GraceGTK guide

Software version: `GraceGTK3` (to be amended)

July 11, 2022

Abstract

This document explains the usage of `GraceGTK3` a 2D plotting tool for numerical data.[1]

Contents

[1] Note that a large part of this document is borrowed to the **Grace User Guide (the guide for `grace-5.1.22`)**.

List of Tables

1 Introduction

1.1 What is GraceGTK?

GraceGTK3 is an easy to use tool to make two-dimensional plots of numerical data[2]
It makes easy to grab numerical data, perform some post-processing (Fourier transform, filtering, *etc*), add objects (rectangles, ellipses *etc*) and obtain a drawing able to make a good figure in a scientific article.
GraceGTK was based on the GTK-2 library, GraceGTK3 on GTK-3.

GraceGTK3 home is https://sourceforge.net/projects/gracegtk

1.2 Installation

The natural operating system is Unix/Linux, but the use of MinGW/Msys allows to build a standalone distribution for MS Windows.
GraceGTK home site distribute source tarballs for Linux users and a binary package for MS Windows. Mac OS versions depend on another site [3].

See the **Installation guide** (section 15) for detailed instructions.

2 Getting started

For a jump-in start, you can browse the demos ("Examples" menu). These are ordinary GraceGTK3 projects, so you can play with them and modify them.[4]

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

1. Start the GUI version: ggrace (return).
2. Load your data clicking the Import toolbar button.
3. Open the Explorer by clicking TreeView button.
4. Adjust what you want by selecting items in the drawing tree and using the contextual right pane menus. Default behaviour[5] 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.
5. Data can be manipulated in a spreadsheet or using Compute menu items.
   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.
6. When you like your plot, select File/Print Setup, adjust what you want and 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.

2.1 General concepts

2.1.1 Input data file

GraceGTK3 reads ASCII text files containing columns of data:

- Several options allows a large variety of column separators.
- 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 origin of date formats.

[2] GraceGTK3 is a fork of grace-5.1.22 (http://plasma-gate.weizmann.ac.il/Grace/) using the **GTK** Application Programming Interface (API) instead of Motif API. The name *Grace* is used when the text is valid for GraceGTK3 as well as for grace-5.1.22 and GraceGTK3 when relevant only for this later.

[3] http://pdb.finkproject.org

[4] Also, read the grace-5.1.22 *Tutorial* Tutorial.html.

[5] This "instant update" feature can be disabled in the Edit/Preferences menu.

- The end of a dataset is marked by a line with a `&`.
- Blank lines and lines beginning with `#` (comment lines) are ignored.
- For large binary files, NetCDF import/export may be used (see sec 3.3.11)

You can mix data and commands in a command file: see section 8 for details on `GraceGTK3` language.

2.1.2 `GraceGTK3` 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 *project file* contains all the information necessary to restore a plot (commands and data). The default extension of the file is `.agr` and the commands lines begins by a `@`.
  Sometimes, we prefer to use the `.com` extension for batch scripts[6].
- A *parameter file* is a "style sheet" containing *only* the parameters needed to draw similar graphs with other compatible data. The default extension of the file is `.par` and for each line the leading `@` may be ommitted.
- A *differential project file* is a project file where all the command lines not changing default values are omitted, thus the result depends upon init files (see sec. 2.3.2).

See section 8 for details on `GraceGTK3` language.

2.1.3 Datasets, sets and block data

- A *dataset* (often called simply a *set* in the following) is a collection of points stored in the computer memory with x and y coordinates, up to four optional data values and one optional character string.
- 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).
- The *block data* is a collection of vectors build when numeric data is parsed[7]. columns may be combined into a dataset using the block data option when importing data (see 3.3.8) or later.

2.1.4 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 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 4.2).
- Fixed Graph (i.e. XY graph with X/Y ratio = 1)
- Pie chart

Graphs are numbered from zero and graph zero always exists. You may hide it if needed.

A graph is active if it contains sets or geometric objects.

It is possible to renumber graphs, wiping inactive ones with the `PACK` command, or associate a name to a graph using a variable (see sec. 8.2.3).

Sets are of various *types* listed in table 1.
Not all set types, however, can be plotted on any graph type. The table 2 summarises the constraints.
Sets are numbered from zero. It is possible to renumber sets in a graph, wiping killed ones with the `PACK G`*n* command.

[6] This is the default setting for the `Commands` window. See `READ BATCH` command and advanced scripting example.
[7] Up to now, only one block data is allowed.

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. Draw vertical bars between Hi and Low, with two horizontal ticks at Open and Close.
XYC	3	Same as XY and z is used to define the color of segments of the curve.
XYZ	3	Same as XY and z is used to define numeric annotations.
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) (deprecated, prefer XYCSYM)
XYCSYM	3	x, y, z, color of the symbols maps z values.
XYCMAP	3	x, y, *color* interpolated in a dedicated color map. (see 7.1 and 8.13.6)
XYCOLPAT	4	x, y, color index, pattern index (currently used for Pie charts only)
XYVMAP	4	Vector map (x, y, V_x, V_y)
XYCVMAP	5	Colored vector map (x, y, V_x, V_y, z) z is color mapped
XYBOXPLOT	6	Box plot (x, median, upper/lower limit, upper/lower whisker)

Table 1: Set types

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

Table 2: Graph/Set type connection

2.1.5 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 coordinates is determined by both the graph type and the axis scaling.
The *loctyp* of a geometric object defines if the dimensions and the place of the object are defined in viewport or world coordinates.

2.1.6 Zoom

Two types of zooming are possible.

- Zooming in user world, i.e. changing scale of the world rectangle visible in the viewport window. In other words, the scale of the transform between world and viewport coordinates is changed, but the size of the viewport on the screen remains the same.
  This can be obtained using the left toolbar buttons.
- Zooming the page viewport, i.e. changing scale transform between viewport and screen; this is equivalent to changing page size.
  This can be obtained via +/- shortcuts or `Menubar/View` entries.

2.1.7 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.

2.1.8 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 or a comment.

Polylines are segments connecting points and can be closed or opened, drawn as zigzags or smooth X-spline interpolated curves. It is possible to use a polyline to define a new set in a graph.

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.

Images can be imported from several formats. See details in section 3.12.3

2.1.9 Compounds

Compounds are used to glue together in a single branch existing geometric objects, allowing to move, scale, 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).
More details about compounds are given in section 6.2.

2.1.10 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.
Beware that, when you merge two different command files using the old Grace-5 syntax, the numbers may overlay and in that case, you have to manually change the numbers attached to different objects before merging.
Use the `begin/end` syntax to overcome this drawback.

2.1.11 TreeView

The various elements of the drawing are linked by a tree structure that can be visualised in the left pane of the `Explorer`. Each graph is the head of a branch of the tree containing axes, sets, legend box and optionally, geometric objects.
Templates are the starting point to create new objects.

2.1.12 Layers

The drawing is done on one or several *layers*: this allows a fine control of the foreground/background issue. Deepest layer is the one with the biggest number. 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.

2.1.13 Colors and patterns

A unique *colormap* is defined in `GraceGTK3` for a general usage, as well as a set of patterns to fill objects (see 8.2.6). Up to now, no transparency, shading or brightening effects are available.
For various geometric objects, it is possible to choose the colors for the boundary, the pattern and the background.
Note that the colormap is not reinitialised when a new project is loaded: it is sometime useful to stop `GraceGTK3` and restart a new instance of the program if the preceding one has modified the default map.
`XYC, XYCMAP, XYCVMAP` sets (see 7.1) are special cases: the colors of the colored map are interpolated from values taken in several predefined color maps.

2.1.14 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` .

2.1.15 Devices: output formats

Grace allows the user to choose between several output device formats to produce its graphics. Available formats are provided by the Cairo graphic library, i.e PDF, EPS, PNG, SVG and JPEG.

2.1.16 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*`:$GRACEGTK_HOME/`*pathname*

2.2 Invocation

The name of the executable is `ggrace` (or `ggrace.exe` in a Windows environment).
At start, the program

- scan environment variables (sec 2.3.1)
- read the `gracegtkrc` file (sec. 2.3.2),
- create graph 0 (`G0`),
- decode command line options[8]
- read the template file `Defaults.agr` (sec. 2.3.2),
- and, if any, read the project file.

[8] Note that all the options had not been systematically tested when `grace-5.1.22` had been changed in `GraceGTK3` . 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 `GraceGTK3` .

2.2.1 Command line options

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

-barebones
Turn off all toolbars

-batch ***param_file***
Execute the parameter file[9] *param_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 and create graphs from 0 to *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

-netcdf ***file***
Assume data *file* is in netCDF format. This option is present only if the netCDF support was compiled in

-netcdfxy ***X_var Y_var***
If -netcdf was used previously, read from the netCDF file *X_var Y_var* variables and create a set. If *X_var* name is "null" then load the index of Y to X. This option is present only if the netCDF support was compiled in

-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

[9] i.e. a command file without leading @

-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 section of the guide

2.2.2 Example

The following command will run `GraceGTK3` in batch mode on the project file `projet.agr` and write an image in PDF format (using the Cairo driver) in file `image.pdf`:

```
ggrace -hardcopy -hdevice PDFcairo -printfile image.pdf projet.agr
```

Note that if the file `image.pdf` already exists it is not overwritten i.e. this command will not gives the wanted result and no message will be displayed due to the batch (`-hardcopy`) mode [10].

[10] Due to historical reason, the `-batch` option has not the same meaning

2.3 Customization

2.3.1 Environment variables

- `GRACEGTK_HOME`
  Set the location of `GraceGTK3` . 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.
  After the installation is done, the executables are in `$GRACEGTK_HOME/bin`, thus this directory must be in your PATH.
- `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`.
  An alternative is to use `gracegtkrc` file (see below).
- `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`
  `GraceGTK3` documentation is in PDF format: specify here the viewer to use. Default is `firefox %s` for Unix/Linux systems and your default viewer for Windows systems.
  The HTML version of the `GraceGTK3` 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 10.5 for detailed info)
- `LC_NUMERIC`
  Default is that `GraceGTK3` check the value of `LC_NUMERIC` to display e.g. the ticks label of axes[11].
  It is possible to disable this feature
  - by using the `Preference` window or
  - by setting "`DEFAULT LOCALE off`" in the default template file (see below).

2.3.2 Init files

Upon start-up, `GraceGTK3` loads two init files:

`gracegtkrc` and `templates/Default.agr` .

The first one is intended to configure all instances of `GraceGTK3` and the second may be used to particularize at the project level. Both are searched for in the magic path (see 2.1.16); *once found, the rest of the path is ignored*. Statements are commands recognized by the interpreter (without leading @ for `gracegtkrc` and with leading @ for `Default.agr`).
Note that many `GraceGTK3` examples are saved as differential project files thus rely on the default template files posted with the distribution to run correctly.

2.3.3 GUI: size and positions

To set the size of the main window (in pixels):

`GUI SIZE` $(width, heigth)$

To place the dialog boxes and override the choice of the window manager[12]:

[11] E.g. using comma as decimal separator instead of point in some environments.
[12] This command is active only *before* the window is first opened.

`GUI` "*title*" (x, y)

title is the string displayed at the top of the window, e.g.
`GUI "GraceGTK: Explorer" (50 ,60)`
Usually (x, y) are the coordinates of the upper left corner of the window.

2.3.4 GUI: colors

`GraceGTK3` defines colors used into the menus independently of the general GTK theme inherited from the windows manager.
Default colors (*wheat* and *sky blue*) may show poor results with some themes (e.g. dark ones). It should be changed by editing the init file `gracegtk.css`.

3 Guide to the graphical user interface

`GraceGTK3` main 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 displays on the left a *toolbar* with various buttons frequently used, and on its right the *canvas*, i.e. the drawing area.

3.1 Using mouse

3.1.1 Using left button (button 1)

- In the left toolbar a click selects an action to be performed[13] (zoom, copy, move,...) . Then, selecting one object in the canvas allows to perform the action on this object[14].
- 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 3.12 for details.
- In the TreeView, selecting a `template` allows to create a new object.
- In the canvas, the nature of the action of the mouse is the one selected previously in the left toolbar. In the "no action" state (see below), a click allows to change the focus from one graph to another.
- 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.
- Remember that the `Close` button in popups actually hide them but does not change the status in the canvas. For example, if you want to draw a polyline, after pressing the `"Create & place with mouse"` button, it is possible to close the `Explorer` before drawing the lines.
- Only changes in selection triggers callbacks (i.e. update the right pane in the explorer) and it is sometimes necessary to force change even if the item is already selected to obtain the wanted result.

3.1.2 Using mouse button 2

Only two cases:

- to place the last point when defining a polyline or a region with the mouse,
- to finalise gluing objects into a compound.

3.1.3 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.

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

- In the TreeView, it fires menus depending upon the item(s)[15] 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 picked in the right pane (visible or not) 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 6.2.
- In sets lists, it allows also to open the explorer if needed and select the clicked set in the tree view, create new sets,..
- `Manage spin button`: 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` entry which allows to fix manually the settings and also to set the value of the variable. Moreover, the entries in this popup are scanned by the command interpreter, allowing to use variables and formulas.
- `Font tool` can also be used to enter some strings in the right pane: see section 5.5.
- `Time tool` may be useful for `Explorer/axis` start/stop when date formats are in use (see section 5.6).
- `Use to create a set` may be used to transform a polyline into a set in a graph.
- `Use to create a polyline` may be used to make a new polyline with the apex of a rectangular box as points. This is useful to define rounded boxes based on the smoothing option for polylines.

3.1.4 Using mouse wheel

Use the wheel to scroll windows displaying a scroll bar or to change value in spin boxes.

3.2 Left toolbar

Along the left-hand side of the canvas is the `ToolBar` (if shown). It is armed with several buttons to provide quick and easy access to the more commonly used Grace functions.
The `Examples/GraceGTK3 /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.

[15] 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.

- `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.
- 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?

3.3 File menu

3.3.1 New

Wipe out present drawing to restart a new one.

3.3.2 Open project

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

3.3.3 Save

Save the drawing in the present opened project file. Datasets are saved in ASCII with the format defined in the *Save as* popup.
If no file is opened, save in the file named `Untitled$`

3.3.4 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. A new experimental `Begin/end` scheme is proposed: geometric objects are not numbered, this should make more easy to merge two project files. 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.

3.3.5 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.

3.3.6 Load parameters...

Open a *Read parameters* window to read a parameters file (see section 2.1.2).

3.3.7 Save parameters...

Open a *Write parameters* window to write a parameters file (see section 2.1.2).

3.3.8 Import ASCII data

Open a *Read data file* window to read ASCII data (see section 2.1.1), i.e. by default a `.dat` file.
Data is parsed line by line and stored into a block data. The first successful parsing fix the number of columns and type of each column (number, date, string).
Two blocks are in use: one is recorded when you press the `Open` button at the bottom of the dialog window, and used to create the set(s), the other is just to preview selected file in the dataset preview popup without destroying the main one.
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*: a click on the `Open` button pops up a dialog that allows to select the abscissas and ordinates at will to define a set. It should be noted that the block data is stored as long as you do not override it 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[16] at a time by the use of a format string called *Sel.* Each character in the *Sel.* string[17] stands for one column in the data file and have the following meaning
  - x: abscissas (*should appear only once)*
  - y: ordinates
  - z: z values (*if relevant with respect to set type, should appear only once*),
  - s: string[18] [19]
  - - : (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 1 and 2).

If the source contains date fields, they should be automatically detected. Several formats are recognized (see appendix 14.2 (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.

3.3.9 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.* For large datasets, you may consider to use NetCDF binary format (see 3.3.12).
A warning is displayed if a file with the same name already exists.

3.3.10 Import Xfig files

Files in the `Xfig FORMAT3.2` may be imported into `GraceGTK3` . Differences between the two programs induce some lost in the information. Some details about the import process are given below, that may require an adaptation "by hand" after import.

[16] Not effective for all types of sets.
[17] Spaces and tabs are ignored
[18] If the string begin by a digit, it is normally interpreted as a number, "s" allows to force the decoding.
[19] Only the first column of strings is used

- The size of text is roughly approximated and line width, patterns scale and text size may be changed, because `Xfig` rescales them and `GraceGTK3` does not.
- `Xfig` may define new colors with number *xxx*: `GraceGTK3` include them in its color map with name `xfig`*xxx* using its own numbering: thus several imports may define different colors with the same name.
- `GraceGTK3` does not resets its color map when a `New project` is issued, thus the number of colors may increase unduly if you don't restart `GraceGTK3` after several imports.
- `Xfig` defines "shades" and "tints" for colors, features that does not exists in `GraceGTK3` , thus colors are not changed by this parameter,
- `Xfig` allows a layer at depth zero, `GraceGTK3` not, thus depth are augmented by one during the import process. Objects on the same layer may be drawn in a different order, thus the visibility relative to the foreground/background issue of several objects with the same depth may be changed.
- a bug in `GraceGTK3` : bounding boxes of arcs are defined for closed arcs and does not take into account en extend < 360 degrees, this can enlarge unduly the bounding box of the compound containing the `Xfig` figure, perturbing the definition of the aspect ratio of the figure.

In my Linux distribution, `Xfig` library files of various object are installed in the `/usr/share/xfig/Libraries` directory.

3.3.11 `Import NetCDF files`

NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data[20] [21]
Only variables of dimension one are taken into account.
If the `Load GraceGTK files globally` button is checked and the file had been produced by GraceGTK with the right option, all the sets saved are imported at a time with some additional information such as legends or annotations.
If not, only one set is created at a time and the user is asked to assign the variables to the columns of the dataset to be created.

3.3.12 `Export NetCDF files`

Import/export of data in binary format may be a touchy business (evils are named 32/64 bits, big endian / little endian, buffer size,...) and a good solution, allowing also exchanges through the Web is to use the NetCDF library.
`GraceGTK3` export datasets of all active datasets at a time in NetCDF format, together with annotations and legends. NetCDF data format is binary, thus it is more efficient than ASCII and recommended for large datasets[22].
It is also a good method to exchange data with other programs.
A part of the NetCDF files content may be displayed with the `Information` button[23].

Files exported by `GraceGTK3` use the following scheme (see `netcdf.c`).

- Each datasets column is a `NC_DOUBLE` one dimensional variable with name `G`*xx*`_s`*yy*`_`*cc* where *xx* is the graph number, *yy* the set and *cc* the column.
- If annotations are present, they are padded to a fixed length with zero and concatenated into a single string variable named `G`*xx*`_s`*yy*`_annotation`.
- All variables have at least 3 attributes: "gno", "setno" and "col". If "col" is less than zero, its value is used to discriminate annotations and legends.

3.3.13 `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.
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

[20] For details about the library, see http://www.unidata.ucar.edu/software/netcdf .
[21] This feature of `grace-5.1.22` had been implemented (and extended) only since `GraceGTK3` -0.9.1. It is available only if compiled with the **NetCDF library**.
[22] Note that only data will be saved: parameters and objects (strings, rectangles,...) must be saved in a separate file.
[23] A complete dump may be done using the `ncdump` utility distributed with the NetCDF library.

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.

3.3.14 `Print`

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

3.3.15 `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.

3.4 Edit menu

3.4.1 `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.

3.4.2 `Explorer`

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

3.4.3 `Datasets stats`

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).

3.4.4 `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.

Use the `Duplicate` entry of the menu launched with mouse button 3 in TreeView or set lists to create a new graph similar to an existing one. Use the `Create new` entry to create new sets by formula, text editor or from the current block data.

3.4.5 `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.

3.4.6 `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.

3.4.7 Autoscale graphs

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.

3.4.8 Graph operations...

Similar to `Set operations` but for graphs: allow to copy, move or swap graphs as a whole.
Use `Graph duplicate`, launched with mouse button 3 in TreeView or graph lists to create a new graph similar to an existing one.
Use `Project/Layout` to create new empty graphs.

3.4.9 Regions

`Region master`

- In the `Main` tab, the *define* buttons allow to define a region this the mouse. For polygons, use the button 2 to finish and close the polygon.
- The `Points` tab allow to move the points with the mouse or typing numerical values.
- The `Operations` tab allows to create or modify sets using a selected region.

Others dialogs remains mainly for Grace-5 compatibility.

`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). 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.

3.4.10 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:

1. select a set in the `Link to sets` part of the window,
2. use the `Files...` button to copy a file name into the entry,
3. push the `Link` button: *the link should be displayed in the upper part of the window*,
4. repeat the process if you want to link several sets to files or pipes.

Then, pushing the `Update` button will reread all the linked files.
It is also possible to use pipes instead of files[24].
Currently, only simple XY sets can be used for hotlinks.

3.4.11 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].

3.4.12 Clear locator fixed point

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

[24] Except in a MS Windows environment.

3.4.13 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. This dialog may be also raised by clicking mouse button 3 in locator display.

3.4.14 Colors

This popup may be used to modify the GraceGTK3 colormap. It allows to add colors or to modify existing ones.
Note that it is not a good idea to change values in the default colormap.

3.4.15 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.
The **Max drawing path length** avoid to try to draw graphs with a too large number of points. Increase it if you want to draw sets with a number of points exceeding this limit[25].
Note that in the rendering process Grace try to purge too dense points that will be superposed in the drawing at drawing time. See also **Data/Transformation/Prune data** menu to decimate a given set.

3.5 Data menu

3.5.1 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 $n-1$.

3.5.2 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 3).

3.5.3 Import/ASCII

Is here for backward compatibility with **grace-5.1.22** . See **Import ASCII** dialog.

3.5.4 Export/ASCII

Is here for backward compatibility with **grace-5.1.22** . See **Export ASCII** dialog.

3.5.5 Digitize a curve

Allows to digitize manually a curve displayed on the screen. See section 5.4

3.6 Compute menu

3.6.1 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 8.3 and 8.9.
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
```

[25] the **-maxpath** options may also be used at the invocation with the command line.

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 3: Extractable features

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.

3.6.2 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 this latter case, abscissas of the set must form a strictly monotonic array. The list at the bottom of the window is the list of sets of the current graph. If you want to use a set from another graph, change the focus.

3.6.3 Fourier transformation

The Fourier transform window is now inspired by `grace-5.99` with some changes:
try `Examples GraceGTK3 /Fourier` and look at section 10 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.

3.6.4 Filters

There are several ways to perform filtering on a dataset: moving average (see `Running average` below), use Fourier transform to apply a custom filter or a given window in the frequency domain (see `Fourier transformation`), perform a non linear fit (see `Non-linear fit`), perform a direct convolution with another set (see `Linear convolution` below) and finally use this menu to perform in a single operation the forward and backward Fourier transforms to apply Butterworth or Chebyshev filters. It is also possible with this menu to load in a set the frequency response of these filters. More details are given in section 11.

3.6.5 Wavelet transformations

Compute Daubechies, Haar and Bsplines 1D transforms for datasets with 2^n points. More details are given in section 12.

3.6.6 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.

3.6.7 Differences/derivation

The differences popup is used to compute the `Plain difference` with the stride s, i.e.

$$\delta y_i = y_{i+s} - y_i. \tag{1}$$

Depending upon your choice: `left, center` or `right` the abscissas of the i-th point of the computed dataset are x_i, $(x_i + x_{i+s})/2$ or x_{i+s}.
If you choose `Derivative`, a first order approximation of the derivative is computed:

$$y'_i \simeq \frac{\delta y_i}{x_{i+s} - x_i}. \tag{2}$$

3.6.8 Seasonal differences

The seasonal differences popup is for compatibility with `grace-5.1.22` . It allows to compute the opposite of the left justified plain difference above, i.e. $\delta y_i = y_i - y_{i+s}$. Beware that the "period" s is entered in terms of index in the set and not in terms of abscissa!

3.6.9 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.

3.6.10 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 in the range $-1, 1$. For $s < 0$, 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.

3.6.11 Regression

The regression popup can be used to fit a set against polynomials or some specific functions (y=A*xˆB, 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.

3.6.12 Non-linear fit

The `Non-linear curve fitting` popup allows to fit a set:

- with the Levenberg-Marquardt algorithm[26] you can fit against your own formula with up to ten unknowns named `A0, A1, ..., A9` or choose a predefined one. The library of functions proposed by Nicola Ferralis is incorporated in `GraceGTK3` 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 9.5 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.
- with the LOESS algorithm[27] the fit is done locally in a sliding window and you have only to choose between a linear or quadratic local approximation.
- The *By hand* selection allows to adjust manually the starting parameters before using one of the two method above.
  Try `Examples/GraceGTK/Non-linear fit/By hand`.

In the `main tab` you select the method and choose the main parameters for each method and 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[28], and choose whether to load the fitted values, the residuals or the function itself. Details are given in section 9.

3.6.13 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` 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.
Note that the transformation acts on the (x, y) world coordinates of the set, thus is meaningful only if the `Graph fixed` option (i.e. same scale ratio for Ox and Oy) is in use.

3.6.14 Sample points

This popup provides several methods to sample points in a set.

- Choosing a starting point and a step you can select one point over n.
- You can also select only the points that satisfy a boolean expression you specify (e.g. `abs(y) < 1`).
- Selecting all minimas, maximas or extremas for y in a curve. [29]

3.6.15 Prune data (decimation)

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.

[26] 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/

[27] LOESS, or LOWESS (locally weighted scatterplot smoothing) http://en.wikipedia.org/wiki/Lowess

[28] 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.

[29] Note that no interpolation is done and that the precision is the one of the initial sampling.

3.7 Plot menu

This menu is present for backward compatibility with `grace-5.1.22` . It is another way to access menus provided by the `Explorer` or some `File` menubar entries.

3.8 View menu

3.8.1 `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.

3.8.2 `Show locator bar, Show status bar, Show tool bars`

Toggle the display of the corresponding elements.

3.8.3 `Page setup`

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

3.8.4 `Redraw`

This menu item triggers a redrawing of the canvas.

3.8.5 `White background`

is the default setting for the screen.

3.8.6 `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.

3.8.7 `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, i.e. not all the GUI is updated.*

3.9 Window menu

3.9.1 `Commands`

This window allows to execute directly *commands* in `GraceGTK3` language[30].

- Only lines selected[31] are executed when you press the `Apply` button.
- Command lines only are allowed (no data lines, use spreadsheet to edit individual values in a set) thus, the leading `@` may be omitted and are stripped by the interpreter if present, but it is a good practise to keep them in order to copy/paste in your scripts.
- 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. Commands reflecting interactions with the GUI are printed in the window when this option is selected.
- You can edit lines into this window with usual GTK conventions; e.g. Ctr-A may be used to select all the lines in the window.
- You can read in files and save the content of the window with the `"File"` menu at the top of the command window.

3.9.2 `Font tool`

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

[30] See section 8 for details on the `GraceGTK3 .agr` language.
[31] This may be done by dragging the mouse or, if all the lines are to be selected, by Ctrl-A.

3.9.3 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.
- The results may be simultaneously printed into a file using the -result comand line option or the OPEN RESULTS FILE command.
- The Edit/List user variables allows to display a list of all user defined variables, and for scalar or string variables, display their values.

3.10 Examples menu

The command files in GraceGTK3 scripting language are in the examples directory[32]: it is a very good way to get an idea of the possibilities of GraceGTK3 .

3.11 Help menu

Display the GraceGTK3 guide in the pdf format and others html files (FAQ, ...) lying in the doc directory[33].

3.12 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 three buttons at the bottom.

3.12.1 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[34].
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 strings 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).

3.12.2 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[35] and in that case to define them in world coordinates.
The Label tab in the menu for lines allows to define a label displaying the length of the line or a user defined string and to display a color rectangle behind the label.
Strings may be adorned by a surrounding box[36].
When defining a polyline with mouse, the backspace key allows to kill the last defined point[37].

[32] In a standard Linux installation, /usr/local/gracegtk/examples
[33] In a standard Linux installation, /usr/local/gracegtk/doc
[34] Note that if more than one element is selected, the apply button is inactived.
[35] Note that polylines are not intended as a replacement of XY sets and should have a moderate number of points.
[36] N.B.: This feature does not work properly if the string is ended by a subscript or a superscript, or is broken in several lines. In that cases, you have to define the box by hand.
[37] A Gimp behaviour

3.12.3 Images

It is possible to import images with various formats (JPEG, PNG,...) as geometric objects using the `Image` *Load file* entry in the Explorer and clicking the "`Create & place with mouse`" button.
The known formats depend upon the version of the `gdk-pixbuf` library linked. The combo box in the file dialog used to load an image file gives the list of the file name extensions recognised.
The `Appearance` parameters can be used to draw a box around the image.

N.B.: up to now, several limitations for images apply:

- only viewport coordinates are allowed for these objects,
- images cannot be incorporated into graphs or compounds,
- only the command to load the image file is saved into `*.agr` files: it is the user responsibility to save the image file,
- Images are stored in memory by `GraceGTK3` in their original size and the `Keep aspect ratio` or `Use original size` options are just GUI features to display or rescale them.
  Images are loaded from GUI with `Free` scaling option but if you have changed the aspect ratio at loading, it is always possible to restore the original one by setting the `Keep aspect ratio` *on* and using mouse to rescale the image.

3.12.4 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 `GraceGTK3` 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 3.4.5.
- The `Graphs` tab allows actions on several graphs at a time which are not the settings performed with the mouse 3 collective updates (see 5.3)
- The `Functions` tab allows to add easily new functions in graphs. The `X` and `Y` entries recognise `$t` as the array created by the `Start/Stop/Nb points` entries.

`TimeStamp` menu is used to control the appearance of the time stamp and set the `GraceGTK3` 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, e.g. to attribute different colors with only one click.

`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` [38] 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

[38] `Degrees` is for polar graphs only), `linear` corresponding to a radians scale.

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, ...).
For ticks with a date format terminated by seconds, the `Precision` parameter allow to choose the number of digits used to display fractions of seconds.
`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.

`Image` template is used to add images in the drawing (see sec. 3.12.3).

The examples may give you a good help to have a better understanding of the detailed behaviour of the program.

3.13 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 Some hints

4.1 How to create new sets

4.1.1 By formula

When `Project` is selected in TreeView, the `Create set` tab allows to create new sets by formula.
A mesh of regularly spaced values stocked in local variable `$t` is first created, than formulas for x and y will be applied.
Of course, it is also possible to use the `Commands` window (see 3.9.1) to write a small script to define a new set.

4.1.2 Using spreadsheet or editor

Menus attached to mouse button 3 for Graphs or Sets allows also to open an editor or a simple spreadsheet to type values by hand.

4.2 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.

4.3 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, if running in the interactive mode you are warned by a message and asked to proceed .

5 Comforts

5.1 Shortcuts

Some keyboard shortcuts are available in addition to ones provided by your desktop manager or GTK standard settings: use `Help/Shortcuts list` menu to display a list.
Up to now, they are not reconfigurable without recompilation[39].

5.2 Drag and drop

It is possible to drag-and-drop files into `GraceGTK3` ; the action taken depends about the extension of the file:

- `*.agr` files are read immediately as `GraceGTK3` project files,
- `*.par` files are recognized as parameter files and the `Load parameters` dialogue is opened,
- `*.fig` files are recognized as xfig files and the user must define with the mouse the rectangle containing the figure to finalize the import,
- `*.nc` files are recognized as NetCDF files and the `Import NetCDF file` dialog is opened,
- `*.dat` files are recognized as data files and the `Import ASCII data` dialog is opened,
- image files recognized by your GTK version open the Explorer menu dialogue to import and image file

Drag-and-drop operations from `GraceGTK3` to another application are not supported.

5.3 Collective updates

As explained in section 3.1.3, mouse button 3 can be used to fire popups allowing to update parameter(s) of several objects at a time.

5.4 Digitizing tool

This popup allows by picking manually points on an image displayed on the screen to define, first a polyline, and then a new set.

- The `"Load a new image"` button allows to launch the `Explorer`, then choose an image file, press the `"Create & place with mouse"` button[40] and place the image on the screen with the mouse.
- The `"Create a polyline"` button launch also the `Explorer` and allows to create a polyline in view coordinates interpolating points that you will click on the image displayed on the screen.

[39] Mainly `gg.c` and `gg_events.c`
[40] At that time, it is possible to close the `Explorer` popup.

- In order to transform this polyline into a set, you have to specify the scaling by typing values of abscissas and ordinates on some points on the axes and clicking onto the corresponding points in the image.
- Then, the `Apply` or `Accept` buttons will create a new set in the graph of your choice and perform the scaling.
- It is the responsibility of the user to kill the image or the polyline transformed into a set by the last ooperation.

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 its precedent state, i.e. if a previous connection to an entry exists, it remains active.
Since release 0.8.0, the default set of default fonts had been augmented to meet `xfig` file import requirements.
Beware that some of them does not display all usual characters e.g. the font `PazoMathBlackboardBold`, to be used in math prose display only capital letters and number 1.
Bug: the `WNCYR10` panel displays characters before % that are not displayed in the canvas: use the end of table instead.

5.6 Time tool

As explained in section 14.2, by default Grace use internally Julian day as date unit and numbers displayed in `Explorer/Axis` main tab for world start/stop and tick steps are displayed with this unit. These entries are scanned by the command interpreter, e.g. a step of 1 hour can be typeset as `1/24`, but this is not always practical to specify world start/stop.
The *Time tool*, raised with mouse button3 will display the calendar values, i.e. year, month,... and convert them in Julian days if you click on `Apply` or `Accept` button.
This menu item is sensitive only when a date format is selected for labels ticks.
Note that the *Place at rounded position* option for tick marks will work on the Julian days scale and give strange result in usual date formats, thus it is recommended to inactive it in that case.

5.7 Follow-me mode

This mode is enabled by a check button in the `Window/Commands` popup. Its goal is to help the user to write its own scripts by displaying the command(s) corresponding to the actions performed with the mouse[41].
The usage of the `Commands` window is described in section 3.9
Note that it is only partial and will not gives always a registration of all your actions.
Another method to help you to write script is to use the `File/Save as` menu with the `Differential` button checked on.

5.8 Undo and Redo

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.

[41] Note that the commands displayed must be independent of the context, thus it is the numbered form that will be used (see section 8.1) and if you want to reuse the command in an unnumbered script file, you have the choice to suppress the numbers and place the command in the right place in the script or add the new command at the end of the file to avoid unwanted interactions.

- 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 in that case.

5.9 Spreadsheet dataset editor

This popup does not pretend to compete with a complete spreadsheet program[42].

- Computation with formulas must be done via `GraceGTK3` pop ups that may be fired by buttons in spreadsheet.
- When points are located out of the world windows of the graph, rows are displayed in red.
- Column width may be automatically adjusted by double clicking on the edge of its label.

6 Comparison with `grace-5.1.22`

`GraceGTK3` try to be upward compatible with `grace-5.1.22` and reproduce whenever it is possible `grace-5.1.22` menus, except for geometric objects benefiting of a more developed interface.
A lot of new features had been added since the fork and we just emphasize some points.

6.1 Examples and `GraceGTK3` language

Default settings for `GraceGTK3` are slightly different from standard `grace-5.1.22` distribution and some old examples may display differently.
In order to obtain short command files easy to read, many new `GraceGTK3` examples rely on default settings and may not work has expected if you use your own display settings.

6.2 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.
To access one component of the compound, select it alone in the TreeView or select the compound in the TreeView, press the edit button in the right pane and use the mouse.
Two methods are possible to manage compounds.

- Using mouse in the canvas. If you click buttons in the toolbar at the left of the canvas ,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 template), with the same loctyp. Thus, if you want to glue objects into a compound belonging to a graph, create the new objects with the templates rows, change attachment from template to wanted graph, if needed change loctyp and finally select and glue them with mouse 3 button.
  - `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 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.

Note that the scaling of a compound that does not keep the aspect ratio will change the aspect of a circle into an ellipse, but that the subtype of that `Arc` is unchanged, i.e. a direct scaling of this element will restore it as a circle, unless you change manually its subtype.
Note that elements in a compound may belong to different layers: the `layer` entry in the right pane menu is used only if you want to collapse the elements in a single layer, using the appropriate button.
Note that it is impossible to change directly the `loctyp` of a compound: you must first break the compound (and sub compounds), change individually the attachment and `loctyp` of the elements and rebuild the compound(s).

6.3 Comments and legends

In `grace-5.1.22` , the origin 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 `GraceGTK3` this is done automatically by default, but you can disable this feature in the Edit/Preference menu.

[42] It had been forked from gtkextra-3.3.4

6.4 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.

7 2D graphs

`GraceGTK3` can draw colours vectors and 2D maps.

7.1 Color map and contour sets (XYCMAP)

`GraceGTK3` can represent $z = f(x, y)$ if x and y values form an **ordered regular** rectangular grid. Two graphic representations may be combined:

- association of a color to each value of z, i.e. a colormap,
- drawing of 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[43].
Two different sort of datasets may be used

- x, y, z are three vectors of equal length and the program check the changes in x and y to determine if the rectangular grid is given left to right or the opposite, also upward or downward;
- only z is given and the corners of the grid as well as the ordering of z data are given separately by the command `XYCMAP`:

  $$set\ \texttt{XYCMAP}\ (col, up, left, x1, y1, x2, y2)$$

  $col, up, left$ are boolean with the following meaning:
  - if $col =$ `TRUE`, z read/write columns first (Fortran default)
  - if $up =$ `TRUE`, z read/write y upward,
  - if $left$ `TRUE`, z read/write x leftward,

  $x1, y1, x2, y2$ are coordinates of two opposite corners.

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 when a `XYCMAP` set is duplicated, the contour lines and labels are not copied with the other components of a set.
The computation of contour lines[44] 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 [7].

N.B. use CAIRO drivers to print these figures, old `grace-5.1.22` drivers are not able to render `XYCMAP`.

7.2 Coloured symbols (XYCSYM and XYCOLOR)

If the data is not scattered into a regular rectangular grid, `XYCMAP` does not work but it is possible to use `XYCSYM` or `XYCOLOR`: the color of symbols drawn at (x, y) is mapped from the z value for each (x, y, z)[45].

N.B. Prefer CAIRO drivers to print `XYCSYM`, old `grace-5.1.22` drivers may not render these sets correctly.

[43] See section 8.13.6 for specific interpreter commands.
[44] It is made using algorithm toms 626 (www.netlib.org) proposed by A. Preusser.
[45] Note that the more sophisticated `XYCSYM` allowing to choose the colormap in the same way as for `XYCMAP` should be preferred to the old deprecated `grace-5.1.22 XYCOLOR` type.

8 Command interpreter

Preferred extensions are `.agr` for command/data files and `.dat` for data files.
The user is encouraged to examine the files in the `examples` directory. In this directory, files whose name begins with `ng_` are new `GraceGTK3` examples.

The input stream is parsed in a line-by-line manner, except if the last significant character[46] of the line is a backslash (\): in that case, the reading is continued on the next line.
The parsing of commands is *case-insensitive* i.e. all commands except strings are converted in upper case before parsing.
In command lines, there may be several statements per line, separated by semicolon (`;`) and no `@` after the semicolon.

The analysis is performed in three pass.

1. First, leading blanks or tabs and 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, if none of the characters above is found, 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 `GraceGTK3` standard headings.
     After that, an error message is displayed.
2. A lexical analysis is the performed to identify graphs and sets e.g. G0.S0 is decoded here by function yylex().
3. The semantic analysis is then performed as described in the `pars.yacc` file. You can refer to this file if you get a mysterious error.

8.1 Structure of a script file

The command language is an extension of `grace-5.1.22` language where graphs, sets and objects are often identified by their *id* number, a method that made difficult to merge two project files, thus a new form is introduced by `GraceGTK3` where the structure of the file reflects the tree view of the drawing and the numbers are omitted[47].
Although it is possible, the mixing of the two forms is strongly dissuaded and can produce unexpected results if lines with numbered forms ares interleaved with unnumbered ones.

8.2 Definitions

`GraceGTK3` language is based on numerous reserved words but a grammar very simple.

8.2.1 Version number

Saved project files start with the version number of the program used to create the file:
`@ VERSION` $1xxyyzz$ where: xx is the *major* and yy the *minor* revision numbers and zz the *patchlevel*.
The leading 1 is used to differentiate `GraceGTK3` from Grace.

8.2.2 Constant

`PI` is the π constant.

8.2.3 Variables

It is possible to define user's variables if the name is not already used[48].
A list of all the defined user variables may be obtained with the `LIST_USER_VARIABLES` command or

[46] Spaces and tabs are the insignificant ones.
[47] In fact, the situation is more complex: `grace-5.1.22` does not use always numbering.
[48] Switching from one project to another one does not reset the user space name, thus it is a good practise to add a command line `@ CLEAR ALLVARS`, destroying all previously user created variables.

interactively in the Window/Console popup with `Edit/List user variables` menu entry. .
Three types of numerical variables are allowed[49]. We note:

- *var* : for a scalar variable,
- *vvar* : for a vector variable (one index array),
- *mvar* : for a matrix variable (two indexes array, stored in memory with the C language convention, i.e. rows first.).

The indices in arrays start at zero, as it is the case in the C language.
The other types are:

- *svar* : string variables (see sec. 8.3.4),
- *setvar* : gives a name to a set, i.e. points toward graph and set id numbers (see sec. 8.10.4),
- *graph* : gives a name to a graph, i.e. point toward a graph id

The table below gives the syntax to define and clear variables or get vector length:

Statement	Description	Example

`DEFINE` *var*
`DEFINE` *var expr*
`DEFINE` *vvar*`[]`
`DEFINE` *vvar*`[`*n*`]`
	define new scalar variable *var*
	define *var* with initial value *expr*
	define new vector variable *vvar* of zero length
	define new vector variable *vvar* of length *n*
		`DEFINE myvar`
		`DEFINE myvar 1.0`
		`DEFINE myvvar[]`
		`DEFINE myvvar[10]`

`DEFINE` *mvar*`[`*m*`][`*n*`]`	define new matrix with *m* rows and *n* columns	`DEFINE mymvar[10][5]`
`DEFINE` *svar* "*string*"	define new string variable. (See sec. 8.3.4)	`DEFINE mysvar "init"`
`DEFINE NEW SET` *setvar* `IN G`*n*	define a new set and a variable name. (See sec. 8.10.4)	
`DEFINE NEW GRAPH` *gvar*	define a new graph and a variable name.	
`CLEAR` *var* `CLEAR ALLVARS` `FORGET` *setvar*	undefine variable *var* and deallocate associated storage (idem for other types of variable) undefine all user defined variables undefine variable but keep the set alive	`CLEAR myvar`
vvar `LENGTH` *n*	reallocate vector variable *vvar*	`myvvar LENGTH 25`
vvar`.LENGTH`	returns the length of vector *vvar*	
vvar[*i*] *vvar*[*i* : *j*] *vvar*[*i* : *j* : *k*]	returns the *i*-th element of a vector variable returns a vector from *i* to *j* inclusive, ($i <= j$) returns a vector with *vvar*[*i*] as first element, *vvar*[*i* + *k*] the second,... until the index is $> j$ or < 0	 `myvvar[3:5]` `myvvar[3:5]`

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

Reserved: `G`*n*, `S`*n* and `P`*n* where *n* is one or more digits [0-9] are reserved to graph, sets and polylines. Note that *n* may be replaced by a $ (a dollar) or a _ (an underscore) with the meaning explained below. Also `A`*n* is reserved to fit parameters (see sec. 9.4).

8.2.4 Graphs, sets and data columns

Graphs (designed by *graph* in tables) may be selected by the following:

Expression	Description	Example
`GRAPH[`*id*`]`	*graph_id*	GRAPH[2]
`G`*id*	*graph_id*	G2
graph	the name of a *graph* user defined variable	mygraph

Graph selection

Sets (designed by *selectset* or *set* in tables) may be selected by:

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

Expression	Description	Example
graph.SETS[*id*]	the set *id* in graph *graph*	GRAPH[0].SETS[1]
graph.S*nn*	the set *nn* in graph *graph*	G0.S1
SET[*id*]	the set *id* in the current graph	SET[1]
S*nn*	the 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$
setvar	the name of a *setvar* user defined variable	tmpset

Set selection

Referring to a set by its id-number is not pregnant and may be not convenient if you want write commands files applying to multiple selections. Moreover S$ or S_ are changed when you operate on the sets, thus a good method is to use *setvar* variables.
Datacolumns into a set are vectors and may be selected by[50]:

Expression	Description	Comment	Example
X	the first column	-	X
Y	the second column	= Y0	Y
Y*n*	$(n+2)$-th column	$n \in [0,4]$	Y3

By default, selection apply to the current set; if you want to select another one, you may concatenate selectors, e.g.

G1.S2.Y3

8.2.5 Geometric objects

Geometric objects are selected by their *objtyp* followed if needed by their *id* number. The *objtyp* is one of the following:

ARC, BOX, LINE, POLYLINE, STRING.

The *id* number of a geometric object must be unique into a project thus no further identification out of the number is needed.

8.2.6 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

Default color map is defined in file Default.agr.
Note that GraceGTK3 new examples rely on the defaults defined in this file, thus may behave unexpectedly if you have modified it.

8.3 Expressions

8.3.1 Elements and operators

GraceGTK3 can compute the value of algebraic expressions and also concatenate strings using the '.' operator. The following types of expressions are defined:

[50] Note that Y0 is a synonym to Y and Z to Y1.

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[5:10]`
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)`

Note that scalar variables are always stored internally as `double` and that *iexpr*, *nexpr* and *indx* results in the nearest integer of this value.

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 *expr*2 if *expr*1 evaluates to `TRUE`, and *expr*3 if *expr*1 evaluates to `FALSE`.

8.3.2 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 expressions operators are applied element by element, i.e. $u * v$ is not the scalar product of the two vectors, but produces a vector w with $w_i = u_i \, v_i \quad \forall i$[51].
- `TP` is a product of vectors of different length (say l_u and l_v):

  $$w = u \ \texttt{TP} \ v$$

  gives a vector of length $l_w = l_u l_v$ with $w_k = u_i v_j$ with $k = i l_v + j$[52].
  Note that this product is not associative.
- For vectors or matrices expressions with a scalar, the operator applies the scalar to all the elements of the array.
  Note that for matrices, only the following is defined:
  `M+a, a+M, M-a, a-M, a*M, M*a, M/a, M%a`.
- For matrices, homogeneous expressions with addition and subtraction only are defined; multiplication, division and power are not in order to avoid confusion between element by element array operations and operations on matrices (as linear operators).
- The `TRANSPOSE(M)` function is defined.

For assignations, the following rules for mixed expressions 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`

Of course, these assignation works only is the dimensions of vectors and matrices are compatibles.
If you have a doubt about the meaning of an operation or want to add new rules, the `Grammar rules` section of file `pars.yacc` is the good place.

[51] Thus, the scalar product may be obtained with `SUM`$(u * v)$

[52] This may be used to define `XYCMAP` sets

8.3.3 Aliasing symbol names

The command

`ALIAS` *newname oldname*

allows to define an alias for a symbol recognized by the interpreter, e.g. `A0, A1,...` may be aliased by more pregnant names to perform a non linear fit.
This may be also useful to refer to a set.

8.3.4 Using string variables

As stated above, a string variable *svar* is defined and initialized by:

`DEFINE` *svar* "*string*"

Normally a *svar* evaluate into a *sexpr* and its value may be changed by assignation:

svar = *sexpr*;

A substring may be obtained using

svar[*nexpr*:*nexpr*]

If the first index is less than zero, an error is issued, if the second is greater than the length of the string, a warning is issued and it is truncated to the string length.

If a string represent a formula, it can be evaluated numerically by

`x = EVALUATE (`*sexpr*`)`

where the left hand variable `x` can be a scalar or an array (i.e. a component of a set or and array created with `DEFINE`). **Note** that the left hand variable is used to determine the size of the array (or if it is a scalar), **thus nothing else is allowed in the statement**.
The function

`REPLACE(`*str*, *old*, *new*`)`

replace everywhere in *str* the substring *old* by *new* and evaluate as a *sexpr*.

8.4 Flow control

Statement	Description	Example
`EXIT(`*status*`)`	cause normal program termination with exit status *status*	`EXIT(0)`
`EXIT`	cause normal program termination; same as `EXIT(0)`	`EXIT`
`IF ( `*iexpr*`) {`	flow control (see note)	`IF ( a < b) {`
`} ELSE {`	flow control (see note)	
`}`	end of a IF block (see note)	

Note: each of the three commands: `IF (` *nexpr* `) {, } ELSE {` and } must be alone on one line.

8.5 Project and files management

When a file is opened inside another one, the control returns to the calling one at the end of the read.
The following commands are used:

Statement	Description	Example
`NEW` `NEW FROM` *file*	Start a new project with `Default.agr` template Start a new project with *file* template	 `NEW FROM "mytemplate.agr"`
`LOAD` *file* `READ BATCH` *file* `RETURN`	load existing project *file* Read procedure file. (see note 1) (see note 1)	`LOAD "foo.agr"` `READ BATCH "myproc.agr"`
`GETP` *file*	Load existing parameter *file* (see note 2)	`GETP "foo.par"`
`SAVEALL` *file* `SAVEP` *file*	save project to *file* save parameters to *file*	`SAVEALL "foo.agr"` `SAVEP "foo.par"`
`READ` *file*	Read set(s) see table 14.	
`OPEN RESULTS FILE` *file* `CLOSE RESULTS FILE`	Results are also printed in *file*. (see note 3) Close the results file.	
`CD` *dir* `GETWD`	Change the working directory (see note 4) Returns the working directory as a string	
`CLOSE` `CLOSE` *file*	close a real time input idem	
`OPEN LOG FILE` *file* `CLOSE LOG FILE`	Open "*file*" for debugging (see note 5) Close the log file.	

Notes

1. `LOAD` *reset project as* `NEW` *do, but* `READ BATCH` *does not, allowing to embed procedures.*
   `RETURN` *in an embedded file, stop reading the file, close it and return to the upper one.*
2. `GETP` *load a* `.par` *file containing only commands without a starting* `@`.
3. Same as the command line option `-results`.
   Only one opened file is allowed at a time. If the file already exists, in interactive mode, the permission to overwrite is asked and in batch mode, it is not opened.
4. The *working directory* is internal to `GraceGTK3` and the user shell one is not changed.
5. All the lines passed to the scanner are written into the file.

8.6 GUI interaction

Statement	Description	Example
`HELP` ***url*** `HELP`	open a HTML document pointed to by *url* open User's Guide	`HELP "doc/FAQ.html"` `HELP`
`REDRAW`	refresh the canvas to reflect the current project state	`REDRAW`
`SLEEP` ***n***	sleep for n seconds	`SLEEP(3)`
`BEEP` *onoff*	allows console beep on error (default *on*)	
`GUI COLOR` $(n, red, green, blue)$	 see sec 2.3.4	
`GUI SIZE` (x, y)	see sec 2.3.4	
`GUI` *title* (x, y)	pop up positions: see sec 2.3.4	
`ECHO` ***sexpr*** `ECHO` *expr*	write in the status message zone (see note) idem with %g format	
`MESSAGE` *sexpr*	Display a message in the **console window**	

Note about `ECHO`: beware that the program usually will overwrite your message.

8.7 Printing

For producing "hard copy", several parameters can be set via the command interpreter. They are summarized in table 4.

Command	Description
`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)
`HARDCOPY DEVICE` "*devname*"	set device *devname* as current hardcopy device
`PAGE SIZE` *xdim*, *ydim* `PAGE SIZE` "*stdsize*" *orientation*	set page dimensions of all devices in pixels[53] 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*

Table 4: Commands to set device parameters

[53] It is assumed that 1 pixel = 1/72 inch

8.8 Preferences and date

See table 5.

REFERENCE DATE (*julian_day*)	see section 14.2
DATE AUTO *onoff* DATE WRAP *onoff* DATE WRAP YEAR *year*	*on*: auto-update of time stamp *on*: use only two digits
DEFAULT LABEL COPY *onoff* DEFAULT CURSOR *nexpr* DEFAULT LINESTYLE *nexpr* DEFAULT LINEWIDTH *expr* DEFAULT COLOR *color* DEFAULT PATTERN *nexpr* DEFAULT CHAR SIZE *expr* DEFAULT FONT *font* DEFAULT FONT TOOL *onoff* DEFAULT FORMAT *format* DEFAULT LOCALE *onoff*	*on*: add "(copy of...)" to label of copied items 0:pointer 1: crosshair Pop Font Tool when a string is clicked
LINK PAGE *onoff* AUTO REDRAW *onoff*	*on*: all graph are scrolled synchronously
FOCUS *onoff* FOCUS CLICK FOCUS SET FOCUS FOLLOWS	*on*: display focus markers click to change graph focus focus cannot be changed by mouse focus follows mouse
MAX POINTS *nexpr*	change the max drawing path length (see 3.4.15)

Table 5: Commands to manage preferences and dates

8.9 Functions

This section describes the functions hard coded in `GraceGTK3` .
You may add your own functions by using the DL module feature (see sec. 13.3).

8.9.1 Strings functions

Function	Description	Example
`ECHO` *`sexpr`* `ECHO` *expr*	write in the status message zone idem with %g format	

`GETENV (`*`sexpr`*`)`
 	
`STRCMP (`*`sexpr,sexpr`*`)`
	call getenv(), i.e. returns the value of an environment variable
	call C function strcmp() : compare two strings
		`GETENV ("GRACEGTK_HOME")`
		
		`STRCMP (s1 ,"first")`

8.9.2 Elementary functions

The list is given in table 6.

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 6: Elementary functions

8.9.3 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

8.9.4 Statistical functions

The list is given in table 7.

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 7: Statistical functions

8.9.5 Special math functions

The list is given in table 8.

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 8: Special math functions

8.10 Defining drawables: WITH, TARGET, BEGIN and END

8.10.1 WITH

- WITH *graph_id* is used to define a new graph or make it current when it already exists.
  Note that once the graph is current, the commands in tables 11, 12 and 13 addresses this particular graph without further reference to it.
  If a data sequence follows, a new set is created into this graph.
- WITH *selset* where *selset* may be Gx.sy or sy
  - if set already exists, make it current
  - if not, create set(s) from the last existing one in graph up to y.

  N.B. If you don't want to give a number for the set, use DEFINE NEW SET (See 8.10.4).
- WITH *geom_typ* [*id*] where *id* is optional is used to define a new geometric object (see table 19).
  If the object *id* already exists, it is made current.

8.10.2 TARGET

This command is used to configure the parser before reading ASCII numeric data.

- TARGET *graph_id* will direct subsequent numeric data towards the block data and this graph,
- TARGET *selectset* will direct subsequent numeric data towards the block data and this set,
- TARGET *selectset.datacolumn* will direct subsequent numeric data only towards the data column of this set. The block data is not affected. The set must be a valid set and the column length will be extended if needed. It is the responsibility of the user to read all the columns with the correct length needed for a given type of set.
- TARGET *vvar* will direct subsequent numeric data towards the 1D array *vvar*. This array must be already defined and its length will be extended if needed.
- TARGET POLYLINE *id* will direct subsequent numeric data towards this polyline.

Note that for large arrays, the use of the NetCDF binary format may be preferred (see 8.15.4).

8.10.3 BEGIN and END

The above commands, inherited from grace-5.1.22 , presents severe drawbacks: the *id number* of elements overlap if you want to merge files and also to build a compound, you have to use the *id number*. This can be avoided if a BEGIN/END scheme reproducing the tree structure of the drawing is used and no further reference to the given element is made outside the BEGIN/END pair.
Thus GraceGTK3 defines

- BEGIN GRAPH ... END GRAPH
- BEGIN SET ... END SET
- BEGIN *objtyp* ... END *objtyp*

Note that the *follow-me* mode is designed on a line-by-line scheme and must not depend onto the preceding and following lines, thus does not use this scheme but the numbered one.

8.10.4 Aliasing names of sets

Referring to a set by its id-number is often cumbersome and it is possible[54] to give a name to a set by defining a *setvar* variable.
Every time a command definition refer to *set* or *selectset*, you may use the variable name.
To define such variables:

- ALIAS *setvar set* gives a name to an existing set,
  e.g. ALIAS myset G1.s2 will associate myset to this set.
  Beware that no attempt to manage the validity of the link is done, thus if you kill the set using the KILL command, the variable always exist but point toward a pair (graph_id, set_id) inactivated.
- DEFINE NEW SET *setvar* IN *graph*$_n$ creates at the same time a new set in graph Gn and a new variable.

[54] Since 0.9.5

To undefine these variables:

- `CLEAR` *setvar* also kill the set,
- `FORGET` *setvar* only undefine the variable, but keep the set alive.

Combining these features with `S$` or `S_` allows to create scripts independent of graph and set numbering.

8.11 Procedures to create new sets or transform old ones

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.

8.11.1 Simple operations

Statement	Description	Example
`DEFINE NEW SET` *setvar* `IN G`*n*	see 8.10.4	
`WITH` *selset*	see 8.10.1	`WITH G1.s2`
`READ` *file*	see 8.13.1	`READ "mydata"`
`COPY` *src* `TO` *dest*	Copy *src* to *dest*	`COPY S0 TO S1`
`MOVE` *src* `TO` *dest*	Move *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`
`SWAP` (set, col_1, col_2)	Swap col_1 and col_2 in *set*	`SWAP (G0.S0 ,X ,Y)`
set `POINT` x, y	Add a point at the end of *set* (*deprecated*)	`G0.S0 POINT 10.0 ,5.0`
set `DATAPOINT` $x, y, [y1, ..]$	Add a point at the end of *set*(*see note*)	`S$ DATAPOINT 1 ,10 ,11 , "a row"`
set `POINT` x, y, n	Insert point at index n of *set*	
set `XYZ POINT` x, y, z	(*deprecated, see note below*)	
set `DROP` $n1, n2$	Drop points $n1$ to $n2$	`G0.S0 DROP 5 , 10`
set`.x`$[n] = expr$	Move point n by changing abscissa	`G0.s1.x[6] = 0.5`
set`.y`$[n] = expr$	Move point n by changing ordinate	`G0.s1.y[7] = 0.4`
`APPEND` *set*2 `TO` *set*1	Append sets (same graph) then kill *set*2	
`JOIN` $s1 : s2 : s3 \ldots$	Join sets then kill $s2, s3 \ldots$ (same graph)	`JOIN G0.s1:s2:s3`
`SPLIT` *set* n	Split *set* int sets of length n	

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

Note: To add a point at the end of a set, `POINT` and `XYZ POINT` are deprecated by `DATAPOINT` which allows to give the two mandatory values x and y and also the others values according to the type of *set* as well as an optional string used for annotations.
`POINT` should be used only to insert a point after a given row ,say n. If the dataset needs more than 3 columns, e.g. 4, the last one (i.e. Y2) can be set with the following command:

```
S0.Y2[n] = expr
```

The more sophisticated operations accessed by the `Compute` menu are described in table 10.
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`.

8.11.2 Data transformations

	Using *npoints*, run on *set* and create a new set	
RUNAVG (*set*, *npoints*)	Running average	RUNAVG (S0, 100)
RUNMED (*set*, *npoints*)	Running median	RUNMED (S0, 100)
RUNMIN (*set*, *npoints*)	Running minimum	RUNMIN (S0, 100)
RUNMAX (*set*, *npoints*)	Running maximum	RUNMAX (S0, 100)
RUNSTD (*set*, *npoints*)	Running standard deviation	RUNSTD (S0, 100)
RUNAVG (*set*, *npoints*, *set2*, *region*, *invr*)		
	extra arguments: *set2*= output, *region* number or -1 if no region is used, *invr*: inverse region	
idem for RUNMED, RUNMIN, RUNMAX, RUNSTD		
DIFFERENCE (*set*, *set2*, *xplace*, *type*, *stride*)	see Differences/derivation	
DIFFERENCE (*set*, *xplace*, *type*, *stride*)		
DIFFERENCE (*set*, *xplace*)	*type* = 0: plain difference, 1: derivative *xplace* = 0: left, 1: centered, 2: right if *set2* is not specified, a new set is created if *stride* is not specified, *stride* = 1.	
SAMPLE (*set*, *set2*, *start*, *step*)	see Sample points	
SAMPLE (*set*, *set2*, "*logical_expression*")		
SAMPLE (*set*, *set2*, *type*)	*type*: MIN, MAX or MINMAX	SAMPLE (G0.s0,G0.s1 ,MAX)

Table 9: Commands to compute averages,...

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*1, *set*2, *s*, *p*)	interpolate *set* using XY-splines with parameters *s* and *p*. (see Interpolation/Splines)	XYSPLINE (G0.s0 ,G0.s1 ,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)
HISTOGRAM(*set*, *bins*, *cumulative*, *normalize*, *set*2)	Same as above, but result go into *set*2	
HISTOGRAM(*set*, *setx*, *cumulative*, *normalize*, *set*2)	Same as above, but *setx* abscissas are used as bins values	
XCOR(*set*1, *set*2, *maxlag*, *covar*) XCOR(*set*1, *set*2, *maxlag*, *covar*, *dest*)	calculate cross-correlation (or -covariance if the *covar* flag is set) of *set1* with *set2* with maximum lag *maxlag*. If set *dest* is not specified or invalid, a new one is created	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)
REGRESS(*set*, *deg*)	Regression of degree *deg*, result in a new set	REGRESS (S0, 1)
FIT...	see section 9.4	
NONLFIT...	Levenberg nonlinear fit see section 9.4	
LOESS...	LOESS nonlinear fit see section 9.4	
	Levenberg non linear fit on set *set* with *nsteps* steps. See Data menu Non linear fit	
NONLFIT(*set*, *weight*, *nsteps*)	*idem* + a weight array *weight*	
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)
LOESS(*set*, *family*, *degree*, *span*, *baronoff*, *level*, *nbar*, *dest*)	see section (9.4.2)	
FFT(*set*, *filter*) FFT(*set*, *intyp*, *filter*, *xscale*, *outtyp*) FFT(*set*, *intyp*, *filter*, *xscale*, *outtyp*, *outseg*) FFT(*set*, *set*2, *outtyp*, *norm*, *inseg*, *intyp*, *dcdump*, *filter*, *padding*, *round2n*, *xscale*, *outseg*)	see section (10.4.2)	
FILTER	see section 11.3	
WAVELET(*set*, *type*, *stride*, *k*, *invflag*)	see section (12.5)	
WAVELET(*set*, *type*, *stride*, *k*, *invflag*, *setout*)	idem	

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.
Arguments *strict*, *cumulative*, *normalize*, *covar*, *trace* and *negate* are of type *onoff*: ON or OFF.
Argument *restriction* is a *vexpr* and *region* a region selector.

Table 10: Commands to transform datasets

8.11.3 Miscellaneous

- `@ GEOTRANSFORM` ($set1, set2, order, deg, cx, cy, tx, ty, sx, sy$)
  Apply a geometric transformation with parameters:

  1. $set1$ the set to transform,
  2. $set2$ the resulting set,
  3. $order$ a string describing the order used to apply elementary transforms by a permutation of 3 letters `R` for rotation, `T` for translation, `S` for scaling, (e.g. `"RTS"`),
  4. deg the angle of the rotation in degrees,
  5. cx and cy, the center of the rotation,
  6. tx and ty, the translation vector,
  7. sx and sy, the scaling factors.

  Note that the transformation is applied to the abscissas and ordinates of $set1$ in the world coordinates, thus that the set must be of the `XY` type and this transformation is useful only if `graph fixed` option is used, i.e. if the y/x scaling ratio of axes is unity.

- `@ DECIMATE` ($set1, set2, type, worldview, dx, dy, dxtype, dytype$)
  Reduce the size of datasets by decimation based on a distance criterion (associated with the `Data/Transformation/Prune` popup).

  1. $set1$ the set to transform,
  2. $set2$ the resulting set,
  3. $type$ of the surface element used to check if it contains several points:
     0: interpolation (ordered sets only), 1: circle ,2: ellipse ,3: rectangle,
  4. $worldview$ the coordinate system used to give the size (`VIEW` or `WORLD`),
  5. dx, dy: the size of the element,
  6. $dxtype, dytype$: 0: linear, 1: logarithmic.

8.12 Graphs

8.12.1 General graphs operation

See table 11.
`Graph` *type*: `XY ,CHART, POLAR, FIXED` and `PIE`: see section 2.1.4.
`Overlay` *hints*: 0=disabled, 1="X and Y axes different", 2="Same X axis scaling", 3="Same Y axis scaling", 4="Same X and Y axis scaling".

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)`
`PACK GRAPHS` renum. graphs wiping inactive ones	
`AUTOSCALE` scale the current 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 S0`
`AUTOSCALE` *graph* scale a given graph	`AUTOSCALE G1`
`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	`TYPE XY` (see note)
graph `TYPE` *type* Sets *type* of *graph*	`G0 TYPE POLAR`
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`
`OVERLAY` $(g_1, g_2, hints)$ Overlay g_1 onto g_2.	*hints*: see section 8.12.1

Table 11: Commands for general graphs operation

8.12.2 Axis parameters

See table 12.

8.12.3 Titles, frame and legend

See table 13.

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 x_{min}	Sets minimum value of current graph's x axis to x_{min}	WORLD XMIN -10
WORLD XMAX x_{max}	Sets maximum value of current graph's x axis to x_{max}	WORLD XMAX 22.5
VIEW XMIN x_{min}	Sets left edge of current graph at $x = x_{min}$ in the viewport	VIEW XMIN .2
VIEW XMAX x_{max}	Sets right edge of current graph at $x = x_{max}$ 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
XAXIS *onoff*	Sets axis visibility	XAXIS ON
XAXES SCALE *type*	Sets 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 TYPE ZERO *onoff*	Sets if axis ordinate is zero	XAXIS TYPE ZERO FALSE
XAXIS TICK *onoff*		
XAXIS TICK MAJOR *dx*	Sets distance between major ticks	XAXIS TICK MAJOR 2.0
XAXIS TICK MINOR TICKS *n*	Number of minor ticks between two majors ticks	
XAXIS TICK OFFSETX *dx*		
XAXIS TICK OFFSETY *dy*		
XAXIS TICK ALT *onoff*		
XAXIS TICK MIN x_{min}		
XAXIS TICK MAX x_{max}		
XAXIS LABEL "*label*"		
XAXIS LABEL LAYOUT [Para|Perp]		
XAXIS LABEL CHAR SIZE *size*		
XAXIS LABEL FONT *font*		
XAXIS LABEL color *color*		
XAXIS LABEL PLACE [Normal|Opposite|Both]		
XAXIS LABEL PLACE [Auto|Spec]		
XAXIS LABEL PLACE x, y	If Spec is used	
To be finished ...		

Table 12: Commands to set axis parameters

XAXIS may be changed into YAXIS, ALTXAXIS or ALTYAXIS.

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"
FRAME *type*	0:closed 1: half open up,...	
FRAME *layer*		FRAME 48
FRAME LINESTYLE *nexpr*	0:none 1:solid, 2:dotted,...	
FRAME LINEWIDTH *expr*		
FRAME COLOR *color*		
FRAME PATTERN *nexpr*		
FRAME BACKGROUND COLOR *color*		
FRAME BACKGROUND PATTERN *nexpr*		
LEGEND onoff	Toggle legend display	LEGEND ON
LEGEND LOCTYPE *type*	*type* of coordinates for box	LEGEND LOCTYPE WORLD
LEGEND *xloc, yloc*	Set upper left corner of legend box	LEGEND .5,.75
LEGEND G*n* ANCHOR *xloc, yloc*	idem for graph G*n*	LEGEND G1 ANCHOR .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 "red"
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 13: Commands for graphs titles, frame and legend

8.13 Sets

8.13.1 General sets operation

See table 14.

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"
PACK G*n*	renum. sets in graph *n*, wiping killed ones	PACK G0
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
set LINK *srctype name*	Hot links *set* to a file or pipe	G0.s0 LINK FILE "fn.dat"
set LINK *onoff*	Tags the set as linked/unlinked	

Table 14: Commands for set operation

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

8.13.2 Points operation

See section 8.11.1

8.13.3 Sets parameters

Command	Description	Example
set HIDDEN onoff	To hide the set	s0 HIDDEN FALSE
set LAYER *nexpr*	Choose the layer $\in [1, 99]$	s0 LAYER 45
set TYPE *settype*	The type of the set	s0 TYPE XY
set SYMBOL *nexpr*	Number of symbol to be used	s0 SYMBOL 10
set SYMBOL *sexpr*	Name of symbol to be used	s0 SYMBOL "star"
set SYMBOL SIZE *expr*	Size of the symbol	s0 SYMBOL size 1.0
set SYMBOL COLOR *color*	Color of the symbol	s0 SYMBOL color "red"
	Symbol fill	
set SYMBOL FILL PATTERN *nexpr*	Pattern	s0 SYMBOL FILL pattern 0
set SYMBOL FILL COLOR *color*	Color	s0 SYMBOL FILL color 1
	Symbol outline	
set SYMBOL LINESTYLE *nexpr*	Style	s0 SYMBOL linestyle 1
set SYMBOL LINEWIDTH *expr*	Width	s0 SYMBOL linewidth 2.0
set SYMBOL PATTERN *nexpr*	Pattern	s0 SYMBOL pattern 1
set SYMBOL CHAR *nexpr*	Character number	s0 SYMBOL char 65
set SYMBOL CHAR FONT *nexpr*	Character font	s0 SYMBOL char font 0
set SYMBOL SKIP *nexpr*	Skip n points	s0 SYMBOL skip 1
set SYMBOL START *nexpr*	Start symbols at point n	s0 SYMBOL start 1
set LINE TYPE *nexpr*	Line connection $\in [0, 6]$	s0 LINE type 1
set LINE TYPE *typ*, *method*, *len*	Spline conn. (see note)	
set LINE TYPE *typ*, *method*, *len*, s, p	idem	
set LINE LINESTYLE *nexpr*	Line style $\in [0, 8]$	s0 LINE linestyle 1
set LINE LINESTYLE *style*	by name (see note)	s0 LINE linestyle "solid"
set LINE LINEWIDTH *expr*	Line width	s0 LINE linewidth 1.5
set LINE COLOR *color*	Line color	s0 LINE color 4
set LINE pattern *nexpr*	Line pattern	s0 LINE pattern 1
set BASELINE onoff	On: draw the baseline	s0 BASELINE off
set BASELINE TYPE *nexpr*	If $n > 0$, draw the baseline	s0 BASELINE type 0
set DROPLINE onoff		
	On: draw lines from data to baseline	s0 DROPLINE off
set FILL TYPE *nexpr*	0: none, 1:polygon, 2:baseline	s0 FILL type 2
*set*1 FILL TO *set*2	fill between *set*1 and *set*2	s0 FILL TO s2
set FILL RULE *nexpr*	0: winding, 1:evenodd	s0 FILL rule 0
set FILL COLOR *color*	Color to fill	s0 FILL color 1
set FILL PATTERN *nexpr*	Pattern	s0 FILL pattern 1
set LEGEND onoff	Display set in legend box	s0 LEGEND on
set LEGEND "*legend*"	Set legend (or label)	s0 LEGEND "First run"

Table 15: Commands to change sets parameters (*except annotations and error bars*)

Notes: the numbers to be used for colors, patterns, linestyles, etc, can be inferred from the graphical interface and colors numbers may vary with the settings in the default file.

1. Symbol types: *0:none, 1:circle, 2:square, 3:diamond, 4:triangle_ up, 5:triangle_ left, 6:triangle_ down, 7:triangle_ right, 8:plus, 9:cross, 10:star, 11:character.*
2. FILL TO is in fact automatically associated to fill type 3 and *set*1 and *set*2 must belong to the same graph.
3. Line type of connections are *0:None, 1:Straight, 2:Left stairs, 3:Right stairs, 4:Segments, 5:3-Segments and 6:Spline.*
4. Line styles: *0:none, 1:solid, 2:dotted, 3:dashed, 4:long_ dashed, 5:dot_ dashed, 6:dot_ long_ dashed, 7:dot_ dot_ dashed, 8:dot_ dashed_ dashed.*
5. Methods for spline connections are: *0:Linear, 1:Cubic spline, 2:Akima spline, 3:X-spline and 4:XY-spline;* in the last case, s and p are the parameters governing the XY-spline interpolation.

8.13.4 Annotations

Command	Description	Example
set AVALUE onoff	Display annotations	
set AVALUE TYPE *nexpr*		
		s0 AVALUE type 2
set AVALUE CHAR SIZE *expr*		s0 AVALUE char size 1.0
set AVALUE FONT *nexpr*		s0 AVALUE font 0
set AVALUE COLOR *color*		s0 AVALUE color 1
set AVALUE ROT *expr*		s0 AVALUE rot 0.0
set AVALUE FORMAT *format*	See format list below	s0 AVALUE format general
set AVALUE PREC *nexpr*		s0 avalue prec 3
set AVALUE PREPEND *string*		s0 avalue prepend "y = "
set AVALUE APPEND *string*		s0 avalue append " km"
set AVALUE OFFSET x, y		s0 avalue offset 0.0 , 0.20
set AVALUE SKIP *nexpr*		s0 avalue skip 0
set AVALUE SKIP *nexpr*, *start*		s0 avalue skip 0 , 0

Table 16: Commands to change sets annotations

AVALUE TYPE index means:
0: none, 1:x, 2:y, 3:(x, y) ,4:string ,5:z ,6:the index of the data point.
FORMAT is one of:
decimal, exponential, general, power, scientific, engineering, computing, ddmmyy, mmddyy, yymmdd, mmyy, mmdd, monthday, daymonth, months, monthsy, monthl, dayofweeks, dayofweekl, dayofyear, hms, mmddhms, mmddyyhms, yymmddhms, degreeslon, degreesmmlon, degreesmmsslon, mmsslon, degreeslat, degreesmmlat, degreesmmsslat, mmsslat.

8.13.5 Error bars

Command	Description	Example
set ERRORBAR onoff		s0 ERRORBAR ON
set ERRORBAR PLACE {NORMAL | OPPOSITE | BOTH}		s0 ERRORBAR place BOTH
set ERRORBAR COLOR *color*		s0 ERRORBAR color 1
set ERRORBAR PATTERN *nexpr*		s0 ERRORBAR pattern 1
set ERRORBAR SIZE *expr*		s0 ERRORBAR size 1.0
set ERRORBAR LINEWIDTH *expr*		s0 ERRORBAR linewidth 1.0
set ERRORBAR LINESTYLE *nexpr*		s0 ERRORBAR linestyle 1
set ERRORBAR RISER LINEWIDTH *expr*		s0 ERRORBAR RISER linewidth 1.0
set ERRORBAR RISER LINESTYLE *nexpr*		s0 ERRORBAR RISER linestyle 1
set ERRORBAR RISER CLIP onoff		s0 ERRORBAR RISER CLIP off
set ERRORBAR RISER CLIP LENGTH *expr*		s0 ERRORBAR RISER CLIP LENGTH 0.1

Table 17: Commands to change errobars parameters

8.13.6 Commands for XYCMAP and XYCSYM sets

XYCMAP needs to specify the size of the rectangular grid in number of points:

selectset `TYPE XYCMAP` *nx ,ny*

If only the z vector is given (see 7.1), the regular rectangular grid must be defined with its corners:

selectset `XYCMAP :` *x1 ,y1 ,x2 ,y2*

`XYCSYM` specification of type obey to common rule.
The *Colorbar* is an object (see table 21, of course not all command are pertinent) with special commands:

Command	Description
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	choose the color map
set`CMAP "name"`	choose the color map

Color map 0 is the current `GraceGTK3` 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".*

XYCMAP may include contour curves. See the commands table 18:

Command	Description	Remarks
set `XYCMAP` $(col, up, left, x1, y1, x2, y2)$	see note 1	`see sec.` 7.1
set `XYCMAP :` $x1, y1, x2, y2$	deprecated	`use command above`
set `CONTOUR` *onoff*	show/hide	`G1.s0 CONTOUR ON`
set `CONTOUR COLOR` *color*	color of contour lines	`G1.s0 CONTOUR COLOR "white"`
set `CONTOUR LINEWIDTH` *width*		`G1.s0 CONTOUR LINEWIDTH 2.5`
set `CONTOUR LABEL` x, y, z	define a label on a curve	

Table 18: Command defining parameters of contour curves

Note 1 $col, up, left$ are boolean with the following meaning:

- if $col =$ `TRUE`, z read/write columns first (Fortran default)
- if $up =$ `TRUE`, z read/write y upward,
- if $left$ `TRUE`, z read/write x leftward,

$x1, y1, x2, y2$ are coordinates of two opposite corners. **Note 2** that the `LABEL` command is not designed for user, but for internal (save) use: if no curve crosses the point of coordinates (x, y), it is ineffective.

8.13.7 Enquiry commands

These commands return as a number the value of various parameters for a given set.

Command	Description
GET *set* LENGTH	See note 1
GET *set* HIDDEN	False=0, True=1
GET *set* LAYER	
GET *set* ID	the set number
GET *set* GRAPH	
GET *set* LINE COLOR	
GET *set* LINE LINESTYLE	
GET *set* LINE LINEWIDTH	
GET *set* LINE TYPE	
GET *set* SYMBOL SIZE	
GET *set* SYMBOL COLOR	
GET *set* SYMBOL FILL COLOR	
GET *set* SYMBOL FILL PATTERN	
GET *set* SYMBOL CHAR NUM	
GET *set* SYMBOL CHAR FONT	
GET *set* SYMBOL SKIP	
GET *set* SYMBOL START	

Note 1: Usually, data columns have the same length, but if it is not the case, the command returns min(lx,ly), except for XYCMAP sets, based on 2D arrays. In that case, the returned length is the length of the second column (y) because the length of x is needed to loop on array elements.

Some commands have also a short form, but to keep the grammar consistent this syntax cannot exist for all the parameters above.

Command
set.LENGTH
set.HIDDEN
set.LAYER
set.ID
set.GRAPH
set.SYMBOL

These commands return as a string the value, e.g. the name of the color is returned, not its id number:

Command
TYPE (*set*)
LINE COLOR (*set*)
SYMBOL COLOR (*set*)
FILL COLOR (*set*)
LEGEND STRING (*set*)
= LEGEND *set*[*]

[*] = LEGEND *is deprecated and does not work in strings assignations to a variable. Since 1.0.0*

8.14 Regions

R*n*	region n, $n \in [0,4]$.	RO
R*n* *onoff* R*n* TYPE *rtype* R*n* LINE $x1, y1, x2, y2$ R*n* XY x, y	activate or deactivate region n *rtype* can be ABOVE, BELOW, LEFT, RIGHT, POLYI, POLYO, HORIZI, VERTI, HORIZO, VERTO add point (x, y) to region coord.	
LINK R*n* TO *graph*		
R*n* *color* R*n* LINESTYLE *nexpr* R*n* LINEWIDTH *expr*		

8.15 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 19: *objtyp* and *selobj* definition

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

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

Table 20: WITH command

N.B.

- Some parameters have no meaning when used with some types of objects, but they will be accepted by the interpreter, often 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 21.
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	*selobj*					
		Arc	Box	Line	Polyline	String	Timestamp
selobj : $x1, y1, x2, y2$	Object coordinates (see note 1)	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 BACKGROUND FILL *onoff*		x	x			x	x
selobj BACKGROUND FILL *color*		x	x			x	x
selobj BOX COLOR *color*	Color of boundaries of box					x	x
selobj CHAR SIZE *size*				x		x	
selobj COLOR *color*		x	x	x	x	x	
selobj FILL COLOR *color*		x	x			x	x
selobj FILL PATTERN *color*		x	x			x	x
selobj FONT n	Font selection			x		x	x
selobj FONT "*name*"				x		x	x
selobj HIDDEN *onoff*		x	x	x	x	x	x
selobj JUST n	justification					x	x
selobj LAYER *nexpr*		x	x	x	x	x	x
selobj LINESTYLE *style*	*style* 0: no line, 1: continuous, etc	x	x	x	x	x	x
selobj LINEWIDTH *width*		x	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 (see note 2)	x	x			x	
selobj TYPE n	Arc subtype	x					
LOAD IMAGE FILE $filename, x_1, y_1, x_2, y_2$	see section 3.12.3						
IMPORT XFIG $filename, x_1, y_1, x_2, y_2$	see sections 8.15.3 and 3.3.10						

Table 21: Commands related to geometrical objects.

Note 1. The colon (:) between *selobj* and the coordinates is needed.

Note 2. Angles are in degrees.

8.15.1 Polylines

Polyline points are usually defined using the `TARGET` command (see. 8.10.2). To move only one point:

Command	Example
P*id*.x[*n*] = *expr*	P1.x[2] = 3.4
P*id*.y[*n*] = *expr*	P0.y[3] = 4.3
POLYLINE *id* MOVE POINT *n* : *x*, *y*	POLYLINE 1 move point 4 : 1.2 , 3.4

8.15.2 Compounds

Compounds are defined in section 2.1.9, the GUI interactions are described in section 6.2 and commands are summarized in table 22.
To define a compound in a script, two forms are possible:

- `BEGIN COMPOUND` create a new empty compound and the following objects are glued into this compound, until a `END COMPOUND` is encountered.

- `NEW COMPOUND` *selobj* defines a compound with *selobj* as first object glued. then, you can add objects using: `COMPOUND` *id type_of_object id_of_object*

Note that strings are not rescaled when the compound scaling is not unity, thus the user may have to adjust manually the size and place, taking into account the justification rule used.

`BEGIN COMPOUND`	The objects that follow will be glued into a new compound.
`END COMPOUND`	End of the BEGIN COMPOUND sequence
`NEW COMPOUND` *selobj*	Create a new compound with *selobj* as first element
`COMPOUND` *id selobj*	Add *selobj* to compound *id*
`COMPOUND` *id* `HIDDEN` *onoff*	
`COMPOUND` *id* `LOCTYPE` *worldview*	
`COMPOUND` *id* `BREAK`	Breaks the compound
`COMPOUND` *id* : x_1, y_1, x_2, y_2	Scales the compound (see note)

Table 22: Commands related to compounds of geometrical objects.

Note. The colon (:) between *id* and the coordinates is needed. Since 0.8 fit the compound into rectangle with corners x_1, y_1, x_2, y_2 (the doc was in error from 0.8 to 1.0.1)

8.15.3 Importing Xfig files

See section 3.3.10 for details about the process.
The command syntax is:

`IMPORT XFIG` *filename*, x_1, y_1, x_2, y_2

where (x_1, y_1) is the lower left corner and (x_2, y_2) the upper right one of the bounding box of the compound produced by the command.
Note that only viewport coordinates are allowed. If one coordinate is less than zero, it is computed by the program to keep the same aspect ratio as the original `Xfig` object.
Example:

```
@ IMPORT XFIG "xmix.fig" , 0.15 , -1 , 0.50 , 0.51
```

will scale the object to a width of 0.35 and a lower left corner at (0.15 , 0.50).

8.15.4 Importing and exporting NetCDF files

See sections 3.3.11 and 3.3.12 for details about the process.

`IMPORT NETCDF` *filename*, *selectset*, *settype*

is used to import a NetCDF file and gives the type of the new set to be created. It is necessary to associate at least a variable name in the NetCDF file to a dataset column before with the command

NETCDF *datacolumn*, *varname*

If no association is done, default is to use the index of the array. Giving only indexes does not allows to specify the number of points thus is an error.
Example:

```
@ NETCDF X "Index"
@ NETCDF Y "longitude"
@ IMPORT NETCDF "pres_temp_4D.nc" , G0.s0 ,XY
```

All the active sets may be exported at a time in NetCDF format with the command

EXPORT NETCDF *filename*

The file produced may be read in with

IMPORT NETCDF *filename*

Note that this simple import commands works only if the file is conform to GraceGTK3 format.

9 Non-linear fit

Three methods are available to fit a $y = f(x)$ curve to a dataset [55]: Levenberg-Marquardt and LOESS (or LOWESS) will adjust automatically the values of the parameters for you and a "by hand" one to change manually the values with spin buttons and display immediately the result.
In addition to the curve, some statistic is printed into the console window and the fitting values can be saved with the `Edit` local menu.
It is also possible to write the results to file in a batch with different methods:

- use the `-results` command line option, (see 2.2.1),
- use the `OPEN results file` command (see 8.5).

9.1 Levenberg-Marquardt algorithm

This is the method already use by `grace-5.1.22` .
It is mainly a gradient method to minimize the euclidean distance between the dataset and the fit[13].
The fit is a function with up to 10 unknown coefficients named `A0, A1,...,A9`. A formula using these coefficients is provided by the user and the iterations stops when the algorithm is stabilized within `tol` parameter value.
N.B. The `tol` *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.*
It is possible to define the starting values for each unknown, to assign a weight to each data point and a restriction interval.
If we write (x_i, y_i) the data points and $\tilde{y}_i$ the ordinates of the fit, the following quantities[56] are printed:

- Chi-square i.e. $\chi^2 = \sum_i |\tilde{y}_i - y_i|^2$
- Correlation coefficient[57] [6]: if $\bar{y}$ is the mean of y and $\check{y}$ the mean of $\tilde{y}$

$$r = \frac{\sum_i (y_i - \bar{y})(\tilde{y}_i - \check{y})}{\sqrt{\sum_i (y_i - \bar{y})^2} \sqrt{\sum_i (\tilde{y}_i - \check{y})^2}} \tag{3}$$

- The Theil U coefficient computed by `grace-5.1.22` and `GraceGTK3` is defined by

$$U = \frac{\sqrt{\sum_i (\tilde{y}_i - y_i)^2}}{\sqrt{\sum_i y_i^2}}. \tag{4}$$

  Note that this coefficient, different from the one described in Wikipedia[12], is sometime written UII [2].
- If $y_i \neq 0 \ \ \forall i$ the RMS error is defined as

$$rms = \sqrt{\frac{1}{n} \sum_i \frac{(\tilde{y}_i - y_i)^2}{y_i^2}} \tag{5}$$

9.2 LOESS

It is based on a local regression method[14] [5]. Up to now, only a simple curve fitting is allowed in `GraceGTK3` [58].
Compared to the preceding one, its advantage is that you don't have to assume an algebraic form for the fit.
The method allows to choose between two distributions for the noise in the data: `Gaussian` or `Symmetric` (i.e. uniform) and the local approximation can be linear or quadratic.
The width of the sliding window is given by its ratio (named `span`) with the length of $[x_{min}, x_{max}]$.
It is possible to compute confidence intervals and draw a XYDYDY set to visualise these intervals. The number of bars to compute is given separately because this computation is not necessary for each point and can be time consuming for large sets.
We recommend to read the original documentation copied from `http://www.netlib.org`, i.e. the file `cloess.pdf` in the `doc` directory.
N.B. Up to now, the restriction by regions apply only to the Levenberg-Marquardt method.

[55] Note that the abscissas must be sorted.
[56] These are the one already computed by `grace-5.1.22` and determined here by reverse engineering (without any guarantee). Probably this can be enhanced but needs somebody more competent than me to do that.
[57] Also called Pearson coefficient.
[58] If you want to perform more sophisticated statistical treatments on your data, you can try to use the R-project[9] [8]

9.3 Adjusting "by hand"

Levenberg method can failed if the starting parameters are not well chosen. This option is proposed to help you to choose them. It is possible to draw the curve with the current values of parameters `A0, A1,...` and also to change their value with spin buttons.
Each time the value of one of the parameters is changed, a new curve is computed.
The set selected as source is used only to give the number of points to be used and the destination selector works as usual:

- if no set is selected, a new set is created each time,
- if a valid set is selected, the new curve replace the old one.

Note that set selectors are not automatically refreshed, thus, you will have to click on the graph selector to refresh the list if you want to display the newly created sets in the list .
This method allows to determine suitable parameters values before using the Levenberg-Marquardt algorithm.
You can try `Examples/GraceGTK/Non-linear fit/By hand`

9.4 Commands

The best way to compose a script is to experiment first interactively with the follow-me mode ON.

9.4.1 Levenberg-Marquardt commands

The `FIT` commands are used to set parameters used by the `NONLFIT` function.

- `FIT FORMULA` "*string*" — Defines the formula for the fit.
- `FIT TITLE` "*string*" — Gives a title to the fit.
- `FIT WITH` n `PARAMETERS` — The number of parameters `A0, A1,...An`
- `FIT PREC` *expr* — The tolerance parameter to stop the iterations.
- `FIT MAX STEPS` *nsteps* — The maximum number of iteration steps.
- `FIT TRACE` *onoff* — Whether to draw curve at each iteration step.
- `FIT LOAD` *type* — *type* is one of `VALUES, RESIDUALS ,FORMULA`.
- `FIT WEIGHT FORMULA` "*string*" — Defines the custom weighting formula.
- An = *expr* — The initial value of parameter An.
- An `CONSTRAINTS` *onoff*
- An`MIN =` *expr*
- An`MAX =` *expr*
- `NONLFIT` $(set, nsteps)$
- `NONLFIT` $(set, warray, nsteps)$ — Use weight array *warray*
- `NONLFIT` $(set, dest, Rn, onoff)$ — Restrict to region n, `ON`= negated.
- `NONLFIT` $(set, dest, weight)$ — *weight* 1: $= 1/Y$, 2: $= 1/Y^2$, 3:$= 1/dY^2$, 4: custom.
- `NONLFIT` $(set1, set2, Rn, weight)$ — Combines restriction and weighting..

9.4.2 LOESS command

`LOESS (set ,family ,degree ,span ,baronoff ,level ,nbar ,dest)`
The arguments are:

1. `set` the dataset to be fitted,
2. `family`: `GAUSSIAN` or `SYMMETRIC` for noise hypothesis,
3. `degree` of approximating functions (1: linear, 2: quadratic)
4. `span` ratio: 1.0 mean that all the x interval is used,
5. `baronoff`: `OFF, ON` to compute confidence bars,
6. `level`: the level of confidence ($\in [0.1, 0.999]$),
7. `nbar`: the number of confidence bars to be computed,
8. `dest`: the destination set for the fit. A new set is created if it does not already exists.
   Note that a new XYDYDY set is always created if confidence bars are computed.

Example file: `doc/ng_nonlloess.agr`

9.5 Library of functions available for Levenberg method

Nicola Ferralis have proposed a set of formulas to be used with the Levenberg-Marquardt method.

9.5.1 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

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

9.5.2 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

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

9.5.3 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.

9.5.4 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.

9.5.5 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

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

10 Fourier transformation

10.1 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.

10.2 Classical Fourier transformation

Let us define the Fourier integral by

$$F(X) \;=\; a \int_{-\infty}^{+\infty} f(x)\, \exp(\pm icXx)\, dx \quad \Leftrightarrow \quad F \;=\; \mathcal{F}^{\pm}_{a,c}\, f, \tag{6}$$

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.

10.3 Discrete Fourier Transform

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

$$\tilde{F}_k \;=\; \sum_{j=0}^{N-1} f_j\, \exp\left(\pm i 2\pi\, \frac{k}{N}\, j\right). \tag{7}$$

Let us identify (7) with a discrete computation of (6) with the method of rectangles for a function $\tilde{f}$ null outside $[0, N-1]$, with $f_j = \tilde{f}(j)$ and $\tilde{X}_k = k/N$:

$$\tilde{F}(\tilde{X}) \;=\; \int_{-\infty}^{+\infty} \tilde{f}(\tilde{x}) \exp\left(\pm i 2\pi\, \tilde{X}\tilde{x}\right)\, d\tilde{x}. \quad \Leftrightarrow \quad \tilde{F} = \mathcal{F}^{\pm}_{1,2\pi}\, \tilde{f} \tag{8}$$

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 (6), we get

$$F(X) \;=\; a \int_{-\infty}^{+\infty} f(A + b\tilde{x})\, \exp[\pm icX(A + b\,\tilde{x})]\, b\, d\tilde{x} \tag{9}$$

$$\;=\; ab\, \exp(\pm icAX) \int_{-\infty}^{+\infty} \tilde{f}(\tilde{x})\, \exp(\pm ibcX\,\tilde{x})\, d\tilde{x} \tag{10}$$

We can identify (8) with the integral in (10) if $bcX \equiv 2\pi\, \tilde{X}$.
Thus,

$$X_k = \frac{2\pi}{bc}\, \frac{k}{N} \quad \Rightarrow \quad F(X_k) \;=\; ab\, \exp(\pm icAX_k)\, \tilde{F}_k \tag{11}$$

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

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

10.4 Fourier transform in GraceGTK

The abscissas must be monospaced.

10.4.1 Choice of parameters

GraceGTK interface gives the following choices.

General

- *Compute* allows to choose the quantity to be computed: the *magnitude* or the *phase* (in radians), the *real* or *imaginary* part, the *complex* values with a cartesian or polar representation.
- *Forward* transform: the sign in the exponential is minus.

- *Backward* transform: the sign in the exponential is plus.
- *Normalize* is useful when you choose to compute the DFT (see below): it allows to divide the values of the $\tilde{F}$ by the length of the buffer (*forward, backward*) or by its square root (*symmetric*).

Input

- *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
- *Imaginary part*: 3 cases:
  - *None*: the function to be transformed is real valued,
  - *3d column of the set*: the second column of the set selected as source contains the real part of the function and the 3d column (i.e. `Y1`) the imaginary one,
  - *Another set*: the imaginary part of the function is in a set in the same graph as the real part; the spin button allows to choose its id number.
- *Dump DC component* allows to subtract the mean value of the input function before taking the transform,
- *Apply window* allows to multiply the input dataset y_j by various functions with values W_j, i.e. to compute

$$y_j \;=\; W_j \; y_j \qquad \forall j \in [0, N-1] \tag{12}$$

before taking the Fourier transform. Thus, taking a normal forward Fourier transform and then, a backward one with this windowing, it is possible to perform a digital filtering on the dataset. Details on available functions are given below.

Output

- X-scale: allows to choose a and c in formula (6):
  - *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"*[59] and *"Centered"*.

Details on input windows. It is a feature inherited from `grace-5.1.22` .

Let us note $\omega_i = 2j\,\pi/(N-1)$. The set of available functions is the following:

- *Triangular*:

$$W_j \;=\; 1 - \left| \frac{j - \frac{1}{2}(N-1)}{\frac{1}{2}(N-1)} \right|$$

- *Parzen*:

$$W_j \;=\; 1 - \left| \frac{j - \frac{1}{2}(N-1)}{\frac{1}{2}(N+1)} \right|$$

- *Welch*:

$$W_j \;=\; \left(\frac{j - \frac{1}{2}(N-1)}{\frac{1}{2}(N+1)} \right)^2$$

- *Hanning*

$$W_j \;=\; \frac{1}{2}(1 \;-\; \cos\omega_j)$$

- *Hamming*

$$W_j \;=\; 0.54 \;-\; 0.46\cos\omega_j$$

- *Blackman*

$$W_j \;=\; 0.42 \;-\; 0.5\cos(\omega_j) \;+\; 0.08\cos(2\omega_j)$$

- *Flattop*

$$W_j \;=\; 0.2810639 \;-\; 0.5208972\cos(\omega_j) \;+\; 0.1980399\cos(2\omega_j)$$

- *Kayser* is present only if GSL library is used. See GSL documentation for details.

Note this citation from [10], chapter 16, p. 286

Several different windows are available, most of them named after their original developers in the 1950s. Only two are worth using, the ***Hamming window*** *and the* ***Blackman window****.*

[59] Of course, this option is meaningful only if the input is real.

10.4.2 Call in GraceGTK scripts

Four forms are possibles, the leading two are not compatible with Grace-5.

1. FFT ($set_1, set2, outtyp, norm, inseg, intyp, dcdump, filter, padding, round2n, xscale, outseg$)
2. FFT ($set_1, intyp, filter, xscale, outtyp, outseg$)
3. FFT ($set_1, intyp, filter, xscale, outtyp$)
4. FFT ($set_1, filter$)

The name of the function can be also INVFFT to compute the backward transform.
The use of DFT is deprecated but if FFTW is not available, the computation is done by a DFT function, i.e. very slowly for a large dataset.
The first form is used by the *Follow-me* mode and the arguments are sorted as in the GUI window.
The arguments are:

- set_1: the ident of the set to be transformed, e.g. G1.S0
- set_2: the ident of the resulting set. If the set does not exist, it is created, but its graph should already exist.
- *outtyp*: MAGNITUDE, PHASE, COEFFICIENTS or COMPLEX
- *norm*: NONE, SYMMETRIC ,FORWARD ,BACKWARD
- *inseg*: INDEX, INSEGX
- *intyp*: allows to specify the imaginary part of the input function:
  - REAL: the input is real,
  - COMPLEX: the 3d column of the source set is the imaginary part,
  - Gn.Sm or Sm: a set belonging to the same graph as the source set containing the imaginary part (Gn is not used).
- *dcdump*: OFF, ON
- *filter*: NONE, TRIANGULAR, WELCH, HANNING, HAMMING or PARZEN
- *xscale*: INDEX, FREQUENCY or PERIOD. The last one is for *angular frequency*.
- *outseg*: POSITIVE, HALF or CENTER

10.4.3 Examples

Examples are given in doc/examples directory:

ng_fourier.agr, ng_fourier2.agr and ng_fourier3.agr

and can be launched via Examples GraceGTK menu.
A good method to get the full list of parameters for the FFT function is to use the Follow-me mode.

10.5 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.

11 Digital filters

11.1 Introduction

Applying a filter to a dataset is nothing else than computing an approximation of the data thus spline interpolation or non linear fit can be used to perform filtering, but this section describe only the use of some classical filters described in Digital Signal Processing (DSP) books. One characteristic of these filters is that it is possible to implement them efficiently into an hardware processing.
Only linear filtering of 1D real data is considered here.

We strongly recommend to beginners further reading [60]. Details on some formlae used are given in section 17.6. Note that the goal is to filter a given dataset, not to design a filter conforming to a given pattern[61]

We recall first some definitions useful to understand `GraceGTK3` interface. The *passband* refers to those frequencies that are passed, while the *stopband* contains those frequencies that are blocked. The *transition band* is between. A fast *roll-off* means that the transition band is very narrow. The division between the passband and transition band is called the *cutoff frequency*. The stopband *ripple* gives a measure of the alteration of frequencies in the stopband.

11.1.1 Filtering procedures

Filtering a dataset resume in a convolution between the data and the *impulse response* of the filter. This convolution can be performed in several ways:

- direct convolution of data samples y_n by filter weights w_m [62]:

$$\tilde{y}_n \;=\; \frac{1}{M}\sum_{j=0}^{M-1} w_j \; y_{n-j}, \tag{13}$$

- recursive convolution, i.e., use of values of $\tilde{y}_m$ already known for $m < n$ to compute $\tilde{y}_n$,
- use of the property that a convolution product is transformed in an ordinary one by a Fourier transformation. Thus, performing a FFT on the dataset, multiplying the result by the *frequency response* of the filter, and returning to time representation by a backward FFT gives the wanted result. When the Laplace transform is used in place of the Fourier one, the function to use is called the *transfer function*.

11.1.2 Filters gender

The intended use of filter allows to distinguish filters of various *gender*:
low-pass, high-pass, pass-band, stop-band and *custom*.
It is easy to transform a low-pass filter into a high-pass, a pass-band or a stop-band one by means of a change of variable in the transfer function, thus in the following, except if stated explicitly, a low-pass filter is assumed.

11.1.3 Cutoff scaling

Without scaling, the cutoff values are given in Hertz if the time scale is in seconds, in kHz if the scale is in milliseconds and so on... There is a connection between the sampling rate of the data and the ability to filter a signal, thus it is also customary to give the cutoff(s) frequencies scaled in the range $[0, 0.5]$: the two methods (scaled or not) are available in `GraceGTK3` .

11.2 Filters in `GraceGTK3`

We just give here material to use a filter; more details about the formulae used to compute some transfer functions are given in section 17.6.

11.2.1 Moving Average filters

Defined by taking $w_j = 1, \forall j$ in eq. 13, this very simple filter is optimal to reduce white random noise, but very poor to separate a band of frequency from another.
It is applied via the `Data/Transformations/Running properties...` menu.

[60] A comprehensive course is [11]. It can be accessed on the WEB at [10]
[61] You can use software like Scilab, Scipy or others for that (e.g. using the Remez algorithm).
[62] Note that side effects (e.g. $n - j < 0$) needs a discussion.

11.2.2 Windowed-Sinc filters

Defined by a rectangular function in the frequency domain, giving a sinc (i.e. $\sin(x)/x$) function as impulse response this filter is also called a *brickwall* filter. In fact sampling and finite size produces unwanted side effects and various more sophisticated windows are used: *Blackman, Hamming, ...*: see **"Details on input windows"** in section 10.4.1.
Up to now, only the simple brickwall can be accessed by the `Data/Transformation/Filters/via FFT` menu; you have to use `Data/Transformation/Fourier transforms` menu if you want to use more sophisticated windows.

11.2.3 Butterworth and Chebyshev filters

These filters are named from their use of the Butterworth or the Chebyshev polynomials. The *order* of the filter is the degree of the polynomials, i.e. the number of poles of the transfer function.

- *Butterworth* filters does not exhibit ripple at the price of a wider transition band.
- The *Chebyshev type 1* filters have a non null ripple only in the passband.
- The *Chebyshev type 2* filters have a non null ripple only in the stopband.
- *Elliptic* (or *Cauer*) filters, so called because their poles are located on an ellipse, have ripples in the two bands (*not yet implemented*).

These filters use the Fourier method and are applied by the

`Data/Transformations/Filers via Fourier`

menu. The transfer functions are normalized to be unity at frequency zero or infinity.
The cutoff frequencies can be specified in Hertz (if the unit of the signal x-axis is in second) or in normalized units int the range $[0, 0.5]$. The meaning of *cutoff frequency* is not uniform:

- For Butterworth low and high-pass filters, the cutoff frequency is for an attenuation of 3 dB,
- For Butterworth pass-band and stop-band filters, the cutoff frequencies ω_1, ω_2 are used in the change of variables that transform a low-pass filter (see section 17.6.1) and are no longer the 3 dB ones.
- For Chebyshev filters, the cutoff frequency is the edge of the ripple box, i.e. the frequency when the magnitude of the transfer function crosses the line $y = 1$.

More details are given in the Developper's section (subsections 17.6.1 and 17.6.2)

11.3 Call in `GraceGTK` scripts

11.3.1 Commands to apply filters

Moving average filtering command, `RUNAVG` is described in table 9.
The Fourier commands described in subsection 10.4.2 are to be used to apply (in two passes) windowed-sinc filters.
For brickwall, Butterworth and Chebyshev filters:

1. `FILTER BRICKWALL` ($set_1, set_2, gender, cut_1, cut_2, scaling$)
2. `FILTER BUTTERWORTH` ($set_1, set_2, gender, order, cut_1, cut_2, scaling$)
3. `FILTER CHEBYSHEV` ($type, set_1, set_2, gender, order, cut_1, cut_2, scaling, ripple$)

are used to apply a filter.

- *type*: only for Chebyshev filters: must be 1 or 2.
- set_1: the ident of the set to be transformed, e.g. `G1.S0`
- set_2: the ident of the resulting set. If the set does not exist, it is created, but its graph should already exist.
- *gender*: a string short name[63]:
  - `"lp"`: low-pass
  - `"hp"`: high-pass
  - `"bp"`: band-pass
  - `"sb"`: stop-band
- cut_1, cut_2: cutoff frequencies: for low-pass and high-pass filters only cut_1 is used.

[63] Short names are the same as in Scilab.

- *scaling*: `ON` or `OFF`: see subsection 11.1.3.
- *ripple*: must be in the range $[0.01, 0.99]$.

Example: `@ FILTER CHEBYSHEV (1 ,G0.s0 ,G0.s1 ,"lp" ,5 ,0.2 ,0.4 ,ON ,0.15)`
The follow-me mode can help you to write scripts.

11.3.2 Commands to load the frequency response

To load in a set the magnitude or the phase of the transfer function:

1. `FILTER LOAD BRICKWALL` $(set, gender, cut_1, cut_2, scaling, load\_mode)$
2. `FILTER LOAD BUTTERWORTH` $(set, gender, order, cut_1, cut_2, scaling, load\_mode)$
3. `FILTER LOAD CHEBYSHEV` $(type, set_1, gender, order, cut_1, cut_2, scaling, load\_mode, ripple)$

Arguments:

- *set* is the set to load into; if it does not exist, it is created (the graph must already exist)
- *load_mode*: 0=Magnitude (dB), 1=Magnitude, 2=Phase (deg), 3=Phase (radians)

Others arguments have the same meaning as above.

12 Wavelet transformations

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.

12.1 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* [6] gives a comprehensive account on the subject.

12.2 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 $k = 100\,i + j$.

12.3 Wavelets parameters

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

12.4 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 `GraceGTK3` , 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 want 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.

12.5 Call in `GraceGTK` scripts

12.5.1 Creating a new set

The following function creates a new set in the same graph 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)`

12.5.2 The set for output is specified

`WAVELET (`$set, type, stride, k, invflag, outset$`)`
Example : `@ WAVELET (G0.s0 ,DAUBECHIES ,1 ,4 ,0 ,0 ,G1.s0)`

13 Advanced topics

13.1 Fonts

13.1.1 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: ../fonts/FontDataBase The fonts beyond ZapfDingbats had been added since release 0.8.0 to meet `xfig` import requirements[64].
Note that the two last are not yet used in Default.agr.

13.1.2 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`

13.1.3 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
    ...
```

[64] Files downloaded from LaTeXdistributions. Fonts `cmr10, cmi10, cmb10` and `cmss10` are the modernised version of Computer Roman LaTeX fonts from the `cm-super` package and adapted with the `fontforge` program to our needs.

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 :-).

13.1.4 `GraceGTK3` 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[65]. 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.

13.2 Interaction with other applications

13.2.1 Using pipes

Not finished ...

13.2.2 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 23 for a list of C functions.
See table 24 for a list of Fortran functions.

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

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

`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 23: grace_np library C functions.

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 24: Fortran functions provided by the grace_np library.

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
```

13.3 DL modules

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

13.3.1 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 25: Grace types for external functions

It is expected that the result is a `double`. In your script, if the variable is a scalar, the result will be a scalar and if the variable is a *vexpr*, the interpreter will perform a loop to call the function to each elements of the array, resulting in a new *vexpr*.
Up to now, calling a function with pointer to vector(s) or matrix(es) as argument(s) is not implemented.

13.3.2 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 25. 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[66] 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"
```

14 Specifications

14.1 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 26. 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.

[66] 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 26: Typesetting control codes.

14.2 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).

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 3.4.15 (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 2.3.2 (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 3.4.15 (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.

15 Installation guide

Note that `GraceGTK3` is in a alpha state and it is a better idea to compile yourself the source.

15.1 Binary installation

15.1.1 MS Windows

Windows© binaries compiled under Windows 10 / MinGW are available at `GraceGTK3` home site.

- You need MinGW/Msys to be installed and and execute `ggrace.exe` in the MinGW console.
- If you want to access GraceGTK from elsewhere, you have to configure yourself the environment variables (see sec 2.3.1) and the execution path.

Note that some features are not yet available under MS Windows:

- Printing: you may write into a file (e.g. in PDF) and use a reader (e.g. Acrobat Reader) as previewer/printer interface.
- Piping is not available.

15.1.2 Unix/Linux

No binaries for these OS are distributed by the `GraceGTK3` home site: it is intended that you compile from sources.

15.1.3 Mac OS X

To be done ...

15.2 Installing from sources

15.2.1 For Linux inpatients

- Make sure you have development packages `GTK+3 and BLAS` and also `gcc, gfortran, FFTW` installed,
- ```
  ./configure
  make
  make install
  export GRACEGTK_HOME=/usr/local/gracegtk3
  export PATH=$PATH:/usr/local/gracegtk3/bin
  ```
- and run `ggrace`...
  ```

15.2.2 More detailed instructions

Requirements.

- You need an ANSI C compiler (`gcc` is just fine), and preferably[67] a Fortran one (e.g. gfortran),
- You need to have installed a recent version of pkg-config available from `http://www.freedesktop.org/software/pkgconfig/`.
- a development version of GTK-3 (e.g. `apt install libgtk-3-dev`) [68].
- If the Fortran compiler have a standard name (e.g. gfortran, ifort...) it is detected by `configure` directly.
  If not, you must set `FC` and `F77` environment variables
- BLAS library is used by LOESS non linear fit.
  For Debian and Ubuntu Linux distributions, the package `libblas-dev` is required.
- If you want to compile your own changes to some parts of `GraceGTK3` , you will need `bison` (the parser generator) and the GNU `autotools`.
- If you want to use CUPS support for printing management, you need a development version of CUPS (providing cups-config)
- If you want to compile other versions (HTML...) of the documentation, you need `linuxdoc` (e.g. `apt install linuxdoc-tools`

Output drivers and extra libraries.

Output drivers are the Cairo drivers
Some features will be available only if additional libraries are installed:

- 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.
  `GraceGTK3` 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 10.5.
- 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 import/export requires version 3 or later.
  The library home is `http://www.unidata.ucar.edu/software/netcdf`.

Compilation.

`cd gracegtk-`*x.y.z*
`./configure`
should produce `Make.conf` and `config.h` to have working `Makefile`(s).
`make`
should compile the various elements[69,70,71].

If something goes wrong.

- First, try to see if the problem has been described already in the `FAQ` (in the `doc` directory). Questions specific to `GraceGTK3` are at the end of the file.
- If the `Makefile` does not allows to compile has expected, run `make distclean`, then rerun the `configure` script.
- Execute `./configure --help` to check if some options may solve the problem. Note that `-with-`*xxx*`-library=OBJ` only gives the path for the object files, not the include ones. If a include file is not found[72], the simpler is to copy it into the `src` directory or in a standard location (e.g. `/usr/include`).

[67] Fortran is required to draw level contours curves and to preform LOESS non linear fits.
[68] `apt install` examples are given for a Debian-11 distribution
[69] The `doc` directory is not in the compilation path because the documents are already compiled (`GraceGTK.pdf,...`): it is only if you want to change something or obtain a different format that you may have to recompile it.
[70] In a MS Window environment, the `grace_np` directory, is not compiled.
[71] `GraceGTK3` is deliberately developed using an old environment in order to make it available to users who don't have a up-to-date one. It appears that default settings for recent versions of the `gcc` compiler emits a lot of new warnings in order to enforce good programming practises. For third parties bundled libraries (cephes, T1lib) stability is privileged and these warnings will remain. For the `src` directory, I try to eliminate them, but users should not be afraid to see such warnings.
[72] The reason for the failure of the script may be found in the `config.log` file.

- If you don't get a working `Makefile`, try to copy (or soft link) `ac-tools/configure.in` in the main directory and to run `autoconf` to regenerate the configure file.
- Note that the only work of the `configure` script is to produce two files: `Make.conf` and `config.h`: it is possible to edit directly the files or the various Makefiles before running `make` to solve a problem.
- Note that `configure.in` is a hand-written file, thus `automake` should not been run here.

Configuration without installation.

In order to get a working executable of `GraceGTK3` *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 2.3.1.

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.

Under Linux:

```
make install
```

install by default in `/usr/local/gracegtk3` (of course you need to have the right permissions). Then you must include in your profile file

```
export GRACEGTK_HOME=/usr/local/gracegtk3
export PATH=$PATH:/usr/local/gracegtk3/bin
```

Upon start-up, `GraceGTK3` loads its init file `gracegtkrc` See section 2.3.2 for details.
You can also customize some environment variables, mainly for GUI size and help viewer configuration: see section 2.3.1.

Under MS Windows:

no installation procedure to integrate the program into Windows `Programs` directory is proposed: you may run the executable `ggrace.exe` directly from the `src` directory or copy it in another directory and set the `GRACEGTK_HOME` and `PATH` accordingly.

Removing installation or installing a new version.

Just remove the installation directory (Unix/Linux default: `/usr/local/gracegtk3`).
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.

Mac OS

See *The Fink Project* (`http://pdb.finkproject.org`) at page `http://pdb.finkproject.org/pdb/package.php/gracegtk`.

16 Copyright statements

`GraceGTK3` is distributed under the GNU GPL licence, in continuation of `grace-5.1.22` .

This later had been 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[73] 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.

Credits

```
For Grace:
Copyright ((C)) 1991-1995 Paul J Turner, Portland, OR
Copyright ((C)) 1996-2007 Grace Development Team
Maintained by Evgeny Stambulchik
```

[73] 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.

```
 For Gracegtk3:
Copyright ((C)) 2009-2022 Patrick Vincent and adds from the Grace Development Team
Copyright ((C)) 2018-2022 Patrick Vincent and Gerald Dumas.

GNU GPL:
                        All Rights Reserved

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

For some libraries required to build `GraceGTK3` (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` .

17 Developper's section

Some hints to help to enter into the program.
See also file `GTK_DEV` in `GraceGTK3` home directory.

17.1 Program structure

From a logical point of view `GraceGTK3` may be divided mainly into

- a kernel performing computations,
- a GUI (files prefixed with `gg_` and files in `../glade` directory,,
- a command interpreter (file `pars.yacc`)

17.1.1 The kernel

- The effort to rationalize the name space is not yet achieved.
- 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` .

17.1.2 The display drivers

The drivers scheme is inherited from `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` : allows a direct access to colors without the cmap indirection[74] .
3. `devdrawpixel`: draw one pixel,
4. `devdrawpolyline`: draw a polygon (or polyline),
5. `devfillpolygon`

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

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 `drawg_project()` .
The drivers are registered in `main` and initialised by a call to `initgraphics`.
The list of drivers is build by the registering functions listed in `devlist.h` .
TO DO: use Freetype2 fonts. T1Lib is always used due to problem with interpreter escape sequences to draw mathematical formulas.

17.1.3 The canvas display driver

The driver in file `driver_terminal.c` provides additional functions to

- take into account canva resizing,
- manage screen refreshment when an event change a part of the displayed window.
- `driver_handles.c` display rubber lines or rectangles used when the user drag objects with the mouse.

17.1.4 The command interpreter

The grammar is defined in `pars.yacc` and the interpreter defined by a call to Bison.
A first parsing done in `nn_uniread()` separates data from commands, then the command 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()`

17.2 Filenames

17.2.1 Naming rules

Following prefixes are use:

- `ge_` `GraceGTK3` explorer, also called TreeView,
- `gg_` GUI components,
- `gw_` widgets like component that may be used in other context,
- `nn_` some kernel files added to `grace-5.1.22` ,
- files with none of them are either inherited from `grace-5.1.22` or new `GraceGTK3` files.

17.2.2 Understanding their use

Introducing a new object induces changes in many files. The following list may be used as a reminder if you want to hack the program. We take the introduction of polylines as an example.

- `defines.h`
  - Add `Q_Polyline` item in `Qtype` enumeration. Later, a *grep* on this constant will localize lines involved to manage this type of object.
  - Add `x` and `y` pointers in `QDobject` structure and add variables for spline interpolation.
- `ng_objects.c` Methods to create, destroy, find, duplicate, *etc*
- `ng_objs_draw.c` Draw the lines.
- `ge.c, ge_obj.c` and `glade/ Polyline.glade`.
  The explorer GUI window (create, update and apply).
- `driver_events.c` and `driver_handles.c` Mouse interaction.
- `pars.yacc` Commands grammar,
- `nn_cmd.c` and `nn_cmd_obj.c` undo/redo actions and new saving syntax.
- `params.c` Old saving syntax to save object parameters.

- `template.c, nn_cmd.c` and `nn_cmd_obj.c`
  Create template and manage differential saving with respect to template settings.
- `doc/GraceGTK.tex` Documentation of the new object.
- In other files, action specific to polylines may be defined and found using a `grep` on `Q_Polyline`.
- `README` News.

Defining data transformation involves less files but more specialized. Let us take Fourier transform as an example. Note that here, Glade tools is not used.

- `fourier.h` Define constant, structures and function prototypes.
- `fourier.c` Mainly an interface to FFTW library and command script.
- `gg_fourier_win.c` GUI window.
- `gg.c`, `gg.xml` and `gg_protos.h` Define the menubar entries
- `pars.yacc` Define grammar of commands,
- In `examples/` directory `ng_fourier*.agr` define new examples, `Makefile` and `dotest` takes into account them.
- `doc/GraceGTK.tex` Documentation.

17.3 Array `objs` and the drawing tree

Any `QDobject` have a `Qtype typ` and is stored in the global array `objs` (see `globals.h`).
Arrays `g[]` for graphs and `p[]` for sets used in `grace-5.1.22` are replaced by pointers `g` and `p` in `QDobject`.

In the `QDobject` structure, the pointers `self ,elder ,brother ,child` are indexes in the array `objs[]` and are managed through the functions defined in `nn_tree.c`

This heterodox method has some *pros*:

- it had allowed a smooth transition between the pure array management of `grace-5.1.22` and the tree view of `GraceGTK3` ,
- it is more easy to scan all the objects with a simple loop than with a tree traversal,
- It is safe from obsolescence that occur too often in GTK development.

The main *con* is that reentrancy is excluded [75].
Note that the tree is a binary tree, thus a difference is made between `elder` and `father`.
Note that `objs[]` is a dynamical arrays, i.e. may be reallocated when its size is varied. Consequently, addressing its elements through memory pointers is safe only if you are sure that a reallocation will not occur i.e. new object is not created.
The drawing tree displayed in the left Explorer pane is a `GtkTreeView` that copy the kernel tree defined by `objs[]`.
Running `./configure --enable-debug --enable-maintainer`
allows to display toolbar buttons to call various help functions.

17.4 Miscellaneous

17.4.1 Glade utilities

To manage a smooth transition a set of utilities are defined in file `gb.c`.

17.4.2 Colors

The kernel of `GraceGTK3` inherits from Grace the `cmap_table` colormap.
For coloured sets (e.g. `XYCMAP`), colors are interpolated at drawing time by functions defined in file `zc.c`.
Anti-aliasing of character strings is done by adding shades into the `cmap_table` with `COLOR_AUX` type and this was confusing when loading `Xfig` file where new colors may be defined. Thus a distinction between `main_color` identifier and `cindex` in the `cmap_table` had been introduced.

For geometric object, the user can set up to 3 different colors defined in `QDobject`:

- `color` used to draw lines or boundaries for lines, polylines, arcs or boxes. For objects that display a string, this is the color used to paint the glyphs but stored in the `scolor` member of the structure.

[75] e.g. only one canvas at a time is possible

- `fillcolor` is used by objects that can display a pattern (arcs, boxes and legend boxes, strings drawn into a box) to draw the pattern (i.e. a uniform colour if pattern=1, or the motif for others).
- `fillbgcolor` is used as background if `opaque` option is on. It is drawn before the pattern and is obscured for pattern=1.

17.4.3 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.

17.4.4 Management of handles

Anw scheme is used, based on the size of the diagonal of the bounding box.

17.4.5 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 callbacks in the Glade files or `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.

17.5 Compilation

17.5.1 General

The compilation scheme of `GraceGTK3` try to obtain a working but restricted program if some libraries are not found. This is obtained by supplying bundled versions for T1lib and a small part of GSL, or not compiling all the drivers, loess directory or undo/redo feature.

17.5.2 The `configure` script

The `configure` script is build by the `autoconf` GNU program.
The scheme used is not up-to-date compared to Autoconf documentation, but in the continuity of `grace-5.1.22` . You must copy the template file `configure.in` from the `ac-tools` subdirectory into the main directory, then run `autoconf`, This produces two files:

- `Make.conf`, included in the various `Makefiles`,
- `config.h`, included in many C source files.

Each file have a template:

- `Make.conf.in` defines which environment variables will be set by the `configure` script in `Make.conf`
- `config.h.in` defines the macros to be used by `cpp`.

This scheme has the advantages that if the `configure` script fails in defining some variables, you can hack them directly and that the compilation can be done independently in each subdirectory[76] but an adaptation to a modern scheme should be a good thing.

17.6 Transfer functions of filters

For a more complete description, see [3] and [4]

[76] Note that the link will not be redone automatically if it is not the `src` directory.

17.6.1 Butterworth filters

Les us note

- H the transfer function in the Laplace s-plane $(s = i\omega)$,
- G the frequency response and G_0 the DC gain,
- ω_c the angular frequency at cutoff,
- N the filter order.

The frequency response of a Butterworth low-pass filter satisfy the relation:

$$G(\omega) \;=\; |H(i\omega)| \;=\; \frac{G_0}{\sqrt{1+\left(\frac{\omega}{\omega_c}\right)^{2n}}}, \tag{14}$$

and the poles of the transfer function are

$$\tilde{s}_k \;=\; \omega_c\, e^{i(2k+1)\frac{\pi}{2N}+i\frac{\pi}{2}} \qquad k = 0, 1, 2, \ldots, N-1. \tag{15}$$

If $\theta_k = (2k+1)\frac{\pi}{2N}$,

$$\tilde{s}_k \;=\; i\,\omega_c\, e^{i\theta_k}.$$

Let us define $s_k = \tilde{s}_k/\omega_c$. The transfer function may be written in terms of its poles as

$$H(s) = \frac{G_0}{\prod_{k=1}^{N}(s-\tilde{s}_k)/\omega_c} \;=\; \frac{G_0}{\prod_{k=1}^{N}(s/\omega_c - s_k)} \;=\; \frac{G_0}{B_N(i\omega/\omega_c)}, \tag{16}$$

where the B_N are the normalized Butterworth polynomials.
If $b = \omega/\omega_c$ and N is even $(N = 2K)$ grouping the pairs of complex conjugate poles gives:

$$\begin{aligned} B_N(ib) &= \prod_{k=1}^{K}(ib - s_k)(ib - \bar{s}_k) && (17)\\ &= \prod_{k=1}^{K}[|s_k|^2 - b^2 - ib\,(s+\bar{s}_k)] && (18)\\ &= \prod_{k=1}^{K}[|s_k|^2 - b^2 - 2ib\,\Re(s_k)] && (19)\\ &= \prod_{k=1}^{K}(1 - b^2 + 2ib\sin\theta_k) && (20)\\ & && (21) \end{aligned}$$

If N is odd $(N = 2K+1)$, the pole $s_{(N+1)/2} = -1$ is on the real axis and has no conjugate, thus we get:

$$B_n(ib) \;=\; (1+ib)\prod_{k=1}^{K}(1 - b^2 - 2ib\cos\theta_k). \tag{22}$$

The transfer function is obtained by changing the variable of the low-pass one in the following way:

- For the high-pass filter

$$s \mapsto 1/s \qquad \Leftrightarrow \qquad \frac{i\omega}{\omega_1} \;\mapsto\; \frac{\omega_1}{i\omega}$$

- For the band-pass filter

$$i\,\frac{\omega}{\omega_1} \;\mapsto\; i\left(\frac{\omega}{\omega_2} - \frac{\omega_1}{\omega}\right)$$

- The stop-band filter is obtained from the pass-band one by $s \mapsto 1/s$

17.6.2 Chebyshev Filters type I

The frequency response of a type 1 low-pass Chebyshev filter satisfy the relation:

$$G(\omega) = |H(i\omega)| \;=\; \frac{G_0}{\sqrt{1+\varepsilon^2 T_N^2\left(\frac{\omega}{\omega_c}\right)}}$$

where ϵ is the *ripple* factor and T_N the Chebyshev polynomial of type 1 defined by

$$T_N(\cos\theta) \;=\; \cos(N\theta) \quad \forall\theta. \tag{23}$$

Thus, the poles of the transfer function correspond to the values of θ satisfying:

$$1+\varepsilon^2\cos^2(N\theta) \;=\; 0 \;\Rightarrow\; \cos(N\theta) \;=\; \frac{\pm i}{\epsilon}, \tag{24}$$

thus,

$$\theta_k \;=\; \frac{1}{N}\left[\operatorname{acos}\left(\frac{\pm i}{\epsilon}\right) \;+\; k\pi\right],$$

Changing the arc cosinus into a asinh[77] we write:

$$\theta_k \;=\; \theta_k' + i\theta" \;=\; \frac{2k+1}{N}\,\frac{\pi}{2} \;\pm\; \frac{i}{N}\operatorname{asinh}\left(\frac{1}{\epsilon}\right) \tag{25}$$

Thus, compared to the Butterworth case, θ_k has an imaginary part $\theta"$ and the poles s_k are given by:

$$s = i\frac{\omega}{\omega_c} \quad \Rightarrow s_k \;=\; i\,\cos(\theta_k) \tag{26}$$

$$\Rightarrow \quad s_k^{\mp} \;=\; i\,[\cos(\theta_k')\cos(i\,\theta") \;\pm\; \sin(\theta_k')\sin(i\,\theta")] \tag{27}$$

$$\Rightarrow \quad s_k^{\pm} \;=\; i\,\cos(\theta_k')\cosh(\theta") \;\pm\; \sin(\theta_k')\sinh(\theta") \tag{28}$$

If the order N of the filter is even ($N = 2K$), the transfer function is

$$H(s) \;=\; \prod_{k=1}^{K} \frac{1}{|s_k|^2 - b^2 - 2ib\,\Re(s_k)} \tag{29}$$

Normalized to get a DC gain equal to one it gives

$$H(s) \;=\; \prod_{k=1}^{K} \frac{|s_k|^2}{|s_k|^2 - b^2 - 2ib\,\Re(s_k)} \tag{30}$$

If the order N of the filter is odd ($N = 2K+1$), the pole $s_{(N+1)/2} = -\sinh(\theta")$ is on the real axis and the normalized transfer function is

$$H(s) \;=\; \frac{1}{1+i\,b/\sinh(\theta")} \prod_{k=1}^{K} \frac{|s_k|^2}{|s_k|^2 - b^2 - 2ib\,\Re(s_k)} \tag{31}$$

17.6.3 Chebyshev Filters type II

The frequency response of a type 2 low-pass Chebyshev filter satisfy the relation:

$$G(\omega) = |H(i\omega)| \;=\; \frac{G_0}{\sqrt{1+\frac{1}{\varepsilon^2 T_N^2(\omega_c/\omega)}}}.$$

[77] For $x \in \mathbf{R}$ let us show the identity:

$$\begin{aligned}
\operatorname{acos}(i\,x) - \frac{\pi}{2} \;&\stackrel{?}{=}\; -i\operatorname{asinh}(x) \\
\Leftrightarrow \quad \sin\left[\operatorname{acos}(i\,x) - \frac{\pi}{2}\right] \;&\stackrel{?}{=}\; -\sin\left[i\operatorname{asinh}(x)\right] \\
\Leftrightarrow \quad -\cos\left[\operatorname{acos}(i\,x)\right] \;&\stackrel{?}{=}\; -i\,\sinh\left[\operatorname{asinh}(x)\right] \\
\Leftrightarrow \quad -i\,x \;&=\; -i\,x \qquad \text{Q.E.D.}
\end{aligned}$$

The poles u_k of the transfer function will be the inverse of the poles of the type I filter given by eq. (28), thus we have $u_k = 1/s_k$.
Transfer functions of type II Chebyshev filter exhibit also zeroes z_k. If we write

$$|H(i\omega)|^2 = G_0^2\,\epsilon^2\,\frac{T_N^2\,(\omega_c/\omega)}{1+\epsilon^2 T_N^2\,(\omega_c/\omega)}.$$

and:

$$T_N(-1/(is_z) = 0.$$

we get

$$z_k = i/\cos\theta'_k. \tag{32}$$

If the order N of the filter is even ($N = 2K$), the normalized transfer function is

$$H(s) = \prod_{k=1}^{K} \frac{z_k^2 - b^2}{|u_k|^2 - b^2 - 2ib\,\Re(u_k)} \times \frac{|u_k|^2}{z_k^2} \tag{33}$$

If the order N of the filter is odd ($N = 2K+1$), the pole $s_{(N+1)/2} = -\sinh(\theta")$ is on the real axis and the denominator of normalized transfer function (33) is multiplied by the factor

$$1 + i\,b\sinh(\theta")$$

17.7 Digitizing

The computation below is elementary but a little cumbersome, thus it is worth to give it here. Viewport base vectors are noted $\vec{i}, \vec{j}$ and the origin O and the base vectors for the image curve are $\vec{u}, \vec{v}$, with origin O'. We note $\vec{T} = \overrightarrow{OO'}$ the translation vector, P a point of the polyline and write the position of the various points in the two coordinate systems:

$$\overrightarrow{OP} = \vec{T} + X\vec{u} + Y\vec{v} = x\vec{i} + y\vec{j} \tag{34}$$
$$\overrightarrow{OX_1} = \vec{T} + X_1\vec{u} = x_{x_1}\vec{i} + y_{x_1}\vec{j} \tag{35}$$
$$\overrightarrow{OX_2} = \vec{T} + X_2\vec{u} = x_{x_2}\vec{i} + y_{x_2}\vec{j} \tag{36}$$
$$\overrightarrow{OY_1} = \vec{T} + Y_1\vec{v} = x_{y_1}\vec{i} + y_{y_1}\vec{j} \tag{37}$$
$$\overrightarrow{OY_2} = \vec{T} + Y_2\vec{v} = x_{y_2}\vec{i} + y_{y_2}\vec{j} \tag{38}$$

With obvious notation, we get

$$(36) - (35) \Rightarrow \delta X\,\vec{u} = \delta x_x\,\vec{i} + \delta y_x\,\vec{j} \tag{39}$$
$$(38) - (37) \Rightarrow \delta Y\,\vec{v} = \delta x_y\,\vec{i} + \delta y_y\,\vec{j} \tag{40}$$

and get the components of $\vec{u}$ and $\vec{v}$ in viewport coordinates:

$$u_x = \frac{\delta x_x}{\delta X}, \quad u_y = \frac{\delta y_x}{\delta X}, \quad v_x = \frac{\delta x_y}{\delta Y}, \quad v_y = \frac{\delta y_y}{\delta Y}. \tag{41}$$

Thus, the components of the translation vector can be written:

$$T_x = x_{x_1} - X_1\,u_x \tag{42}$$
$$T_y = y_{x_1} - X_1\,u_y \tag{43}$$

and eq. (34) gives the linear system:

$$X\,u_x + Y\,v_x = x - T_x \tag{44}$$
$$X\,u_y + Y\,v_y = y - T_y \tag{45}$$

with X and Y as unknowns. The determinant is

$$D = u_x\,v_y - u_y\,v_x$$

and the solution:

$$X = [(x - T_x)\,v_y - (y - T_y)\,v_x]/D \tag{46}$$
$$Y = [(y - T_y)\,u_x - (x - T_x)\,u_y]/D \tag{47}$$

Index

Index of Commands and Functions

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] Friedhelm Bliemel and David. B. MacKay. Theil's forecast accuracy coefficient: A clarification. *Journal of Marketing Research*, 10(4):444–447, 1973. http://www.jstor.org/pss/3149394.

[3] C. Sidney Burrus. Butterworth filter properties. http://cnx.org/content/m16903/1.2/, 2012.

[4] C. Sidney Burrus. Chebyshev filter properties. http://cnx.org/content/m16906/1.2/, 2012.

[5] NIST/SEMATECH. e-handbook of statistical methods . http://www.itl.nist.gov/div898/handbook/pmd/section1/pmd144.htm, 2012.

[6] W. H. Press, Q. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in FORTRAN*, chapter 13.10 ,14.5. Cambridge University Press, 1992.

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

[8] Loess smoothing. http://research.stowers-institute.org/efg/R/Statistics/loess.htm.

[9] The r project for statistical computing. http://www.r-project.org/.

[10] Steven W. Smith. http://www.dspguide.com/pdfbook.htm.

[11] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 2011.

[12] Wikipedia. Theil index. http://en.wikipedia.org/wiki/Theil_index, 2012.

[13] Wikipedia:Levenberg. Levenberg-Marquardt algorithm . http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm, 2012. [Online].

[14] Wikipedia:LOESS. Local regression. http://en.wikipedia.org/wiki/Lowess, 2012.