

Grace User's Guide (for Grace-5.99.1)

The Grace Team

28.01.2007

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

Contents

1	Introduction	4
1.1	What is Grace?	4
1.2	Copyright statement	4
2	Installation guide	5
2.1	Installing from sources	5
2.2	Binary installation	6
3	Getting started	6
3.1	General concepts	7
3.1.1	Project tree	7
3.1.2	Coordinate frames	7
3.1.3	Frames	7
3.1.4	Graphs	7
3.1.5	Sets	8
3.1.6	Regions	8
3.1.7	Project files	9
3.1.8	Datasets	9
3.1.9	Input File formats	9
3.1.10	Real Time Input	9
3.1.11	Hotlinks	10
3.1.12	Devices	10
3.1.13	Magic path	10
3.1.14	Dynamic modules	10
3.2	Invocation	10
3.2.1	Operational mode	10
3.2.2	Command line options	11
3.3	Customization	12
3.3.1	Environment variables	12
3.3.2	Default template	13

3.3.3	X resources	13
4	Guide to the graphical user interface	13
4.1	GUI controls	13
4.1.1	File selection dialogs	14
4.1.2	List selectors	14
4.1.3	Pen chooser	15
4.1.4	Spreadsheet data set editor	15
4.2	The main window	15
4.2.1	The canvas	15
4.2.2	Toolbar buttons	16
4.3	File menu	16
4.3.1	New	16
4.3.2	Open	16
4.3.3	Save	16
4.3.4	Save as	16
4.3.5	Revert to saved	16
4.3.6	Print setup	17
4.3.7	Print	17
4.3.8	Exit	17
4.4	Edit menu	17
4.4.1	Explorer	17
4.4.2	Arrange frames	18
4.4.3	Autoscale graphs	18
4.4.4	Preferences	18
4.5	Data menu	19
4.5.1	Data set operations	19
4.5.2	Transformations menu	19
4.5.3	Feature extraction	21
4.5.4	Import menu	21
4.5.5	Export menu	21
4.6	View menu	22
4.6.1	Show/hide	22
4.6.2	Page zoom	22
4.6.3	Page rendering	22
4.6.4	Redraw	22
4.6.5	Update all	22

4.7	Tools menu	22
4.7.1	Font tool	22
4.7.2	Console	22
4.7.3	Dataset statistics	23
4.8	Help menu	23
4.8.1	On context	23
4.8.2	User's guide	23
4.8.3	FAQ	23
4.8.4	Changes	23
4.8.5	Examples	23
4.8.6	Web support	23
4.8.7	License terms	23
4.8.8	About	24
5	Command interpreter	24
5.1	General notes	24
5.2	Definitions	24
5.3	Variables	24
5.4	Numerical operators and functions	24
6	Advanced topics	27
6.1	Fonts	27
6.1.1	Font configuration	27
6.1.2	Font data files	29
6.1.3	Custom fonts	29
6.2	Interaction with other applications	30
6.2.1	Using pipes	30
6.3	FFTW tuning	30
6.4	DL modules	30
6.4.1	Function types	30
6.4.2	Examples	31
6.4.3	Operating system issues	33
7	References	34
7.1	Typesetting	34
7.2	Device-specific limitations	34
7.3	Device-specific settings	34
7.4	Dates in Grace	38

7.5 Xmgr to Grace migration guide	39
---	----

1 Introduction

1.1 What is Grace?

Grace is a WYSIWYG tool to make two-dimensional plots of scientific 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/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.

As of now, most of the Grace codebase has been re-written from scratch.

1.2 Copyright statement

```
Copyright ((C)) 1996-2007 Grace Development Team
Portions Copyright ((C)) 1991-1995 Paul J Turner, Portland, OR
```

```
Maintained by Evgeny Stambulchik
```

```
All Rights Reserved
```

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

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

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

For 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
T1lib	LGPL
Expat	MIT/X
Xbae widget	BSD-like
Tab widget	BSD-like
ListTree widget	LGPL

Table 1: Third-party license terms

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 implementation of the M*tif API, version 1.2 or above (2.1 is highly recommended). 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:
 - Multi-level undo functionality is based on the libundo library, version 0.8.0.
 - All raster backends (PNG, JPEG, ...) are based on the (*Xmi library* <<http://www.gnu.org/software/libxmi/libxmi.html>>), version 1.2.
 - 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 to be installed, which is available *here* <<http://www.pdflib.com/>> , version 6.0.0 or above.
 - If your computer has the FFTW library (see the *FFTW Home page* <<http://www.fftw.org>>) installed when Grace is compiled, Grace will link itself to this, and drop all conventional FFT's and DFT's. All Fourier transforms will be routed through this package, resulting in higher accuracy and better (much better in many cases) performance. You'll need version 2.1.*, since the FFTW-3 API is not supported as of yet.
 - 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`.

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

We neither provide nor support binary builds. Please contact your vendor. Practically all general-purpose GNU/Linux and other Open Source distributions have a Grace package in either main or contributed sections.

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.

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

1. Start the GUI version: `xmgrace` (return).
2. Load your data with Data/Import/ASCII. 'Load as': 'Single set' for two-column ASCII data, 'Block data' for multi-column ASCII data.
3. Fire up project explorer (Edit/Explorer).
4. Click on the top-level node of the project tree and select/check the page size.

5. Adjust the scales, axis labels and tick marks in "Axis" quark properties.
6. Adjust lines, symbols, legends in "Set" properties.
7. Continue exploring the quark tree for finer tuning.
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/Save'.
10. Select 'File/Print setup' and then 'File/Print' to get a hardcopy. That's it!

3.1 General concepts

3.1.1 Project tree

A plot in Grace is built in a tree-like manner (reflected 1:1 in the Explorer tool) out of "Quarks". Quark is an object that may have its own presentation properties plus a number of children quarks.

3.1.2 Coordinate frames

Some quarks impose a coordinate transform; these are "Project", "Frame", and "Graph". Correspondingly, there are three types of coordinates in Grace: the **viewport**, the **frame**, and the **world coordinates**. A child position is *always* defined in the coordinate frame of its parent, so e.g. points of data sets are defined in the graph's world coordinates.

The frame coordinates go from 0 to 1 in both X and Y dimensions of a Frame.

The viewport coordinates correspond to the image of the plot drawn on the canvas (or printed on, say, PS output page). 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.

The transformation converting the world coordinates into the frame and further into the viewport ones is determined by both the graph type and the axis scaling.

3.1.3 Frames

Frame is a rectangular placeholder for one or more graphs (if there are more than one graph in a frame, such graphs are called **overlaid**), and optional annotating objects (texts, boxes, etc). As well, Frame is responsible for displaying legend box holding legend entries for all of its children graphs.

3.1.4 Graphs

A graph may hold (every element is optional) Axes, Sets, and additional annotative objects.

The graph type can be any of:

- XY Graph

- XY Chart
- Polar Graph
- Fixed Graph
- Pie chart

3.1.5 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:

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

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

3.1.6 Regions

Regions are sections of a 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 covered by region.

Regions are auxiliary objects which are not displayed in print outputs.

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

3.1.7 Project files

A project file contains all information necessary to restore a plot created by Grace. Each plot is represented on a single page (this may change in future), but may have an unlimited number of graphs. You create a project file of your current plot with File/Save, Save as.

3.1.8 Datasets

A dataset is an array of points with X and Y coordinates, up to four optional data values (which, depending on the set presentation type, can be displayed as error bars or like) and one optional character string.

3.1.9 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. Depending on configuration, Grace can also read NetCDF files (see 1 (configuration)).

3.1.10 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 instance: 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.

3.1.11 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).

3.1.12 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), suitable for printing
- 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, depends on extra libraries, see 1 (configuration))
- JPEG (depends on extra libraries, see 1 (configuration))
- PNG (depends on extra libraries, see 1 (configuration))

Grace can also be instructed to launch conversion programs automatically based on file name. As an example you can produce many vector formats using `pstoedit`, or almost any image format using the `netpbm` suite (see also the *FAQ* <FAQ.html>).

3.1.13 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`

3.1.14 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").

3.2 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*

Set autoscale type

-barebones

Turn off all toolbars

-block *block .data*

Assume data file is block data

-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

-hardcopy

No interactive session, just print and quit

-hdevice *hardcopy device name*

Set default hardcopy device

-hdevice-options *option string*

Set options for default hardcopy device

-install

Install private colormap

-maxpath *length*

Set the maximal drawing path length

-noask

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

-noinstall

Don't use private colormap

-noprint

In batch mode, do not print

-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

-printfile *file*

Save print output to file

-results *results file*

Write results of some data manipulations to results_file

-saveall *save file*

Save all graphs to save_file

-seed *seed value*

Integer seed for random number generator

-settype *xy|xydx|...*

Set the type of the next data file

-timer *delay*

Set allowed time slice for real time inputs to delay ms

-version

Show the program version

-wd *directory*

Set the working directory

-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

Invocation string of an HTML viewer for on-line browsing of help documents. Must include "%s" to be replaced with an URL string.

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 Default template

Whenever a new project is started, Grace loads the default template, `templates/Default.xgr`. The file is searched for in the magic path (see 3.1.13 (magic path)); once found, the rest of the path is ignored.

3.3.3 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.toolBar`
Enables button toolbar
- `XMgrace.statusBar`
Enables status bar
- `XMgrace.locatorBar`
Enables locator bar
- `XMgrace.instantUpdate`
Enables instant update feature of Explorer

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 pre-defined 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.

- hide
- show
- delete
- bring to front
- move up
- move down
- send to back
- selector operations
 - select all
 - unselect all
 - invert selection
 - update list

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

Frame selector An extension of the 4.1.2 (list selector). The additional operations that can be performed on frames through the frame selector's popup menu are:

- create new

Graph selector An extension of the 4.1.2 (list selector). The additional operations that can be performed on graphs through the graph selector's popup menu are:

- create new
- focus to selected

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

Set selector An extension of the 4.1.2 (list selector). The additional operations that can be performed on sets through the set selector's popup menu are:

- edit
 - in spreadsheet (see 4.1.4 (Spreadsheet data set editor))
 - in text editor
- create new
 - by formula
 - in spreadsheet (see 4.1.4 (Spreadsheet data set editor))
 - in text editor
 - from block data

Double-clicking on a list entry will open the spreadsheet editor (see 4.1.4 (Spreadsheet data set editor)) on the set data.

4.1.3 Pen chooser

Pen chooser lets you select a pen (a color+pattern combination) for drawing a line or filling in an object. When you click on the button, a dialog with color and pattern selectors will appear. When applied, the pen chooser will itself be painted by the pen. A faster way to alter pen's color is to press the right mouse button on it, which will invoke a color-change menu.

4.1.4 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.

4.2 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>U: Refresh hotlinks
- Ctrl <Key>Z: Zoom

Clicks and double clicks Mouse wheel (if you got one) scrolls the canvas vertically. With <Ctrl> pressed, the scroll direction is horizontal. Alternatively, simply drag the canvas with the mouse.

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 an object will invoke Explorer with the relevant Quark selected.

Dragging an object with mouse is possible while <Ctrl> is pressed.

Pressing 3-rd mouse button on an object pops up a context-sensitive menu.

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; most of them operate on the current graph.

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 3.1.7 (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 Explorer

This is the most complex dialog. It allows to tune presentation properties of any plot element.

The left pane of the dialog shows the current project's tree structure. By clicking on a node, a "plugin" dialog corresponding to the quark selected appears in the right pane. A few quarks of the same type can be selected; in this case, changes will be applied to all of them.

Project properties The project properties dialog lets you set page dimensions and the background color.

Frame properties The frame properties dialog governs location of the frame rectangle and legend properties.

Graph properties The main tab includes the properties you will need more often, in particular, properties related to the graph's coordinate frame. For each axis, you can define the axis range and a scale types: linear, logarithmic, or reciprocal, and you can invert the axis (which normally increases from left to right and from bottom to top).

The "Locator" tab allows you to customize the display of the locator, mainly its type and the format and precision of the display.

Set properties 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* given set) is entered in the set properties popup, this is not sufficient to display it. Displaying *all* legends is a frame-level decision, so the toggle is in the main tab of the 4.4.1 (frame properties) plugin dialog.

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.

Axis properties 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.

Annotated texts Not written yet...

Drawing objects Not written yet...

Regions This dialog is incomplete as of yet.

4.4.2 Arrange frames

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

The frame selector at the top allows one to select a number of frames the arrangement will operate on. If the number of selected frames isn't equal to **M** times **N**, new frames may be created or extra frames killed if needed. These options are controlled by the respective checkboxes below the frame selector.

The order in which the matrix is filled in with the frames can be selected (first horizontally then vertically or vice versa, with either of them inverted).

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.

If you don't want the regular layout this arrangement gives you, you can change it afterwards.

4.4.3 Autoscale graphs

Using this entry, you can autoscale a graph 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.

4.4.4 Preferences

The preferences popup allows you to set miscellaneous properties of your Grace session, such as GUI behavior, cursor type, etc.

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. 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.

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:

$$y = y - 0.653 * \sin (x \text{ deg}) + s2.y$$

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 you may want to use FFTW *wisdom* files. Then, you should set several 3.3.1 (environment variables) to name them.

Running properties The running properties 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.

Differences/derivatives The differences popup is used to compute approximations of the first derivative of a function with finite differences. a period to data of the preceding period (namely $y[i] - y[i + \text{period}]$). 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.

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 a_0, a_1, \dots, a_9 to denote the fit parameters (as an example you can say $y = a_0 * \cos(a_1 * x + a_2)$). 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 (notice that "dY" in the preset "1/dY²" one actually refers to the third column of the data set; use the "Custom" function if this doesn't make sense for your data set), 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 $\text{abs}(C(0)) = 1$.

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

Sample points This popup provides a way to sample a set by selecting 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.

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 3.1.5 (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 View menu

4.6.1 Show/hide

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

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

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

4.6.2 Page zoom

Smaller Zoom out page.

Larger Zoom in page.

Original size Reset page zoom.

4.6.3 Page rendering

Set the rendering properties of the display device (font anti-aliasing and color transformations).

4.6.4 Redraw

This menu item triggers a redrawing of the canvas.

4.6.5 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 4.7.2 (Console) tool.

4.7 Tools menu

4.7.1 Font tool

Not written yet...

4.7.2 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.

The dialog also provides an interface to the command driven version of Grace. Here, commands can be typed in 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 using the keyboard arrow buttons. Depending on the state of the "Options/Auto redraw" and "Options/Auto update" checkboxes, each command entered

will cause an immediate resync of the canvas and/or the GUI, respectively. For a reference on the Grace command interpreter, see 5 (Command interpreter).

4.7.3 Dataset statistics

Using the dataset statistics popup, you can view statistical properties of datasets (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.

4.8 Help menu

4.8.1 On context

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

4.8.2 User's guide

Browse the Grace user's guide.

4.8.3 FAQ

Frequently Asked Questions with answers.

4.8.4 Changes

The list of changes during the Grace development.

4.8.5 Examples

The whole tree of submenus each loading a sample plot.

4.8.6 Web support

Home page Open the official Grace home page.

Forums Go to Grace forums, where you can participate in discussions.

Report an issue Use this to send your suggestions or bug reports.

4.8.7 License terms

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

4.8.8 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, $0.1 + 1.9$, $\text{PI}/\text{asin}(1)$
nexpr	Non-negative iexpr	$2 - 1$
indx	Non-negative iexpr	
qstr	Quoted string	"a string"

Table 4: Basic types

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 5: Data column selections

Not finished yet...

5.3 Variables

Variable	Description
datacolumn	data column of current set
set.datacolumn	data column of set
vvar	user-defined array

Table 6: Vector 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.

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

Table 7: Scalar variables

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
%	modulus
^	raising to power

Table 8: Arithmetic operators

Operator	Description
AND or &&	logical AND
OR or	logical OR
NOT or !	logical NOT

Table 9: 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 10: Comparison operators

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 11: 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
gptr(a, b, x)	gamma distribution function
gptrc(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 12: Statistical functions

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	n0210031.pfb
Times-Italic	Times-Italic	n0210231.pfb
Times-Bold	Times-Bold	n0210041.pfb
Times-BoldItalic	Times-BoldItalic	n0210241.pfb
Helvetica	Helvetica	n0190031.pfb
Helvetica-Oblique	Helvetica-Oblique	n0190231.pfb
Helvetica-Bold	Helvetica-Bold	n0190041.pfb

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
dawson(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 13: Special math functions

Helvetica-BoldOblique	Helvetica-BoldOblique	n0190241.pfb
Courier	Courier	n0220031.pfb
Courier-Oblique	Courier-Oblique	n0220231.pfb
Courier-Bold	Courier-Bold	n0220041.pfb
Courier-BoldOblique	Courier-BoldOblique	n0220241.pfb
Symbol	Symbol	s0500001.pfb
ZapfDingbats	ZapfDingbats	d0500001.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

6.3 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 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
<code>f_of_i</code>	a function of 1 int variable
<code>f_of_d</code>	a function of 1 double variable
<code>f_of_nn</code>	a function of 2 int parameters
<code>f_of_nd</code>	a function of 1 int parameter and 1 double variable
<code>f_of_dd</code>	a function of 2 double variables
<code>f_of_nnd</code>	a function of 2 int parameters and 1 double variable
<code>f_of_ppd</code>	a function of 2 double parameters and 1 double variable
<code>f_of_pppd</code>	a function of 3 double parameters and 1 double variable

Table 14: 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 $y = \text{pow}(x, 2)$ 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 14 (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 slashes 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.

Example:

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

prints roughly

$$F_{\epsilon}(e) = \sin(e) \cdot e^{-x} \cdot \cos(e)$$

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/xkb` settings, or with the help of the font tool ("Tools/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, EPS and all raster drivers (PNM/JPEG/PNG) are full-featured devices
- PDF driver:
 - bitmapped text strings are not transparent
 - kerning is not implemented
- MIF driver: the driver is a brand new one and still in beta test
 - some of patterned fills not implemented
 - bitmapped text strings not implemented
 - kerning is not implemented
- SVG driver: the driver is a brand new one and still in beta test
 - patterned fills not implemented
 - bitmapped text strings not implemented
 - kerning is not implemented

7.3 Device-specific settings

Some of the output devices accept several configuration options. A few options can be passed in one command, separated by commas.

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

Table 15: Control codes.

Command	Description
level1	use only PS Level 1 subset of commands
level2	use also PS Level 2 commands if needed
colorspace:grayscale	set grayscale colorspace
colorspace:rgb	set RGB colorspace
colorspace:cmyk	set CMYK colorspace (PS Level2 only)
docdata:7bit	the document data is 7bit clean
docdata:8bit	the document data is 8bit clean
docdata:binary	the document data may be binary
embedfonts:none	don't embed fonts
embedfonts:but13	embed all but the 13 standard fonts
embedfonts:but35	embed all but the 35 standard fonts
embedfonts:all	embed all fonts
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 16: PostScript driver options

Command	Description
level1	use only PS Level 1 subset of commands
level2	use also PS Level 2 commands if needed
colorspace:grayscale	set grayscale colorspace
colorspace:rgb	set RGB colorspace
colorspace:cmyk	set CMYK colorspace (PS Level2 only)
docdata:7bit	the document data is 7bit clean
docdata:8bit	the document data is 8bit clean
docdata:binary	the document data may be binary
embedfonts:none	don't embed fonts
embedfonts:but13	embed all but the 13 standard fonts
embedfonts:but35	embed all but the 35 standard fonts
embedfonts:all	embed all fonts

Table 17: EPS driver options

Command	Description
compatibility:PDF-1.3	set compatibility mode to PDF-1.3 (Acrobat4)
compatibility:PDF-1.4	set compatibility mode to PDF-1.4 (Acrobat5)
compatibility:PDF-1.5	set compatibility mode to PDF-1.5 (Acrobat6)
compression:value	set compression level (0 - 9)
fpprecision:value	set floating point precision (4 - 6)

Table 18: 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 19: 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 20: 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 21: 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 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 be modified for a single session using the 4.4.4 (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 3.3.2 (Default template)).

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

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

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

You can also provide a hint about the format ("ISO8601", "european", "us") using the -datehint command line flag or the 4.4.4 (Edit->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 4.4.4 (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 Universal Time (UTC). If you find yourself in a situation where 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. You can use the `"grconvert"` utility that comes with Grace to convert such files to Xmgr-4.1.2-style projects.
3. 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.
4. 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.

5. 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.**
6. System (shell) variables GR_* renamed to GRACE_*
7. Smith plots don't work now. They'll be put back soon.