended as a replacement of XY sets and should have a moderate number of points.

(for polar graph only). and you can invert the axis direction (which normally increases from left to right and from bottom to top).
The main tab includes the properties you will need more often (axis label, tick spacing and format for example), and other tabs are used to fine tune some less frequently used options (fonts, sizes, colors, placements, stagger, grid lines, special ticks, ...).
`Special` tab allows to choose individual positions and labels for ticks. The default is `None`, i.e. are defined in the `Main` and `Tick marks` tabs. If you change the settings, the list is populated by the default list i.e. major and minor ticks are displayed.
The `Nb` spin box allows to choose the number of ticks to use and you can change positions and labels of ticks, then `Apply` to finalise your choice.
Beware that returning to `None` will destroy the present list and return to the default one.
It is possible to use an algebraic expression for the location and specials for strings label, e.g. to display a tick at location $3\pi/2$ and display the label:

`3*pi/2` and `3\f{Symbol}p/2`

`Set` menu is used to control set appearance and add or delete points with the mouse. The tabs displayed vary with the type of the set.
The `Line` tab allows various types of connection for data points, including spline interpolation. For spline interpolation a temporary set is created at drawing time and immediately killed after drawing is done. If you want to create a permanent interpolating set use `Data/transformation` menu.
More details about spline interpolation are given in Interpolation/Splines section about data transformation.

`Leg.Box` allows to display the *legend* of sets i.e the line type and symbol used as well as a comment.

`Line, Arc, String and Box` templates are used to create new objects, attached directly to the page (by choosing -1 as graph number) or to a given graph. In the latest case, it is possible to choose between viewport coordinates and graph coordinates. The difference appears when you move or rescale the graph. Also, automatic scaling take into account object in world coordinates and not in viewport coordinates.

The examples can be a good help to have a better understanding of the detailed behaviour of the program.

4.5 Using layers

When all the drawing is made on the same layer, the right hand column of check boxes used to toggled layers visibility is not displayed and this layer is putted automatically visible.
If you want to display an object in front of another, use the spin boxes in the Explorer right pane menus to attach it to a upper layer.
Objects glued into a *compound* can be attached to different layers, thus compounds does not have a given depth, but the *Collapse* button allows to put all its objects on the same layer.
It is the same for *graphs*. Note that *graph frame, title* and *subtitle* can be drawn on different layers.

4.6 Some hints

4.6.1 Polar graphs

The interface for polar type is not well polished.

- The X-axis is in fact the angle axis and the Y-axis the modulus one.
- For the angle (or phi) axis,
  - *scale*=`linear` display tick labels in radians and the x-autoticking choose rounded values in radians,
  - *scale*=`degrees` display tick labels in degrees and the x-autoticking choose rounded values as 30° multiples.

4.6.2 Set types

Beware that changing the type of sets can produce a loss of data: if the new set use less data columns than the old one, and the data had not been imported with the `block data` popup, the columns in excess are lost. In both cases, you are warned by a message and asked to proceed if in the interactive mode.

5 Behavior different from grace-5.1.22

Some differences with grace-5.1.22 .

5.1 Arcs

grace-5.1.22 users already know the line, box, string and ellipse objects. In GraceGTK , the ellipse type is replaced by the more general arc object, i.e. an arc of ellipse with possibly arrows at its ends.

5.2 Compounds

The new compound type is similar to compounds in xfig: several geometric objects can be glued together and selected, hide or shown as well as moved as a single block.
No direct scaling of the block is available.
To access one component of the compound, select it alone in the TreeView.
Two methods are possible to manage compounds.

- Using mouse in the canvas. If you click buttons in the canvas left toolbar, as for other objects, any of the eligible objects are selectable. If you click buttons in the Explorer right pane, only the item selected in the tree is selectable.
- Use TreeView and mouse button 3 for following actions.
  - Create a new compound and glue can be used with several items selected. All the items must be geometric, attached on the same level to the same graph (or to templates), with the same loctyp. Thus, if you want to create new objects and glue them into a compound belonging to a graph, create them with the template(s), change attachment, if needed change loctyp and finally select and glue them.
  - Break compound: obvious.
  - Prune works only when one item is selected. The result is to use templates nodes as a sort of clipboard: object is moved into a template branch, the compound shift is added to its coordinates and the loctyp is set as coord. view.
  - Insert into a compound is the inverse of Prune: you are asked to choose an existing compound then, the object is moved into it and parameters are set according to the compound ones.

5.3 Comments and legends

In grace-5.1.22 , the nature of new sets is automatically described in the comment string, not in the legend displayed in the graph: you have, attached to the same graph (or to the templates to load comments as legends to see them. In GraceGTK this is done automatically by default, but you can disable this feature in the Edit/Preference menu.

5.4 Command window

This window is now directly editable and only selected lines are executed when the Apply or Accept buttons are clicked.
All the command lines must begin with a @.
A "Follow me" mode had been added: see Commands in Window menu.

5.5 Font tool

It is a tool to help writing strings with Grace escape sequences and non ASCII characters that will be displayed onto the canvas with a pretty printing.
Normally, it is fired clicking mouse button 3 into the entries where it usage makes sense. More often, if instant update is on, clicking the Apply button into the Font tool windows will be enough to take into account the changes, except for the Special tab in Set menu, where this will change only the Label entry and it is necessary to click also the Apply button of the Explorer.
When the Font tool is called by the Menubar/Window/Font tool menu entry, it is displayed into it precedent state, i.e. if a previous connection to an entry exists, it remains active.

5.6 Non-linear fit

The library of functions proposed by Nicola Ferralis for the non linear popup is incorporated in `GraceGTK` and accessed via the "Library" menu.
N.B. This has been programmed and written in a «blind» manner and needs some checking and testing.

5.6.1 Gaussian Functions

Gaussian simple

$$y = A_0 + \frac{A_3}{A_2}\,2\sqrt{\frac{\ln(2)}{\pi}}\;e^{-4\ln(2)\left(\frac{x-A_1}{A_2}\right)^2} \qquad \text{where}$$

- A_0: Baseline offset,
- A_1: Center of the peak,
- A_2: Full width at half maximum,
- A_3: Peak area.

The center and initial amplitude of the peak can be set from user input (via mouse coordinates).

Gaussian Function (Chromatography)

$$y = A_0 + \frac{1}{\sqrt{2\pi}}\,\frac{A_3}{A_2}\;e^{-\frac{1}{2}\left(\frac{x-A_1}{A_2}\right)^2} \qquad \text{where}$$

- A_0: Baseline offset,
- A_1: Center of the peak (retention time),
- A_2: Standard deviation of the peak,
- A_3: Peak area.

The center and initial amplitude of the peak can be set from user input (via mouse coordinates).

Gaussian double

$$y = A_0 + 2\sqrt{\frac{\ln(2)}{\pi}}\left(\frac{A_3}{A_2}\,e^{-4\ln(2)\left(\frac{x-A_1}{A_2}\right)^2} + \frac{A_6}{A_5}\,e^{-4\ln(2)\left(\frac{x-A_4}{A_5}\right)^2}\right) \qquad \text{where}$$

- A_0: Baseline offset,
- A_1, A_4: Center of the peaks 1,2,
- A_2, A_5: Full width at half maximum of peaks 1, 2,
- A_3, A_6: Area of peaks 1, 2.

Gaussian triple

$$y = A_0 + 2\sqrt{\frac{\ln(2)}{\pi}}\left(\frac{A_3}{A_2}\,e^{-4\ln(2)\left(\frac{x-A_1}{A_2}\right)^2} + \frac{A_6}{A_5}\,e^{-4\ln(2)\left(\frac{x-A_4}{A_5}\right)^2} + \frac{A_9}{A_8}\,e^{-4\ln(2)\left(\frac{x-A_7}{A_8}\right)^2}\right)$$

5.6.2 Lorentzian Functions

Lorentzian simple

$$y = A_0 + \frac{2}{\pi}\,\frac{A_2 A_3}{4(x-A_1)^2 + A_2^2} \qquad \text{where}$$

- A_0: Baseline offset,
- A_1: Center of the peak,
- A_2: Full width at half maximum,
- A_3: Peak area.

Lorentzian double

$$y = A_0 + \frac{2}{\pi}\left(\frac{A_2 A_3}{4(x-A_1)^2 + A_2^2} + \frac{A_5 A_6}{4(x-A_4)^2 + A_5^2}\right)$$

Lorentzian triple

$$y = A_0 + \frac{2}{\pi}\left(\frac{A_2 A_3}{4(x-A_1)^2 + A_2^2} + \frac{A_5 A_6}{4(x-A_4)^2 + A_5^2} + \frac{A_8 A_9}{4(x-A_7)^2 + A_8^2}\right)$$

5.6.3 Peak Functions

Pseudo Voigt 1

$$y = A_0 + A_3 \left[\frac{2}{\pi} \frac{A_4 A_2}{4 * (x - A_1)^2 + A_2^2} + \sqrt{\frac{4 \ln(2)}{\pi}} \frac{1 - A_4}{A_2} e^{-4 \ln(2) \left(\frac{x - A_1}{A_2}\right)^2} \right] \quad \text{where}$$

Gaussian and Lorentzian have the same width and

- A_0: Baseline offset,
- A_1: Center of the peak,
- A_2: Full width at half maximum,
- A_3: Amplitude,
- A_4: Profile shape factor.

Pseudo Voigt 2

$$y = A_0 + A_3 \left[\frac{2}{\pi} \frac{A_5 A_2}{4 * (x - A_1)^2 + A_2^2} + \sqrt{\frac{4 \ln(2)}{\pi}} \frac{1 - A_5}{A_4} e^{-4 \ln(2) \left(\frac{x - A_1}{A_2}\right)^2} \right] \quad \text{where}$$

Gaussian and Lorentzian have different widths.

Doniach-Sunjic

$$y = A_0 + A_3 \cos\left[\frac{\pi}{2} A_4 + \frac{1 - A_4}{(A_2^2 + (x - A_1)^2)^{(1 - A_4)/2}} \arctan\left(\frac{x - A_1}{A_2}\right) \right] \quad \text{where}$$

- A_0: Baseline offset,
- A_1: Center of the peak,
- A_2: Full width at half maximum,
- A_3: Peak area,
- A_4: Asymmetry parameter.

Asymmetric double sigmoidal function

$$y = A_0 + A_3 \frac{1}{1 + e^{-\frac{x - A_1 + \frac{1}{2} A_2}{A_4}}} \left[1 - \frac{1}{1 + e^{-\frac{x - A_1 - \frac{1}{2} A_2}{A_5}}} \right] \quad \text{where}$$

- A_0: Baseline offset,
- A_1: Center of the peak,
- A_2: Width 1,
- A_3: Amplitude,
- A_4: Width 2.
- A_5: Width 3.

Log Normal Function

$$y = A_0 + A_3 e^{-\frac{(\ln x - \ln A_1)^2}{2 A_2}} \quad \text{where}$$

- A_0: Baseline offset,
- A_1: Center of the peak,
- A_2: Width,
- A_3: Amplitude.

Gram-Charlier A-Series

$$y = A_0 + \frac{A_3}{A_2 \sqrt{2\pi}} \left[1 + \frac{A_4}{6} (X^3 - 3X) + \frac{A_5}{24} (X^4 - 6X^3 + 3) \right] e^{-\frac{1}{2} X^2} \quad \text{where } X = \frac{x - A_1}{A_2}$$

- A_0: Baseline offset,
- A_1: Center of the peak,

- A_2: Standard deviation,
- A_3: Peak Area,
- A_4: Skew,
- A_5: Excess.

Edgeworth-Cramer Series

$$y = A_0 + \frac{A_3}{A_2\sqrt{2\pi}}\left[1 + \frac{A_4}{6}(X^3 - 3X) + \frac{A_5}{24}(X^4 - 6X^3 + 3) + \frac{A_5^2}{720}(X^6 - 15X^4 + 45X^2 - 15)\right] e^{-\frac{1}{2}X^2}$$

where

$$X = \frac{x - A_1}{A_2}$$

and parameters as for Gram-Charlier A-Series.

Inverse Polynomial Function

$$y = A_0 + \frac{A_3}{1 + A_4X^2 + A_5X^4 + A_6X^6} \qquad \text{where } X = 2\,\frac{x - A_1}{A_2}$$

- A_0: Baseline offset,
- A_1: Center of the peak,
- A_2: Standard deviation,
- A_3: Peak Area,
- A_4, A_5, A_6: Parameters.

5.6.4 Periodic Peak Functions

Sine Function

$$y = A_0 + A_3 \sin\left(\pi\frac{x - A_1}{A_2}\right)$$

- A_0: Baseline offset,
- A_1: Center,
- A_2: Width,
- A_3: Amplitude,
- A_4, A_5, A_6: Parameters.

Sine Square Function

$$y = A_0 + A_3 \sin^2\left(\pi\frac{x - A_1}{A_2}\right)$$

and parameters as for Sine Function.

Sine Damp Function

$$y = A_0 + A_3 \sin\left(\pi\frac{x - A_1}{A_2}\right) e^{-x/A_4}$$

and parameters as for Sine Function.

5.6.5 Baseline Functions

Exponential Decay 1

$$y = A_0 + A_3\, e^{-\frac{x - A_1}{A_2}}$$

Exponential Decay 2

$$y = A_0 + A_3\, e^{-\frac{x - A_1}{A_2}} + A_6\, e^{-\frac{x - A_4}{A_5}}$$

Exponential Growth 1

$$y = A_0 + A_3\, e^{\frac{x - A_1}{A_2}}$$

Exponential Growth 2

$$y = A_0 + A_3\, e^{\frac{x-A_1}{A_2}} + A_6\, e^{\frac{x-A_4}{A_5}}$$

Hyperbolic

$$y = A_0 + \frac{A_1\, x}{A_2 + x}$$

Bradley

$$y = A_0\, \ln(-A_1\, \ln x)$$

Logarithm 3 Parameters

$$y = A_0 - A_1 \ln(x + A_2)$$

Weibull Probability Density

$$y = \frac{A_0}{A_1} \left(\frac{x}{A_1}\right)^{(A_0-1)} e^{-(x/A_1)^{A_0}}$$

Weibull Cumulative Distribution

$$y = 1 - e^{-(x/A_1)^{A_0}}$$

6 2D graphs

`GraceGTK` can draw vector and colours maps.

6.1 Color map and contour sets

`GraceGTK` can represent $z = f(x, y)$ if x and y values form a regular rectangular grid in two ways:

- associate a color to each value of z, i.e. a colormap,
- draw level curves.

The set must be defined as a *XYCMAP* set and the contour line option is possible only if a Fortran compiler had been detected and used during the compilation of the program.
Note that you must supply two new parameters when the set is defined: nx and ny, the number of different values for x and y.
In the `Explorer`, the `Color map` tab shows a `Contour line` frame: by default, level curves are not computed, i.e. the `Hide` check box is active. To draw level curves, deactivate the box and click `Apply`.
The levels defined by the colorbar ticks are accessed by selecting the `Colorbar` associated to the set.
It is possible to add labels on the level curves with the mouse by clicking first the appropriate button in the Contour lines frame, then clicking on the contour curve at the place chosen. Beware that contour lines and labels are not duplicated with the other components of a set.
The computation of contour lines[16] is based on a triangulation of the surface that can take some time, thus we store the resulting mesh and try to avoid to recompute it. If something goes wrong in refreshing the level curves, use the `Kill` button to force a new computation.
More details about computation of contour lines can be found in [3].

N.B. only the Postscript `grace-5.1.22` driver had been adapted to deal with `XYCMAP` colours: use CAIRO drivers instead for other formats.

7 Undo and Redo

If you have compiled with the `libundo` library, it is possible to use the `Undo/Redo` buttons in the `Edit` menu. The method is inspired by the follow-me mode: each time a change in the parameters is detected, two command lines (using the grace .agr language) are recorded: one with the old values for the undo action, one with the new ones for the redo action.
N.B.

- This feature works only when two reciprocal commands exists and if the registering is programmed: e.g. data transformations and killings are excluded.
- An attempt is made to discard intermediate values when the parameters are varied at high frequency by using spin buttons arrows, but, the delay to start repetition makes that several "undo" are anyway recorded recorded in that case.

[16] It is made using algorithm toms 626 (www.netlib.org) proposed by A. Preusser.

8 Command interpreter

Preferred extensions are `.agr` for command/data files and `.dat` for data files.
The input stream is parsed in a line-by-line manner i.e. an instruction cannot spread over several lines.
First, leading blanks and tabs as well as empty lines are discarded, then, the first significant character is used to determine the type of the line or change the mode:

- `#` means a comment line,
- `@` means a command line,
- `!` means a heading line for columns of data (see `Import ASCII data`),
- `&` is the mark for the end of a data sequence.
- After that, the default is to try to parse the line as a data line, i.e. a list of numbers and quoted strings. If this later fails and the line is in the first ten, it is simply discarded. This behavior allows to read data files produced by other programs with non `GraceGTK` standard headings.

The parsing of commands is case-insensitive.
In command lines, there may be several statements per line, separated by semicolon (`;`) and no `@` after the semicolon.
The command language is an extension of `grace-5.1.22` language, with all the subsequent constraints. The user is encouraged to examine the files in the `examples` directory. In this directory, files whose name begins with `ng_` are new `GraceGTK` examples.

`GraceGTK` language is based on numerous reserved words but a simple grammar without elaborated constructs spreading on several lines.

8.1 Definitions

8.1.1 Version number

Saved project files start with: `@ VERSION` *nexpr* where $nexpr = 1xxyyzz$ where: xx is the *major* and yy the *minor* revision numbers and zz the *patchlevel*. The leading 1 is used to differentiate `GraceGTK` from Grace.

8.1.2 Constant

`PI` is the π constant.

8.1.3 Variables

Three types of numerical variables are possibles[17]:

- *var* : scalar variable
- *vvar* : vector variable (one index array)
- *mvar* : matrix variable (two indexes array)

The indices in arrays start at zero, as it is the case in the C language.
It is possible to define user's variables if the name is not already used[18]. The table below gives the syntax to define and clear variables.

Statement	Description	Example
`DEFINE` *var*	define new scalar variable *var*	`DEFINE myvar`
`DEFINE` *vvar*`[]`	define new vector variable *vvar* of zero length	`DEFINE myvvar[]`
`DEFINE` *vvar*`[`*n*`]`	define new vector variable *vvar* of length *n*	`DEFINE myvvar[10]`
`DEFINE` *mvar*`[`*m*`][`*n*`]`	define new matrix with *m* rows and *n* columns	
`CLEAR` *var*	undefine new variable *var* and deallocate associated storage	`CLEAR myvar`
`CLEAR ALLVARS`	undefine all user defined variables	
vvar `LENGTH` *n*	reallocate vector variable *vvar*	`myvvar LENGTH 25`

Note that vectors length is dynamic, i.e. it can be changed after its definition, but that matrices dimensions are static.

[17] For simplicity we use *vector* and *matrix* although it is simply arrays without reference to there exact mathematical definition.

[18] Switching from one project to another one does not reset the user space name, thus it is a good practise to `CLEAR ALLVARS`, i.e. all user created variables at the end of a project file.

We use a special denomination for some particular variables created by the program:

Name	Type	Description
datacolumn	vector	a data column of the current ("active") set
set.datacolumn	vector	a data column of set
vvariable[i:j]	vector	segment of a vector variable (elements from i-th to j-th inclusive, i <= j)
vvariable[i]	scalar	i-th element of a vector variable

The selection of a *datacolumn* of a dataset is done with the following syntax

Expression	Description	Types	Example
X	the first column	-	X
Y	the second column	-	Y
Y*n*	$(n + 2)$-th column	$n = 0$ - 4	Y3

Before selecting a *datacolumn*, you must select a *graph* and a *set* in the graph:

Expression	Description	Types	Example
GRAPH[*id*]	graph *id*	indx *id*	GRAPH[0]
G*nn*	graph *nn*	*nn*: 0-99	G0

Graph selection

Expression	Description	Example
graph.SETS[*id*]	set *id* in graph *graph*	GRAPH[0].SETS[1]
graph.S*nn*	set *nn* in graph *graph*	G0.S1
SET[*id*]	set *id* in the current graph	SET[1]
S*nn*	set *nn* in the current graph	S1
S_	the last implicitly (i.e. as a result of a data transformation) allocated set in the current graph	S_
S$	the active set in the current graph	S$

Set selection

8.1.4 Colors, patterns and regions

Expression	Description	Example
COLOR *"colorname"*	a mapped color *colorname*	COLOR "red"
COLOR *id*	a mapped color with ID *id*	COLOR 2
MAP COLOR *id* TO *(r,g,b), sexpr*	define new color or remap old one	MAP COLOR 5 TO (255, 255, 0), "yellow"
PATTERN *id*	pattern with ID *id*	PATTERN 1
R*n*	region n , $n \in [0, 4]$.	R0

In the following, *color* can be the *id* number as well as the the quoted *colorname*.

8.2 Expressions

8.2.1 Elements and operators

GraceGTK can compute the value of algebraic expressions and also concatenate strings using the '.' operator. The following types of expressions are defined:

Name	Description	Examples
expr	Any numeric expression	1.5 + sin(2)
iexpr	Any expression that evaluates to an integer	25, the integer cast of 0.1 + 1.9, PI/asin(1)
nexpr	Non-negative *iexpr*	2 - 1
indx	Non-negative *iexpr*	
vexpr	Vector expression	"2*x"
mexpr	Matrix expression	mx[][0:ylen-1] = mesh(0,2*pi,360)
sexpr	String expression	"a string", "a " . "string", "square root of 4 = " . sqrt(4)

The following operators can be used in numerical expressions:

+	addition
-	substraction
*	multiplication
/	division
%	modulus
^	raising to power

Arithmetic operators

AND	or	&&
OR	or	||
NOT	or	!

Logical operators

EQ	or	==	equal
NE	or	!=	not equal
LT	or	$<$	less than
LE	or	$<=$	less than or equal
GT	or	$>$	greater than
GE	or	$>=$	greater than or equal

Comparison operators

Another conditional operator is the "? :" (or ternary) operator, which operates as in C and many other languages:

$$(expr1) \ ? \ (expr2) \ : \ (expr3);$$

This expression evaluates to $expr2$ if $expr1$ evaluates to `TRUE`, and $expr3$ if $expr1$ evaluates to `FALSE`.

8.2.2 Grammar rules

In the following, let us write `a` for scalars, `m ,n` for integers positive or nul, `u, v, w` for vectors and `M`, for matrices.

- Algebra for scalars obeys the usual rules.
- For homogeneous vectors or matrices expressions operators are applied element by element, i.e. $v * w$ is not the scalar product of the two vectors, but produces a vector w with $w_i = u_i \, v_i \quad \forall i$
- For vectors or matrices expressions with a scalar, the operator applies the scalar to all the elements of the array.

For assignations, the not obvious following rules are defined:

- `w = v[m:n]`
- `M[][n] = v` and `M[n][] = v`
- `M[m:n][] = v` and `M[][m:n] = v`
- `v = M,  M = v` and `M = a`

If you have a doubt or want to add new rules, the `Grammar rules` section of file `pars.yacc` is the good place.

8.3 Functions

8.3.1 Elementary functions

The list is given in table 5.

Function	Description
abs(x)	absolute value
acos(x)	arccosine
acosh(x)	hyperbolic arccosine
asin(x)	arcsine
asinh(x)	hyperbolic arcsine
atan(x)	arctangent
atan2(y, x)	arc tangent of two variables
atanh(x)	hyperbolic arctangent
ceil(x)	greatest integer function
cos(x)	cosine
cosh(x)	hyperbolic cosine
exp(x)	e^x
fac(n)	factorial function, n!
floor(x)	least integer function
irand(n)	random integer less than n
ln(x)	natural log
log10(x)	log base 10
log2(x)	base 2 logarithm of x
maxof(x, y)	returns greater of x and y
mesh(n)	mesh array $(0 \ldots, n-1)$
mesh(x_1, x_2, n)	mesh array of n equally spaced points between x_1 and x_2 inclusive
minof(x, y)	returns lesser of x and y
mod(x, y)	mod function (also x % y)
rand	pseudo random number distributed uniformly on (0.0,1.0)
rand(n)	array of n random numbers
rint(x)	round to closest integer
rsum(x)	running sum of x
sgn(x)	signum function
sin(x)	sine function
sinh(x)	hyperbolic sine
sqr(x)	x^2
sqrt(x)	x^0.5
tan(x)	tangent function
tanh(x)	hyperbolic tangent

Table 5: Elementary functions

8.3.2 Min, max and others

Function	Description
MIN(x)	min value of array x
MAX(x)	max value of array x
AVG(x)	average of array x
SD(x)	standard deviation of array x
SUM(x)	sum of all elements of array x
INT(x, y)	integral of $y\ dx$
IMIN(x)	index of min value of array x
IMAX(x)	index of max value of array x

8.3.3 Statistical functions

The list is given in table 6.

Function	Description
`chdtr`(df, x)	chi-square distribution
`chdtrc`(v, x)	complemented Chi-square distribution
`chdtri`(df, y)	inverse of complemented Chi-square distribution
`erf`(x)	error function
`erfc`(x)	complement of error function
`fdtr`$(df1, df2, x)$	F distribution function
`fdtrc`(x)	complemented F distribution
`fdtri`(x)	inverse of complemented F distribution
`gdtr`(a, b, x)	gamma distribution function
`gdtrc`(a, b, x)	complemented gamma distribution function
`ndtr`(x)	Normal distribution function
`ndtri`(x)	inverse of Normal distribution function
`norm`(x)	gaussian density function
`pdtr`(k, m)	Poisson distribution
`pdtrc`(k, m)	complemented Poisson distribution
`pdtri`(k, y)	inverse Poisson distribution
`rnorm`$(xbar, s)$	pseudo random number distributed $N(xbar, s)$
`stdtr`(k, t)	Student's t distribution
`stdtri`(k, p)	functional inverse of Student's t distribution

Table 6: Statistical functions

8.3.4 Special math functions

The list is given in table 7.

Function	Description
`ai`(x), `bi`(x)	Airy functions (two independent solutions of the differential equation $y''(x) = xy$)
`beta`(x)	beta function
`chi`(x) `ci`(x)	hyperbolic cosine integral cosine integral
`dawsn`(x)	Dawson's integral
`ellie`(ϕ, m) `ellik`(ϕ, m) `ellpe`(m) `ellpk`(m)	incomplete elliptic integral of the second kind incomplete elliptic integral of the first kind complete elliptic integral of the second kind complete elliptic integral of the first kind
`expn`(n, x)	exponential integral
`fresnlc`(x) `fresnls`(x)	cosine Fresnel integral sine Fresnel integral
`gamma`(x)	gamma function
`hyp2f1`(a, b, c, x) `hyperg`(a, b, x)	Gauss hyper-geometric function confluent hyper-geometric function
`i0e`(x) `i1e`(x)	modified Bessel function of order zero, exponentially scaled modified Bessel function of order one, exponentially scaled
`igam`(a, x) `igamc`(a, x) `igami`(a, p)	incomplete gamma integral complemented incomplete gamma integral inverse of complemented incomplete gamma integral
`incbet`(a, b, x) `incbi`(a, b, y)	incomplete beta integral Inverse of incomplete beta integral
`iv`(v, x) `jv`(v, x) `k0e`(x) `k1e`(x) `kn`(n, x)	modified Bessel function of order v Bessel function of order v modified Bessel function, third kind, order zero, exponentially scaled modified Bessel function, third kind, order one, exponentially scaled modified Bessel function, third kind, integer order
`lbeta`(x)	natural log of |`beta`(x)|
`lgamma`(x) `psi`(x) `rgamma`(x)	log of gamma function psi (digamma) function reciprocal gamma function
`shi`(x) `si`(x)	hyperbolic sine integral sine integral
`spence`(x)	dilogarithm
`struve`(v, x)	Struve function
`voigt`(γ, σ, x)	Voigt function (convolution of Lorentzian and Gaussian)
`yv`(v, x)	Bessel function of order v
`zeta`(x, q) `zetac`(x)	Riemann zeta function of two arguments Riemann zeta function

Table 7: Special math functions

8.4 Procedures to transform and create new sets

Methods of directly manipulating the sets and their data, corresponding to simple actions accessed trough `Edit/Set operations` and `Data/Data sets operations` are given below

Statement	Description	Example
`COPY` *src* `TO` *dest*	Copies *src* to *dest*	`COPY S0 TO S1`
`MOVE` *src* `TO` *dest*	Moves *src* to *dest*	`MOVE G0.S0 TO G1.S0`
`SWAP` *src* `AND` *dest*	Interchanges *src* and *dest*	`SWAP G0.S0 AND G0.S1`
`KILL` *set*	Kills *set*	`KILL G0.S0`
`REVERSE` *set*	reverse abscissas of *set*	`REVERSE G0.S0`
`SORT` *set*, *col*, *dir*	Sort points with respect to *col* and direction *dir*	`SORT G0.S0 Y ASCENDING`
set `POINT` *x*, *y*	Add a point at the end of *set*	`G0.S0 POINT 10.0 ,5.0`
set `DROP` *n*1, *n*2	Drop points *n*1 to *n*2	`G0.S0 DROP 5 , 10`
`APPEND` *set*2 `TO` *set*1	Append sets from the same graph	

To `SORT`, the *col* can be `X, Y`, the *dir* `ASCENDING, DESCENDING`

The more sophisticated operations accessed be `Data/Transformation` menu are described in table 8.
To evaluate expressions, you can directly submit them to the command interpreter like you would in the evaluate expression window.
As an example, S1.X = S1.X * 0.000256 scales the x-axis coordinates. The functionality of the 'Sample points' menu entry can be obtained through RESTRICT.

Not finished yet...

RUNAVG(*set*, *npoints*)	running average of *set* using *npoints* number of points	RUNAVG (S0, 100)
RUNMED(*set*, *npoints*)	running median of *set* using *npoints* number of points	RUNMED (S0, 100)
RUNMIN(*set*, *npoints*)	running minimum of *set* using *npoints* number of points	RUNMIN (S0, 100)
RUNMAX(*set*, *npoints*)	running maximum of *set* using *npoints* number of points	RUNMAX (S0, 100)
RUNSTD(*set*, *npoints*)	running standard deviation of *set* using *npoints* number of points	RUNSTD (S0, 100)
INTERPOLATE(*set*, *mesh*, *method*, *strict*)	interpolate *set* on a sampling *mesh* using *method*. *strict* flag controls whether result should be bound within the source set	INTERPOLATE (S0, S1.X, ASPLINE, OFF)
XYSPLINE(*set*, *s*, *p*)	interpolate *set* using XY-splines with parameters *s* and *p*. (see Interpolation/Splines)	XYSPLINE (G1.s0 ,0.5 ,0.002)
HISTOGRAM(*set*, *bins*, *cumulative*, *normalize*)	calculate histogram of *set* on defined *bins*. *cumulative* and *normalize* flags control whether to calculate cumulative and normalized (aka PDF) histograms, respectively. Data points are placed at upper limit of the bin	HISTOGRAM(S0, MESH(0, 1, 11), OFF, ON)
XCOR(*set*1, *set*2, *maxlag*, *covar*)	calculate cross-correlation (or -covariance if the *covar* flag is set) of *set1* with *set2* with maximum lag *maxlag*.	XCOR (S0, S0, 50, OFF)
LINCONV(*set*1, *set*2)	calculate linear convolution of the array of ordinates of *set1* with that of *set2*.	LINCONV (S0, S1)
RESTRICT(*set*, *restriction*)	filter *set* according to logical *restriction*. The original set will be overwritten	RESTRICT (S0, S0.X < 0)
RESTRICT(*set*, *region*, *negate*)	filter *set* by keeping only points lying inside/outside *region*. The original set will be overwritten	RESTRICT (S0, R1, OFF)
NONLFIT(*set*, *nsteps*)	Non linear fit on set *set* with *nsteps* steps. See Data menu Non linear fit	
NONLFIT(*set*, *wheight*, *nsteps*)	*idem* + a weight array *wheight*	
NONLFIT(*set*, *params*, *nsteps*, *formula*, *trace*)	*params* is an array for starting values of parameters. See examples/ng_nonlfitb.agr	NONLFIT (G0.s0 ,nlparms ,10 ,"y = A0 + A1*x + A2*x^2)" ,ON)
FFT(*set*, *filter*) FFT(*set*, *intyp*, *filter*, *xscale*, *outtyp*) FFT(*set*, *intyp*, *filter*, *xscale*, *outtyp*, *outseg*)	See section (9.4.2)	
WAVELET(*set*, *type*, *stride*, *k*, *invflag*)	see section (10.5)	

The *set*(s) arguments are any set selector, and *npoints* is a *nexpr*. The *mesh* argument is a *vexpr*.
To INTERPOLATE, the *method* can be LINEAR, SPLINE ,ASPLINE, or XSPLINE.
strict, *cumulative*, *normalize*, *covar*, *trace* and *negate* arguments are *onoff*.
restriction is a *vexpr* and *region* a region selector.

Table 8: Commands to transform datasets

8.5 Page loading, sizing, saving and printing

The following commands are used:

Statement	Description	Example
LOAD "*file*"	load project *file*	LOAD "foo.agr"
SAVEALL "*file*"	save project to *file*	SAVEALL "foo.agr"
PRINT	execute print job	PRINT

For producing "hard copy", several parameters can be set via the command interpreter. They are summarized in table 9.

Command	Description
PAGE SIZE *xdim*, *ydim*	set page dimensions of all devices (in pp)
PAGE SIZE "*stdsize*", *orientation*	set page dimensions of all devices: *stdsize* can be "Letter" ,"A0", ... "A9", "B0",...,"B9" (into double quotes) *orientation* is `portrait` or `landscape` (without double quotes)
PAGE RESIZE *xdim*, *ydim* PAGE RESIZE "*stdsize*", *orientation*	same as above plus rescale the current plot accordingly
DEVICE "*devname*" PAGE SIZE *xdim*, *ydim*	set page dimensions (in pp) of device*devname*
DEVICE "*devname*" DPI *dpi*	set device's dpi (dots per pixel)
DEVICE "*devname*" FONT *onoff*	enable/disable usage of built-in fonts for device *devname*
DEVICE "*devname*" FONT ANTIALIASING *onoff*	enable/disable font aliasing for device *devname*
DEVICE "*devname*" OP "*options*"	set device specific options (see 12.3 (Device-specific settings))
HARDCOPY DEVICE "*devname*"	set device *devname* as current hardcopy device
PRINT	execute print job
PRINT TO "*filename*"	set print output to *filename* (but do not print)
PRINT TO DEVICE	set print output to hardcopy device (but do not print)

Table 9: Commands to set device parameters

8.6 Flow control and GUI interaction

Statement	Description	Types	Example
EXIT(*status*)	cause normal program termination with exit status *status*	iexpr *status*	EXIT(0)
EXIT	cause normal program termination; same as EXIT(0)		EXIT
HELP *url*	open a HTML document pointed to by *url*	sexpr *url*	HELP "doc/FAQ.html"
HELP	open User's Guide		HELP
REDRAW	refresh the canvas to reflect the current project state		REDRAW
SLEEP *n*	sleep for *n* seconds	expr *n*	SLEEP(3)

8.7 Graphs

8.7.1 General graphs operation

See table 10.

Command — Description	Example
ARRANGE(*nrows*, *ncols*, *offset*, *hgap*, *vgap*) Arrange existing graphs (or add extra if needed) to form an *nrows* by *ncols* matrix, leaving *offset* at each page edge with *hgap* and *vgap* relative horizontal and vertical spacings nexpr *nrows*, *ncols*, expr *offset*, *hgap*, *vgap*	ARRANGE(2, 2, 0.1, 0.15, 0.2)
ARRANGE(*nrows*, *ncols*, *offset*, *hgap*, *vgap*, *hvinv*, *hinv*, *vinv*) Same as above, plus additional *hvinv*, *hinv*, and *vinv* flags allowing to alter the order of the matrix filling nexpr *nrows*, *ncols*, expr *offset*, *hgap*, *vgap*, onoff *hvinv*, *hinv*, *vinv*	ARRANGE(2, 2, 0.1, 0.15, 0.2, ON, OFF, ON)
ARRANGE(*nrows*, *ncols*, *offset*, *hgap*, *vgap*, *hvinv*, *hinv*, *vinv*, *snake*) Same as above, plus additional *snake* flag allowing to fill the matrix in a snake-like fashion nexpr *nrows*, *ncols*, expr *offset*, *hgap*, *vgap*, onoff *hvinv*, *hinv*, *vinv*, *snake*	ARRANGE(2, 2, 0.1, 0.15, 0.2, ON, OFF, ON, ON)
AUTOSCALE — scale the graph	AUTOSCALE
AUTOSCALE XAXES — scale the graph in x only	AUTOSCALE XAXES
AUTOSCALE YAXES — scale the graph in y only	AUTOSCALE YAXES
AUTOSCALE *set* — scale to a specific set	AUTOSCALE SO
AUTOTICKS — autotick all axes	AUTOTICKS
FOCUS *graph* Makes *graph* current and unhides it if necessary	FOCUS G0
KILL *graph* — Kills *graph*	KILL G0
WITH *graph* — Makes *graph* current	WITH G0
TYPE *type* — Sets *type* of current graph *graph* TYPE *type* — Sets *type* of *graph*	TYPE XY (see table 2) G0 TYPE XYDY
graph onoff — (De)Activates selected *graph*	G0 ON
graph HIDDEN *onoff* — Hides selected *graph*	G1 HIDDEN TRUE
graph AUTOSCALE TYPE *atype* — Sets autoscale type	G1 AUTOSCALE TYPE SPEC

Table 10: Commands for general graphs operation

8.7.2 Axis parameters

See table 11.

8.7.3 Titles and legends

See table 12.

Command — Description	Example
AUTOSCALE ONREAD *type* Set automatic scaling on read according to *type*	AUTOSCALE ONREAD NONE
WORLD $x_{min}, y_{min}, x_{max}, y_{max}$ Sets world scaling	WORLD -1.0 ,-1.0 ,1.0 ,10.0
WORLD XMIN *xmin* Sets minimum value of current graph's x axis to *xmin*	WORLD XMIN -10
WORLD XMAX *xmax* Sets maximum value of current graph's x axis to *xmin*	WORLD XMAX 22.5
VIEW XMIN *xmin* Sets left edge of current graph at x=*xmin* in the viewport	VIEW XMIN .2
VIEW XMAX *xmax* Sets right edge of current graph at x=*xmax* in the viewport	VIEW XMAX 1.0
VIEW $x_{min}, y_{min}, x_{max}, y_{max}$ Sets graph's viewport	VIEW 0.15, 0.15, 1.15, 0.85
XAXES SCALE *type* — Set scaling of the x axes to *type*	XAXES SCALE NORMAL
XAXES INVERT onoff If ON, draws xmin to xmax from right to left	XAXES INVERT OFF
XAXIS onoff — Set axis visibility	XAXIS ON
XAXIS TYPE ZERO onoff — Set if axis ordinate is zero	XAXIS TYPE ZERO FALSE

Table 11: Commands to set axis parameters

Command	Description	Example
TITLE *title*	Sets the title of current graph	TITLE "Foo"
TITLE FONT *font*	Selects font of title string	TITLE FONT 1
TITLE SIZE *size*	Sets size of title string	TITLE SIZE 1.5
TITLE COLOR *color*	Sets color of title string	TITLE COLOR 1
SUBTITLE *subtitle*	Sets the subtitle of current graph	SUBTITLE "Bar"
SUBTITLE FONT *font*	Selects font of subtitle string	SUBTITLE FONT "Times-Italic"
SUBTITLE SIZE *size*	Sets size of subtitle string	SUBTITLE SIZE .60
SUBTITLE COLOR *color*	Sets color of subtitle string	SUBTITLE COLOR "blue"
LEGEND onoff	Toggle legend display	LEGEND ON
LEGEND LOCTYPE *type*	Position legend in *type* coordinates	LEGEND LOCTYPE WORLD
LEGEND *xloc, yloc*	Set location of legend box (upper left corner)	LEGEND .5,.75
LEGEND FONT *font*	Set legend font type	LEGEND FONT "Helvetica"
LEGEND CHAR SIZE *size*	Sets size of legend label characters (1 is normal)	LEGEND CHAR SIZE .30
LEGEND *color*	Set color of legend text	LEGEND COLOR 1
LEGEND VGAP *gap*	Sets vertical gap between legend entries	LEGEND VGAP 1
LEGEND HGAP *gap*	Sets horizontal gap between symbol and description	LEGEND HGAP 4
LEGEND LENGTH *length*	Sets *length* of legend	LEGEND LENGTH 5
LEGEND INVERT onoff	Determines relationship between order of sets and order of legend labels	LEGEND INVERT true
LEGEND BOX onoff	Determines if the legend bounding box is drawn	LEGEND BOX off
LEGEND BOX COLOR *color*	Sets color of legend bounding box	LEGEND BOX COLOR 1
LEGEND BOX PATTERN *pattern*	Sets pattern of legend bounding box	LEGEND BOX PATTERN 2
LEGEND BOX LINESTYLE *style*	Sets line style of bounding box	LEGEND BOX LINESTYLE 1
LEGEND BOX LINEWIDTH *width*	Sets line width of bounding box	LEGEND BOX LINEWIDTH 1.5
LEGEND BOX FILL onoff	Determines if the legend bounding box is filled	LEGEND BOX FILL false
LEGEND BOX FILL COLOR *color*	Sets color of legend box fill	LEGEND BOX COLOR 3
LEGEND BOX FILL *pattern*	Sets pattern of legend box fill	LEGEND BOX FILL PATTERN 1

Table 12: Commands for graphs titles and legend

8.8 Sets

8.8.1 General sets operation

See table 13.

Command	Description	Example
READ *file*	Reads *file* as a single set	READ "foo.dat"
READ *settype file*	Reads *file* into a single set of type *settype*	READ xydy "bar.dat"
READ NXY *file*	Reads *file* as NXY data	READ NXY "gad.dat"
READ BLOCK *file*	Reads *file* as block data	READ BLOCK "zooks.dat"
KILL BLOCK	Kills the current block data and frees the associated memory	KILL BLOCK
BLOCK *settype columns*	Forms a data set of type *settype* using *columns* from current block data file.	BLOCK xydxdy "0:2:1:3"
set TYPE *settype*	Defines the type of the set	G0.s0 TYPE XY
WRITE *set*	writes *set* to stdout	WRITE G0.S1
WRITE *set* FORMAT *formatstring*	writes *set* to stdout using format specification *formatstring*	WRITE G0.S1 FORMAT "%18.8g"
WRITE *set* FILE *file*	writes *set* to *file*	WRITE G0.S1 FILE "data.dat"
WRITE *set* FILE *file* FORMAT *formatstring*	writes *set* to *file* using format specification *formatstring*	WRITE G0.S1 FILE "data.dat" FORMAT "%18.8g"
COPY *src* TO *dest*	Copies *src* to *dest*	COPY S0 TO S1
MOVE *src* TO *dest*	Moves *src* to *dest*	MOVE G0.S0 TO G1.S0
SWAP *src* AND *dest*	Interchanges *src* and *dest*	SWAP G0.S0 AND G0.S1
KILL *set*	Kills *set*	KILL G0.S0

Table 13: Commands for set operation

Note that XYCMAP is a special case: *set* TYPE XYCMAP *nx* , *ny*

8.8.2 Commands for XYCMAP sets

This type of sets implies some changes compared to the rules for other types of sets.
First the command defining the type of the set is

selectset `TYPE XYCMAP` *nx ,ny*

where nx and ny are the width and height in number of points of the rectangular grid.
Second, the *Colorbar* is an object (see table 17, of course not all command are pertinent) with special commands. Also, it is possible to choose the colormap used.

Command	Description	Remarks
set `COLORBAR ZSCALE` $z1, z2$	the scale in z	
set `COLORBAR NTICKS` n	the number of ticks and level for contours	
set `COLORBAR AUTOSCALE`	computes the z scale for the colors	
set `CMAP` n *set* `CMAP` *"name"*	choose the color map choose the color map	

Color map 0 is the current `GraceGTK` colormap used when you fix a line, symbol or pattern color. The other names of color maps are borrowed to Scilab (http://www.scilab.org): *"Jet colormap", "Hot colormap", "Grey colormap", "Winter colormap", "Spring colormap", "Summer colormap", "Autumn colormap", "Rainbow colormap", "Bone colormap", "Copper colormap", "Ocean colormap" and "White colormap".*
Contour curves use the commands in table 14:

Command	Description	Remarks
set `CONTOUR` *onoff*	show/hide	`G1.s0 CONTOUR ON`
set `CONTOUR COLOR` < *n* | *"colorname"* >	show/hide	`G1.s0 CONTOUR COLOR "white"`
set `CONTOUR LINEWIDTH` *width*		
set `CONTOUR LABEL` x, y, z	define a label on a curve	

Note that the `LABEL` command is not designed for user, but for internal (save) use: if no curve crosses the point, it is ineffective.

Table 14: Command defining parameters of contour curves

8.9 Geometric objects

This section deal about arcs, boxes, lines, polylines and circles. Compared to `grace-5.1.22` The `ELLIPSE` object is replaced by the more general `ARC`.
Let us define *objtyp* and *selobj* .

Name	Values	Remarks
objtyp	ARC, BOX, LINE ,POLYLINE ,STRING	
selobj	*objtyp id*	for update

Table 15: *objtyp* and *selobj* definition

These non terminal symbols can be used to create new objects, make an object current or update objects parameters. N.B.

Command	Action	Example
WITH *objtyp*	Creates a new object	`WITH Line`
WITH *selobj*	The object *id* is made current	`WITH Box 2`

Table 16: WITH command

- Some parameters have no meaning when used with some values of *selobj*, but they will be accepted by the interpreter without any message.
- Some old `grace-5.1.22` commands with *objtyp* in place of *selobj* are not in the table but are still working.

Commands are summarized in table 17.
Anyway, you are encouraged to hack the examples files and use the follow-me mode to write your own scripts. In particular, Fourier examples show how to create sets from formulas.

Command	Description	Remarks	*selobj*				
			Arc	Box	Line	Poly	String
selobj : $x1, y1, x2, y2$	Object coordinates	the : is needed	x	x	x	x	
selobj ANCHOR $x1, y1$							x
selobj "the string itself"							x
selobj sexpr	Name of the object		x	x	x	x	
selobj ARROW n	0:no arrows, 1:at start, 2:end, 3:both		x		x	x	
selobj ARROW LENGTH *length*			x		x	x	
selobj ARROW TYPE *type*			x		x	x	
selobj ARROW LAYOUT a, b			x		x	x	
selobj CHAR SIZE *size*					x		x
selobj COLOR *color*			x	x	x	x	x
selobj FILL COLOR *color*				x			x
selobj FILL PATTERN *color*				x			x
selobj FONT n	Font selection				x		x
selobj HIDDEN *onoff*			x	x	x	x	x
selobj JUST n	justification						x
selobj LINESTYLE *style*	*style* 0: no line, 1: continuous, etc		x	x	x	x	x
selobj LINEWIDTH *width*			x	x	x	x	x
selobj LOCTYPE *worldview*	Object coordinates are in world ones or in view ones		x	x	x	x	x
selobj ROT *angle*	Rotation	in degrees					x
selobj TYPE n	Arc subtype		x				
NEW COMPOUND *selobj*	Create a new compound with *selobj* as first element		x	x	x	x	x
COMPOUND *id selobj*	Add *selobj* to compound *id*		x	x	x	x	x
COMPOUND *id* HIDDEN *onoff*							
COMPOUND *id* LOCTYPE *worldview*							
COMPOUND *id* BREAK	Breaks the compound						
COMPOUND *id* : $x1, y1, x2, y2$	Object coordinates: up to now only $(x1, y1)$ are used (for translation, $(x2, y2)$ are reserved for scaling)	the : is needed					

Table 17: Commands related to geometrical objects.

9 Details on Fourier transformation in GraceGTK

9.1 Few words about FFTW

In short, this package allows one to do non-power-of-2 length FFT's along with the normal ones. It seems to work very efficiently for any set length which factors into $2^a 3^b 5^c 7^d$ for integer a, b, c, d. The great feature here is that set lengths which are powers of 10 (e.g. 1000, 10000) and integer multiples of these (500, 2000, 2500, 5000, etc.) can be computed with no significant penalty (maybe 20%) over power-of-2 transforms. Very often, real datasets come in these sizes, and not in powers of 2.

9.2 Classical Fourier transformation

Let us define the Fourier integral by

$$F(X) \;=\; a \int_{-\infty}^{+\infty} f(x)\, \exp(\pm icXx)\, dx \quad \Leftrightarrow \quad F \;=\; \mathcal{F}^{\pm}_{a,c}\, f, \tag{1}$$

where a and c are two real constants and $i^2 = -1$. Various values for the two constants and the sign in the exponential are commonly used.

9.3 Discrete Fourier Transform

The discrete Fourier transform (DFT) is defined in FFTW library by

$$\tilde{F}_k \;=\; \sum_{j=0}^{N-1} f_j\, \exp\left(\pm i2\pi\, \frac{k}{N}\, j\right). \tag{2}$$

Let us identify (2) with a discrete computation of (1) for a function $\tilde{f}$ null outside $[0, N-1]$, with $f_j = \tilde{f}(j)$ and $\tilde{X}_k = k/N$:

$$\tilde{F}(\tilde{X}) \;=\; \int_{-\infty}^{+\infty} \tilde{f}(\tilde{x}) \exp\left(\pm i2\pi\, \tilde{X}\tilde{x}\right)\, d\tilde{x}. \quad \Leftrightarrow \quad \tilde{F} = \mathcal{F}^{\pm}_{1,2\pi}\, \tilde{f} \tag{3}$$

Let us now connect $\tilde{F}$ to the wanted transform F.
Noting $x = A + b\,\tilde{x}, \quad b > 0$ and substituting $\tilde{x}$ to x into (1), we get

$$F(X) \;=\; a \int_{-\infty}^{+\infty} f(A + b\tilde{x})\, \exp[\pm icX(A + b\,\tilde{x})]\, b\, d\tilde{x} \tag{4}$$

$$\;=\; ab\, \exp(\pm icAX) \int_{-\infty}^{+\infty} \tilde{f}(\tilde{x})\, \exp(\pm ibcX\, \tilde{x})\, d\tilde{x} \tag{5}$$

We can identify (3) with the integral in (5) if $bcX \equiv 2\pi\, \tilde{X}$.
Thus,

$$X_k = \frac{2\pi}{bc}\, \frac{k}{N} \quad \Rightarrow \quad F(X_k) \;=\; ab\, \exp(\pm icAX_k)\, \tilde{F}_k \tag{6}$$

For a function f null outside $[A, B]$, the coefficient b is given by

$$B = A + b(N-1) \quad \Rightarrow \quad b = (B-A)/(N-1).$$

9.4 Fourier transform in GraceGTK

9.4.1 Choice of parameters

GraceGTK interface gives the following choices.

- *Forward* transform: the sign in the exponential is minus.
- *Backward* transform: the sign in the exponential is plus.
- *Input segment*: as explain above, classical Fourier transforms and DFT are related but not identical. If you choose
  - *Index*, you compute only the DFT,
  - *[Xmin ,Xmax]*, you compute the classical Fourier transform
- X-scale: allows to choose a and c in formula (1):

  - *index* means $a = 1,\ c = 1$,
  - *frequency* means $a = 1,\ c = 2\pi$
  - *angular frequency* means $a = 1/\sqrt{2\pi},\ c = 1$.
- *Output segment*
  The DFT is periodical, i.e. $\tilde{F}_{k+N} = \tilde{F}_k$, thus it is possible to choose the abscissas interval for the computed transform. Moreover, the transform of a real function shows the Hermitian symmetry, i.e. $\tilde{F}_{N-k} = \tilde{F}_k^*$ and in that case the full period is not needed, thus the program offers three possible answers: *"Positive", "Positive half length"* and *"Centered"*.

9.4.2 Call in GraceGTK scripts

Three forms are possibles, the first is not compatible with Grace-5.

1. `FFT(set ,intyp ,filter ,xscale ,outtyp ,outseg)`
2. `FFT(set ,intyp ,filter ,xscale ,outtyp)`
3. `FFT(set ,filter)`

The arguments are:

- `set`: the set ident, e.g. `G1.S0`
- `intyp`: `REAL` or `COMPLEX`
- `filter`: `NONE, TRIANGULAR, WELCH, HANNING, HAMMING` or `PARZEN`
- `xscale`: `INDEX, FREQUENCY` or `PERIOD`
- `outtyp`: `MAGNITUDE, PHASE, COEFFICIENTS` or `COMPLEX`
- `outseg`: `POSITIVE, HALF` or `CENTER`

9.4.3 Examples

Two examples are given in doc/examples directory:

`ng_fourier.agr` and `ng_fourier2.agr`

and can be launched via `Examples GraceGTK` menu.

9.5 FFTW tuning

When the `FFTW` capabilities are compiled in, Grace looks at two environment variables to decide what to do with the `FFTW` 'wisdom' capabilities. First, a quick summary of what this is. The FFTW package is capable of adaptively determining the most efficient factorization of a set to give the fastest computation. It can store these factorizations as 'wisdom', so that if a transform of a given size is to be repeated, it does not have to re-adapt. The good news is that this seems to work very well. The bad news is that, the first time a transform of a given size is computed, if it is not a sub-multiple of one already known, it takes a LONG time (seconds to minutes).

The first environment variable is `GRACE_FFTW_WISDOM_FILE`.

- If this is set to the name of a file which can be read and written (e.g., `$HOME/.grace_fftw_wisdom`)then Grace will automatically create this file (if needed) and maintain it.
- If the file is read-only, it will be read, but not updated with new wisdom.
- If the symbol `GRACE_FFTW_WISDOM_FILE` either doesn't exist, or evaluates to an empty string, Grace will drop the use of wisdom, and will use the fftw estimator (`FTW_ESTIMATE` flag sent to the planner) to guess a good factorization, instead of adaptively determining it.

The second variable is `GRACE_FFTW_RAM_WISDOM`. If this variable is defined to be non-zero, and `GRACE_FFTW_WISDOM_FILE` variable is not defined (or is an empty string), Grace will use wisdom internally, but maintain no persistent cache of it. This will result in very slow execution times the first time a transform is executed after Grace is started, but very fast repeats. I am not sure why anyone would want to use wisdom without writing it to disk, but if you do, you can use this flag to enable it.

10 Wavelet transformations in GraceGTK

The code is based on the GNU Scientific Library (GSL) http://www.gnu.org/software/gsl/. If the library is not installed, a bundled small part of the library is used. The first subsection reproduce here the introductory section of GSL documentation.

10.1 Continuous and discrete wavelet transforms

The continuous wavelet transform and its inverse are defined by the relations,

$$w(s,\tau) = \int_{-\infty}^{\infty} f(t) * \psi^*_{s,\tau}(t)dt$$

and

$$f(t) = \int_0^{\infty} ds \int_{-\infty}^{\infty} w(s,\tau) * \psi_{s,\tau}(t)d\tau$$

where the basis functions $\psi_{s,\tau}$ are obtained by scaling and translation from a single function, referred to as the *mother wavelet*.
The discrete version of the wavelet transform acts on equally-spaced samples, with fixed scaling and translation steps (s,τ). The frequency and time axes are sampled dyadically on scales of 2^j through a level parameter j. The wavelet ψ can be expressed in terms of a scaling function φ,

$$\psi(2^{j-1},t) = \sum_{k=0}^{2^j-1} g_j(k) * \bar{\varphi}(2^j t - k)$$

and

$$\varphi(2^{j-1},t) = \sum_{k=0}^{2^j-1} h_j(k) * \bar{\varphi}(2^j t - k)$$

The functions ψ and φ are related through the coefficients

$$g_n = (-1)^n h_{L-1-n}, \qquad g_n = (-1)^n h_{L-1-n}, \qquad n = 0 \ldots L-1,$$

where L is the total number of coefficients. The two sets of coefficients h_j and g_i define the scaling function and the wavelet.
The centered forms of the wavelets align the coefficients of the various sub-bands on edges. Thus the resulting visualization of the coefficients of the wavelet transform in the phase plane is easier to understand.
More details and references can be found in GSL documentation and Wikipedia. Also, the *Numerical recipes* [2] gives a comprehensive account on the subject.

10.2 Wavelets families

Available families in GSL are the following.

Daubechies This is the Daubechies wavelet family of maximum phase with $k/2$ vanishing moments. See http://en.wikipedia.org/wiki/Daubechies_wavelet

Haar Simple, but discontinuous. See http://en.wikipedia.org/wiki/Haar_wavelet

Bspline This is the biorthogonal B-spline wavelet family of order (i,j). The k parameter is defined by $k = 100\,i + j$.

10.3 Wavelets parameters

GLS allows only the following values for the wavelet parameter k for the different families of wavelet:

Daubechies $k = 4, 6, \ldots, 20$ with k even,

Haar the only legal value is 2,

Bspline $k = 103, 105, 202, 204, 206, 208, 301, 303, 305, 307, 309$.

10.4 Some hints

- The main interest of wavelet transforms is to compress the information into a small number of coefficients. If the wavelet and its parameters are well chosen, a dense vector v gives a vector w with a small number of significant components. In `GraceGTK` , if v has a large number of components, w have the same number of components i.e. is also large. In order to save only the useful information, selecting the set of wavelet coefficients, you can
  - pop up `Data/Transformation/Sample points..` menu,
  - choose `Sample type: Expression`

- give formula `y = (abs(y) > xx) ? y : 0;` where `xx` is the threshold of your choice
- and thus create a new set that you can save with `File/Export ASCII data` .
  The abscissas of the new set are the index of the components above threshold, and the ordinates the significant components of w.

- Beware that the abscissa range for the transformed set is the index and that the x scale is often different from the one of the original data. Thus creating a new graph and moving the coefficients to this graph is often a good option.
- Remember also that if you wont to retrieve the original data with same scale by backward transformation, you will have to rescale abscissas properly.
- If your data have not 2^j points, it is possible (but not very satisfactory) to interpolate it on a regular mesh before taking the wavelet transform.

10.5 Call in GraceGTK scripts

The following function creates a new set with the wavelet coefficients:
`WAVELET (`$set, type, stride, k, invflag$`)`
where

set is the set selector for the input vector (e.g. `G0.s0`),

type is one of the following
`DAUBECHIES, DAUBECHIES_CENTERED, HAAR, HAAR_CENTERED, BSPLINE, BSPLINE_CENTERED`,

stride is the stride to use in the input data vector (1 means all the components)

k is the parameter described above,

invflag 0: forward, 1: backward transform.

Example : `@ WAVELET (s_ ,DAUBECHIES ,1 ,4 ,0 ,0)`

11 Advanced topics

11.1 Fonts

11.1.1 Font configuration

The file responsible for the font configurations of Grace is `fonts/FontDataBase`. The first line contains a positive integer specifying the number of fonts declared in that file. All remaining lines contain declarations of one font each, composed out of three fields:

1. Font name. The name will appear in the font selector controls. Also, backend devices that has built-in fonts, will be given the name as a font identifier.
2. Font fall-back. Grace will try to use this in case the real font is not found.
3. Font filename. The file with the font outline data.

Here is the default `FontDataBase` file:

```
14
Times-Roman             Times-Roman             n021003l.pfb
Times-Italic            Times-Italic            n021023l.pfb
Times-Bold              Times-Bold              n021004l.pfb
Times-BoldItalic        Times-BoldItalic        n021024l.pfb
Helvetica               Helvetica               n019003l.pfb
Helvetica-Oblique       Helvetica-Oblique       n019023l.pfb
Helvetica-Bold          Helvetica-Bold          n019004l.pfb
Helvetica-BoldOblique   Helvetica-BoldOblique   n019024l.pfb
Courier                 Courier                 n022003l.pfb
Courier-Oblique         Courier-Oblique         n022023l.pfb
Courier-Bold            Courier-Bold            n022004l.pfb
Courier-BoldOblique     Courier-BoldOblique     n022024l.pfb
Symbol                  Symbol                  s050000l.pfb
ZapfDingbats            ZapfDingbats            d050000l.pfb
```

11.1.2 Font data files

For text rastering, three types of files are used.

1. `.pfa`-/`.pfb`-files: These contain the character outline descriptions. The files are assumed to be in the `fonts/type1` directory; these are the filenames specified in the `FontDataBase` configuration file.
2. `.afm`-files: These contain high-precision font metric descriptions as well as some extra information, such as kerning and ligature information for a particular font. It is assumed that the filename of a font metric file has same basename as the respective font outline file, but with the `.afm` extension; the metric files are expected to be found in the `fonts/type1` directory, too.
3. `.enc`-files: These contain encoding arrays in a special but simple form. They are only needed if someone wants to load a special encoding to re-encode a font. Their place is `fonts/enc`

11.1.3 Custom fonts

It is possible to use custom fonts with Grace. One mostly needs to use extra fonts for the purpose of localization. For many European languages, the standard fonts supplied with Grace should contain all the characters needed, but encoding may have to be adjusted. This is done by putting a `Default.enc` file with proper encoding scheme into the `fonts/enc` directory. Grace comes with a few encoding files in the directory; more can be easily found on the Internet. (If the `Default.enc` file doesn't exist, the IsoLatin1 encoding will be used). Notice that for fonts having an encoding scheme in themselves (such as the Symbol font, and many nationalized fonts) the default encoding is ignored.

If you do need to use extra fonts, you should modify the `FontDataBase` file accordingly, obeying its format. However, if you are going to exchange Grace project files with other people who do not have the extra fonts configured, an important thing is to define reasonable fall-back font names.

For example, let us assume I use Hebrew fonts, and the configuration file has lines like these:

```
    ...
Courier-Hebrew              Courier                 courh___.pfa
Courier-Hebrew-Oblique      Courier-Oblique         courho__.pfa
    ...
```

My colleague, who lives in Russia, uses Cyrillic fonts with Grace configured like this:

```
    ...
Cronix-Courier              Courier                 croxc.pfb
Cronix-Courier-Oblique      Courier-Oblique         croxco.pfb
    ...
```

The font mapping information (Font name <-> Font fall-back) is stored in the Grace project files. Provided that all the localized fonts have English characters in the lower part of the ASCII table unmodified, I can send my friend files (with no Hebrew characters, of course) and be sure they render correctly on his computer.

Thus, with properly configured national fonts, you can make localized annotations for plots intended for internal use of your institution, while being able to exchange files with colleagues from abroad. People who ever tried to do this with MS Office applications should appreciate the flexibility :-).

11.1.4 `GraceGTK` internal encoding

GTK API assume a UTF-8 encoding and `grace-5.1.22` programming is Latin1 by default thus, the strings are converted back and forth between the two when displayed through a GTK widget[19]. Note that only strings that are normally drawn on the canvas and can contains non ASCII characters are concerned. This can produce strange results if you use a different coding scheme.
The font tool use a workaround: character in the 128-255 range are coded with their number (e.g. `\#{e0}`) and decoded by the command interpreter.

11.2 Interaction with other applications

11.2.1 Using pipes

Not finished ...

[19] Except of course for the canvas where characters are drawn through T1lib library.

11.2.2 Using grace_np library

The grace_np library is a set of compiled functions that allows you to launch and drive a Grace subprocess from your C or Fortran application. Functions are provided to start the subprocess, to send it commands or data, to stop it or detach from it.
See table 18 for a list of C functions.
See table 19 for a list of Fortran functions.

There is no Fortran equivalent for the GracePrintf function, you should format all the data and

---

`int GraceOpenVA (char *exe, int buf_size, ...)`
launch a Grace executable *exe* and open a communication channel with it using *buf_size* bytes for data buffering. The remaining `NULL`-terminated list of options is command line arguments passed to the Grace process

`int GraceOpen (int buf_size)`
equivalent to `GraceOpenVA("ggrace", buf_size, "-nosafe", "-noask", NULL)`

`int GraceIsOpen (void)`
test if a Grace subprocess is currently connected

`int GraceClose (void)`
close the communication channel and exit the Grace subprocess

`int GraceClosePipe (void)`
close the communication channel and leave the Grace subprocess alone

`int GraceFlush (void)`
flush all the data remaining in the buffer

`int GracePrintf (const char* format, ...)`
format a command and send it to the Grace subprocess

`int GraceCommand (const char* cmd)`
send an already formated command to the Grace subprocess

`GraceErrorFunctionType GraceRegisterErrorFunction (GraceErrorFunctionType f)`
register a user function *f* to display library errors

Table 18: grace_np library C functions.

commands yourself before sending them with GraceCommandF.

The Grace subprocess listens for the commands you send and interprets them as if they were given in a batch file. You can send any command you like (redraw, autoscale, ...). If you want to send data, you should include them in a command like "g0.s0 point 3.5, 4.2".

Apart from the fact it monitors the data sent via an anonymous pipe, the Grace subprocess is a normal process. You can interact with it through the GUI. Note that no error can be sent back to the parent process. If your application send erroneous commands, an error popup will be displayed by the subprocess.

If you exit the subprocess while the parent process is still using it, the broken pipe will be detected. An error code will be returned to every further call to the library (but you can still start a new process if you want to manage this situation).

Function	Arguments	Description
`integer GraceOpenF`	(integer *buf_size*)	launch a Grace subprocess and open a communication channel with it
`integer GraceIsOpenF`	(void)	test if a Grace subprocess is currently connected
`integer GraceCloseF`	(void)	close the communication channel and exit the Grace subprocess
`integer GraceClosePipeF`	(void)	close the communication channel and leave the Grace subprocess alone
`integer GraceFlushF`	(void)	flush all the data remaining in the buffer
`integer GraceCommandF`	(character*(*) *cmd*)	send an already formatted command to the Grace subprocess
`GraceFortranFunctionType GraceRegisterErrorFunctionF`	(GraceFortranFunctionType *f*)	register a user function *f* to display library errors

Table 19: grace_np library F77 functions.

Here is an example use of the library, you will find this program in the distribution.

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <grace_np.h>

#ifndef EXIT_SUCCESS
#  define EXIT_SUCCESS 0
#endif

#ifndef EXIT_FAILURE
#  define EXIT_FAILURE -1
#endif

void my_error_function(const char *msg)
{
    fprintf(stderr, "library message: \"%s\"\n", msg);
}

int
main(int argc, char* argv[])
{
    int i;

    GraceRegisterErrorFunction(my_error_function);

    /* Start Grace with a buffer size of 2048 and open the pipe */
    if (GraceOpen(2048) == -1) {
        fprintf(stderr, "Can't run Grace. \n");
        exit(EXIT_FAILURE);
    }

    /* Send some initialization commands to Grace */
    GracePrintf("world xmax 100");
    GracePrintf("world ymax 10000");
    GracePrintf("xaxis tick major 20");
    GracePrintf("xaxis tick minor 10");
    GracePrintf("yaxis tick major 2000");
    GracePrintf("yaxis tick minor 1000");
    GracePrintf("s0 on");
    GracePrintf("s0 symbol 1");
    GracePrintf("s0 symbol size 0.3");
    GracePrintf("s0 symbol fill pattern 1");
    GracePrintf("s1 on");
    GracePrintf("s1 symbol 1");
    GracePrintf("s1 symbol size 0.3");
    GracePrintf("s1 symbol fill pattern 1");

    /* Display sample data */
    for (i = 1; i <= 100 && GraceIsOpen(); i++) {
        GracePrintf("g0.s0 point %d, %d", i, i);
        GracePrintf("g0.s1 point %d, %d", i, i * i);
        /* Update the Grace display after every ten steps */
        if (i % 10 == 0) {
            GracePrintf("redraw");
            /* Wait a second, just to simulate some time needed for
               calculations. Your real application shouldn't wait. */
            sleep(1);
        }
    }
```

```
    if (GraceIsOpen()) {
        /* Tell Grace to save the data */
        GracePrintf("saveall \"sample.agr\"");

        /* Flush the output buffer and close Grace */
        GraceClose();

        /* We are done */
        exit(EXIT_SUCCESS);
    } else {
        exit(EXIT_FAILURE);
    }
}
```

To compile this program, type

```
cc example.c -lgrace_np
```

If Grace wasn't properly installed, you may need to instruct the compiler about include and library paths explicitly, e.g.

```
cc -I/usr/local/grace/include example.c -L/usr/local/grace/lib -lgrace_np
```

11.3 DL modules

Grace can access external functions present in either system or third-party shared libraries or modules specially compiled for use with Grace.

11.3.1 Function types

One must make sure, however, that the external function is of one of supported by Grace types:

Grace type	Description
`f_of_i`	a function of 1 `int` variable
`f_of_d`	a function of 1 `double` variable
`f_of_nn`	a function of 2 `int` parameters
`f_of_nd`	a function of 1 `int` parameter and 1 `double` variable
`f_of_dd`	a function of 2 `double` variables
`f_of_nnd`	a function of 2 `int` parameters and 1 `double` variable
`f_of_ppd`	a function of 2 `double` parameters and 1 `double` variable
`f_of_pppd`	a function of 3 `double` parameters and 1 `double` variable
`f_of_ppppd`	a function of 4 `double` parameters and 1 `double` variable
`f_of_pppppd`	a function of 5 `double` parameters and 1 `double` variable

Table 20: Grace types for external functions

11.3.2 Examples

Caution: the examples provided below (paths and compiler flags) are valid for Linux/ELF with gcc. On other operating systems, you may need to refer to compiler/linker manuals or ask a guru.

Example 1 Suppose I want to use function `pow(x,y)` from the Un*x math library (libm). Of course, you can use the "^" operator defined in the Grace language, but here, for the sake of example, we want to access the function directly.
First run `ggrace` and open `Window/Commands`. Paste the command to make `pow` accessible:

```
@ USE "pow" TYPE f_of_dd FROM "/usr/lib/libm.so"
```

select the line and "`Apply`".
We can now compare "`pow`" with "`^`": paste in the command window:

```
@ with g0
@ title "Fonction pow"
@ g0.s0 length 100
@ g0.s0.x = mesh(-2 ,3 ,100)
@ g0.s0.y = x^2
@ g0.s1 length 100
@ g0.s1.x = mesh(-2 ,3 ,100)
@ g0.s1.y = pow(x,2)
@  s1  symbol 8
@  s1  line type 0
@ autoscale
```

select this sequence and "Apply".
Another method to build sets from functions is to use TreeView/Project/Functions menu.

Example 2 Now, let us try to write a function ourselves.
We will define function my_function which simply returns its (second) argument multiplied by integer parameter transferred as the first argument.
In a text editor, type in the following C code and save it as "my_func.c":

```
double my_function (int n, double x)
{
    double retval;
    retval = (double) n * x;
    return (retval);
}
```

OK, now compile it:

```
gcc -c -fPIC my_func.c
gcc -shared my_func.o -o /tmp/my_func.so
```

(You may strip it to save some disk space):

```
strip /tmp/my_func.so
```

That's all! Ready to make it visible to Grace as "myf" - we are too lazy to type the very long string "my_function" many times.

```
USE "my_function" TYPE f_of_nd FROM "/tmp/my_func.so" ALIAS "myf"
```

Example 3 A more serious example. There is a special third-party library available on your system which includes a very important for you yet very difficult-to-program from the scratch function that you want to use with Grace. But, the function prototype is NOT one of any predefined table 20. The solution is to write a simple function wrapper. Here is how:
Suppose, the name of the library is "special_lib" and the function you are interested in is called "special_func" and according to the library manual, should be accessed as
void special_func(double *input, double *output, int parameter).
The wrapper would look like this:

```
double my_wrapper(int n, double x)
{
    extern void special_func(double *x, double *y, int n);
    double retval;
    (void) special_func(&x, &retval, n);
    return (retval);
}
```

Compile it:

```
gcc -c -fPIC my_wrap.c
gcc -shared my_wrap.o -o /tmp/my_wrap.so -lspecial_lib -lblas
strip /tmp/my_wrap.so
```

Note that I added -lblas assuming that the special_lib library uses some functions from the BLAS. Generally, you have to add *all* libraries which your module depends on (and all libraries those libraries rely upon etc.), as if you wanted to compile a plain executable.
Fine, make Grace aware of the new function

```
USE "my_wrapper" TYPE f_of_nd FROM "/tmp/my_wrap.so" ALIAS "special_func"
```

so we can use it with its original name.

Example 4 An example of using Fortran[20] modules.
Here we will try to achieve the same functionality as in Example 2, but with the help of gfortran.

```
      DOUBLE PRECISION FUNCTION MYFUNC (N, X)
      IMPLICIT NONE
      INTEGER N
      DOUBLE PRECISION X
C
      MYFUNC = N * X
C
      RETURN
      END
```

As opposite to C, there is no way to call such a function from Grace directly - the problem is that in Fortran all arguments to a function (or subroutine) are passed by reference. So, we need a wrapper:

```
double myfunc_wrapper(int n, double x)
{
    extern double myfunc_(int *, double *);
    double retval;
    retval = myfunc_(&n, &x);
    return (retval);
}
```

Note that most of f77 compilers by default add underscore to the function names and convert all names to the lower case, hence I refer to the Fortran function `MYFUNC` from my C wrapper as `myfunc_`, but in your case it can be different!
Let us compile the whole stuff:

```
gfortran -c -fPIC myfunc.f
gcc -c -fPIC myfunc_wrap.c
gcc -shared myfunc.o myfunc_wrap.o -o /tmp/myfunc.so -lf2c -lm
strip /tmp/myfunc.so
```

And finally, inform Grace about this new function:

```
USE "myfunc_wrapper" TYPE f_of_nd FROM "/tmp/myfunc.so" ALIAS "myfunc"
```

12 References

12.1 Typesetting

Grace permits quite complex typesetting on a per string basis. Any string displayed (titles, legends, tick marks,...) may contain special control codes to display subscripts, change fonts within the string *etc.*
The escape sequences frequently used can be produced easily with the `Font tool` (see section 5.5). More complex output need the use of sequences in table 21. Example:
F\sX\N(\xe\f{}) = sin(\xe\f{})\#{b7}e\S-X\N\#{b7}cos(\xe\f{})
prints roughly

$$F_x(\epsilon) = \sin(\epsilon).e^{-x}.\cos(\epsilon)$$

using string's initial font (e prints as epsilon from the Symbol font).
Note that an extension named `dvi2gr` and available on Grace site can be used to convert some DVI files produced by LaTeX in Grace strings.

12.2 Device-specific limitations

Grace can output plots using several device backends. The list of available devices can be seen (among other stuff) by specifying the "-version" command line switch.

- GTK, PostScript and EPS are full-featured devices
- Cairo PDF, PNG and SVG drivers have no restrictions different from the Cairo ones but no specific options are available and the output is not at all optimized.

The drivers below does not display correctly xycmap sets.

[20] This example is a little out-of-date because Fortran2003 norm introduce a systematic interface with C called `ISO_C_BINDING` (and of course prefer free form).

Control code	Description
\f{x}	switch to font named "x"
\f{n}	switch to font number n
\f{}	return to original font
\R{x}	switch to color named "x"
\R{n}	switch to color number n
\R{}	return to original color
\#{x}	treat "x" (must be of even length) as list of hexadecimal char codes
\t{xx xy yx yy}	apply transformation matrix
\t{}	reset transformation matrix
\z{x}	zoom x times
\z{}	return to original zoom
\r{x}	rotate by x degrees
\l{x}	slant by factor x
\v{x}	shift vertically by x
\v{}	return to unshifted baseline
\V{x}	shift baseline by x
\V{}	reset baseline
\h{x}	horizontal shift by x
\n	new line
\u	begin underline
\U	stop underline
\o	begin overline
\O	stop overline
\Fk	enable kerning
\FK	disable kerning
\Fl	enable ligatures
\FL	disable ligatures
\m{n}	mark current position as n
\M{n}	return to saved position n
\dl	LtoR substring direction
\dr	RtoL substring direction
\dL	LtoR text advancing
\dR	RtoL text advancing
\x	switch to Symbol font (same as \f{Symbol})
\+	increase size (same as \z{1.19} ; 1.19 = sqrt(sqrt(2)))
\-	decrease size (same as \z{0.84} ; 0.84 = 1/sqrt(sqrt(2)))
\s	begin subscripting (same as \v{-0.4}\z{0.71})
\S	begin superscripting (same as \v{0.6}\z{0.71})
\T{xx xy yx yy}	same as \t{}\t{xx xy yx yy}
\Z{x}	absolute zoom x times (same as \z{}\z{x})
\q	make font oblique (same as \l{0.25})
\Q	undo oblique (same as \l{-0.25})
\N	return to normal style (same as \v{}\t{})
\\	print \
\n	switch to font number n (0-9) (deprecated)
\c	begin using upper 128 characters of set (deprecated)
\C	stop using upper 128 characters of set (deprecated)

Table 21: Typesetting control codes.

- Raster (`grace-5.1.22`) drivers (PNM/JPEG/PNG):
  - only even-odd fill rule is supported
  - patterned lines are not implemented
- PDF (PDFlib) driver:
  - bitmapped text strings are not transparent
  - PDFlibLite does not display correctly xycmap sets.
- MIF driver:
  - some of patterned fills not implemented
  - bitmapped text strings not implemented
- SVG (`grace-5.1.22`) driver:
  - bitmapped text strings not implemented

12.3 Device-specific settings

Some of the output devices accept several configuration options. You can set the options by passing a respective string to the interpreter using the "DEVICE *"devname"* OP *"options"*" command (see 9 (Device parameters)). A few options can be passed in one command, separated by commas.

Command	Description
`grayscale`	set grayscale output
`color`	set color output
`level1`	use only PS Level 1 subset of commands
`level2`	use also PS Level 2 commands if needed
`docdata:7bit`	the document data is 7bit clean
`docdata:8bit`	the document data is 8bit clean
`docdata:binary`	the document data may be binary
`xoffset:`x	set page offset in X direction x pp
`yoffset:`y	set page offset in Y direction y pp
`mediafeed:auto`	default input tray
`mediafeed:match`	select input with media matching page dimensions
`mediafeed:manual`	manual media feed
`hwresolution:on`	set hardware resolution
`hwresolution:off`	do not set hardware resolution

Table 22: PostScript driver options

Command	Description
`grayscale`	set grayscale output
`color`	set color output
`level1`	use only PS Level 1 subset of commands
`level2`	use also PS Level 2 commands if needed
`bbox:tight`	enable "tight" bounding box
`bbox:page`	bounding box coincides with page dimensions

Table 23: EPS driver options

12.4 Dates in Grace

We use two calendars in Grace: the one that was established in 532 by Denys and lasted until 1582, and the one that was created by Luigi Lilio (Alyosius Lilius) and Christoph Klau (Christophorus Clavius) for pope Gregorius XIII. Both use the same months (they were introduced under emperor Augustus, a few years after Julian calendar introduction, both Julius and Augustus were honored by a month being named after each one).

Command	Description
PDF1.3	set compatibility mode to PDF-1.3
PDF1.4	set compatibility mode to PDF-1.4
compression:value	set compression level (0 - 9)
patterns:on	enable use of patterns
patterns:off	disable use of patterns

Table 24: PDF (PDFlib) driver options

Command	Description
format:pbm	output in PBM format
format:pgm	output in PGM format
format:ppm	output in PPM format
rawbits:on	"rawbits" (binary) output
rawbits:off	ASCII output

Table 25: PNM driver options

Command	Description
grayscale	set grayscale output
color	set color output
optimize:on/off	enable/disable optimization
quality:value	set compression quality (0 - 100)
smoothing:value	set smoothing (0 - 100)
baseline:on/off	do/don't force baseline output
progressive:on/off	do/don't output in progressive format
dct:ifast	use fast integer DCT method
dct:islow	use slow integer DCT method
dct:float	use floating-point DCT method

Table 26: JPEG driver options

Command	Description
interlaced:on	make interlaced image
interlaced:off	don't make interlaced image
transparent:on	produce transparent image
transparent:off	don't produce transparent image
compression:value	set compression level (0 - 9)

Table 27: PNG (grace-5.1.22) driver options

The leap years occurred regularly in Denys's calendar: once every four years, there is no year 0 in this calendar (the leap year -1 was just before year 1). This calendar was not compliant with earth motion and the dates were slowly shifting with regard to astronomical events.

This was corrected in 1582 by introducing Gregorian calendar. First a ten days shift was introduced to reset correct dates (Thursday October the 4th was followed by Friday October the 15th). The rules for leap years were also changed: three leap years are removed every four centuries. These years are those that are multiple of 100 but not multiple of 400: 1700, 1800, and 1900 were not leap years, but 1600 and 2000 were (will be) leap years.

We still use Gregorian calendar today, but we now have several time scales for increased accuracy. The International Atomic Time (TAI) is a linear scale: the best scale to use for scientific reference. The Coordinated Universal Time (UTC, often confused with Greenwich Mean Time) is a legal time that is almost synchronized with earth motion. However, since the earth is slightly slowing down, leap seconds are introduced from time to time in UTC (about one second every 18 months). UTC is not a continuous scale ! When a leap second is introduced by International Earth Rotation Service, this is published in advance and the legal time sequence is as follows: 23:59:59 followed one second later by 23:59:60 followed one second later by 00:00:00. At the time of this writing (1999-01-05) the difference between TAI and UTC was 32 seconds, and the last leap second was introduced in 1998-12-31.

These calendars allow to represent any date from the mist of the past to the fog of the future, but they are not convenient for computation. Another time scale is possible: counting only the days from a reference. Such a time scale was introduced by Joseph-Juste Scaliger (Josephus Justus Scaliger) in 1583. He decided to use "-4713-01-01T12:00:00" as a reference date because it was at the same time a Monday, first of January of a leap year, there was an exact number of 19 years Meton cycle between this date and year 1 (for Easter computation), and it was at the beginning of a 15 years *Roman indiction* cycle. The day number counted from this reference is traditionally called *Julian day*, but it has really nothing to do with the Julian calendar.

Grace stores dates internally as reals numbers counted from a reference date. The default reference date is the one chosen by Scaliger, it is a classical reference for astronomical events. It can modified for a single session using the 4.3.3 (Edit->Preferences) popup of the GUI. If you often work with a specific reference date you can set it for every sessions with a REFERENCE DATE command in your configuration file (see 3.3.3 (Default template)).

The following date formats are supported (hour, minutes and seconds are always optional):

1. iso8601 : 1999-12-31T23:59:59.999
2. european : 31/12/1999 23:59:59.999 or 31/12/99 23:59:59.999
3. us : 12/31/1999 23:59:59.999 or 12/31/99 23:59:59.999
4. Julian : 123456.789

One should be aware that Grace does not allow to put a space in one data column as spaces are used to separate fields. You should always use another separator (:/.- or better T) between date and time in data files. The GUI, the batch language and the command line flags do not have this limitation, you can use spaces there without any problem. The T separator comes from the ISO8601 standard. Grace support its use also in european and us formats.

You can also provide a hint about the format ("ISO8601", "european", "us") using the -datehint command line flag or the ref name="Edit->Preferences" id="preferences"> popup of the GUI. The formats are tried in the following order: first the hint given by the user, then iso, european and us (there is no ambiguity between calendar formats and numerical formats and therefore no order is specified for them). The separators between various fields can be any characters in the set: " :/.-T" (one or more spaces act as one separator, other characters can not be repeated, the T separator is allowed only between date and time, mainly for iso8601), so the string "1999-12 31:23/59" is allowed (but not recommended). The '-' character is used both as a separator (it is traditionally used in iso8601 format) and as the unary minus (for dates in the far past or for numerical dates). By default years are left untouched, so 99 is a date far away in the past. This behavior can be changed with the 4.3.3 (Edit->preferences) popup, or with the `DATE WRAP on` and `DATE WRAP YEAR year` commands. Suppose for example that the wrap year is chosen as 1950, if the year is between 0 and 99 and is written with two or less digits, it is mapped to the present era as follows:

range [00 ; 49] is mapped to [2000 ; 2049]

range [50 ; 99] is mapped to [1950 ; 1999]

with a wrap year set to 1970, the mapping would have been:

range [00 ; 69] is mapped to [2000 ; 2069]

range [70 ; 99] is mapped to [1970 ; 1999]

this is reasonably Y2K compliant and is consistent with current use. Specifying year 1 is still possible using more than two digits as follows: "0001-03-04" is unambiguously March the 4th, year

1. The inverse transform is applied for dates written by Grace, for example as tick labels. Using two digits only for years is not recommended, we introduce a *wrap year + 100* bug here so this feature should be removed at some point in the future ...

The date scanner can be used either for Denys's and Gregorian calendars. Inexistent dates are detected, they include year 0, dates between 1582-10-05 and 1582-10-14, February 29th of non leap years, months below 1 or above 12, ... the scanner does not take into account leap seconds: you can think it works only in International Atomic Time (TAI) and not in Coordinated Unified Time (UTC). If you find yourself in a situation were you need UTC, a very precise scale, and should take into account leap seconds ... you should convert your data yourself (for example using International Atomic Time). But if you bother with that you probably already know what to do.

13 Developper's section

Some hints to help to enter into the program.

13.1 Program structure

Like its ancestors, `GraceGTK` has a GUI independent *kernel* using *display drivers* for its output, a *command interpreter* to read command files and a GUI.

13.1.1 The kernel

The kernel is mainly the same as in `grace-5.1.22` but two significant additions:

- Each drawing element visible in the TreeView correspond to a `QDobject` struct (see `defines.h`) element realised in the `objs` array,
- these elements are linked to form a tree and managed by the functions defined in `nn_tree.c` and `ng_objects.c` .

13.1.2 The display drivers

One driver is essential: the GTK driver in the `gg_gtkdrv.c` file. It is used to draw the project on the canvas and replace the X11/Motif driver of `grace-5.1.22` .
Each driver fills a `Device_entry` struct and provides the following drawing functions

1. `devupdatecmap`: update color map,
2. `devsetfrgbcolor`: added to `grace-5.1.22` functions allows a direct access to colors without the cmap indirection[21] .
3. `devdrawpixel`: draw one pixel,
4. `devdrawpolyline`: draw a polygon (or polyline),
5. `devfillpolygon`
6. `devdrawarc`: draw an arc of ellipse,
7. `devfillarc`: fill an arc of ellipse,
8. `devputpixmap`: draw a pixmap, i.e. an array of pixels,
9. `devputtext`: draw a string,
10. `devleavegraphics`: finalise the drawing.

Actually, these names are aliases for the local names used in the drivers files and this aliasing is made effective by a call to `initgraphics()` in `drawgraph()` .
The drivers are registered in `main` and GTK driver is initialised by a call to `gg_init()` in `gg_start_gtk_gui()` function.
The list of drivers is build by the registering functions listed in `devlist.h` .
Three `CAIRO` drivers had been added to old `grace-5.1.22` drivers. They don't need external libraries (`CAIRO` is included in GTK) and allow a correct output of `XYCMAP`, but are not at all optimized; e.g. the `CAIROpdf` driver don't define local fonts, thus letters are drawn pixel by pixel using `T1lib`.
A good thing to do should be to switch to Freetype2 fonts, wich include T1lib fonts and are the natural fonts for `CAIRO`, but this needs to code a new interface in replacement of `t1fonts.c`.

[21] Note that not all the drivers have this functionality, thus the ability to draw xycolormap sets.

13.1.3 The command interpreter

A first parsing done in `uniread()` separates data from commands, then the commands strings are passed to `scanner()` and finally to `parser()`, defined in `pars.yacc` .
Finally, the parser apply the rules defined in the grammar rules section of `pars.yacc` by calling `yyparse()`

13.2 Filenames

In order to keep a link with the past, name space is a little messy.

- The names for the old kernel files remains unchanged.
- New kernel files have prefix `nn_` or `ng_` .
- Undo/redo and follow-me modes are based on `nn_cmd` files.
- Motif to GTK transition was made with new GTK files that have the Motif name prefixed with `gg_` and with some changes, e.g.:
  - `motifutils.c` is splitted in `gg_gtkutils.c` and `gg_frame.c`
  - `xprotos.h -> gg_protos.h`
  - `xmgrace.c -> gg.c`

  After the transition was achieved, the Motif files had been removed.
- Explorer GTK files have prefix `ge_`
- Custom GtkWidgets that can be used in a more general context are prefixed `gw_` .
- Collective update menus are build using `wu_` subsystem.
- New spline interpolation method X-splines is in `xsplines.c` .
- XYCMAP files are `zc.c` and `zc_contours.c` .

13.3 Arrays and the drawing tree

`grace-5.1.22` is based on arrays for graphs, sets and objects. `GraceGTK` keep this structure but add a tree view to it and gather in array `objs[]` the various geometric objects.

13.3.1 Arrays

Drawable	array name	struct name	struct definition	element created by	Qtype
graph	g	graph	graphs.h	realloc_graphs()	Q_Graph
set	g[].p[]	plotarr	"	realloc_graph_plots()	Q_Set
axis	g[].t[]	tickmarks	defines.h	new_graph_tickmarks()	Q_Axis
legend	g[].l	legend	"		Q_LegendBox
others	objs	QDobject	"	obj_next()	

In order to manage the drawables as nodes in a tree, each drawable is associated to an element in the `objs` array. In the `QDobject` structure, the pointers `self ,elder ,brother ,child` are indexes in array `objs[]` and managed through the functions defined in `nn_tree.c`

This heterodox method has some *pros*:

- it allows a smooth transition between the pure array management of `grace-5.1.22` and the tree view of `GraceGTK` ,
- it is more easy to scan all the objects with a simple loop than with a tree traversal,
- the debugging is easier.
  Running `./configure --enable-debug --enable-maintainer`
  allows to display toolbar buttons to print the tree with its pointers and all the `objs` array.

The main *cons* are that you have to manage the synchronisation between the arrays and the tree and that, of course, reentrancy is excluded.
Note that the tree is a binary tree, thus a difference is made between `elder` and `father`.
Note that `g[], p[], obj[]` are dynamical arrays, i.e. are reallocated when their size is varied. Consequently, addressing their elements through memory pointers is safe only if you are sure that a reallocation will not occur.

13.3.2 Drawing tree

The drawing tree displayed in the left Explorer has in fact two instances:

- the GTK one, managed through the `ge_tree` widget,
- the kernel one, coded as a binary tree. The low level function are in `nn_tree.c` file, others functions independent from the graphical interface but taking into account the `Qtype` are mainly in `ng_objects.c`.

13.4 Miscellaneous

13.4.1 Layers

Each leaf of the drawing tree has a layer number. Actually `Q_Graph` display 3 drawing elements with a layer number that does not correspond to a `objs[]` element (the frame, the title and the subtitle) and consequently needs a special management.
For `Q_Graph` and `Q_Compound` which are nodes but not leafs, a button is added to collapse all the branch into a single layer.

13.4.2 Management of handles

Handles are used to select objects or points and act on them with the mouse. There are three possibilities to attach handles to an object (or set or graph):

- to the corner of the bounding box,
- to the ends of a line,
- to each point of the polyline or set,

and two ways to grab the item:

- a click inside the bounding box to edit, move or copy the object,
- a click on the handle to scale the object or move, delete or add a point.

For historical reasons, bounding boxes (bb) of graphs, axis ,sets and legend boxes are not stored in `objs[].bb`, thus these elements are all sets to zero. A unification of bb management is desirable.

13.4.3 Instantaneous update

Instantaneous update needs to add callbacks to each spin box, combo box, *etc* in the Explorer right panes. It is obtained by adding `if (CB) g_signal_connect (...CB...)` to utility functions defined in `gg_frame.c` The pointer `CB` towards the *apply* function of the concerned right pane is set by a call to `gg_frame_connect()` but, when a right pane menu is created or updated, GTK emits signals for elements in the panel: to avoid to create a circular dependency, it is necessary to set the variable `block_instant_update` to `TRUE` when the menu is created or updated. This is simpler then use `gtk_signal_handler_block()` for each item in the menu.

References

[1] Carole Blanc and Christophe Schlick. X-splines : A spline model designed for the end-user. In *Proc. SIGGRAPH'95, Computer Graphics*, pages 377–386, 1995.

[2] W. H. Press, Q. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in FORTRAN*, chapter 13.10. Cambridge University Press, 1992.

[3] A. Preusser. Computing contours by successive solution of quintic polynomial equations,. *ACM TOMS*, 10(4), 1984.

Index