Grace User's Guide (for Grace-5.1.11)

by the Grace Team 08.01.2003

This document explains the usage of **Grace**, a WYSIWYG 2D plotting tool for numerical data.

Contents

1 Introduction

1.1 What is Grace?

Grace is a WYSIWYG tool to make two-dimensional plots of numerical data. It runs under various (if not all) flavors of Unix with X11 and M*tif (LessTif or Motif). It also runs under VMS, OS/2, and Windows (95/98/NT/2000/XP). Its capabilities are roughly similar to GUI-based programs like Sigmaplot or Microcal Origin plus script-based tools like Gnuplot or Genplot. Its strength lies in the fact that it combines the convenience of a graphical user interface with the power of a scripting language which enables it to do sophisticated calculations or perform automated tasks.

Grace is derived from Xmgr (a.k.a. ACE/gr), originally written by Paul Turner.

From version number 4.00, the development was taken over by a team of volunteers under the coordination of Evgeny Stambulchik. You can get the newest information about Grace and download the latest version at the *Grace home page* `http://plasma-gate.weizmann.ac.il/Grace/` .

When its copyright was changed to GPL, the name was changed to Grace, which stands for “GRaphing, Advanced Computation and Exploration of data” or “Grace Revamps ACE/gr”. The first version of Grace available is named 5.0.0, while the last public version of Xmgr has the version number 4.1.2.

Paul still maintains and develops a non-public version of Xmgr for internal use.

1.2 Copyright statement

For certain libraries required to build Grace (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.

Package	License
cephes library	Free
T1lib	LGPL
Xbae	BSD-like
Tab Widget	BSD-like

Table 1: Licenses

2 Installation guide

2.1 Installing from sources

1. Configuration

   - Requirements. Grace usually compiles out of the box in a regular Unix-like environment. You need an ANSI C compiler (gcc is just fine), the X11R5 or above libraries and headers, and an implementaion of the M*tif API, version 1.2 or above. If you want to compile your own changes to certain parts of Grace, you will need a parser generator (`yacc` or, better, `bison`).
   - Extra libraries. Some features will be available only if additional libraries are installed. Those are:
     - The JPEG backend needs the IJG's (*JPEG library* `ftp://ftp.uu.net/graphics/jpeg/`), version 6.x.
     - The PNG backend needs the (*libpng* `http://www.libpng.org/pub/png/libpng.html`) library (version 0.96 or above).
     - The PDF driver requires the PDFlib library of Thomas Merz to be installed, which is available *here* `http://www.pdflib.com/` , version 4.0.3 or above.
     - If your computer has the FFTW library installed when Grace is compiled, Grace will link itself to this, and drop all conventional FFT's and DFT's. All transforms will be routed through this package. Note that there is then no difference between pushing the "FFT" button and the "DFT" button, except that FFT will complain if the length isn't a power of 2, and DFT will not.
       For more information on this package, see the
       FFTW Home page `http://www.fftw.org` . 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.
     - In order to read/write sets in the NetCDF data format, you will also need the *NetCDF libraries* `http://unidata.ucar.edu/packages/netcdf/index.html` .
   - Decide whether you want to compile in a separate place (thus leaving the source tree pristine). You most probably would want it if compiling Grace for more than one OS and keeping the sources in

a central shared (e.g. via NFS) location. If you don't need it, skip the rest of this paragraph and go right to the next step. Otherwise, assuming the sources are in `/usr/local/src/grace-x.y.z` and the compilation will be performed in `/tmp/grace-obj`, do the following:

```
% mkdir /tmp/grace-obj
% cd /tmp/grace-obj
% /usr/local/src/grace-x.y.z/ac-tools/shtool mkshadow \
  /usr/local/src/grace-x.y.z .
```

- The `configure` shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create `Make.conf` in the top directory of the package. It also create `config.h` file containing system-dependent definitions. Finally, it creates a shell script `config.status` that you can run in the future to recreate the current configuration, a file `config.cache` that saves the results of its tests to speed up reconfiguring, and a file `config.log` containing compiler output (useful mainly for debugging `configure`). If at some point `config.cache` contains results you don't want to keep, you may remove or edit it.
- Run `./configure -help` to get list of additional switches specific to Grace
- Run `./configure` <`options`>. Just an example:

```
% ./configure --enable-grace-home=/opt/grace
  --with-extra-incpath=/usr/local/include:/opt/include \
  --with-extra-ldpath=/usr/local/lib:/opt/lib --prefix=/usr
```

would use `/usr/local/include` and `/opt/include` in addition to the default include path and `/usr/local/lib` and `/opt/lib` in addition to the default ld path. As well, all stuff would be put under the /opt/grace directory and soft links made to `/usr/bin`, `/usr/lib` and `/usr/include`. **Note**: If you change one of the `-with-extra-incpath` or `-with-extra-ldpath` options from one run of configure to another, remember to delete the `config.cache` file!!!

2. Compilation

   - Issue `make` If something goes wrong, try to see if the problem has been described already in the **Grace FAQ** (in the `doc` directory).

3. Testing

   - `make tests` This will give you a slide show demonstrating some nice features of Grace.

4. Installation

   - `make install`
   - `make links` The later (optional) step will make soft links from some files under the Grace home directory to the system-wide default locations (can be changed by the `-prefix` option during the configuration, see above).

2.2 Binary installation

1. Getting pre-built packages

2. Installation
3. Running tests

2.3 Alternative packaging schemes (RPM, ...)

Not written yet...

3 Getting started

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

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

1. Start the GUI version: xmgrace (return).
2. Select/check the output medium and canvas size in File/Device Setup.
3. If needed, set the graph size ('Viewport' in Plot/Graph Appearance).
4. Load your data with Data/Import/ASCII. 'Load as': 'Single set' for two-column ASCII data, 'Block data' for multi-column ASCII data.
5. Adjust the scales, axis labels and tick marks in Plot/Axis properties. Acknowledge all changes with 'Apply'.
6. Adjust lines, symbols, legends in Plot/Set appearance.
7. Adjust titles, plot frame and legend display in Plot/Graph Appearance.
8. Data can be manipulated in Data/Transformations. To shift a data set by 20 to the left, e.g., in 'Evaluate Expression' select the same set on the left and the right, and say Formula: y=y-20. As you'll probably notice, Grace can do MUCH more than that. Explore at your leisure.
9. When you like your plot, select File/Print. That's it!

3.1 General concepts

3.1.1 Project files

A project file contains all information necessary to restore a plot created by Grace, as well as some of preferences. Each plot is represented on a single page, but may have an unlimited number of graphs.You create a project file of your current graph with File/Save,Save as.

3.1.2 Parameter files

A parameter file contains the detailed settings of your project. It can be used to transfer these settings to a different plot/project. You generate a parameter file with File/Save menu entry selected from the "Plot/Graph appearance popup". You can load the settings contained in a parameter file with File/Open.

3.1.3 Input File formats

Grace understands several input files formats. The most basic one is ASCII text files containing space and comma separated columns of data. The data fields can be either numeric (Fortran 'd' and 'D' exponent markers are supported) or alphanumeric (with or without quotes). Several calendar date formats are recognized automatically and you can specify your own reference for numeric dates formats. Grace also has a command language (see 5 (command interpreter)), you can include commands in data files using lines having "@" as their first non-blank character. Depending on configuration, Grace can also read NetCDF files (see 1 (configuration)).

3.1.4 Graphs

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

The graph type can be any of:

- XY Graph
- XY Chart
- Polar Graph
- Fixed Graph
- Pie 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).

3.1.5 Datasets

A dataset is a collection of points with x and y coordinates, up to four optional data values (which, depending on the set type, can be displayed as error bars or like) and one optional character string.

3.1.6 Sets

A set is a way of representing datasets. It consists of a pointer to a dataset plus a collection of parameters describing the visual appearance of the data (like color, line dash pattern etc).

The set type can be any of the following:

Not all set types, however, can be plotted on any graph type. The following table summarizes it:

3.1.7 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.

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	Hi/Low/Open/Close plot
XYZ	3	Same as XY; makes no sense unless the annotated value is Z
XYR	3	X, Y, Radius. Only allowed in Fixed graphs
XYSIZE	3	Same as XY, but symbol size is variable
XYCOLOR	3	X, Y, color index (of the symbol fill)
XYCOLPAT	4	X, Y, color index, pattern index (currently used for Pie charts only)
XYVMAP	4	Vector map
XYBOXPLOT	6	Box plot (X, median, upper/lower limit, upper/lower whisker)

Table 2: Set types

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

Table 3: Graph/Set type connection

3.1.8 Real Time Input

Real Time Input refers to the ability Grace has to be fed in real time by an external program. 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.

3.1.9 Hotlinks

Hotlinks are sources containing varying data. Grace can be instructed a file or a pipe is a hotlink in which case it will provide specific commands to refresh the data on a mouse click (a later version will probably allow automatic refresh).

3.1.10 Devices

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

- X11
- PostScript (level 1 and level 2)
- EPS (encapsulated PostScript)
- Metafile (which is Grace format, used at the moment mostly for debugging purposes)

- MIF (Maker Interchange Format used by FrameMaker)
- SVG (Scalable Vector Graphics, a language for describing two-dimensional vector and mixed vector/raster graphics in XML)
- PDF (depends on extra libraries, see 1 (configuration))
- PNM (portable anymap file format)
- JPEG (depends on extra libraries, see 1 (configuration))
- PNG (depends on extra libraries, see 1 (configuration))

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

3.1.11 Magic path

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

3.1.12 Dynamic modules

Grace can access external functions present in either system or third-party shared libraries or modules specially compiled for use with it. The term dynamic refers to the possibility Grace has to open the library at run time to find the code of the external function, there is no need to recompile Grace itself (the functions already compiled in Grace are "statically linked").

3.1.13 Coordinate frames

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. The viewport coordinates correspond to the image of the plot drawn on the canvas (or printed on, say, PS output page). The transformation converting the world coordinates into the viewport ones is determined by both the graph type and the axis scaling.

Actually, there is yet another level in the hierarchy of coordinates - the **device coordinates**. However, you (as a user of Grace) should not worry about the latter. The mapping between the viewport coordinates and the device coordinates is always set in such a way that the origin of the viewport corresponds to the left bottom corner of the device page, the smallest of the device dimensions corresponds to one unit in the viewport coordinates. Oh, and the most important thing about the viewport → device transformation is that it is homotetic, i.e. a square is guaranteed to remain a square, not a rectangle, a circle remains a circle (not an ellipse) etc.

3.2 Invocation

3.2.1 Operational mode

With respect to the user interface, there are three modes of operation that Grace can be invoked in. The full-featured GUI-based version is called `xmgrace`. A batch-printing version is called `gracebat`. A command-line interface mode is called `grace`. Usually, a single executable is called in all cases, with two of the three files being (symbolic) links to a "real" one.

3.2.2 Command line options

-autoscale ***x|y|xy***

Override any parameter file settings

-barebones

Turn off all toolbars

-batch ***batch_file***

Execute batch_file on start up

-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

-fixed ***width height***

Set canvas size fixed to width*height

-free

Use free page layout

-graph ***graph_number***

Set the current graph number

-graphtype ***graph_type***

Set the type of the current graph

-hardcopy

No interactive session, just print and quit

-hdevice *hardcopy_device_name*

Set default hardcopy device

-install

Install private colormap

-legend *load*

Turn the graph legend on

-log *x|y|xy*

Set the axis scaling of the current graph to logarithmic

-mono

Run Grace in monochrome mode (affects the display only)

-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

-npipe *file*

Read data from named pipe on startup

-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

-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 allowd through the batch language

-saveall ***save_ file***

Save all graphs to save_file

-seed ***seed_ value***

Integer seed for random number generator

-source ***disk|pipe***

Source type of next data file

-timer ***delay***

Set allowed time slice for real time inputs to delay ms

-timestamp

Add timestamp to plot

-settype ***xy|xydx|...***

Set the type of the next data file

-version

Show the program version

-viewport ***xmin ymin xmax ymax***

Set the viewport for the current graph

-wd ***directory***

Set the working directory

-world *xmin ymin xmax ymax*

Set the world coordinates for the current graph

-usage|-help

This message

3.3 Customization

3.3.1 Environment variables

- GRACE_HOME Set the location of Grace. This will be where help files, auxiliary programs, and examples are located. If you are unable to find the location of this directory, contact your system administrator.
- GRACE_PRINT_CMD Print command. If the variable is defined but is an empty string, "Print to file" will be selected as default.
- GRACE_EDITOR The editor used for manual editing of dataset values.
- GRACE_HELPVIEWER The shell command to run an HTML viewer for on-line browsing of the help documents. Must include at least one instance of "%s" which will be replaced with the actual URL by Grace.
- GRACE_FFTW_WISDOM_FILE and GRACE_FFTW_RAM_WISDOM These flags control behavior of the FFTW planner (see 6.3 (FFTW tuning) for detailed info)

3.3.2 Init file

Upon start-up, Grace loads its init file, `gracerc`. The file is searched for in the magic path (see 3.1.11 (magic path)); once found, the rest of the path is ignored. It's recommended that in the `gracerc` file, one doesn't use statements which are part of a project file - such defaults, if needed, should be set in the default template (see 3.3.3 (default template)).

3.3.3 Default template

Whenever a new project is started, Grace loads the default template, `templates/Default.agr`. The file is searched for in the magic path (see 3.1.11 (magic path)); once found, the rest of the path is ignored. It's recommended that in the default template, one doesn't use statements which are NOT part of a project file - such defaults, if needed, should be set in the `gracerc` (see 3.3.2 (init file)).

3.3.4 X resources

The following Grace-specific X resource settings are supported:

- XMgrace.invertDraw
  Use GXinvert rather than GXxor for rubber-band lines. If the rubber-banding for zooms and lines, etc. doesn't appear on the canvas, set this resource to yes.

- XMgrace.allowDoubleClick
  When Yes, allow double clicks on the canvas to bring up various popups depending on the location of the pointer when the double click occurs.

- XMgrace.toolBar
  Enables button toolbar

- XMgrace.statusBar
  Enables status bar

- XMgrace.locatorBar
  Enables locator bar

It is also possible to customize menus by assigning key accelerators to any item.

You'll need to derive the item's X resource name from the respective menu label, which is easily done following these rules:

- All non-alphanumeric characters are skipped
- Start with lower case; each new word (if any) continues from the capital letter
- Add the item's type to the end - "Menu" for pulldown menus, "Button" for menu buttons.

For example, in order to make Grace popup the Non-linear curve fitting by pressing Control+F, you would add the following two lines

```
XMgrace*transformationsMenu.nonLinearCurveFittingButton.acceleratorText:  Ctrl+F
XMgrace*transformationsMenu.nonLinearCurveFittingButton.accelerator:  Ctrl<Key>f
```

to your `.Xresources` file (the file which is read when an X session starts; it could be `.Xdefaults`, `.Xsession` or some other file - ask your system administrator when in doubt).

Similarly, it may be desirable to alter default filename patterns of file selection dialogs. The recipe for the dialog's name is like for menu buttons outlined above, with "Button" being replaced with "FSB". E.g., to list all files in the "Open project" dialog ("File/Open..."), set the following resource:

```
XMgrace*openProjectFSB.pattern:  *
```

4 Guide to the graphical user interface

4.1 GUI controls

This section describes interface controls - basic building blocks, used in many popups.

4.1.1 File selection dialogs

Whenever the user is expected to provide a filename, either for reading in or writing some data, a file selection dialog is popped up. In addition to the standard entries (the directory and file lists and the filter entry), there is a pulldown menu for quick directory change to predefined locations (the current working directory, user's home directory and the file system root). Also, a "Set as cwd" button is there which allows to set any directory as you navigate through the directory tree as the current working directory (cwd). Once defined, it can be used in any other file selection dialog to switch to that directory quickly.

4.1.2 List selectors

Various selectors are available in several popups. They all display lists of objects (graphs, sets, ...) and can be used to perform simple operations on these objects (copying, deleting, ...). The operations are available from a popup menu that appears when pressing mouse button 3 on them. Depending on the required functionality, they may allow multiple choices or not. The following shortcuts are enabled (if the result of an action would contradict the list's selection policy, this would be ignored):

- Ctrl+a select all
- Ctrl+u unselect all
- Ctrl+i invert selection

Graph selector The operations that can be performed on graphs through the graph selector's popup menu are:

- focus to
- hide
- show
- duplicate
- kill
- swap
- create new

All this operations are not available in every instance of the selector. For example in the "read sets" popup only one graph can be selected at a time, and the swap operation is disabled.

Double-clicking on a list entry will switch the focus to that graph.

Set selector The operations that can be performed on sets through the set selector's popup menu are:

- hide
- show

- bring to front
- send to back
- duplicate
- kill
- kill data
- swap
- edit
  - in spreadsheet (see 4.4.1 (Spreadsheet data set editor))
  - in text editor
- create new
  - by formula
  - in spreadsheet (see 4.4.1 (Spreadsheet data set editor))
  - in text editor
  - from block data
- pack all sets
- selector operations
  - view set comments
  - show data-less
  - show hidden
  - select all
  - unselect all
  - invert selection
  - update

Double-clicking on a list entry will open the spreadsheet editor (see 4.4.1 (Spreadsheet data set editor)) on the set data.

4.2 The main window

4.2.1 The canvas

Canvas hotkeys When the pointer focus is on the canvas (where the graph is drawn), there are some shortcuts to activate several actions. They are:

- Ctrl <Key>A: Autoscale the current graph
- Ctrl <Key>D: Delete an object

- Ctrl <Key>L: Move current graph legend
- Ctrl <Key>M: Move an object
- Ctrl <Key>T: Place timestamp
- Ctrl <Key>U: Refresh hotlinks
- Ctrl <Key>V: Set the viewport with mouse
- Ctrl <Key>Z: Zoom
- Ctrl Alt <Key>L: Draw a line
- Ctrl Alt <Key>B: Draw a box
- Ctrl Alt <Key>E: Draw an ellipse
- Ctrl Alt <Key>T: Write a text string

Clicks and double clicks A single click inside a graph switches focus to that graph. This is the default policy, but it can be changed from the "Edit/Preferences" popup.

Double clicking on parts of the canvas will invoke certain actions or raise some popups:

- on a focus marker: move selected viewport corner
- on an axis: "Plot/Axis properties" popup
- on a set: "Plot/Set appearance" popup
- on a legend: "Plot/Graph appearance" popup
- on a (sub)title: "Plot/Graph appearance" popup
- on an object (box, line, ...): a popup for editing properties of that object

The double clicking actions can be enabled/disabled from the "Edit/Preferences" popup.

4.2.2 Toolbar buttons

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

- `Draw`: This will redraw the canvas and sets. Useful if "Auto Redraw" has been deselected in the Edit|Preferences dialog or after executing commands directly from the Window|Commands interpreter.
- `Lens`: A zoom lens. Click on the lens, then select the area of interest on the graph with the "rubber band". The region enclosed by the rubber band will fill the entire graph.
- `AS`: AutoScale. Autoscales the graph to contain all data points of all visible (not hidden) sets.
- `Z/z`: Zoom in/out by 5%. The zoom percentage can be set in the Edit/Preferences dialog.

- `Arrows`: Scroll active graph by 5% in the arrow's direction. The scroll percentage can be set in the Edit/Preferences dialog.
- `AutoT`: AutoTick Axes. This will find the optimum number of major and minor tick marks for both axes.
- `AutoO`: Autoscale On set. Click the `AutoO` button, then click on the graph near the set you wish to use for determining the autoscale boundaries of the graph.
- `ZX,ZY`: Zoom along an axis. These buttons work like the zoom lens above but are restricted to a single axis.
- `AX,AY`: Autoscale one axis only. The following buttons deal with the graph stack and there is a good example under Help/Examples/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.
- `Exit`: Pretty obvious, eh?

4.3 File menu

The file menu contains all entries related to the input/output features of Grace.

4.3.1 New

Reset the state of Grace as if it had just started (one empty graph ranging from 0 to 1 along both axes). If some work has been done and not yet saved, a warning popup is displayed to allow canceling the operation.

4.3.2 Open

Open an existing 3.1.1 (project file). A popup is displayed that allow to browse the file system.

4.3.3 Save

Save the current work in a project file, using the name that was used for the last open or save. If no name has been set (i.e., if the project has been created from scratch) act as 4.3.4 (save as).

4.3.4 Save as

Save the current work in a project file with a new name. A popup allows to browse the file system and set the name, the format to use for saving data points (the default value is "%16.8g"), and a textual description of the project. A warning is displayed if a file with the same name already exists.

4.3.5 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.

4.3.6 Print setup

Set the properties of the printing device. Each device has its own set of specific options (see 7.3 (Device-specific settings)). According to the device, the output can be sent either directly to a printer or directed to a file. The global settings available for all devices are the sizing parameters. The size of the graph is fixed. Changing the 'Page' settings changes the size of the canvas underneath the graph. Switching between portrait and landscape rotates the canvas. Make sure the canvas size is large enough to hold your graph. Otherwise you get a 'Printout truncated' warning. If your canvas size cannot easily be changed because, for example, you want to print on letter size paper, you need to adjust the size of your graph ('Viewport' in Plot/Graph Appearance).

4.3.7 Print

Print the project using the current printer settings

4.3.8 Exit

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

4.4 Edit menu

4.4.1 Data sets

Using the data set popup, you can view the properties of datasets. This include its type, length, associated comment and some statistics (min, max, mean, standard deviation). A horizontal scrollbar at the bottom allows to get the two last properties, they are not displayed by default. Also note that if you find some columns are too narrow to show all significant digits, you can drag the vertical rules using Shift+Button 2.

Using the menu on the top of this dialog, you can manipulate existing sets or add new ones. Among the most important entries in the menu, are options to create or modify a set using the spreadsheet data set editor (see 4.4.1 (Spreadsheet data set editor)).

Spreadsheet data set editor The dialog presents an editable matrix of numbers, corresponding to the data set being edited. The set type (and hence, the number of data columns) can be changed using the "Type:" selector. Clicking on a column label pops up a dialog allowing to adjust the column formatting. Clicking on the row labels toggles the respective row state (selected/unselected). The selected rows can be deleted via the dialog's "Edit" menu. Another entry in this menu lets you add a row; the place of the new row is determined by the row containing a cell with the keyboard focus on. As well, just typing in an empty cell will add one or several rows (filling the intermediate rows with zeros).

To resize columns, drag the vertical rules using Shift+Button 2.

4.4.2 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 4.5.1 (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.

4.4.3 Arrange graphs

This entry fires up a popup to lay out several graphs in a regular grid given by **M** rows and **N** columns.

The graph selector at the top 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 checkboxes below the graph selector.

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 (select a graph and double click on the focus marker, see 4.2.1 (clicks and double clicks)).

4.4.4 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 (apparently) graph. The main difficulty is to be sure you operate on the graph you want at all times (you can hide one for a moment if this becomes too difficult).

4.4.5 Autoscale

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

4.4.6 Regions menu

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

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

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

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

Clear This kills a region.

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

4.4.7 Hot links

You can link a set to a file or a pipe using this feature. Once a link has been established, you can update it (i.e., read data again) by clicking on the update button.

Currently, only simple XY sets can be used for hotlinks.

4.4.8 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].

4.4.9 Clear locator fixed point

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

4.4.10 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.

4.4.11 Preferences

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

4.5 Data menu

4.5.1 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 4.4.2 (set operations) popup from the Edit menu. You can sort according to any coordinate (X, Y, DX, ...) 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 4.1.2 (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.

4.5.2 Transformations menu

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

Evaluate expression Using evaluate expression allows you to create a set by applying an explicit formula to another set, or to parts of another set if you use regions restrictions.

All the classical mathematical functions are available (cos, sin, but also lgamma, j1, erf, ...). As usual all trigonometric functions use radians by default but you can specify a unit if you prefer to say cos (x rad) or sin (3 * y deg). For the full list of available numerical functions and operators, see 5.4 (Operators and functions).

In the formula, you can use X, Y, Y1, ..., Y4 to denote any coordinate you like from the source set. An implicit loop will be used around your formula so if you say:

```
x = x - 4966.5
```

you will shift all points of your set 4966.5 units to the left.

You can use more than one set in the same formula, like this:

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

which means you use both X and Y from the source set but also the Y coordinate of set 2. Beware that the loop is a simple loop over the indices, all the sets you use in such an hybrid expression should therefore have the same number of points and point i of one set should really be related to point i of the other set. If your sets do not follow these requirements, you should first homogenize them using 4.5.2 (interpolation).

Histograms The histograms popup allows you to compute either standard or cumulative histograms from the Y coordinates of your data. Optionally, the histograms can be normalized to 1 (hence producing a PDF (Probability Distribution Function).

The bins can be either a linear mesh defined by its min, max, and length values, or a mesh formed by abscissas of another set (in which case abscissas of the set must form a strictly monotonic array).

Fourier transforms This popup is devoted to direct and inverse Fourier transforms. The default is to perform a direct transform on unfiltered data and to produce a set with the index as abscissa and magnitude as ordinate. You can filter the input data window through triangular, Hanning, Welch, Hamming, Blackman and Parzen filters. You can load magnitude, phase or coefficients and use either index, frequency or period as abscissas. You can choose between direct and inverse Fourier transforms. If you specify real input data, X is assumed to be equally spaced and ignored; if you specify complex input data X is taken as the real part and Y as the imaginary part.

If Grace was configured with the FFTW library (see 1 (configuration)), then the DFT and FFT buttons really perform the same transform (so there is no speed-up in using FFT in this case). If you want Grace can to use FFTW *wisdom* files, you should set several 3.3.1 (environment variables) to name them.

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

Differences The differences popup is used to compute approximations of the first derivative of a function with finite differences. The only choice (apart from the source set of course) is the type of differences to use: forward, backward or centered.

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

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

Interpolation/Splines This popup is used to interpolate a set on an array of alternative X coordinates. This is mainly used before performing some complex operations between two sets with the 4.5.2 (evaluate expression) popup.

The sampling array can be either a linear mesh defined by its min, max, and length values, or a mesh formed by abscissas of another set.

Several interpolation methods can be used: linear, spline or Akima spline.

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 a primitive extrapolation. This behaviour can be disabled by checking the "Strict" option on the popup.

The abscissas of the set being interpolated must form a strictly monotonic array.

Regression The regression popup can be used to fit a set against polynomials or some specific functions (y=A*x^B, y=A*exp(B*x), y=A+B*ln(x) and y=1/(A+Bx)) for which a simple transformation of input data can be used to apply linear regression formulas.

You can load either the fitted values, the residuals or the function itself. Choosing to load fitted values or residuals leads to a set of the same length and abscissas as the initial set. Choosing to load the function is almost similar to load the fitted values except that you choose yourself the boundaries and the number of points. This can be used for example to draw the curve outside of the data sample range or to produce an evenly spaced set from an irregular one.

Non-linear fit The non linear fit popup can be used for functions outside of the simple regression methods scope. With this popup you provide the expression yourself using a0, a1, ..., a9 to denote the fit parameters (as an example you can say y = a0 * cos (a1 * x + a2)). You specify a tolerance, starting values and optional bounds and run several steps before loading the results.

The fit characteristics (number of parameters, formula, ...) can be saved in a file and retrieved as needed using the file menu of the popup.

In the "Advanced" tab, you can additionally apply a restriction to the set(s) to be fitted (thus ignoring points not satisfying the criteria), use one of preset weighting schemes or define your own, and choose whether to load the fitted values, the residuals or the function itself. Choosing to load fitted values or residuals leads to a set of the same length and abscissas as the initial set. Choosing to load the function is almost similar to load the fitted values except that you choose yourself the boundaries and the number of points. This can be used for example to draw the curve outside of the data sample range or to produce an evenly spaced set from an irregular one.

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 C(0) = 1.

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

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

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

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

Prune data This popup is devoted to reducing huge sets (and then saving both computation time and disk space).

The interpolation method can be applied only to ordered sets: it is based on the assumption that if a real point and an interpolation based on neighboring points are closer than a specified threshold, then the point is redundant and can be eliminated.

The geometric methods (circle, ellipse, rectangle) can be applied to any set, they test each point in turn and keep only those that are not in the neighborhood of previous points.

4.5.3 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.

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 (FFT if set length a power of 2) to find largest frequency component
Period	Inverse of above
Zero crossing	Time of the first zero crossing, + or - going
Rise time	Assume curve starts at the minimum and rises to the maximum, get time to go from 10% to 90% of rise. For single exponential curves, this is 2.2*time constant
Fall time	Assume curve starts at the maximum and drops to the minimum, get time to go from 90% to 10% of fall
Slope	Perform linear regression to obtain slope
Y intercept	Perform linear regression to obtain Y-intercept
Set length	Number of data points in set
Half maximal width	Assume curve starts from the minimum, rises to the maximum and drops to the minimum again. Determine the time for which the curve is elevated more than 50% of the maximum rise.
Barycenter X	Barycenter along X axis
Barycenter Y	Barycenter along Y axis
X (Y max)	X of Maximum Y
Y (X max)	Y of Maximum X
integral	cumulative sum

Table 4: Extractable features

4.5.4 Import menu

ASCII Read new sets of data in a graph. A 4.1.2 (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).

Reading as "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. Reading as "NXY" means that the first numeric column will provide the abscissas and all remaining columns will provide the ordinates of several sets. Reading as "Block data" means all column will be read and stored and that another popup will allow to select the abscissas and ordinates at will. It should be noted that block data are stored as long as you do not override them by a new read. You can still retrieve data from a block long after having closed all popups, using the 4.1.2 (set selector).

The set type can be one of the predefined set presentation types (see 3.1.6 (sets)).

The data source can be selected as "Disk" or "Pipe". In the first case the text in the "Selection" field is considered to be a file name (it can be automatically set by the file selector at the top of the popup). In the latter case the text is considered to be a command which is executed and should produce the data on its standard output. On systems that allows is, the command can be a complete sequence of programs glued together with pipes.

If the source contains date fields, they should be automatically detected. Several formats are recognized (see appendix 7.4 (dates in grace)). Calendar dates are converted to numerical dates upon reading.

The "Autoscale on read" menu controls whether, upon reading in new sets, which axes of the graph should be autoscaled.

NetCDF This entry exists only if Grace has been compiled with support for the NetCDF data format (see 1 (configuration)).

4.5.5 Export menu

ASCII Save data sets in a file. A 4.1.2 (set selector) is used to specify the set to be saved. The format to use for saving data points can be specified (the default value is "%16.8g"). A warning is displayed if a file with the same name already exists.

4.6 Plot menu

4.6.1 Plot appearance

The plot appearance popup let you set the time stamp properties and the background color of the page. The color is used outside of graphs and also on graphs were no specific background color is set. The time stamp is updated every time the project is modified.

4.6.2 Graph appearance

The graph appearance popup can be displayed from both the plot menu and by double-clicking on a legend, title, or subtitle of a graph (see 4.2.1 (Clicks and double clicks)). The graph selector at the top allows to choose the graph you want to operate on, it also allows certain common actions through its popup menu (see 4.1.2 (graph selector)). Most of the actions can also be performed using the "Edit" menu available from the popup menubar. The 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).

If you need special characters or special formatting in your title or subtitle, you can use Grace escape sequences (the sequence will appear verbatim in the text field but will be rendered on the graph), see 7.1 (typesetting). If you don't remember the mapping between alphabetic characters and the glyph you need in some specific fonts (mainly symbol and zapfdingbats), you can invoke the font tool from the text field by hitting CTRL-e. You can change fonts and select characters from there, they will be copied back in the text field when you press the "Accept" button. Beware of the position of the cursor as you enter text or change font in the font tool, the character or command will be inserted at this position, not at the end of the string!

You can save graph appearance parameters or retrieve settings previously saved via the "File" menu of this popup. In the "Save parameters" dialog, you can choose to save settings either for the current graph only or for all graphs.

4.6.3 Set appearance

The set appearance popup can be displayed from both the plot menu and by double-clicking anywhere in a graph (see 4.2.1 (Clicks and double clicks)). The set selector at the top allows to choose the set you want to operate on, it also allows certain common actions through its popup menu (see 4.1.2 (set selector)). The main tab gathers the properties you will need more often (line and symbol properties or legend string for example), and other tabs are used to fine tune some less frequently used options (drop lines, fill properties, annotated values and error bars properties for example).

You should note that despite the legend string related to *one* set is entered in the set appearance popup, this is not sufficient to display it. Displaying *all* legends is a graph level decision, so the toggle is in the main tab of the 4.6.2 (graph appearance) popup.

If you need special characters or special formatting in your legend, you can use Grace escape sequences (the sequence will appear verbatim in the text field but will be rendered on the graph), see 7.1 (typesetting). If you don't remember the mapping between alphabetic characters and the glyph you need in some specific fonts (mainly symbol and zapfdingbats), you can invoke the font tool from the text field by hitting CTRL-e. You can change fonts and select characters from there, they will be copied back in the text field when you press the "Accept" button. Beware of the position of the cursor as you enter text or change font in the font tool, the character or command will be inserted at this position, not at the end of the string!

4.6.4 Axis properties

The axis properties popup can be displayed from both the "Plot" menu and by double-clicking exactly on an axis (see 4.2.1 (Clicks and double clicks)). The pulldown menu at the top allows to select the axis you want to operate on. The "Active" toggle globally activates or deactivates the axis (all GUI elements are insensitive for deactivated axes). The start and stop fields depict the displayed range. Three types of scales are available: linear, logarithmic or reciprocal, and you can invert the axis (which normally increases from left to right and from bottom to top). The main tab includes the properties you will need more often (axis label, tick spacing and format for example), and other tabs are used to fine tune some less frequently used options (fonts, sizes, colors, placements, stagger, grid lines, special ticks, ...).

If you need special characters or special formatting in your label, you can use Grace escape sequences (the sequence will appear verbatim in the text field but will be rendered on the graph), see 7.1 (typesetting). If you don't remember the mapping between alphabetic characters and the glyph you need in some specific fonts (mainly symbol and zapfdingbats), you can invoke the font tool from the text field by hitting CTRL-e. You can change fonts and select characters from there, they will be copied back in the text field when you press the "Accept" button. Beware of the position of the cursor as you enter text or change font in the font tool, the character or command will be inserted at this position, not at the end of the string!

Once you have set the options as you want, you can apply them. One useful feature is that you can set several axes at once with the bottom pulldown menu (current axis, all axes current graph, current axis all graphs, all axes all graphs). Beware that you always apply the properties of all tabs, not only the selected one.

4.7 View menu

4.7.1 Show locator bar

This toggle item shows or hides the locator below the menu bar.

4.7.2 Show status bar

This toggle item shows or hides the status string below the canvas.

4.7.3 Show tool bar

This toggle item shows or hides the tool bar at the left of the canvas.

4.7.4 Page setup

Set the properties of the display device. It is the same dialog as in 4.3.6 (Print setup).

4.7.5 Redraw

This menu item triggers a redrawing of the canvas.

4.7.6 Update all

This menu item causes an update of all GUI controls. Usually, everything is updated automatically, unless one makes modifications by entering commands in the 4.8.1 (Command) tool.

4.8 Window menu

4.8.1 Commands

Command driven version of the interface to Grace. Here, commands are typed at the "Command:" text item and executed when <Return> is pressed. The command will be parsed and executed, and the command line is placed in the history list. Items in the history list can be recalled by simply clicking on them with the left mouse button. For a reference on the Grace command interpreter, see 5 (Command interpreter).

4.8.2 Point tracking

Not written yet...

4.8.3 Drawing objects

Not written yet...

4.8.4 Font tool

Not written yet...

4.8.5 Console

The console window displays errors and results of some numerical operations, e.g. nonlinear fit (see 4.5.2 (Non-linear fit)). The window is popped up automatically whenever an error occurs or new result messages appear. This can be altered by checking the "Options/Popup only on errors" option.

4.9 Help menu

4.9.1 On context

Click on any element of the interface to get context-sensitive help on it. Only partially implemented at the moment.

4.9.2 User's guide

Browse the Grace user's guide.

4.9.3 Tutorial

Browse the Grace tutorial.

4.9.4 FAQ

Frequently Asked Questions with answers.

4.9.5 Changes

The list of changes during the Grace development.

4.9.6 Examples

The whole tree of submenus each loading a sample plot.

4.9.7 Comments

Use this to send your suggestions or bug reports.

4.9.8 License terms

Grace licensing terms will be displayed (GPL version 2).

4.9.9 About

A popup with basic info on the software, including some configuration details. More details can be found when running Grace with the "-version" command line flag.

5 Command interpreter

5.1 General notes

The interpreter parses its input in a line-by-line manner. There may be several statements per line, separated by semicolon (;). The maximal line length is 4 kbytes (hardcoded). The parser is case-insensitive and ignores lines beginning with the "#" sign.

5.2 Definitions

Name	Description	Examples
expr	Any numeric expression	1.5 + sin(2)
iexpr	Any expression that evaluates to an integer	25, 0.1 + 1.9, PI/asin(1)
nexpr	Non-negative iexpr	2 - 1
indx	Non-negative iexpr	
qstr	Quoted string	"a string"
vexpr	Vector expression	"2*x"

Table 5: Basic types

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

Table 6: Graph selections

Not finished yet...

Expression	Description	Types	Example
graph.SETS[*id*]	set *id* in graph *graph*	indx *id*, graphsel *graph*	GRAPH[0].SETS[1]
graph.S*nn*	set *nn* in graph *graph*	*nn*: 0-99, graphsel *graph*	G0.S1
SET[*id*]	set *id* in the current graph	indx *id*	SET[1]
S*nn*	set *nn* in the current graph	*nn*: 0-99	S1
S_	the last allocated set in the current graph	-	S_
S$	the active set in the current graph	-	S$

Table 7: Set selections

Expression	Description	Types	Example
R*n*	region *n*	*n*: 0-4	R0

Table 8: Region selections

Expression	Description	Types	Example
COLOR "*colorname*"	a mapped color *colorname*	-	COLOR "red"
COLOR *id*	a mapped color with ID *id*	nexpr *id*	COLOR 2

Table 9: Color selections

Expression	Description	Types	Example
PATTERN *id*	pattern with ID *id*	nexpr *id*	PATTERN 1

Table 10: Pattern selections

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

Table 11: Data column selections

5.3 Variables

Variable	Description
datacolumn	data column of current set
set.datacolumn	data column of set
vvar	user-defined array
vvariable [i:j]	segment of a vector variable (elements from i-th to j-th inclusive, i <= j)

Table 12: Vector variables

Variable	Description
vvariable[i]	i-th element of a vector variable
var	user-defined variable

Table 13: Scalar variables

5.4 Numerical operators and functions

In numerical expressions, the infix format is used. Arguments of both operators and functions can be either scalars or vector arrays.

Operator	Description
+	addition
-	substraction
	multiplication
/	division
%	modulus
ˆ	raising to power

Table 14: Arithmetic operators

5.5 Procedures

Methods of directly manipulating the data corresponding to the Data|Transformation menu are described in table 21 ().

Not finished yet...

5.6 Device parameters

For producing "hard copy", several parameters can be set via the command interpreter. They are summarized in table 22 (Device parameters).

<table>
<tr><th>Operator</th><th>Description</th></tr>
<tr><td>AND or &&</td><td>logical AND</td></tr>
<tr><td>OR or ||</td><td>logical OR</td></tr>
<tr><td>NOT or !</td><td>logical NOT</td></tr>
</table>

Table 15: Logical operators

Operator	Description
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

Table 16: Comparison operators

5.7 Flow control

5.8 Declarations

User-defined variables are set and used according to the syntax described in table 24 (User variables).

Not finished yet...

5.9 Graph properties

We divide the commands pertaining to the properties and appearance of graphs into those which directly manipulate the graphs and those that affect the appearance of graph elements—the parameters that can appear in a Grace project file.

5.9.1 Command operations

General graph creation/annihilation and control commands appear in table 25 (Graph operations).

5.9.2 Parameter settings

Setting the active graph and its type is accomplished with the commands found in table 26 (Graph selection parameters).

The axis range and scale of the current graph as well as its location on the plot viewport are set with the commands listed in table 27 (Axis parameters).

The commands to set the appearance and textual content of titles and legends are given in table 28 (Titles and legends).

Not finished yet...

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 ... n - 1)
mesh(x1, x2, n)	mesh array of n equally spaced points between x1 and x2 inclusive
minof(x,y)	returns lesser of x and y
mod(x,y)	mod function (also x % y)
pi	the PI constant
rand	pseudo random number distributed uniformly on (0.0,1.0)
rand(n)	array of n random numbers
rint(x)	round to closest integer
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 17: Functions

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 18: Statistical functions

5.10 Set properties

Again, as with the graphs, we separate those parser commands that manipulate the data in a set from the commands that determine parameters—elements that are saved in a project file.

5.10.1 Commands

Operations for set I/O are summarized in table 29 (Set input, output, and creation). (Note that this is incomplete and only lists *input* commands at the moment.)

The parser commands analogous to the Data|Data set operations dialogue can be found in table 30 (Set operations).

Not Finished yet...

5.10.2 Parameter settings

Not written yet...

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

Table 19: Special math functions

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

Table 20: Aggregate functions

Statement	Description	Types	Example
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	vexpr *mesh*, *method*: one of LINEAR, SPLINE, and ASPLINE, onoff *strict*	INTERPOLATE (S0, S1.X, ASPLINE, OFF)
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	vexpr *bins*, onoff *cumulative*, onoff *normalize*	HISTOGRAM (S0, MESH(0, 1, 11), OFF, ON)
XCOR (set1, set2, maxlag, covar)	calculate cross-correlation (or -covariance if the *covar* flag is set) of *set1* with *set2* with maximum lag *maxlag*.	nexpr *maxlag*, onoff *covar*	XCOR (S0, S0, 50, OFF)
RESTRICT (set, restriction)	filter *set* according to logical *restriction*. The original set will be overwritten	vexpr *restriction*	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	onoff *negate*	RESTRICT (S0, R1, OFF)

Table 21: Transformations

Command	Description
PAGE SIZE xdim, ydim	set page dimensions (in pp) of all devices
PAGE RESIZE xdim, ydim	same as above plus rescale the current plot accordingly
DEVICE *"devname"* PAGE SIZE xdim, ydim	set page dimensions (in pp) of device *devname*
DEVICE *"devname"* DPI dpi	set device's dpi (dots per pixel)
DEVICE *"devname"* FONT onoff	enable/disable usage of built-in fonts for device *devname*
DEVICE *"devname"* FONT ANTIALIASING onoff	enable/disable font aliasing for device *devname*
DEVICE *"devname"* OP *"options"*	set device specific options (see 7.3 (Device-specific settings))
HARDCOPY DEVICE *"devname"*	set device *devname* as current hardcopy device
PRINT TO *"filename"*	set print output to *filename* (but do not print)
PRINT TO DEVICE	set print output to hardcopy device (but do not print)

Table 22: Device parameters

Statement	Description	Types	Example
EXIT(*status*)	cause normal program termination with exit status *status*	iexpr *status*	EXIT(0)
EXIT	cause normal program termination; same as EXIT(0)		EXIT
HELP *url*	open a HTML document pointed to by *url*	qstr *url*	HELP "doc/FAQ.html"
HELP	open User's Guide		HELP
PRINT	execute print job		PRINT
REDRAW	refresh the canvas to reflect the current project state		REDRAW
SLEEP n	sleep for n seconds	expr n	SLEEP(3)
UPDATEALL	update the GUI (graph and set selectors etc) to reflect the current project state		UPDATEALL

Table 23: Flow control

Statement	Description	Types	Example
DEFINE *var*	define new scalar variable *var*		DEFINE myvar
DEFINE *vvar*[]	define new vector variable *vvar* of zero length		DEFINE myvvar[]
DEFINE *vvar*[n]	define new vector variable *vvar* of length n	nexpr n	DEFINE myvvar[10]
CLEAR *var*	undefine new variable *var* and deallocate associated storage		CLEAR myvar
vvar LENGTH n	reallocate vector variable *vvar*	nexpr n	myvvar LENGTH 25

Table 24: User variables

Statement	Description	Types	Example
FOCUS *graph*	Makes *graph* current and unhides it if necessary	graphsel *graph*	FOCUS G0
KILL *graph*	Kills *graph*	graphsel *graph*	KILL G0
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)

Table 25: Graph operations

Statement	Description	Types	Example
WITH *graph*	Makes *graph* current	graphsel *graph*	WITH G0
TYPE *type*	Sets *type* of current graph	graphtype *type*	TYPE XY
graph onoff	(De)Activates selected *graph*	graphsel *graph*, onoff	G0 ON
graph HIDDEN onoff	Hides selected *graph*	graphsel *graph*, onoff	G1 HIDDEN TRUE
graph TYPE *type*	Sets *type* of *graph*	graphsel *graph*, graphtype *type*	G0 TYPE XYDY

Table 26: Graph selection parameters

Statement	Description	Types	Example
WORLD XMIN *xmin*	Sets minimum value of current graph's x axis to *xmin*	expr *xmin*	WORLD XMIN -10
WORLD XMAX *xmax*	Sets maximum value of current graph's x axis to *xmin*	expr *xmax*	WORLD XMAX 22.5
WORLD YMIN *ymin*	Sets minimum value of current graph's y axis to *ymin*	expr *ymin*	WORLD YMIN 0
WORLD YMAX *ymax*	Sets maximum value of current graph's y axis to *ymax*	expr *ymax*	WORLD YMAX 1e4
VIEW XMIN *xmin*	Sets left edge of current graph at x=*xmin* in the viewport	expr *xmin*	VIEW XMIN .2
VIEW XMAX *xmax*	Sets right edge of current graph at x=*xmax* in the viewport	expr *xmax*	VIEW XMAX 1.0
VIEW YMIN *ymin*	Sets bottom edge of current graph at y=*ymin* in the viewport	expr *ymin*	VIEW YMIN .25
VIEW YMAX *ymax*	Sets top edge of current graph at y=*ymax* in the viewport	expr *ymax*	VIEW YMAX .75
XAXES SCALE *type*	Set scaling of the x axes to *type*	*type*: one of NORMAL, LOGARITHMIC, or RECIPROCAL	XAXES SCALE NORMAL
YAXES SCALE *type*	Set scaling of the y axes to *type*	*type*: one of NORMAL, LOGARITHMIC, or RECIPROCAL	YAXES SCALE LOGARITHMIC
XAXES INVERT onoff	If ON, draws xmin to xmax from right to left	onoff	XAXES INVERT OFF
YAXES INVERT onoff	If ON, draws ymin to ymax from top to bottom	onoff	YAXES INVERT OFF
AUTOSCALE ONREAD *type*	Set automatic scaling on read according to *type*	*type*: one of NONE, XAXES, YAXES, XYAXES	AUTOSCALE ONREAD NONE

Table 27: Axis parameters

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

Table 28: Titles and legends

Statement	Description	Types	Example
READ *file*	Reads *file* as a single set	qstr *file*	READ "foo.dat"
READ *settype file*	Reads *file* into a single set of type *settype*	xytype *settype*, qstr *file*	READ xydy "bar.dat"
READ NXY *file*	Reads *file* as NXY data	qstr *file*	READ NXY "gad.dat"
READ BLOCK *file*	Reads *file* as block data	qstr *file*	READ BLOCK "zooks.dat"
BLOCK *settype columns*	Forms a data set of type *settype* using *columns* from current block data file.	xytype *settype*, qstr *columns*	BLOCK xydxdy "0:2:1:3"

Table 29: Set input, output, and creation

Statement	Description	Types	Example
COPY *src* TO *dest*	Copies *src* to *dest*	setsel *src,dest*	COPY S0 TO S1
MOVE *src* TO *dest*	Moves *src* to *dest*	setsel *src,dest*	MOVE G0.S0 TO G1.S0
SWAP *src* AND *dest*	Interchanges *src* and *dest*	setsel *src,dest*	SWAP G0.S0 AND G0.S1
KILL *set*	Kills *set*	setsel *set*	KILL G0.S0

Table 30: Set operations

6 Advanced topics

6.1 Fonts

For all devices, Grace uses Type1 fonts. Both PFA (ASCII) and PFB (binary) formats can be used.

6.1.1 Font configuration

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

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

Here is the default `FontDataBase` file:

```
14
Times-Roman             Times-Roman             n021003l.pfb
Times-Italic            Times-Italic            n021023l.pfb
Times-Bold              Times-Bold              n021004l.pfb
```

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

6.1.2 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`

6.1.3 Custom fonts

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

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

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

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

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

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

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

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

6.2 Interaction with other applications

6.2.1 Using pipes

6.2.2 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.

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).

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
```

Function	Arguments	Description
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("xmgrace", 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 31: 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 32: grace_np library F77 functions.

```
#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);
    }
}
```

6.3 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 is 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 (FFTW_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.

6.4 DL modules

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

6.4.1 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

Table 33: Grace types for external functions

The return values of functions are assumed to be of the `double` type.

Note, that there is no difference from the point of view of function prototype between parameters and variables; the difference is in the way Grace treats them - an attempt to use a vector expression as a parameter argument will result in a parse error.

Let us consider few examples.

6.4.2 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.

The command to make it accessible by Grace is

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

Try to plot y = pow(x,2) and y = x^2 graphs (using, for example, "create new -> Formula" from any 4.1.2 (set selector)) and compare.

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 33 (types). 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 modules.

Here we will try to achieve the same functionality as in Example 2, but with the help of F77.

```
      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:

```
$g77 -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"
```

6.4.3 Operating system issues

OS/2 In general the method outlined in the examples above can be used on OS/2, too. However you have to create a DLL (Dynamic Link Library) which is a bit more tricky on OS/2 than on most Un*x systems. Since Grace was ported by using EMX we also use it to create the examples; however other development environments should work as well (ensure to use the _System calling convention!). We refer to Example 2 only. Example 1 might demonstrate that DLLs can have their entry points (i.e. exported functions) callable via ordinals only, so you might not know how to access a specific function without some research. First compile the source from Example 2 to "my_func.obj"

```
gcc -Zomf -Zmt -c my_func.c -o my_func.obj
```

Then you need to create a linker definition file "my_func.def" which contains some basic info about the DLL and declares the exported functions.

```
LIBRARY my_func INITINSTANCE TERMINSTANCE
CODE LOADONCALL
DATA LOADONCALL MULTIPLE NONSHARED
DESCRIPTION 'This is a test DLL: my_func.dll'
EXPORTS
my_function
```

(don't forget about the 8 characters limit on the DLL name!). Finally link the DLL:

```
gcc my_func.obj my_func.def -o my_func.dll -Zdll -Zno-rte -Zmt -Zomf
```

(check out the EMX documentation about the compiler/linker flags used here!) To use this new library function within Grace you may either put the DLL in the LIBPATH and use the short form:

```
USE "my_function" TYPE f_of_nd FROM "my_func" ALIAS "myf"
```

or put it in an arbitrary path which you need to specify explicitly then:

```
USE "my_function" TYPE f_of_nd FROM "e:/foo/my_func.dll" ALIAS "myf"
```

(as for most system-APIs you may use the Un*x-like forward slashs within the path!)

7 References

7.1 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.

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 34: Control codes.

Example:

F\sX\N(\xe\f{}) = sin(\xe\f{})\#{b7}e\S-X\N\#{b7}cos(\xe\f{})

prints roughly

```
                   -x
F (e) = sin(e).e     .cos(e)
 x
```

using string's initial font and e prints as epsilon from the Symbol font.

NOTE: Characters from the upper half of the char table can be entered directly from the keyboard, using appropriate `xmodmap(1)` settings, or with the help of the font tool ("Window/Font tool").

7.2 Device-specific limitations

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

- X11, PostScript and EPS are full-featured devices
- Raster drivers (PNM/JPEG/PNG):
  - only even-odd fill rule is supported
  - patterned lines are not implemented
- PDF driver:
  - bitmapped text strings are not transparent
- MIF driver:
  - some of patterned fills not implemented
  - bitmapped text strings not implemented
- SVG driver:
  - bitmapped text strings not implemented

7.3 Device-specific settings

Some of the output devices accept several configuration options. You can set the options by passing a respective string to the interpreter using the "DEVICE *"devname"* OP *"options"*" command (see 22 (Device parameters)). A few options can be passed in one command, separated by commas.

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

Table 35: PostScript driver options

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

Table 36: EPS driver options

Command	Description
PDF1.2	set compatibility mode to PDF-1.2
PDF1.3	set compatibility mode to PDF-1.3
compression:value	set compression level (0 - 9)
patterns:on	enable use of patterns
patterns:off	disable use of patterns

Table 37: PDF driver options

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

Table 38: PNM driver options

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

Table 39: JPEG driver options

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

Table 40: PNG driver options

7.4 Dates in Grace

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

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

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

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

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

Grace stores dates internally as reals numbers counted from a reference date. The default reference date is the one chosen by Scaliger, it is a classical reference for astronomical events. It can modified for a single session using the 4.4.11 (Edit->Preferences) popup of the GUI. If you often work with a specific reference date you can set it for every sessions with a REFERENCE DATE command in your configuration file (see 3.3.3 (Default template)).

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

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

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

You can also provide a hint about the format ("ISO8601", "european", "us") using the -datehint command line flag or the ref name="Edit->Preferences" id="preferences"> popup of the GUI. The formats are tried in the following order: first the hint given by the user, then iso, european and us (there is no ambiguity between calendar formats and numerical formats and therefore no order is specified for them). The separators between various fields can be any characters in the set: " :/.-T" (one or more spaces act as one separator, other characters can not be repeated, the T separator is allowed only between date and time, mainly for iso8601), so the string "1999-12 31:23/59" is allowed (but not recommended). The '-' character is used both as a separator (it is traditionally used in iso8601 format) and as the unary minus (for dates in the far past or for numerical dates). By default years are left untouched, so 99 is a date far away in the past. This behavior can be changed with the 4.4.11 (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.

7.5 Xmgr to Grace migration guide

This is a very brief guide describing problems and workarounds for reading in project files saved with Xmgr. You should read the docs or just play with Grace to test new features and controls.

1. Grace must be explicitly told the version number of the software used to create a file. You can manually put "@version VERSIONID" string at the beginning of the file. The VERSIONID is built as MAJOR_REV*10000 + MINOR_REV*100 + PATCHLEVEL; so 40101 corresponds to xmgr-4.1.1. Projects saved with Xmgr-4.1.2 do NOT need the above, since they already have the version string in them. If you have no idea what version of Xmgr your file was created with, try some. In most cases, 40102 would do the trick.

2. The above relates to the ASCII projects only. The old binary projects (saved with xmgr-4.0.*) are not automatically converted anymore. An input filter must be defined to make the conversion work on-the-fly. Add the following line to /.gracerc or the system-wide $GRACE_HOME/gracerc resource file: DEFINE IFILTER "grconvert %s -" MAGIC "00000031" See docs for more info on the I/O filters.

3. Documentation on the script language is severely lacking still.

4. Grace is WYSIWYG. Xmgr was not. Many changes required to achieve the WYSIWYG'ness led to the situation when graphs with objects carefully aligned under Xmgr may not look so under Grace. Grace tries its best to compensate for the differences, but sometimes you may have to adjust such graphs manually.

5. A lot of symbol types (all except *real* symbols) are removed. "Location *" types can be replaced (with much higher comfort) by A(nnotating)values. "Impulse *", "Histogram *" and "Stair steps *" effects can be achieved using the connecting line parameters (Type, Drop lines). "Dot" symbol is removed as well; use the filled circle symbol of the zero size with no outline to get the same effect.

6. Default page layout switched from free (allowing to resize canvas with mouse) to fixed. For the old behavior, put "PAGE LAYOUT FREE" in the Grace resource file or use the "-free" command line switch. **The use of the "free" page layout is in general deprecated, though.**

7. System (shell) variables GR_* renamed to GRACE_*

8. Smith plots don't work now. They'll be put back soon.