Gwyddion user guide

Petr Klapetek, David Nečas, and Christopher Anderson

This guide is a work in progress.

Contents

Chapter 1

Introduction

The latest version of this guide is available on-line at http://gwyddion.net/documentation/user-guide-en/. This guide is available in several languages, namely English, French and Russian. All language versions are listed on-line at http://gwyddion.net/-documentation/.

1.1 Motivation

Gwyddion is a modular program for SPM data analysis. Primarily it is supposed to be used for analysis of height fields obtained by means of scanning probe microscopy techniques (AFM, MFM, STM, NSOM), but generally it can be used for any other height field analysis or image analysis. Gwyddion is Free Software (and Open Source Software), covered by GNU General Public License (GNU GPL).

The main idea behind Gwyddion development is to provide modular program for 2D data analysis that could be easily extended by modules and plug-ins with no need of core recompilation. Moreover, the status of free software enables to provide source codes to developers and users, which makes the further program improvement easier.

Gwyddion can be currently used with Linux/Unix (including Mac OS X) and Microsoft Windows operating systems. Both families of systems can be used also for developement. For graphical interface, Gtk+ widget toolkit is used, therefore it can be basically ported on any system that is supported by Gtk+.

Gwyddion core development is currently funded by Czech Metrology Institute. The project started as a part of the Nanomet initiative (covered by Euromet) in August, 2004. It is supposed that more persons and institutions will participate on development. Project is open for anyone. Welcome...

1.2 Licensing

Gwyddion is covered by GNU General Public License (GNU GPL). The full license text is also included as file `COPYING` in the source distribution (MS Windows installers contain it as file `COPYING.wri`). In brief, this license means that:

- You can freely use the program. You can freely make copies, modify and distribute them. You can download the program and its source code from Gwyddion web pages and modify it as you want.
- If you decide to distribute it, the modified code is still covered by the same license. In particular, you have to offer the source code too.
- The same holds for extensions, e.g. if you write an import module for a new file type or a new data analysis function it has to be licensed under GNU GPL (if you distribute it).

  However, it is also possible to execute third-party programs from Gwyddion and these do not necessarily have to be distributed under the same license – if they are not derived works of Gwyddion (which, admittedly, is not always easy to determine).

The main reasons, why the program is covered by this kind of license are here: first of all, this licensing policy enables us to make modular program that can be easily developed by many persons from different institutions. Second, this license protects the rights of developers that their code, here given to public, cannot be copied and used for closed proprietary products.

Chapter 2

Installation

Gwyddion source code and binaries can be downloaded from the download web page of the project, or alternatively from raw SourceForge.net download page. The installation varies depending on the operating system and the various installation methods will be described in the following sections.

Gwyddion needs or can utilise various software libraries, described in section Build Dependencies. If you install binary packages you usually do not need to concern yourself with the required components as the packager has taken care of it and ensured that all are present. However, it is important if you compile Gwyddion from source code.

To play with Gwyddion you might also want to download the sample Gwyddion files. They are in native Gwyddion format and represent typical AFM data.

2.1 Linux/Unix Packages

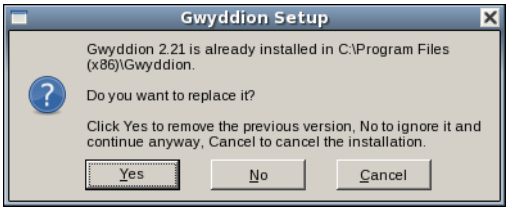
Some GNU/Linux and Unix systems provide binary packages of Gwyddion. The download page of the project also tracks known packages and packaging efforts. For instance, Debian, Ubuntu, Gentoo, openSuSE or FreeBSD offer Gwyddion packages. If your operating system provides such a package and it is recent enough, install it using the standard means of the operating system. Otherwise proceed with compilation from source code.

On Linux distributions using the RPM Package Manager, such as Fedora, openSuSE or Mandriva, you can also build a package yourself from the source code, as described below.

2.2 MS Windows Packages

Note The packaging of MS Windows executables has changed substantially in version 2.23. So, if you upgrade from a pre-2.23 version to version 2.23 or newer, please read the description of the changes.

If you have already installed Gwyddion the installer asks if you want to replace the previous version.



The question what to do with the previous Gwyddion version.

You have three options:

Yes (replace) The already installed version will be replaced. This is the normal upgrade method.

No (keep) The already installed version will be ignored and the installation will proceed as a fresh one. Generally, this is a bad idea as both versions will share settings and registry keys and if you uninstall one the other will be affected. In some cases, though, you can find this option useful.

Cancel The installation will be aborted and the old version will be kept untouched.

In the following steps the installer reminds you of the software components included in the package and their licenses (that are all Free Software), lets you change the installation directory and offers a choice of languages to use for the user interface.

Uninstallation

If you want to uninstall Gwyddion go to *Start* → *Control Panel* → *Add or Remove Programs* and choose Gwyddion. Note that this is valid for Windows XP. The path to the *Add/Remove* window may be slightly different on other Windows OS.

Registry keys

The installer creates the following useful keys under `HKEY_LOCAL_MACHINE\Software\Gwyddion\2.0`:

`InstallDir` Installation directory, e.g. `C:\Program Files\Gwyddion`. Reading this key can be useful for determining where to install extensions.

`Version` Full Gwyddion version as a string.

`Locale` Language of Gwyddion user interface chosen during the installation (more precisely, a locale specification that, among other things, defines the language). You can modify it using **regedit** to choose another language as described below.

The list of available languages and corresponding `Locale` values include:

`Locale`	**Language**
`cs_CZ.UTF-8`	Czech (Czech Republic)
`de_DE.UTF-8`	German (Germany)
`en_US.UTF-8`	English (Unites States)
`fr_FR.UTF-8`	French (France)
`it_IT.UTF-8`	Italian (Italy)
`ru_RU.UTF-8`	Russian (Russia)

Missing features

Gwyddion has a large number of optional features that depend on third party libraries. The MS Windows packages contain most of them but a few are not included at present:

- OpenEXR HDR image import and export.
- Flexible Image Transport System (FITS) file import.
- Pygwy support in the 64bit packages (it is supported only in the 32bit packages).

Enabling pygwy

The Python scripting interface, pygwy, is included in the installer, however, you need to install Python and PyGTK2 separately to use Python scripting. This can be done either prior to Gwyddion installation or any time later. If Python and PyGTK2 is not present pygwy simply does not register itself upon Gwyddion startup.

MS Windows Python installer can be obtained at http://python.org/download/releases/. Since pygwy requires Python 2 install the latest Python 2.7 version, which will probably be `python-2.7.7.msi`.

Three packages are required for PyGTK2: PyGTK, PyCairo and PyGObject. Follow the corresponding download links for these modules at http://www.pygtk.org/downloads.html to obtain the installers `pygobject-2.28.3.win32-py2.7.msi`, `pycairo-1.8.10.win32-py2.7.msi`, and `pygtk-2.24.0.win32-py2.7.msi` or possibly newer versions (if available).

Success has also been reported with the all-in-one installer `pygtk-all-in-one-2.24.2.win32-py2.7.msi` that contains everything. However, using the all-in-one installer means entire Gtk+ will be installed twice (into different locations). Which bits of which installation will be used in pygwy is difficult to tell. Hence this method is not recommended.

2.3 Build Dependencies

The following table lists packages required to build Gwyddion from source code. If your operating system has separate development packages for libraries you need them too. The table does not include common software compilation prerequisites like the C compiler or the **make** utility. Operating system specifics are described in following sections dedicated to building on particular operating systems.

Minimum required versions are listed for some packages. If no specific version is listed, the minimum version is so old that it did not seem useful to determine it exactly. Specific environments may be listed in the Dependency column, meaning the package is useful only in this environment.

Package	Version	Dependency	Required for, Notes
pkg-config	0.16	**Required**	Tracking the locations and compiler and linker flags of the various packages.
GTK+ 2	2.8.0	**Required**	Gwyddion user interface. This entry implies the dependencies of GTK+ itself, such as GLib, Gdk-Pixbuf, Pango or Cairo. Version at least 2.12 is recommended as it enables previews in the file open dialog.
GLib	2.14.0	**Required**	Everything. GLib is a base library also required by GTK+, but Gwyddion needs a slightly newer version than strictly required by GTK+ 2.8.
Pango	1.10	**Required**	All text rendering. Gwyddion needs a slightly newer version than strictly required by GTK+ 2.8. This entry implies pangocairo, which is an optional component of Pango and in principle can be disabled. However, it is normally included in Pango packages.
Cairo	1.2	**Required**	All drawing within GTK+. Gwyddion needs a slightly newer version than strictly required by GTK+ 2.8. Version at least 1.6 is recommended.
GtkGLExt	1.0	Optional	OpenGL 3D data views. This entry implies the dependencies of GtkGLExt itself, such as the platform OpenGL libraries and headers.
FFTW3	3.0 (3.1)	Optional	Speedup of various integral transforms, power spectrum and correlation operations. Version 3.0 is sufficient for 32bit systems, 3.1 is required for 64bit systems.
libunique	1.0	Optional	Remote control based on D-BUS or whatever technology is currently in.
PyGTK2	2.10	Optional	Pygwy, the Gwyddion Python scripting interface. You need PyGTK2 including the compile-time parts, i.e. codegen, to build pygwy.
GtkSourceView 2		Optional	Syntax highlighting in the Python scripting console.
zlib		Optional	Import of SPML data files and import of gzip-compressed data from other file formats (Createc, NRRD, RHK SM4 PRM metadata).
minizip		Optional	Import of APE DAX, NanoObserver, NanoScanTech and OpenGPS data files.
bzip2		Optional	Import of bzip2-compressed data from NRRD.
LibXML2		Optional	Import of SPML and APE DAX data files.
libpng		Optional	Export of height fields to 16bit greyscale PNG images and import from 16bit PNG images. For common 8bit images, you just need PNG support in Gdk-Pixbuf.
OpenEXR		Optional	Import and export of OpenEXR HDR images.
C++ compiler		Optional	Import and export of OpenEXR HDR images and import of other high-depth images.
cfitsio		Optional	Import of Flexible Image Transport System (FITS) files.
desktop-file-utils		Optional, Unix	Basic desktop integration to Freedesktop-conforming environments, such as file associations and installation of Gwyddion to the desktop environments menus.
gtk-mac-integration		Optional, OS X	OS X platform integration such as the global menu.
GConf2 2		Obsolete, Unix	Better GNOME 2 and XFce integration (up to XFce 4.6), in particular automatic generation of SPM file thumbnails in Nautilus and Thunar. Newer versions of these desktop environments use different mechanisms.

Package	Version	Dependency	Required for, Notes
libXmu		Obsolete, X11	Remote control on X11. This is a standard X Window System library and everyone having X probably has its runtime files. However, since the modularisation of X in Xorg 7.0, it is distributed separately and therefore you might not have its development files installed.

2.4 Compilation on Linux/Unix

Gwyddion Unix build system is based on GNU autotools (autoconf, automake, libtool), like most of current Unix Free and Open Source Software. If you have ever compiled software from source code, you very likely met autotools and already know how to proceed. This section shall describe the compilation procedure in enough detail even for the uninitiated though. File `INSTALL` in the top-level directory of the source tarball contains generic GNU autotools installation instructions.

Quick Instructions

If you know the drill:

```
tar -jxvf gwyddion-2.26.tar.xz
cd gwyddion-2.26
./configure
make install
```

Source Unpacking

Unpack the source code tarball with

```
tar -Jxvf gwyddion-2.26.tar.xz
```

replacing 2.26 with the actual version number. It will create directory `gwyddion-2.26` (again, with the actual version number in place of 2.26), **cd** to this directory. All other compilation actions will take place there.

If your operating system does not come with xz you might want to download `gwyddion-2.26.tar.gz` (compressed with gzip) instead and unpack it with

```
tar -zxvf gwyddion-2.26.tar.gz
```

However, modern Unix and Unix-like systems come with both xz and gzip so, the considerably smaller `gwyddion-2.26.tar.xz` should be normally the better choice.

Configuration

Run

```
./configure
```

to configure Gwyddion.

The **configure** shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a `Makefile` in each directory of the package, a couple of header `.h` files containing system-dependent definitions and a few other system-dependent auxiliary files. Finally, it creates a shell script **config.status** that you can run in the future to recreate the current configuration, and a file `config.log`. This file contains the details of the detection process and it is helpful to include it in compilation related bug reports. At the end, **configure** also prints a summary of enabled/disabled optional features, including the reasons why features were disabled.

If **configure** reports missing required packages, install these packages and re-run it. The same applies to the case when **configure** passes but you find you have not installed an optional package you want to compile Gwyddion with. It is possible a package is not found or it is misdetected even if you have installed it, namely when it is installed into a non-standard directory. In this case it is necessary to adjust certain environment variables to make **configure** able to find the packages:

PKG_CONFIG_PATH Most packages come with so called pkg-config files (`.pc`) that describe how programs should compile and link with them. **configure** uses information from these files, therefore `PKG_CONFIG_PATH` must be set to list all non-standard directories with relevant pkg-config files. To add for instance a GTK+ installation in `/opt/gnome` and a FFTW3 installation in `$HOME/opt/fftw3` one can do

```
PKG_CONFIG_PATH=/opt/gnome/lib/pkgconfig:$HOME/opt/fftw3/lib/pkgconfig
export PKG_CONFIG_PATH
```

PATH, LD_LIBRARY_PATH It may be necessary to adjust these variables to include non-standard directories with executables and libraries of relevant packages, respectively.

CPPFLAGS, LDFLAGS It may be necessary to adjust these variables to include non-standard directories with header files and libraries of packages that do not come with pkg-config files, for example for libTIFF in `/usr/local` one can set:

```
CPPFLAGS=-I/usr/local/include
export CPPFLAGS
LDFLAGS=-L/usr/local/lib
export LDFLAGS
```

Option `--prefix` of **configure** sets the base installation directory. Program components will be installed into its `bin`, `lib`, `share`, etc. subdirectories (that will be created if they do not exist). More detailed control is possible with options specifying particular subdirectories as `--bindir`, `--libdir`. The default prefix is `/usr/local/bin`, to install Gwyddion into your home directory you may want to use for instance

```
./configure --prefix=$HOME/opt/gwyddion
```

If you install Gwyddion for personal use it is recommended to use a similar installation directory as no steps need to be performed as root in this case.

Configuration tweaks

Optional features can be enabled/disabled with options such as `--with-foo`/`--without-foo` or `--enable-foo`/`--disable-foo`. For instance compilation with FFTW3 can be disabled with:

```
./configure --without-fftw3
```

By default all optional features are enabled if their prerequisites are found. A brief summary enabled and disabled optional features is printed near the end of **configure** output.

The complete list of **configure** options and important variables can be obtained with:

```
./configure --help
```

Most of these options control inclusion/exclusion of optional features. Some interesting general options are explained below.

User's tweaks

Gwyddion comes with various desktop integration files defining MIME types, menu entries, file associations, thumbnailers, etc. If you install Gwyddion to a system prefix they usually end up in the correct location. However, if you install it somewhere to your home directory then these files need to be placed elsewhere, namely into certain dot-directories in your home.

This can be requested using `--enable-home-installation` option of **configure**. Note that using this option causes installation of files outside the specified prefix.

Packager's tweaks

If Gwyddion is installed into a staging area for a subsequent packaging it is necessary to disable certain post-installation actions that need to be done on the target system, not while packaging.

Updating of Freedesktop files can be disabled with `--disable-desktop-file-update`. Installation of GConf2 schemas can be disabled with `--disable-schemas-install`. Usually, this does not have to be done explicitly as installations into a staging area use non-empty `DESTDIR` (see installation). If `DESTDIR` is found to be non-empty the build system skips post-installation actions automatically.

Developer's tweaks

If you intend to patch or otherwise modify Gwyddion source code pass option `--enable-maintainer-mode` to **configure** to enable various update and rebuild rules that are not used in plain compilation. Depending on the nature of the modifications, some of the additional tools described in section Subversion Checkout, Development may be necessary.

Compilation

Run

```
make
```

and wait until Gwyddion is compiled. If **configure** finished without errors the compilation should pass too.

If you need to do unusual things to compile the package, please try to figure out how **configure** could detect whether and what to do, and e-mail patches or instructions to the bug-report address so they can be considered for the next release.

Installation

Gwyddion has to be installed to be run, it is not possible to run it uninstalled.

Run

```
make install
```

to install Gwyddion to the target directory. If you install Gwyddion to a system directory you have to become root for running this command. This is the *only* command that you might have to run as root during the installation. For example using **sudo**:

```
sudo make install
```

To install Gwyddion to a staging area, for example for packaging, set **make** `DESTDIR` variable to a prefix that will be prepended to all target directories:

```
make install DESTDIR=/var/tmp/gwyddion-buildroot
```

Do *not* override individual directory variables as `bindir`, `libdir`.

If you do not install to a system directory, e.g. to a subdirectory of your home directory, you may need to adjust the following variables during installation:

- `GCONF_SCHEMA_CONFIG_SOURCE` – location of GConf2 schemas
- `KDE4_MODULE_DIR` – location of KDE4 modules

Also, variable `XDG_DATA_DIRS` might need to be adjusted after installation to get full desktop integration.

If you install Gwyddion into `/usr/local` and get error message that `libgwyapp.so.0` cannot be found your system probably lacks standard library directories in the dynamic linker configuration. Notably, this happens on Ubuntu. Edit file `/etc/ld.so.conf` and add the line

```
/usr/local/lib
```

there.

Running

Running Gwyddion does not normally require any additional setup.

The misfeatures of some desktop environments, however, may render Gwyddion unusable and need to be disabled. The hijacking of program main menu in Unity makes most of Gwyddion menus inaccessible. It can be disabled by by unsetting `UBUNTU_MENUPROXY` while running Gwyddion:

```
UBUNTU_MENUPROXY= gwyddion
```

Deinstallation

Run

```
make uninstall
```

in the directory you previously compiled Gwyddion to remove it. If you have lost the source directory meanwhile you can try to unpack, configure and build it exactly as before and then issue **make uninstall**, although this relies on your ability to reproduce the build process.

RPM Packages

It is possible to build RPM packages on RPM-based GNU/Linux distributions directly from source code tarballs with

```
rpmbuild -tb gwyddion-2.26.tar.xz
```

where 2.26 is to be replaced with the actual version as above. This method was tested mainly on Fedora, openSuSE and Mandriva and the RPM spec file contains some specific provisions for these systems. Specific support for other RPM-based systems can be added on request.

2.5 Mac OS X

Much of the previous generic Unix/Linux installation section applies also to OS X. Therefore this section deals mainly with the specifics of OS X installation, some of the steps listed here are explained in more detail in the generic Unix section.

Beside building everything on your own (good luck), at this time there are two ways to install Gwyddion:

- using MacPorts (formerly Darwinports) and building from a Portfile.
- using Fink and compiling Gwyddion the common Unix way,

Preparation

To install and run Gwyddion you need the Xcode Tools and X (SDK and App) installed. They where located on your CDs/DVDs. The Xcode Tools where located on the first DVD as XcodeTools.mpkg below Xcode Tools, the X11SDK is located as X11SDK.pkg below the Packages Folder within Xcode Tools. X11 is localed as X11User.pkg below `System/Installation/Packages` even on the first DVD. If you have an CD Set the Discs may differ. The people from MacPorts recommending using the newest version of XCode. For further information look at the MacPorts Install Page. Also you should have some experience using Terminal.app. All the commands in the the rest of this section are to be entered and run in Terminal.app.

See installation dependencies section for an overview of required and optional packages to install prior to Gwyddion installation. The following table summarises how they are called in the two software collections:

Package	Fink	MacPorts
Gtk+	gtk+2	gtk2
GtkGLExt	gtkglext1	gtkglext
FFTW3	fftw3	fftw-3
LibXML2	libxml2	libxml2

MacPorts

MacPorts is a Port based System for porting and installing Open Source/GNU software to OS X. It's based on using installation files called "Portfiles" which where describing the steps to compile and install an application. So it's far easy to port software to OS X using MacPorts but every computer has to compile the application. Get and install MacPorts. After you installed MacPorts, run

```
sudo port selfupdate
```

to update MacPorts to the latest version.

Usually installing ports with MacPorts is easy. But since X11 is not the native Desktop for OS X, things went a little worse. So it is recommended to install an alternative X11 before installing Gwyddion. The recommended alternatives are XQuartz on Leopard and the Port xorg-server on Tiger. After installing the suggested X11-System, Gwyddion can be then build and installed simply by

```
sudo port install gwyddion
```

To install xorg-server (Tiger) simply type

```
sudo port install xorg-server
```

this is *needed* for the 3D view on tiger. After everything is done, you will find the StartUp-Icon below `/Applications/MacPorts`.

Fink

Get and install Fink. After you installed Fink run

```
apt-get update
```

to update the database of available packages and install Gwyddion with

```
apt-get install gwyddion
```

To install Gwyddion from source code, for instance if you want to install a development version, you need to install the required packages listed in the above table and then follow the generic Unix installation section instructions.

Running

On MacPorts you simply click on the StartUp-Icon and wait until Gwyddion appears. Using Fink or a self-compiled version you should follow the steps below: Start X11.app and type in Terminal.app

```
export DISPLAY=":0"
```

Then run Gwyddion from the folder it was installed to. This is typically `/usr/local/bin` for Fink. So for example for Fink run:

```
/usr/local/bin/gwyddion
```

You can also configure X11.app to run Gwyddion via: Locate X11.app in your dock, open the menu, choose Applications, choose Customize from the next menu. Here you can choose add and enter the name (gwyddion for example) as *Menu Name* and the complete path to gwyddion (e.g. /usr/local/bin/gwyddion) as *Command*. After this you can choose gwyddion from the X11 menu.

2.6 Cross-Compiling for MS Windows

Cross-compiling Gwyddion for MS Windows under Linux is quite similar to normal Unix compilation with certain additional setup and extra steps. Although the process is quite smooth the initial setup may seem a bit complicated. If, in addition, you are not familiar with the normal Unix compilation you might wish to start with that and attempt cross-compilation once you familiarise yourself with the basic procedure.

These instructions describe compilation under Fedora using its MinGW cross-compilation support as this is what Gwyddion developers use. In general, the instructions work on the current version of Fedora. Compilation on other versions and other RedHat-based distributions (CentOS, Scientific Linux, ...) should be similar and relatively straightforward, possibly with some tweaks. Building on, for instance, openSUSE will require modifications. Reports of success (or failure) of cross-compilation of Gwyddion in other distributions and environments and namely improvements to these instructions are welcome.

Full cross-compilation has the following steps:

- configuration for mingw64/mingw32,
- compilation,
- installation into a staging area,
- creation of an installer using NSIS.

A script is available that automatically performs all the steps, as described below.

Setup

Before the first compilation you must set up the cross-compilation environment. This has to be done only once.

Base MinGW Packages

Run as root:

```
yum install mingw{32,64}-{gcc-c++,gtk2,libxml2,minizip,fftw,gtkglext}
```

to install the necessary mingw32 and mingw64 packages. Several more packages will be installed as dependencies of those explicitly given here. Note that, technically, some of the packages are optional dependencies and you can build a MS Windows installer without them (after some adjustments). Neverhteless the standard installers include these packages and the cross-compilation scripts expect them to be present by default.

Gwyddion.net repository

MinGW versions of a few packages used by Gwyddion are not available in Fedora yet (or any more). Presently the only missing package is gtksourceview2 which is only used by pygwy.

You can build these additional packages using the patches and spec files at http://sourceforge.net/projects/gwyddion/files/-mingw32-cross-compile/, however, it should be much easier to just install them using **yum**. For this, download and install the gwyddion.net repository configuration package. The installation makes available all the additional MinGW packages. After installing it you can run

```
yum install mingw32-gtksourceview2
```

Incidentally, the repository also contains a native x86_64 Gwyddion package that you can install to use Gwyddion on Fedora; and a package with cross-compiled Gwyddion libraries that can be used to cross-compile modules.

Wine

Wine is the MS Windows compatibility layer/emulator for Unix. It is used to run NSIS that creates the executable Gwyddion Windows installer. Wine can also be used to run and test the cross-compiled Gwyddion, as described below.

Run

```
yum install wine
```

to install Wine.

NSIS

Nullsoft scriptable install system (NSIS) is used to create the Gwyddion installer. This is a MS Windows program, therefore, it is installed *under Wine*. A cross-compiled version of NSIS might be available in the distribution but we have found the original more reliable.

Download NSIS from its web page and run

```
wine nsis-2.46-setup.exe
```

replacing 2.46 with the actual version. Version 2.46 of NSIS is the oldest that has been tested.

Python

To compile pygwy you need to install Python into Wine. The steps are the same as if you just want to use pygwy, except that all packages listed in Enabling pygwy need to be installed using **msiexec**:

```
wine msiexec /i python-2.7.3.msi
wine msiexec /i pygobject-2.28.3.win32-py2.7.msi
wine msiexec /i pycairo-1.8.10.win32-py2.7.msi
wine msiexec /i pygtk-2.24.0.win32-py2.7.msi
```

or similarly.

Support scripts

Support scripts and data are available in `mingw32-cross-compile` module in the Gwyddion subversion repository. Run

```
svn checkout http://svn.code.sf.net/p/gwyddion/code/trunk/mingw32-cross-compile
```

to check it out from the repository.

The most important tool you obtain is the **cross-build-32** (or **cross-build-64**) script that automates all the cross-compilation steps. Before you use it for the first time, review file `setup32` that defines where various things are located (or `setup64` for the 64-bit target). The default contents looks as follows:

```
source_dir=$HOME/Projects/Gwyddion/gwyddion-mingw
mingw_prefix=/usr/i686-pc-mingw32/sys-root/mingw
target_prefix=$HOME/opt/gwyddion-mingw32
python_dir=$HOME/.wine/drive_c/Python27
nsis_compiler=C:\\Program\ Files\ \(x86\)\\NSIS\\makensis.exe
```

Variable `source_dir` specifies the location of the unpacked or checked-out Gwyddion source code and it will likely need to be adjusted. Variable `target_prefix` specifies the installation directory (staging area) for the cross-compiled Gwyddion. The default value should be reasonable and you do not need to change it unless you want to. The remaining variables, `mingw32_prefix`, `nsis_compiler` and `python_dir`, specify the location of MinGW files, NSIS compiler and Win32 Python, respectively. They do not need to be changed from the default values under normal circumstances although NSIS can be installed in either `Program Files (x86)` or `Program Files` by default depending on Wine configuration. Note `setup` is read by shell so there must not be any spaces around =.

Compilation

The setup was tedious but it was worth it because the compilation is then extremely simple. Run

```
./cross-build-32
```

in `mingw32-cross-compile` directory to build Win32 insstaller. That's all. If it succeeds an executable Gwyddion Windows installer with bundled GTK+ and everything will be created in `$target_prefix`. Similarly, the Win64 installer is built just with

```
./cross-build-64
```

You can make a coffee meanwhile – or study the `cross-build` script (it is actually quite short and clear).

Note the cross-build scripts run `autogen.sh` but do not clean the source code directory. You may wish to do that manually if you compile Gwyddion repeatedly. Especially if you build for both architectures in the same directory, make sure to run

```
make distclean
```

between the builds to get the source directory back to a well-defined state.

Running under Wine

Compiled Gwyddion can be run under Wine. Assuming the default value of `target_prefix`:

```
wine ~/opt/gwyddion-mingw32/bin/gwyddion.exe
```

To run `gwyddion.exe` the dynamic linker must be able to find all the necessary DLLs. This is ensured in a somewhat crude way by script **copysysfiles** that copies all necessary MinGW files from system to `$target_prefix`. Since **copysysfiles** is executed by **cross-build** you normally do not need to execute it manually.

The second step that might be necessary is setting registry key

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\App Paths\gwyddion. ↩
    exe
```

to point to `gwyddion.exe` and the value `Path` to point to the `bin` subdirectory.

Cross-compilation of standalone modules

Cross-compilation of standalone modules requires only Gwyddion headers and Win32 development libraries. While they can be surely obtained by cross-compiling entire Gwyddion is it not necessary to do so. By compiling only the libraries you can avoid the installation of various rarer Gwyddion dependencies. This can be achieved using the patch `gwyddion-2.22-build-only-libs.patch` available among the build scripts.

But it gets even easier, the MinGW Gwyddion libraries are available as a RPM package `mingw32-gwyddion-libs` in the gwyddion.net repository.

Once you have this package installed you can try cross-compilation of the sample standalone module `threshold-example` that is available in the subversion repository (or as a tarball). See `README` therein for some more details.

2.7 Compiling on MS Windows using MinGW

Although the standard MS Windows executables are created using cross-compilation it is also possible to build Gwyddion on MS Windows using the MinGW port of GNU tools to MS Windows. The standard MS Windows excutables also come with almost all the optional features packaged – the notable exception being Python scripting. Getting all these components work in MS Windows requires additional effort. However, the most likely reason for compiling on MS Windows is to obtain all the necessary files to develop and build standalone Gwyddion modules and for this purpose building all the optional components is not necessary.

The procedure is essentially the same as the normal Unix compilation. Some MinGW-specific remarks follow.

It has been reported that the Gtk+ 2.24.10 bundle can be successfully used. After installing it, set in the MSYS shell

```
PKG_CONFIG=PATH-TO-GTK+/gtk+/bin/pkg-config.exe
```

where `PATH-TO-GTK+` needs to be replaced with the actual Gtk+ installation directory.

To compile only the libraries, it may be useful to use the patch `gwyddion-2.22-build-only-libs.patch` described in the cross-compilation section. In addition, it seems that the MinGW libintl redefines `printf()` to `libintl_printf()` which it, however, does not provide. This leads to link failure of `gwyddion.exe`. This can be 'fixed' by simply removing `include/libintl.h` in the Gtk+ directory.

2.8 Subversion Checkout, Development

Gwyddion uses Subversion version control system for source code revision management. The organisation of the repository is described on project web pages. For example the current head revision of the program itself can be checked out with

```
svn checkout http://svn.code.sf.net/p/gwyddion/code/trunk/gwyddion
```

The repository does not contain any generated files, no matter how exotic tools may be necessary to generate them. Therefore, additional packages are required for building from a fresh checkout. There are also certain platform limitations. The additional tools and packages required for development are essentially the same as for compilation from Subversion checkout. More precisely, to build from a fresh checkout all the additional tools are necessary, whereas development may require only a subset of them or even none, depending on the type and extent of the changes in the source code.

ADDITIONAL DEVELOPMENT BUILD DEPENDENCIES

- GNU autoconf $\geq$ 2.60
- GNU automake $\geq$ 1.11
- GNU libtool $\geq$ 1.4
- Python $\geq$ 2.2
- Perl5
- gtk-doc $\geq$ 1.12
- GNU gettext $\geq$ 0.12, including development stuff
- probably GNU versions of most tools: the compiler, binutils, ...

After a fresh checkout, run **`./autogen.sh`** with any arguments you would give to **configure**. Note it automatically adds options `--enable-maintainer-mode` and `--enable-gtk-doc` to ensure the rules for creation and updates of various files are active. Generally, you should always use `--enable-maintainer-mode` if you intend to change the program source code in a non-trivial way.

On some systems, **autogen.sh** can fail even if you have sufficient versions of autotools installed. These systems do not install general **autoconf** or **automake** commands, only versioned commands such as **autoconf261** or **automake19**. This makes it particularly difficult to find for example "**automake** 1.9 or newer" with no limit on how newer it can be. Therefore, **autogen.sh** does not attempt this at all. You can either create unversioned symbolic links to the versioned commands or run **autogen.sh** as follows: **`AUTOCONF=autoconf261 AUTOHEADER=autoheader261 ./autogen.sh`** You may need to set the following variables: `ACLOCAL`, `AUTOCONF`, `AUTOHEADER`, `AUTOM4TE`, `AUTOMAKE`, `LIBTOOLIZE`. In addition, some operating systems may install **autoconf** macros in a place **aclocal** does not find them by default. This can be fixed by setting variable `ACLOCAL_FLAGS` to give **aclocal** additional search paths: **`ACLOCAL_FLAGS="-I /usr/local/share/aclocal" ./autogen.sh`**

It is often necessary to combine these adjustments. For instance on FreeBSD, where all tools are versioned, one typically invokes (broken to lines for easier reading):

```
AUTOCONF=autoconf261 \
AUTOHEADER=autoheader261 \
AUTOM4TE=autom4te261 \
AUTOMAKE=automake19 \
ACLOCAL=aclocal19 \
ACLOCAL_FLAGS="-I /usr/local/share/aclocal" \
CPPFLAGS=-I/usr/local/include \
LDFLAGS=-L/usr/local/lib \
./autogen.sh --prefix=...
```

If **autogen.sh** passes you can compile the program as usual.

MS Windows

Since the standard method to create MS Windows executables is cross-compilation in Linux the recommended method to develop for MS Windows is also to compile in Linux. This can be done either on a different physical computer using **ssh** or in a virtual machine running on the same computer as the host MS Windows system. In both cases the Gwyddion build directory (and other directories) can be shared between the Linux and MS Windows systems using either Samba or a shared directory mechanism of the virtual machine and the compiled executables thus can be directly tested in MS Windows without having to transfer files back and forth.

Chapter 3

Getting Started

This chapter introduces various basic concepts and terms, such as masks or selections, explains the organization of data in Gwyddion and describes the user interface.

The descriptions are relatively thorough and some of the topics near the end of the chaper, such as raw file import or plug-ins, might be considered advanced and not for everyday use. So, despite the name, it is not necessary to read this entire chapter to be able to work with Gwyddion. The Gwyddion user interface is intuitive and much can be discovered by playing. The clarification of the basic ideas and components provided here will hopefully ease the discoverying.

You can also get help for the current function or window from within Gwyddion. Pressing F1 or clicking on the *Help* button will in most windows show a relevant part of the on-line version of this guide in a web browser. The starting page of guide can be displayed by *Info* → *User Guide*. Of course, this works if you are on-line and Gwyddion finds a suitable web browser. See the settings tweaks if you want or need to tweak the setup, e.g. for an off-line version of the guide.

Tip Command *Info* → *Tip of the Day* displays data processing tips and highlights useful features that you might miss.

3.1 Main Window

The main window, also called toolbox, is one of the two Gwyddion windows that appear after program start (with no files given to open), the other is the data browser. Closing the main window causes Gwyddion to exit.

The toolbox contains the set of Gwyddion menus and from several rows of buttons connected with common functions and tools. The menus group the functions as follows:

File associates commands that are used for file loading and saving. Certain global commands (e. g. *Exit*) are located here too. The history of recently opened files can be browsed with *File* → *Open Recent* → *Document History*.

Edit provides history manipulation commands (*Undo*, *Redo*) and editors of miscellaneous global resources, such as gradients and materials for false color and 3D data representation or the default color used for data masking.

Data Process is built automatically from all data processing modules available in the Gwyddion module directory (depending on the operating system). This menu together with *Tools* panel of buttons contain most of the commands you will need at analyzing your SPM data. A subset of these functions is also available in *Data Process* button panel. These buttons serve as shortcuts to commonly used functions from *Data Process* menu. All functions accessible from *Data Process* button panel can be found in the menu too.

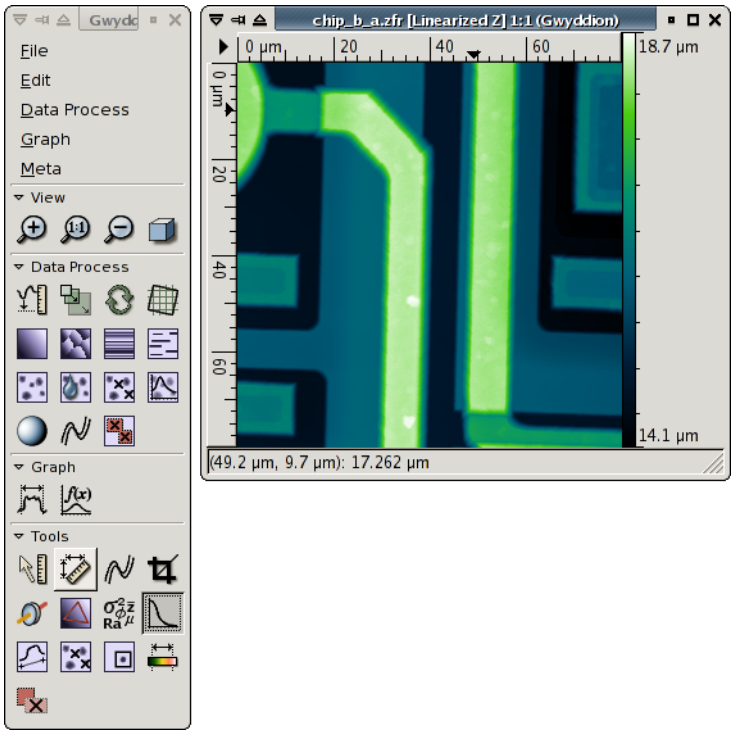
Graph is similar to *Data Process*, except it consists of graph functions. Graph processing includes function fitting, exporting graph data etc. Button panel *Graph* again contains a subset of the commonly used functions from *Graph* menu.

Volume is similar to *Data Process*, except it consists of volume data functions.

Info contains commands that provide various auxiliary information about Gwyddion, such as program version or the list of loaded modules and their functions.

Finally, you can find some rows of buttons in the main window. Buttons in *View* panel offer zooming functions (that are often more easily invoked by keyboard shortcuts or just rezising the data window) and 3D data display. Panels *Data Process* and *Graph* contain selected functions from *Data Process* and *Graph* menus as described above.

Panel *Tools* contains tools, i.e. functions that directly work with selections on data windows. These functions are accessible only from this button panel.



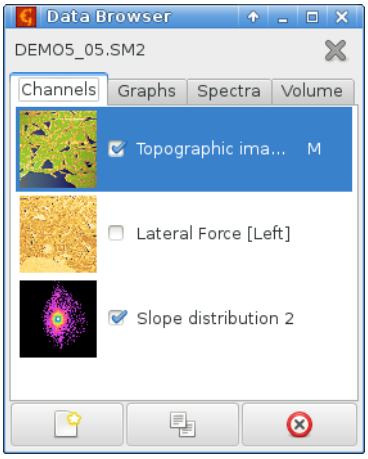
Main window and a data window showing microchip surface (Gwyddion sample file `chip.gwy`).

3.2 Data Browser

Data browser is a window that displays the structure of currently focused file. It shows the content as represented in Gwyddion which may differ somewhat from the organization in the original software.

Gwyddion supports an arbitrary number of two-dimensional data fields per file. Depending on the context, they are also often called channels or height fields in this guide. The dimensions of channels in one file may differ and also the physical dimensions and values can be arbitrary physical quantities.

In addition, one-dimensional data, represented as graphs, and single-point spectra can be present in the same file. The data browser is a tool for browsing and managing all the available data in the file.



Data browser displaying several channels.

Controlling the Browser

Since the data browser always displays the structure of currently focused file, its contents change as you switch between different windows, possibly showing data from different files. There is no difference between native Gwyddion files (`.gwy`) and other files. Once a file is loaded its structure is shown as if it was a Gwyddion file.

The data browser has three tabs, one for each type of data that can be present in the file:

- *Channels*
- *Graphs*
- *Spectra*
- *Volume*

Each list shows names of the data objects and some additional properties that depend on the specific data type. The names can be edited after double-clicking on them.

Individual channels, graphs, spectra or volume data can be deleted, duplicated or extracted to new Gwyddion native file using the buttons at the bottom of the browser. It is also possible to copy them to another file by dragging a data browser row to any window belonging to the target file. To rename a data item, select it and press Enter or triple-click on it with the mouse.

The close button in the top right corner of the data browser closes the current file, discarding all unsaved changes. A file is also closed when all windows displaying data from this file are closed.

If the data browser is closed it can be recalled using the *Info* → *Show Data Browser* command.

Channels

The channel list shows channel thumbnails, check-boxes controlling whether the channel is visible (i.e. displayed in a window) and channel names. Right to the name the presence of presentation, mask or calibration is indicated by the following letters:

- *M* – mask
- *P* – presentation
- *C* – calibration

Graphs

The graph list shows check-boxes controlling whether the graph is visible and graph names. Right to the name the number of curves in the graph is displayed.

Spectra

The spectrum list shows the spectra name and the number of points in the set. Since single-point spectra are displayed and operated on only in connection with a two-dimensional data using the spectroscopy tool, there is no check-box controlling the visibility.

Volume

The volume data list shows the data name and the number of levels in the z direction, i.e. perpendicular to the section displayed in the window.

3.3 Managing Files

Gwyddion uses its custom data format (`.gwy`) to store data. This format has the following important advantages:

- Capability to store the complete state of the individual data, including masks, selections and other properties.
- Arbitrary number of channels, graphs and spectrum sets, with arbitrary dimensions and units of both dimensions and values.
- Double-precision representation of all data, preventing information loss due to rounding.

Therefore, we recommend to use this format for saving of processed files.

Other data file formats are handled with appropriate file loading and saving modules. Beside a large number of file formats used in scanning probe microscopy, graphical file types (PNG, JPEG, TIFF, TARGA) and raw binary and text data can be imported too. If your SPM data format is not supported by Gwyddion yet or it is loaded incorrectly, you are encouraged to write an import module (if you can program) or contact the maintainers to help them improve the support.

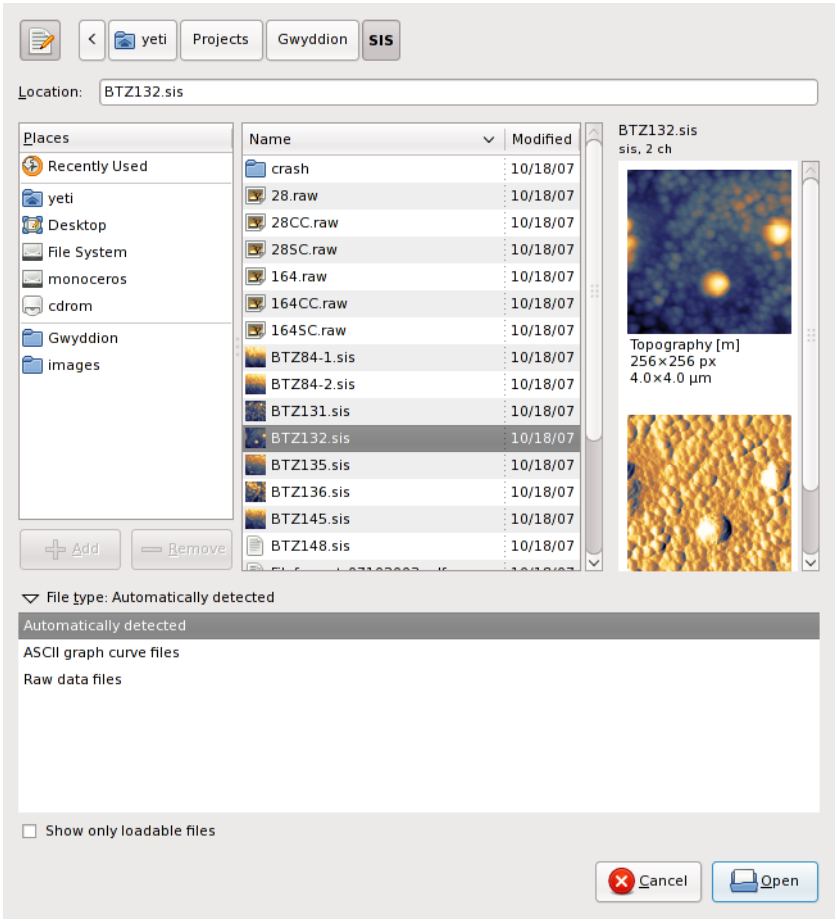
The list of all supported file formats can be found in chapter Summaries and Tables.

File Loading

Files are opened using *File* → *Open*. The file type is detected automatically, based solely on the file content. Since the same extensions such as `.img`, `.afm` or `.dat` are used by many different SPM file types this approach is superior to relying on file extensions.

The only exception is the import of various raw data, either two-dimensional or graph, that must be chosen explicitly in the file open dialog. See sections Raw Data File Import for details of import of raw data and manual extraction of data from unsupported formats and Specific Data Import for import of XYZ data, pixmap image data and graph data.

The list of files in the file open dialog can be limited to only files Gwyddion recognizes as loadable by enabling the *Show only loadable files* option. The file type label then indicates the filtering by appending *(filtered)* to the end. This can be often convenient, on the other hand it can slow down listing of directories with many files.



File open dialog with expanded file type options and channel preview. The small text above the preview shows the module used to load the file (sis) and the number of channels (ch), graphs (gr) and single-point spectra (sps) in the file.

File Merging

File merging, performed by *File* → *Merge*, is similar to normal file loading, except that the selected file (or files) is merged into the current open file. In other words, channels, graphs and spectra, together with all their settings and properties are added to those already present in the current file.

File Saving

Much of the previous paragraphs applies to file saving too. One of the main differences is the reliability of automatic file type determination. While loading can and does examine the file contents, saving depends on file name and extension. Combined with the large number of different file types using the same extension such as `.img`, `.afm` or `.dat` it leads to ambiguities. Select the file type explicitly before saving if you are unsure.

Since the only file type able to fully represent Gwyddion data structures is its native data format, saving to a `.gwy` file is the only proper saving. Saving to other file formats essentially consists of exporting of a limited subset of the data, typically only the active channel (without masks and presentations). Therefore it does *not* change the file name of the current file to the just saved file name.

File → *Save as...* can also be used to export channels to image formats. Just enter `foo.png` as the file name to export a PNG image the current channel, similarly for other formats.

Document History

The history of recently opened files can be accessed with *File* → *Open Recent*. The submenu contains the last 10 recently used files for quick recalling, an extensive recent file history is accessed with the last item *Document History*.

Document history lists the files sorted by the last access time (the most recently accessed at the top), with previews and some additional information about a selected channel. The function of the bottom row of buttons is following:

Prune Removes history entries of files that have been deleted or are no longer accessible for other reasons.

Close Closes the document history window.

Open Opens the selected file. This can be also achieved by activating the selected row, either by double-clicking or with the keyboard.

The history can be searched/filtered by file name using the filter controls above the buttons. The filter is activated by pressing **Enter** in the filter pattern entry. To display all history entries, clear the entry and activate it. The filter pattern is interpreted in two ways:

- If the pattern contains wildcards, i.e. `*` or `?`, it is interpreted as file glob. This means `?` represents a single arbitrary character, `*` represents an arbitrary sequence of zero or more characters, and the file name has to precisely match the pattern. Note directory separators (`/` or `\`) are not treated specially, therefore in the pattern `*.sis` the initial `*` matches all leading directory components. The pattern syntax is described in GPatternSpec documentation.
- If the pattern does not contain any wildcards, it is directly searched as a part of the file name.

Search case sensitivity, controlled by option *Case sensitive*, is useful mainly on systems distinguishing letter case in file names, such as Unix. On systems that do not distinguish the case themselves it is recommended to keep the setting on case insensitive.

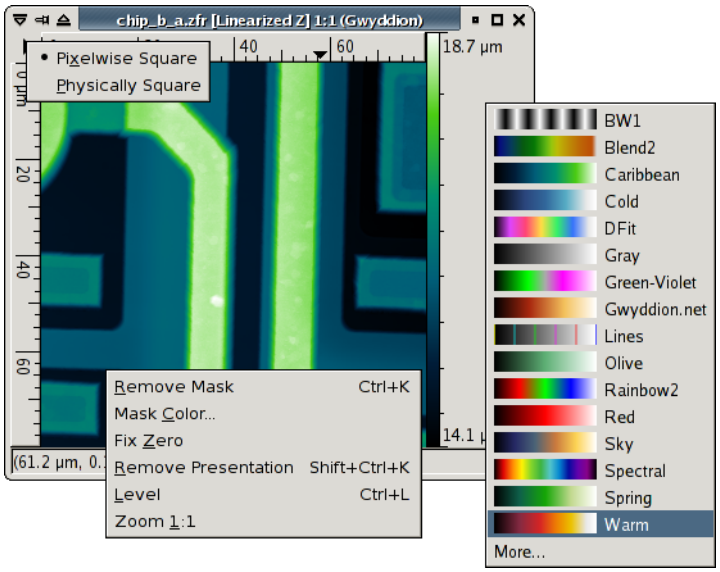
3.4 Data Window

Two-dimensional data are presented in so called data windows. It is the main widget used for working with Gwyddion. The data are presented as a field of false colors corresponding to heights. Color axis that represents mapping of colors to real height values is on the right side of the data window.

The False color palette used to represent height data can be changed by clicking on the color axis with right mouse button (i.e. invoking context menu) and selecting a palette from the list. Most frequently used palettes are available directly in the context menu; however you can reach much more of the possible palettes using the *More* menu entry. Moreover, you can use the color gradient editor to create your own palettes and select which palettes should be displayed in the short list.

There is a context menu available also for the data area. This menu consists of basic data and presentation operations. To reach all the possible operations use *Data process...* menu at the Gwyddion main window, or use some of the tools available at the *Tools* set of buttons at the Gwyddion main window.

The arrow in the upper left corner brings the aspect ratio switch menu. The data can be displayed either with pixels mapped with 1:1 aspect ratio to screen pixels (*Pixelwise Square*), or with physical dimensions mapped 1:1 onto the screen (*Physically Square*). For instance a sample of size 1×1 μm scanned with reduced slow axis resolution and having therefore 512×128 pixels, can be displayed either in 512×128 or 512×512 window.

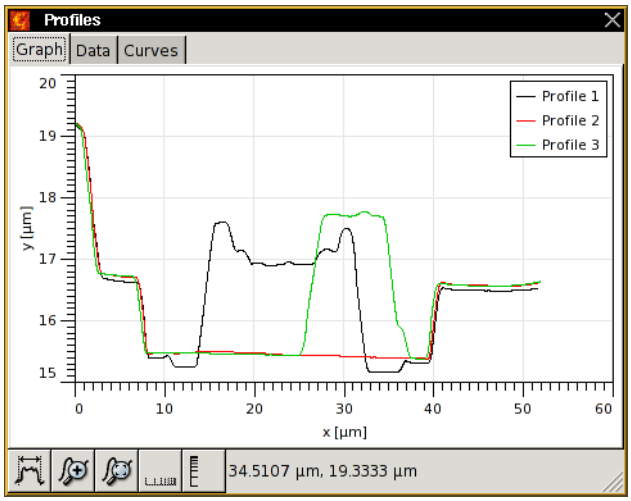


Data window with all three available context menus shown.

3.5 Graph Window

Graph window is used for 1D data processing. They are created by appropriate tools or modules that extract graphs from height field data.

Graph window consist of three tabs: the first two represent graphical and tabular views of the 1D data and the third one shows a list of graph curves. Several tools connected with viewing 1D data are available directly in the graph window toolbar, namely zoom buttons and logarithmic axis buttons. To reach all the possible operations use *Graph* menu in the Gwyddion main window.



A graph window with three profiles.

To edit curve presentation one can either click on the curve in the graph or activate (double-click) the corresponding row in *Curves*. Individual curves can be deleted by selecting them in *Curves* and pressing **Delete**. It is also possible to copy individual curves to other graphs by dragging their curve list rows onto them (provided the graphs are unit-wise compatible).

Clicking on a graph axis brings a dialog with axis properties and graph key properties can be edited after double-clicking on it.

3.6 Tools

Functions from *Data Process* menu and button panel either execute immediately or after asking for parameters and settings in a dialog box. Tools, accessible from *Tools* button panel, work differently. Once selected, they remain active and always follow the current data window until one switches to another tool. In other words one can switch data windows freely while using a tool and

it always shows information from or operates on the current data. Tools also differ by working with selections, e.g. points, lines or rectangles, on data windows. Nevertheless functionally they perform similar tasks as *Data Process* functions – value reading, leveling, statistics, correction, etc.

Tools can be launched only from *Tools* button panel located in the main window. Gwyddion includes these tools:

Read Value Reads values at the position of mouse click.

Distance Measures distances – similarly to Read value this tool enables user to measure horizontal, vertical and Euclidean distance and angle between points in the data field. In addition it displays the difference of data values between points.

Profile Extracts profiles of the data field and puts them to separate graphs. These graphs can be further processed with commands from the *Graph* menu.

Spectro Views and extracts single point spectroscopy data.

Statistical Quantities Computes basic statistical quantities (RMS, Ra, minimum, maximum, projected and surface area, etc.) from a selection of full data field. It can also calculate them only on the masked area, or even combine these two types of selection of area of interest.

Statistical Functions Computes basic statistical functions (distribution of heights or slopes, autocorrelation function, power spectrum density function, etc.) from a selection of full data field.

Row/Column Statistics Somewhat complementary to 1D statistical functions, this tool plots characteristics such as mean, median or surface length for each row (column).

Roughness Evaluates standardized one-dimensional roughness parameters.

Three Point Level Levels data by plane obtained by clicking on three points within data window. The three values can be averaged over a small area around the selected point.

Path Level Row leveling tool equalizing the height along a set of arbitrary straight lines.

Crop Cuts part of the data.

Mask Editor Manual editing of masks: creation, exclusion, intersection, inversion, growing and shrinking, ...

Grain Measurement Measures individual grain parameters.

Grain Remover Removes continuous parts of the mask by clicking on mask point and/or interpolates (removes) data under a continuous part of mask.

Spot Remover Manually removes spots. Select a point on a data window, mark an area to interpolate on the zoomed view and remove the defect using chosen interpolation method.

Color Range Stretches color range or changes false color mapping type. It enables the user to change the false color representation range (by default from data minimum to data maximum).

Filter Basic filters – mean, median, conservative denoise, minimum, maximum and similar simple filters to reduce noise in the data.

Selection Manager Displays selections for a channel and copies them to other channels or files.

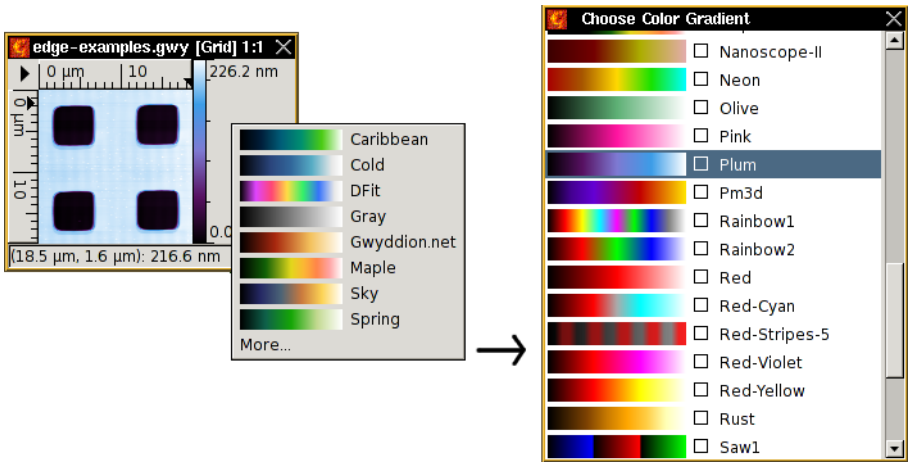
Tool dialogs can be closed (or more precisely hidden, as the current tool is still active even if its dialog is not visible), beside activating the *Hide* button, by pressing **Esc** or clicking the tool's button in the toolbox again.

3.7 False Color Mapping

False color mapping is the basic two-dimensional data visualization method. The color gradient (also called palette) to use can be selected after clicking on the false color map part of a data window with right mouse button.

This quick selection pop-up menu offers the list of preferred color gradients. In addition it allows to invoke the full color gradient list by selecting *More*. Preferred gradients can be chosen by checking the corresponding check buttons in the full list or in the gradient editor list. Selecting a row in the full list changes the gradient to the selected one, double-clicking (or pressing **Enter**) also finishes selection and closes the list window. Gradients of known names can be quickly accessed by starting to type their name. The default color gradient to use (when none is specified in the file) can be also set in the gradient editor.

More control over the way values are mapped to colors is possible with Color range tool.



A data window with the right-click color gradient pop up menu and the full color gradient list.

Color Range Tool

Color range tool is a special tool whose purpose is not to analyse or modify data, but to control the way values are mapped to colors. It offers four basic color mapping types:

Full Data values are mapped to colors linearly, the full data range corresponds to the full color range. This is the default type (unless you have changed the default).

Fixed Data values are mapped to colors linearly, a user-specified data range (which can be smaller or greater than the full range) maps onto the full color range. Values outside this range are displayed with the edge colors. The range can be set by several means:

- by entering desired values numerically in the tool window,
- by selecting a range on the height distribution graph in the tool window,
- by selecting an area on the data window, the range is then set from the minimum to maximum of the data in this area only or
- by pressing buttons *Set to Masked* or *Set to Unmasked* that set the range to the range of values that are under or not under the mask, respectively.

If no range is manually set, fixed range type behaves identically to full range.

Note data processing operations often modify the value range – and as the fixed range remains fixed as you set it, it can result for instance in completely black data display. You may wish or have to update the range manually then, or to switch to another mapping type.

Automatic Data values are mapped to colors linearly, a heuristically determined subinterval of the full value range maps onto the full color range. Values outside this subrange are again displayed with the edge colors.

Adaptive The full data range corresponds to the full color range, however data values are mapped to colors non-linearly. The mapping function is based on inverse cumulative height distribution, therefore flat areas generally get bigger slice of the color gradient and smaller value variations can be seen on them than normally.

The false color map ruler on the right side of data windows does not display any ticks in this mode, only the minimum and maximum value.

A mapping type can be set to be default by checking the *Default* check button when it is active. Newly displayed data windows then use this type, unless a channel explicitly specifies other type to use.

Saving data to `.gwy` file also saves all color mapping settings: mapping type, range and gradient. Gradient is however not physically stored in the file, only referenced by name. In other words, color gradients of the same name are shared among files.

Color Gradient Editor

Color gradient editor can be invoked with *Edit* → *Color Gradients*. It consists of a gradient list similar to the full gradient selector, with an additional button panel, and the actual editor that can be invoked by double-clicking on a gradient you wish to edit or by activating the *Edit* button. Renaming is possible. Only user-created color gradients can be edited or deleted, system gradients installed with Gwyddion are immutable.

The last button in the gradient list control panel makes the currently selected gradient the default. It will be used for all newly displayed data that do not specify any particular color gradient.

Two editing modes are available:

Points The color gradient is defined by a set of points and their associated colors. The points are represented by triangular markers on the gradient displayed in the lower part of the editor window. Moving these markers moves the points, new points can be added by clicking into an empty space, existing points can be removed by dragging them away from the gradient.

Curve The color gradient is defined by red, green and blue curves. The curves are again segmented, but the segments of individual curves do not need to coincide.

3.8 Presentations and Masks

Presentations

Presentations can be used to show the height field in another way than as a false color map of the heights, for instance with shading or with highlighted edges. It is also possible to superimpose an arbitrary data field over another one as the presentation.

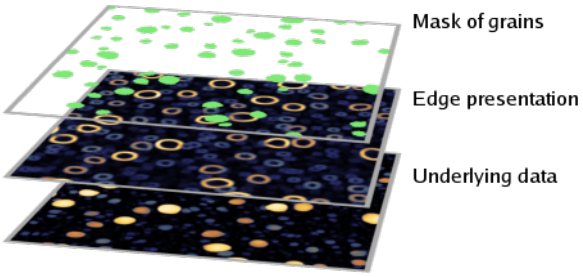
Note the superimposed presentation is really only a presentation, it is never used for calculations. In all data processing functions or tools the results are always computed from the original underlying data. Since presentations can be computationaly intensive to calculate, they are not automatically updated when the underlying data change. The various presentations available are described in section Presentations.

The presence of presentation over the data is indicated by flag *P* in the data browser and also by the empty false color map ruler on the right side of the data window that does not display any ticks nor the minimum and maximum value.

Masks

Masks are used for special areal selections, e.g. grains, defects or facets with certain orientation. Masks can have any shape and within the data window and they are visualized by a color overlayed over the data. The mask color and opacity can be changed in the right-click context menu of the data window.

Since grain marking is the most common use of masks, several functions that operate on marked areas are called “grain” functions, e.g. Grain Statistics. Also, a contiguous part of mask is sometimes called grain in this guide. However, since a mask does not bear any information how it was created all mask functions can be used with masks of any origin.



Visualization of masks and presentations. If you look from above they can be imagined to be stacked as in the picture.

Both masks and presentations can be removed from the data by functions in the right-click menu of the data window, or with keyboard shortcuts.

Data in default false color representation (left), with superimposed mask visualized with a red color (centre) and with shading presentation (right).

Working with Masks

The mask-related functions can be divided into three main groups:

Creation Masks are created by various types of marking functions, namely grain marking functions (Mark by Threshold, Mark by Watershed), defect marking functions (Mask of Outliers, Mark Scars) and feature marking functions (Mask by Correlation, Facet Analysis, Certainty Map). In addition, some general mask editing functions provide means to create masks from scratch.

Masks are also used to mark invalid pixels in files imported from formats that distinguish between valid and invalid pixels since Gwyddion does not have a concept of invalid pixels.

Application In general, the mask-covered area is considered to be the area of interest, i.e. the area to operate on. This applies namely to statistical functions such as the Statistical Quantities tool. Function Remove Data Under Mask replaces the data under mask, while the Remove Grains tool can perform such replacement for individual grains. There are several functions for the examination of grain properties, see section Grain Statistics.

Some functions ask whether to consider the area under mask included or excluded (or ignore the mask), namely leveling functions. Such choice is offered only if a mask is present on the data.

Editing A few basic mask operations, such as inversion or complete removal, are available in *Data Process* → *Mask* menu. More advanced functions include the grain-oriented Remove Grains tool and Grain Filter that provide different means to remove parts of the mask, as well as Mask Editor tool and Mark With focused on general mask editing.

Mask Editor Tool

The Mask Editor is the universal mask modification tool. It provides two groups of functions: editing of the mask by drawing directly in the data window and global operations with the mask such as inversion or growing and shrinking.

The direct mask modification is controlled by the buttons in the *Editor* group. It can be done in two ways: by selecting geometrical shapes that are subsequently filled or erased (using the *Shapes* option), and by freehand drawing operations using drawing tools (option *Drawing Tools*).

Buttons in the *Mode* row select how the geometrical shape drawn in the data window will modify the mask:

Set The mask is set to the drawn shape, discarding any mask already present.

Add The mask is extended by the drawn shape (if there is no mask yet a mask is created).

Subtract The drawn shape is cut out from the mask This function has no effect if there is no mask.

Intersect The mask is set to the intersection of the drawn shape and the already present mask. This function has no effect if there is no mask.

Buttons in the *Shape* row control which shape is drawn on the mask. The choices include rectangles, ellipses and thin lines.

Freehand drawing tools are selected by buttons in the *Tool* row:

Pencil Freehand drawing with a pencil of radius specified by parameter *Radius*. Note this may be slow on slow computers and/or large data fields.

Eraser Freehand erasing with an eraser of radius specified by parameter *Radius*. Note this may be slow on slow computers and/or large data fields.

Fill Bucket-filling of a contiguous unmasked area.

Unfill Bucket-unfilling of a contiguous masked area.

The basic global operation with masks, i.e. inversion, removal and filling the entire data field area with a mask are available in the *Actions* row. Additional operations include:

Grow Extends the mask by *Amount* pixels on each side. More precisely, the mask is extended by one pixel on each side and this is repeated *Amount* times.

Normally, growing does not distinguish between individual parts of the mask. Parts that grow so much that they touch therefore merge. This can be prevented by *Prevent grain merging by growing* which makes individual parts of the mask stop growing once there is only one-pixel space between them.

Shrink Reduces the mask by *Amount* pixels from each side. More precisely, the mask is reduced by one pixel from each side and this is repeated *Amount* times.

The reduction may or may not occur from the data field borders. This is controlled by the *Shrink from border* check box.

Fill Voids Makes the grains single-connected, i.e. without any holes, by filling the holes in grains.

Mark With

Data Process → *Mask* → *Mark With*

Mark With can create or modify masks using another mask or data of the same dimensions. The operations that can be applied to the current mask are the same as in the Mask Editor tool: creation, union, subtraction and intersection. The source of the other mask can be one of the following:

Mask This is the simplest case, a mask can be combined with another mask using the specified logical operations.

Data In the Data mode, another height field is used as the other mask source. The mask consists of pixels within a range of heights, specified as relative values within the total range. To use pixels outside a certain range for the masking, set the upper bound to a smaller value than the lower bound.

Presentation The Presentation mode differs from Data mode only in that a presentation is used instead of the data.

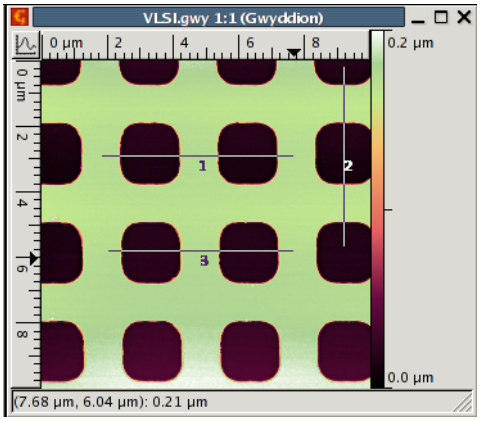
This is an exception to the rule stating that presentations are never used for further processing. Sometimes it can be useful to mark, for instance, edges found on the data even though the corresponding presentation visualizes a quantity weird from the physical point of view.

3.9 Selections

All interactive tools and some other processing methods allow to select geometrical shapes on data with mouse: points, lines, rectangles, circles/ellipses. Existing selections can be similarly modified by dragging corners, endpoints, or complete selections. When mouse cursor is moved near to an editable point of a selection, is changes its shape to indicate the possibility to edit this point.

Each tool typically uses only one type of selection and when it is activated on a data window, it sets the selection mode to this type. Selections of other types than currently displayed are remembered and they are recalled when a tool which uses them is activated again. E.g. when you select several lines with Profile extraction tool, then switch to Statistical quantities (the lines disappear) and select a rectangular area to calculate its statistical characteristics, and then switch back to Profile extraction, the rectangle disappears and the lines appear again.

Tools that use the same type of selection – e.g. both Statistical functions and Statistical quantities use rectangular selection – share it. To calculate height distribution of the same rectangle you have selected for statistical quantities, it is sufficient to switch the tool.



Data window with three selected lines, two horizontal and one vertical.

If you save data in Gwyddion native file format (`.gwy`), all selections are saved together with data and recalled the next time the file is opened and appropriate tool chosen.

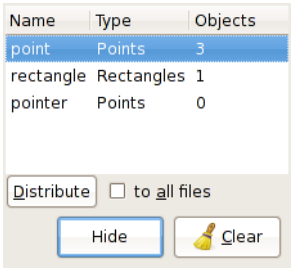
Pressing Shift during selection restricts the degrees of freedom of the shape, making it easier to draw shapes from a specific subset. Specifically, pressing Shift restricts

- rectangular selections to perfect squares,
- elliptical selections to perfect circles,
- directions of line selections to multiples of 15°.

Selection Manager

The selection manager is a special tool that displays the list of all selections in a channel and enables to copy them to other channels.

For each selection, the tool shows the name, which is how the selection is identified in the `.gwy` file; the selection type and the number of objects (points, lines, rectangles, ...) selected. Usually, there is at most one selection of any type because they are shared among the tools as described above. Nevertheless, sometimes there are special or private selections present as shown on the following figure displaying two point-wise selections.



Selection Manager showing several selections present in the data.

Selection chosen in the list is displayed in the data window.

It is possible to delete individual selections by choosing them in the list and pressing Delete – this is equivalent to clearing the selection in the corresponding tool. The *Clear* button removes all selections.

However, the most interesting function of Selection Manager is selection copying. There are two ways to copy a selection to another channel:

- Dragging a row from the selection list onto a data window copies the selection to this data window.
- Clicking the *Distribute* button copies the selection to all other channels in the file. Or, if *to all files* is enabled, to all channels in all open files.

Selections are copied only to channels with compatible lateral units. This means that a selection in a normal channel with meters as the lateral units will not be distributed to a two-dimensional PSDF channel or a two-dimensional slope distribution.

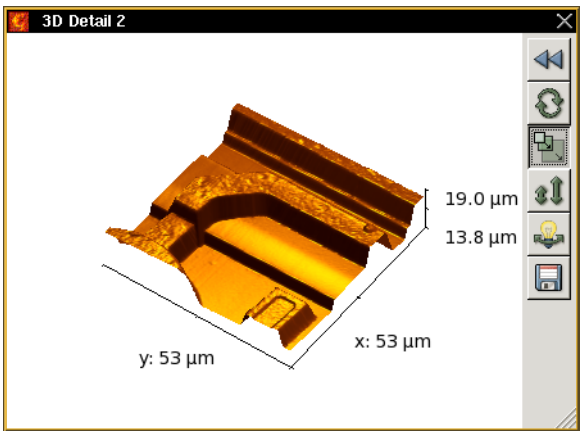
If the physical dimensions of the target data are not sufficient to contain all the objects of the copied selection then only those objects that fit are copied (this can also mean nothing is copied).

3.10 OpenGL 3D Data Display

Three-dimensional OpenGL display of the current data window can be invoked with the button with symbol of cube in *View* button row of main window.

This feature is optional, i.e. it can be disabled at compile time. It can also happen that while Gwyddion is capable of 3D data display your system does not support it. In both cases an attempt to invoke 3D view gives an error message explaining which of the two cases occured. In the former case you have to ask the producers of Gwyddion executables to build them with 3D support or build Gwyddion yourself from source code. If it is the latter case, refer to your operating system guide on how to enable OpenGL 3D capabilities. If you experience a poor performance of the 3D view in MS Windows try disabling desktop composition for Gwyddion (in *Compatibility* tab of the Gwyddion shortcut).

The 3D window has two possible forms: with basic and expanded controls. It starts with basic controls only, this form is displayed on the following figure. It can be switched to the expanded form (and back) with an expander button in the upper right corner. Clicking on the view with right mouse button brings a quick color gradient/GL material selector.



Three-dimensional OpenGL data display window with basic controls.

Basic Controls

Basic 3D window contains interaction mode controls at the right side of the view. By default, dragging the view with mouse rotates it horizontally and vertically. All possible modes are listed below:

- Rotation – this is the default. Dragging the view horizontally rotates it around z-axis, vertical drag rotates it around horizontal axis parallel with the plane of view.
- Scale – dragging the view right and down enlarges it, drag in the opposite direction makes it smaller.
- Z-scale – dragging the view up (down) increases (decreases) the z-scale, making the hills and valleys more or less pronounced.
- Light rotation – this possibility is available only in lighting visualization mode. Dragging the view changes position of light source similarly to rotation of data in normal rotation mode.

The basic controls also include an image export button.

When basic controls are shown it is possible to switch between the modes using keys **R** (rotation), **S** (scale), **V** (value scale) and **L** (light rotation).

Full Controls

In expanded controls the mode buttons are located in top row, however their function does not change. In addition, there are several tabs with options below them:

- *Basic* – controls to set rotations and scales numerically and to switch on and off axes, axis labels, and perspective projection.
- *Light & Material* – visualization settings. Gwyddion 3D view has two basic visualization modes: gradient, in which the data are simply colored with a false color scale exactly like in normal 2D view; and material, in which the data are presented as an OpenGL material rendered according to light position. This tab also contains controls to set light position numerically.
- *Labels* – fine tuning of sizes, positions, and other properties of axis labels.

Saving Images

The 3D view can be saved into a bitmap image with the *Save* button. The output is currently always a PNG (Portable Network Graphics) image with exactly the same size and contents as displayed on the screen. Entering a different file extensions than `.png` still produces an image in PNG format, albeit with a confusing extension.

Note due to the peculiarities of certain operating systems, graphics drivers and windowing environments, artefacts may sometimes appear on the exported image in parts corresponding to obscured parts of the 3D view. If you encounter this problem, make sure the 3D view is not obscured by other windows during the image export.

OpenGL Material Editor

OpenGL material editor can be invoked with *Edit* → *GL Materials*. The controls in the material list are the same as in the color gradient editor list and the material management works identically. The actual editor is of course different. It allows to edit four quantities defining the material:

- ambient color $k_{\mathrm{a},\alpha}$ (where $\alpha = \mathrm{red}, \mathrm{green}, \mathrm{blue}$), controlling the reflection of ambient light that is assumed coming uniformly from all directions,
- diffuse color $k_{\mathrm{d},\alpha}$, describing the diffuse reflection which is independent on the direction of incident light and whose apparent brightness is independent of the viewing angle,
- specular color $k_{\mathrm{s},\alpha}$, controlling the specular reflection with reflected light intensity dependent on the angle between the observing direction and the direction of light that would be reflected by an ideal mirror with the same normal, and
- shininess s, a numeric exponent determining how much the specular reflection resembles an ideal mirror, smaller values mean rougher surfaces, higher values mean smoother surfaces.

If we denote **L** the normal vector pointing from the observed surface point to the light source, **V** the normal vector to the observer, **N** the normal vector to the surface and **R** the normal vector in the direction of ideal mirror reflection, the observed light intensity in OpenGL lighting model can be expressed as

$$I_\alpha = k_{a,\alpha} I_{a,\alpha} + k_{d,\alpha} I_{d,\alpha} (\mathbf{N} \cdot \mathbf{L}) + k_{s,\alpha} I_{s,\alpha} (\mathbf{R} \cdot \mathbf{V})^s$$

where $I_{\mathrm{a},\alpha}$, $I_{\mathrm{d},\alpha}$ and $I_{\mathrm{s},\alpha}$ are the ambient, diffuse and specular light source intensities, respectively.

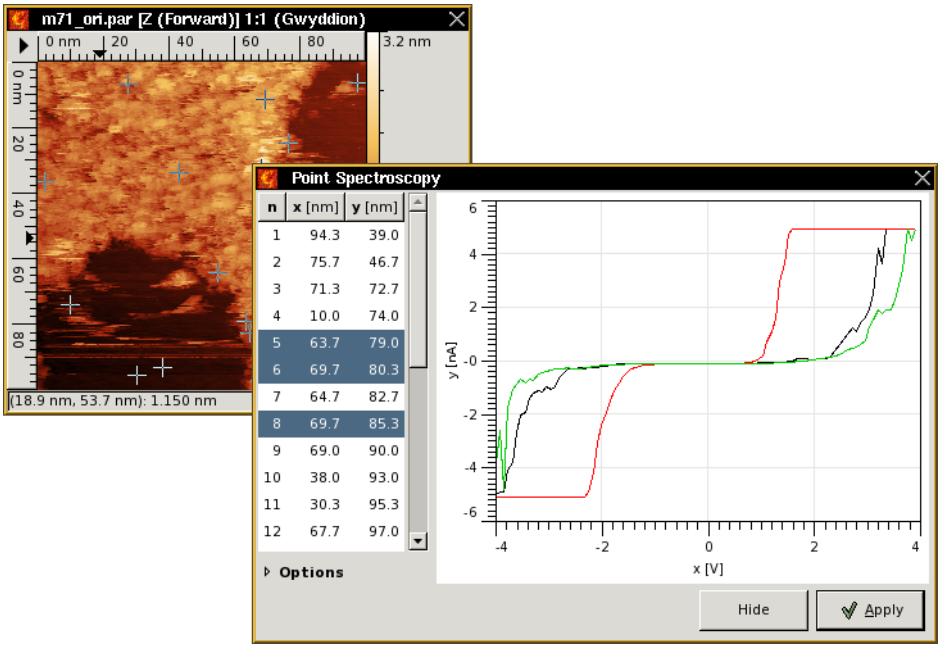
3.11 Single Point Spectra

Gwyddion currently offers some basic visualization and extraction means for single point spectroscopy data (we will generally refer to any curves measured in or otherwise attached to individual points of the sample as to "spectra" here). If spectra import is supported for a file type, they will appear in the *Spectra* tab of the data browser. Standalone spectra files can be added to the two-dimensional data using file merging.

Point Spectroscopy Tool

The primary spectra visualization and extraction tool is the Point spectroscopy tool. It displays a list of measurement points and shows their positions on the data window. Individual curves can be selected either in the list or by selecting the corresponding crosses on the data window. If the file contains more than one spectrum set, one can choose among them by selecting the desired one in the *Spectra* tab list of the data browser.

The *Apply* button then extracts the selected set of curves into a graph that can be further analysed by graph functions, such as force-distance curve fitting.

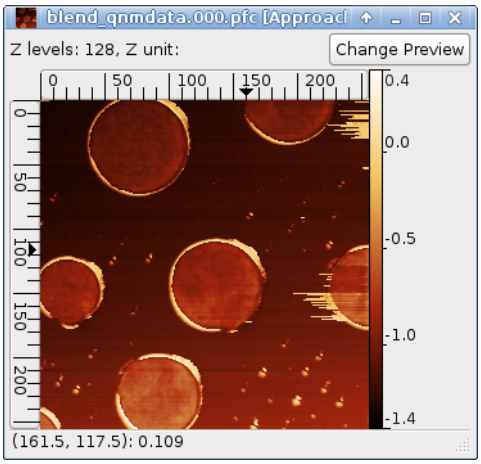


A data window with measurement points displayed and the point spectroscopy tool showing curves from three selected points.

3.12 Volume Data

Volume data are three-dimensional data representing, for instance, grid spectroscopy (spectra in each image point) or time evolution of the image. So, depending on the point of view, they can be imagined as either a stack of identically-sized images or as a set of curves forming a regular grid (image).

Gwyddion currently offers some basic visualization and extraction tools in the *Volume* menu. Volume data windows, as shown in the following figure, provide the most basic visualisation of volume data: an "preview" image, representing one plane of the volume data or other associated two-dimensional data. The x and x coordinates of the image correspond to the x and x coordinates of the volume data; the z coordinates can be imagined as orthogonal to the screen.



A volume data window displaying the associated two-dimensional preview data.

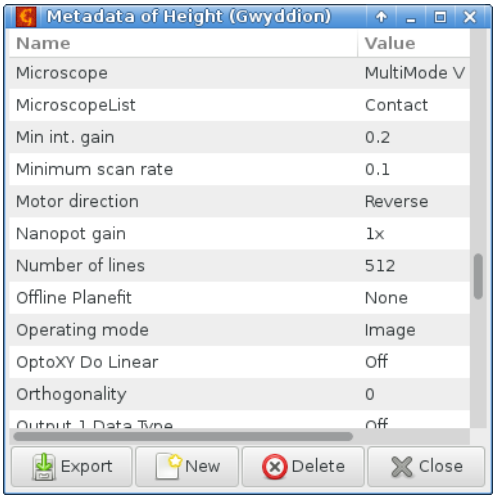
Button *Change Preview* near the top of the window can be used to change the preview image. The preview can either be calculated as a summary quantity characterising data along the invisible z direction (minimum, maximum, mean, ...), chosen as a section of the volume data, alternatively just any two-dimensional data of the same dimensions can be shown as the preview.

3.13 Metadata

Auxiliary information and values describing certain data and the conditions it was measured on are called metadata in Gwyddion. They may include the SPM mode, tip bias, scanning frequency, measurement date and time or user comments.

Metadata are always per-channel. The metadata for the current channel or volume data can be displayed with *Metadata Browser* command in the right-click context menu (in version 2.32 or newer) or using *Meta* → *Metadata Browser* (in older versions). The browser lists all available metadata as Name, Value pairs. It also enables to modify and delete values or add new ones. It is possible to export all metadata of a channel to a text file with *Export* button.

The level of metadata support differs widely between file formats. For file formats that are well documented and/or allow to import all metainformation generically lots of auxiliary data including obscure hardware settings can be listed. On the other hand it is possible that no metadata is imported from your files, for example when they contain no auxiliary data or it is not known how to read it.



Metadata browser showing the metadata of a Nanoscope file.

3.14 Image Export

Microscopy data often need to be rendered into image formats, usually for presentation purposes, occasionally to open them in programs that do not support any SPM data format. Both is achieved the same way in Gwyddion: by selecting *File* → *Save As*

and choosing a file name corresponding to an image format, e.g. `channel.png` or `channel.tiff`. The reference section High-Depth Image Formats describes the export of data as high-depth greyscale images that should be still usable as quantiative data. For supported formats you can enable it using the *Export as 16 bit grayscale* check-box at the top of the image export dialog. The rest of this section discusses the creation of nice images for publications and presentations.

Gwyddion can render images into a number of formats, including for instance PNG, PDF, SVG, EPS, TIFF, BMP, PPM, TARGA and JPEG. Depending on the intent, some may be more suitable than others. Generally, the following choices can be recommended:

- PDF (Portable Document Format) for high-quality rendering suitable for printing, with all text and lines perfectly sharp and clean at any scale. Gwyddion also suports output to EPS (Encapsulated PostScript) for the same purpose, however, some features currently seem to work better in the PDF output.
- PNG (Portable Network Graphics) for web pages, low-resolution previews, thumbnails and icons, and also if you have to use a raster image format because it is the only option. PNG is a modern and widely supported raster image format with good loseless compression and a number of nifty features.
- SVG (Scalable Vector Graphics) for subsequent editing and processing. SVG is a modern vector graphics format. You can open in a vector image editor such as Inkscape and modify it or combine with other images – keeping all text and lines perfectly sharp and clean.

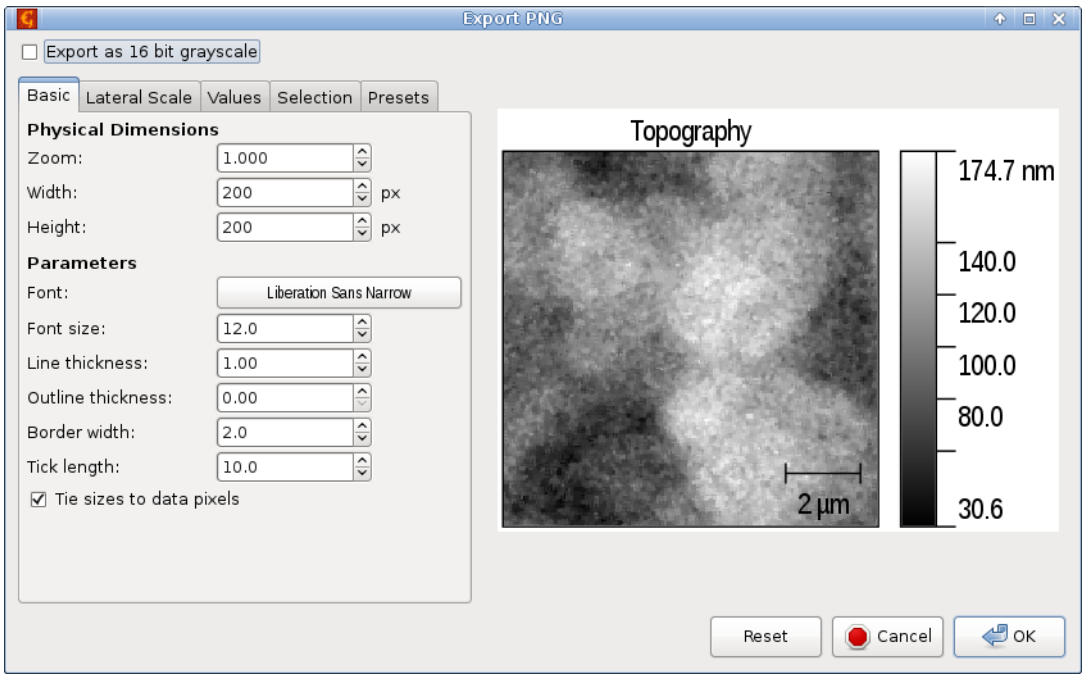


Image export dialog screenshot showing the Basic *options page.*

The image export options are divided into several groups, as seen in the above screenshot.

Basic

Basic options specify various overall sizes and scales. The *Physical Dimensions* part differs for raster and vector images. For raster images the physical dimensions are specified as follows:

Zoom Scaling of data pixels to image pixels. The default zoom of 1 means data pixels correspond exactly to image pixels. Upscaling is possible with zoom larger than 1, and downscaling with zoom smaller than 1. For data with non-square pixels displayed with physical aspect ratio, the zoom is applied to the shorter side of the pixel, whichever it is.

Width Width of the rectangle corresponding to data in the exported image (not width of the entire image), in pixels.

Height Height of the rectangle corresponding to data in the exported image (not height of the entire image), in pixels.

Vector images do not have finite pixel resolution, therefore, the physical dimensions can be instead given as follows:

Pixel size Size of one data pixel in millimetres.

Pixels per inch Number of data pixels per one inch.

Width Width of the rectangle corresponding to data in the exported image (not width of the entire image), in millimetres.

Height Height of the rectangle corresponding to data in the exported image (not height of the entire image), in millimetres.

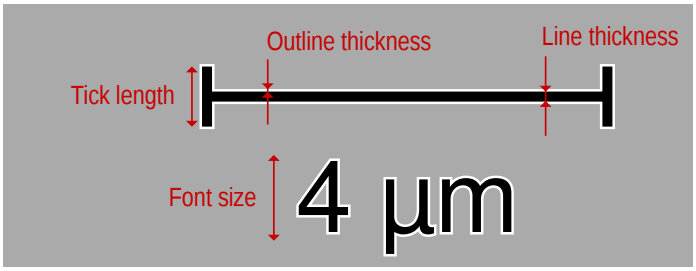


Illustration of the basic size parameters.

The remaining size and scale parameters, denoted *Parameters*, are common to both image types. However, for raster images the values are in pixels while for vector images they are in typographical points. More precisely, this is true if they are given as absolute, i.e. the option *Tie sizes to data pixels* is not selected. If this option is selected all sizes are relative to data pixels, i.e. they scale together with the image data when the physical dimensions change. The parameters, illustrated in the figure above, include:

Font The font used to draw all labels.

Font size Font size, in pixels or typographical points as described above.

Line thickness Thickness of lines: borders, ticks, inset scale bar and selections.

Outline thickness Thickness of outlines that can be drawn around the inset scale bar and selections.

Border width Width of empty border around the entire image.

Tick length Length of ruler and false color map ticks. This setting also controls the inset scale bar ticks and crosses drawn for the point-like selection.

Lateral Scale

Settings in the *Lateral Scale* page control how the lateral dimensions are visualised. There are two basic choices displayed in the figure below, rulers and an inset scale bar. The lateral scale can be also disabled entirely. The inset scale bar has the following settings:

Length The bar length can be set manually to an arbitrary value which does not result in a too short or too long bar. Press Enter to update the preview when you modify the length. Button *Auto* selects a suitable length automaticaly (this is also done when the manually entered length is not found reasonable).

Placement The bar can be placed along the upper or lower edge and aligned to either side or centered.

Horizontal gap Horizontal gap between the bar and the closer vertical edge of the data area (meaningful only if the bar is not centered).

Vertical gap Horizontal gap between the bar and the closer horizontal edge of the data area.

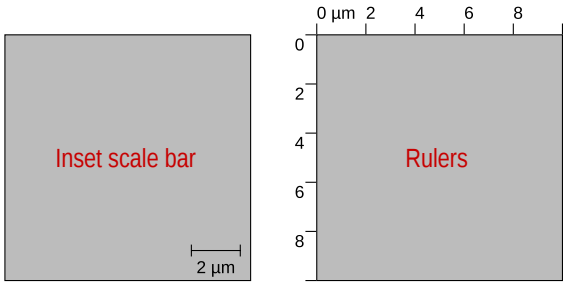
Color Color with which the bar is drawn. Buttons *Black* and *White* allow choosing quickly the basic colors.

Outline color Color of the outlines. Note if the outline width is zero (the default), no outlines are drawn, hence changing the color has no effect.

Opacity Opacity with which the inset scale bar is drawn. This setting permits drawing the bar as semi-transparent.

Draw ticks If enabled, the bar has vertical ticks at its ends. When disabled, the bar is just a line.

Draw label If enabled, the bar length is displayed under the bar.



Lateral scale visualisation types.

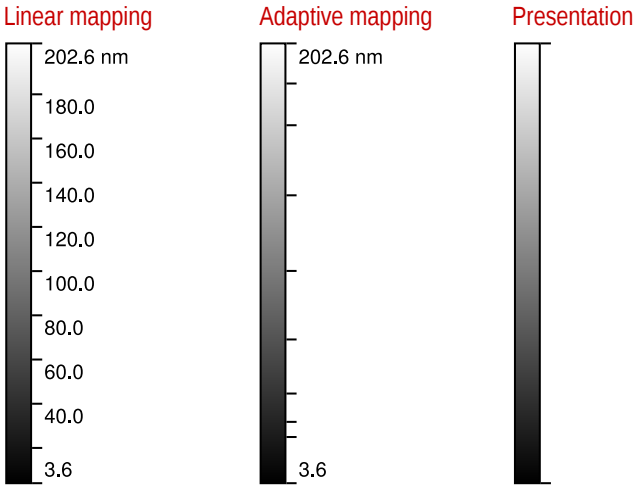
Value

Settings in the *Value* page control the rendering of values and false color mapping. Two basic settings control the rendering of field data:

Interpolation The interpolation type can be noticeable particularly for large zooms. In the case of vector image formats, the final rendering is done when then the image is viewed or printed. Hence the available interpolations are limited to two types. Round, in which each data pixel is drawn as a sharp rectangle, and Linear, in which values in the rendered image are linearly interpolated between original data pixels. In the case of raster images, you can choose from the full set of interpolations supported by Gwyddion because the interpolation is done during the export.

Draw mask If enabled, the mask is drawn on top the data using the same color as in the data window.

The value scale can be rendered as a false color ruler or disabled. The ruler is drawn somewhat differently depending on the false color mapping type, as illustrated in the following figure.



Rendering of the false color map scale depending on the mapping type. For the standard linear mapping, whether full or fixed range, ticks with values are drawn. For an adaptive mapping the color gradient is drawn the same, but tick positions correspond to the adaptive mapping and interior ticks are drawn without labels (to avoid overlapping labels). If a presentation is shown, the values are considered arbitrarily scaled thus no values and interior ticks are displayed.

The channel title can be optionally added to the top of the image or along the false color ruler. Using the check-box *Put units to title*, the placement of value units can be chosen between the title and false color ruler. Note if the no title is drawn then placing units to the title disables them entirely.

Settings *Horizontal gap* for the false color ruler and *Gap* for the title control the gaps between them and the corresponding image edges as shown in the following figure. If the channel title is drawn along the false color ruler the gap can be negative, moving the title a bit inside the ruler. For titles along the top edge, negative gap values are ignored.

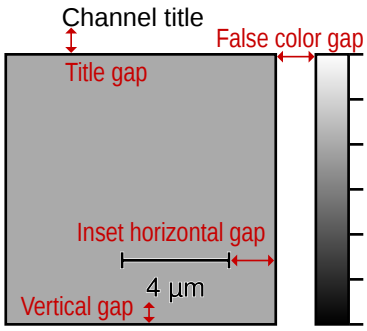


Illustration of the various gap settings (for channel title, in the position at the top).

Selection

Any kind of selection stored with the data can be also shown in the image. If the check-box *Draw selection* is enabled you can choose the selection to draw from the list below. The colors are specified in the same manner as for the inset scale bar:

Color Color with which the selection is drawn. Buttons *Black* and *White* allow choosing quickly the basic colors.

Outline color Color of the outlines. Note if the outline width is zero (the default), no outlines are drawn, hence changing the color has no effect.

Opacity Opacity with which the selection is drawn. This setting permits drawing the selection as semi-transparent.

Beside the colors, some selection types have further options, for instance whether individual shapes are numbered. If you are using a tool that has some shapes selected on the data, the kind of the selection to draw and the specific options are preset to match the current tool. This is usually the most convenient way to get the selection drawn as you intend. However, any existing selection can be drawn and the options adjusted manually if you wish.

Presets

Different sets of image rendering options are useful on different occasions. A set of options can be saved as a preset and recalled later. The list in the *Presets* page shows all saved presets that can be managed using the buttons below:

Load Loads the currently selected preset, i.e. sets the image rendering options according to the preset. The inset scale bar length can be set to the automatic value if the stored length is not found reasonable. Also the selection type and its options are kept intact, only the colors are set according to the preset.

Store Stores the current options under the name given as *Preset name*. The name also serves as a file name so it is advisable to avoid odd characters in the preset name. If a preset of the same name already exists, it is overwritten.

Rename Renames the currently selected preset to the name given as *Preset name*. The preset does not have to be loaded for renaming. It is not possible to overwrite an existing preset by renaming.

Delete Deletes the currently selected preset.

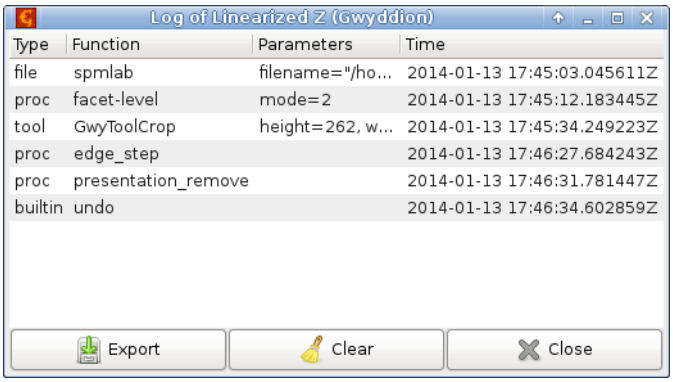
3.15 Logging

Gwyddion records data modification operations for each channel or volume data in so-called log. When the data are saved to a `.gwy` file, the log is saved along with them. The log can be displayed by selecting the *View Log* command in the right-click context menu of the channel or volume data window. This is useful for recalling later what corrections you applied, how a mask or presentation was created, etc. It should be noted that the logs are informational; they are neither intended nor suitable for auditing purposes.

The log view is live: if you keep it open you can see individual data processing operations appearing there as you perform them.

A simple log example is shown in the following figure. For each operation, the type, name, parameters and time are recorded. The type can be for instance file import, data processing function or tool application. The function names correspond to those shown in the module browser (*Info* → *Module Browser*), where they are listed in *Registered functions* for each module; or in the on-line module browser. The parameter list represents the settings of the function at the time it was used. Since the log is

only informative, the parameters may or may not allow a precise reconstruction of the operation. For typical simple operations, they should be sufficient. In the case of complex or interactive operations involving multiple channels, the log entry may not be detailed enough. The time is recorded in the local time zone when the operation is performed; it is not recalculated when you send the files around the world and display elsewhere.



Log viewer showing a simple data processing operation log for a channel, starting with file import from an SPM vendor format and continuing with the application of data processing functions, tools and undo.

The entire log can be exported to a text file using the *Export* button. It not possible to modify the log entries as that would defeat the purpose of logging somehow, but you can clear the entire log with *Clear*.

Disabling logging

In some circumstances, you may wish to store or publish `.gwy` files without logs. Therefore, logging is controllable on several levels:

- It can be enabled and disabled globally using *Edit* → *Logging Enabled*. When logging is disabled no new log entries are added. Existing logs are not removed though and you can still view them. They are also still saved to `.gwy` files.
- The log for a specific channel or volume data can be cleared with button *Clear* in the viewer.
- All logs in the current file can be removed using *File* → *Remove All Logs*. As with any other file modification, the file needs to be saved afterwards for the log removal to have any effect on the on-disk file. And, of course, if logging is enabled and you start modifying the data, new logs will be created and the new data operations recorded in them.

3.16 Raw Data File Import

Both raw ASCII and binary data files and files in unsupported formats can be imported with rawfile module – with some effort. Raw data import can be explicitly invoked by selecting *Raw data files* type in the file open dialog. It can be also set to appear automatically when you try to open a file in an unknown format. This is controlled in the raw file dialog by option *Automatically offer raw data import of unknown files*.

Information

Its first tab, *Information*, allows to set basic file information:

Horizontal size*, *Vertical size Horizontal and vertical data resolution (number of samples).

Square sample Fixes horizontal and vertical resolution to the same value.

Width*, *Height Physical sample dimensions.

Identical measure Keeps the ratio between physical dimension and number of samples equal for horizontal and vertical direction, that is the data has square pixels.

Z-scale (per sample unit) The factor to multiply raw data with to get physical values.

Data Format

On the second tab, *Data Format*, particular data format can be chosen. There are two independent possibilities: *Text data* and *Binary data*.

Text files are assumed to be organized by lines, each line containing a one data row, data being represented by integers or floating point numbers in standard notation. Following options are available:

Start from line The line data starts at, that is the number of lines to ignore from file start. All types of end-of-line markers (Unix, MS-DOS, Macintosh) are recognized.

Each row skip The number of fields to ignore at the begining of each line.

Field delimiter, Other delimiter If delimiter is *Any whitespace*, then any nonzero number of whitespace characters counts as field delimiter. If a whitespace character is selected, the delimiter must be this character. Otherwise field are separated by specified character or string and all whitespace around delimiters is ignored.

Decimal separator is comma By default, floating point numbers are assumed to use decimal point. This option changes it to comma.

Following options are available for binary files:

Binary data You can either select one of predefined standard data formats, or *User defined* to specify a format with odd number of bits per sample or other peculiarities.

Byte swap pattern How bytes in samples are swapped. This option is only available for predefined formats larger than one byte. Its bits correspond to groups of bytes to swap: if the j-th bit is set, adjacent groups of 2^j bits are swapped.

For example, value 3 means sample will be divided into couples (bit 1) of bytes and adjacent couples of bytes swapped, and then divided into single bytes (bit 0) and adjacent bytes swapped. The net effect is reversal of byte order in groups of four bytes. More generally, if you want to reverse byte order in groups of size 2^j, which is the common case, use byte swap pattern $j-1$.

Start at offset Offset in file, in bytes, the data starts at.

Sample size Size of one sample in bits for user defined formats. E.g., if you have a file with only 4 bits per sample, type 4 here. For predefined formats, their sample size is displayed, but it is not modifiable.

After each sample skip The number of bits to skip after each sample.

Usually, samples are adjacent to each other in the file. But sometimes there are unused bits or bytes between them, that can be specified with this option. Note for predefined types the value must be a multiple of 8 (i.e., only whole bytes can be skipped).

After each row skip The number of bits to skip after each sample in addition to bits skipped after each sample.

Usually, rows are adjacent to each other in the file. But sometimes there are unused bits or bytes between them, that can be specified with this option. Note for predefined types the value must be a multiple of 8 (i.e., only whole bytes can be skipped).

Reverse bits in bytes Whether the order of bits in each byte should be reversed.

Reverse bits in samples Whether the order bits in each sample should be reversed for user defined samples.

Samples are signed Whether samples are to be interpreted as signed numbers (as opposed to unsigned). For predefined formats, their signedness is displayed, but it is not modifiable.

Presets

Import settings can be saved as presets that allow to easily import the same file – or the same file type – later.

Button *Store* saves current import settings under the name in *Preset name* field. *Rename* renames currently selected preset to specified name, *Delete* deletes selected preset, and *Load* replaced current import setting with selected preset.

3.17 Specific Data Import

Import of several other types of data is not automatic and it requires human intervention.

Graphics Formats

Importing data from image formats such as PNG, TIFF, JPEG or BMP is similar to import from raw/unknown file formats, only simpler.

It is simpler because the file structure is known and the file format is automatically detected. Hence the file type does need to be selected explicitly. However, the data interpretation is still unknown and must be specified manually. The Pixmap import dialog therefore resembles the *Information* tab of raw data import, requiring you to set the physical dimensions and value scale.

Note the physical dimensions suggested there are not obtained from the file, they are simply the last values used. Some SPM data format are based on an image format (typically, TIFF is used as the base) and contain the information about physical scales and units, albeit stored in a manufacturer-specific way. In this case a separate import module can be written for this particular format to load the files automatically with correctly scaled values.

See the reference section High-Depth Image Formats for the details of support for high-depth images and the possibility of using them for data representations.

Graph Curves

Simple two-column text files containing curve data can be imported as graph curves. In some cases, these files are recognized automatically. They can also be explicitly selected as *ASCII graph curve files* in the file open dialog, causing the import module to try harder to load the file as a graph data.

The import dialog shows a preview of the graph and permits to set the units and labels.

XYZ Data

Three-column text files containing XYZ data are imported by selecting the *XYZ data files* file type. Again, they can be recognized automatically but requesting this format explicitly makes the module to try harder to load the file as XYZ data.

Since Gwyddion only works with data in a regular grid irregular XYZ data must be interpolated to a regular grid upon import. In fact, the XYZ data import module serves two different purposes:

- loading of data in a regular grid that were just saved as XYZ data – if the data is found to be in a regular grind only a very simple import dialog is presented where you can set the units because the import is straightforward;
- regularization and interpolation of irregular XYZ data – this case is much less straightforward and the rest of this section will discuss the options you have and some of the pitfalls.

The import dialog permits to set the basic parameters as the regularized data resolution and range and lateral and value units. However, the most important option is *Interpolation type*:

Round This interpolation is analogous to the Round interpolation for regular grids. The interpolated value in a point in the plane equals to the value of the nearest point in the XYZ point set. This means the Voronoi triangulation is performed and each Voronoi cell is “filled” with the value of the nearest point.

Linear This interpolation is analogous to the Linear interpolation for regular grids. The interpolated value in a point is calculated from the three vertices of the Delaunay triangulation triangle containing the point. As the tree vertices uniquely determine a plane in the space, the value in the point is defined by this plane.

Field The value in a point is the weighted average of all the XYZ point set where the weight is proportional to the inverse fourth power of the mutual distance. Since all XYZ data points are considered for the calculation of each interpolated point this method can be very slow.

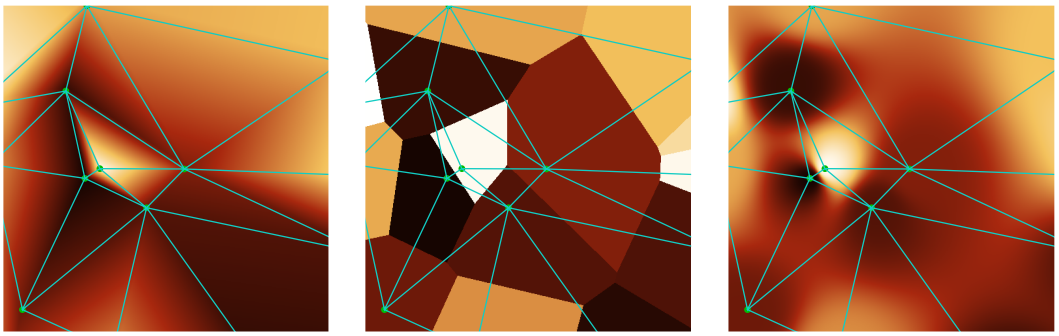
The former two interpolation types are based on Delaunay/Voronoi triangulation which is not well-defined for point sets where more than two points lie on a line or more than three lie on a circle. If this happens the triangulation fails and the import module displays an error message.

The values outside the convex hull of the XYZ point set in the plane are influenced by *Exterior type*:

Border The point set is not amended in any way and the values on the convex hull simply extend to the infinity.

Mirror The point set is amended by points "reflected" about the bounding box sides.

Periodic The point set is amended by periodically repeated points from around the opposite side of bounding box.



Delaunay triangulation displayed on linear (left), round (centre) and field (right) interpolation of a irregular set of points.

3.18 Plug-ins

Plug-ins are external programs that can be executed by Gwyddion to either perform some operation on the data or to read or write data in a third-party file format. In general, plug-ins are programs that can register themself within Gwyddion (for example printing something on standard output) to enable Gwyddion to create plugin menu choice and can be used for data processing (or IO operation).

Generally it is preferable to extend Gwyddion functionality by modules, because modules are dynamic libraries linked directly to Gwyddion at run-time allowing much more versatile interaction with the application, and they are also faster (for the same reason). For example, plug-ins generally cannot make use of existing Gwyddion data processing functions and cannot modify data in-place, a new window is always created for the result. Programming of modules is also no harder than programming of plug-ins, maybe it is even easier (assuming you know C).

Warning The plug-in mechanism is deprecated. It will remain supported in Gwyddion 2.x, however, it will not be extended or improved. The recommended method to extend Gwyddion by routines in another language is to use language bindings, at this moment a Python interface is available. The recommended method to run third-party programs is to write a small specialized C module that knows how to communicate with these programs.

Chapter 4

Data Processing and Analysis

The number and breadth of data manipulation and analysis functions is one of the Gwyddion main strengths. The description of their principles and applications is the purpose of this chapter, with focus on the explanation of how they work and what they calculate or perform. Elements of their user interface are also occasionaly described where it seems useful, nevertheless, it is assumed the reader is familiar with the basic organization of data and controls in Gwyddion, as described in the previous chapter.

4.1 Basic Operations

Value-reading and basic geometrical operations represent the core of any data processing program. Gwyddion offers a wide set of functions for data scaling, rotation, resampling or profile extraction. This section describes these simple but essential functions.

Basic 2D Data Operations

Within basic modules it is possible to perform the following operations with 2D data field:

- Resample the data to chosen dimensions or scale using selected interpolation method by *Data Process* → *Basic Operations* → *Scale*
- Crop the data using the Crop tool either in place or putting the result to a new channel (with option *Create new channel*). With *Keep lateral offsets* option enabled, the top left corner coordinates of the resulting image correspond to the top left corner of the selection, otherwise the top left corner coordinates are set to $(0, 0)$.
- Extend the data with *Data Process* → *Basic Operations* → *Extend*. Extending is essentially the opposite of cropping. Of course, adding more real data around the image borders is only possible by measuring more data. So this function offers, instead, several simple artificial extension methods such as mirrored and unmirrored periodic continuation or repetition of boundary values.
- Rotate data by 90 degrees or by user-specified amount using some of the rotate functions: *Data Process* → *Basic Operations* → *Rotate Clockwise*, *Rotate Anticlockwise* or *Rotate*.
- Flip the data horizontally (i.e. about the vertical axis) and vertically (i.e. about the horizontal axis) with *Data Process* → *Basic Operations* → *Flip Horizontally* and *Flip Vertically*, respectively.
- Flip the data about the centre (i.e. about both axes) with *Data Process* → *Basic Operations* → *Flip Both*.
- Invert the data values using the *Invert Value* function: *Data Process* → *Basic Operations* → *Invert Value*. The values are inverted about the mean value which keeps the mean value unchanged.
- Limit the data range by cutting values outside a specified range by *Data Process* → *Basic Operations* → *Limit Range*. The range can be set numerically or taken from the false color map range previously set using the Color range tool and it is also possible to cut off outliers farther than a chosen multiple of RMS from the mean value.
- Upsample the data to make pixels square with *Data Process* → *Basic Operations* → *Square Samples*. Most scans have pixels with 1:1 aspect ratio, therefore this function has no effect on them.
- Tilt the data by specified gradient or angle using *Data Process* → *Basic Operations* → *Tilt*.
- Change physical dimensions, units or value scales and also lateral offsets using *Data Process* → *Basic operations* → *Dimensions and Units*. This is useful to correct raw data that have been imported with wrong physical scales or as a simple manual recalibration of dimensions and values.

Reading Values

The simplest value reading method is to place the mouse cursor over the point you want to read value of. The coordinates and/or value is then displayed in the data window or graph window status bar.

Read Value Tool

Tool *Read Value* offers more value reading possibilities: it displays coordinates and values of the last point of the data window the mouse button was pressed. It can average the value from a circular neighbourhood around the selected point, this is controlled by option *Averaging radius*. When the radius is 1, the value of a single pixel is displayed (as the simplest method does). Button *Set Zero* shifts the surface to make the current z the new zero level.

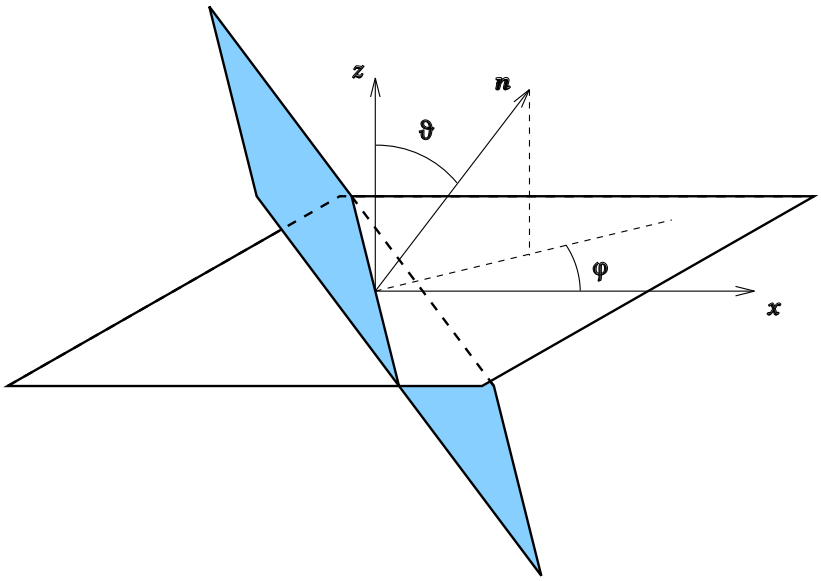
Read Value can also display the inclination of the local facet. *Averaging radius* again determines the radius of the area to use for the plane fit.

Inclinations

In all Gwyddion tools, facet and plane inclinations are displayed as the spherical angles (ϑ, φ) of the plane normal vector.

Angle ϑ is the angle between the upward direction and the normal, this means that $\vartheta = 0$ for horizontal facets and it increases with the slope. It is always positive.

Angle φ is the counter-clockwise measured angle between axis x and the projection of the normal to the xy plane, as displayed on the following figure. For facets it means φ corresponds to the downward direction of the facet.

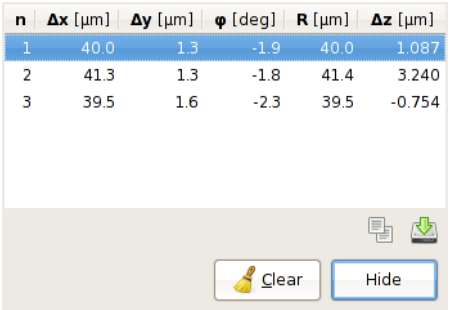


Surface facet (displayed blue) orientation measured as the counterclockwise angle from x-axis to the projection of facet normal vector **n** *to xy plane.*

Distance Tool

Distances and differences can be measured with the *Distance* tool. It displays the horizontal (Δx), vertical (Δy) and total planar (R) distances; the azimuth φ (measured identically to inclination φ) and the endpoint value difference Δz for a set of lines selected on the data.

The distances can be copied to the clipboard or saved to a text file using the buttons below the list.

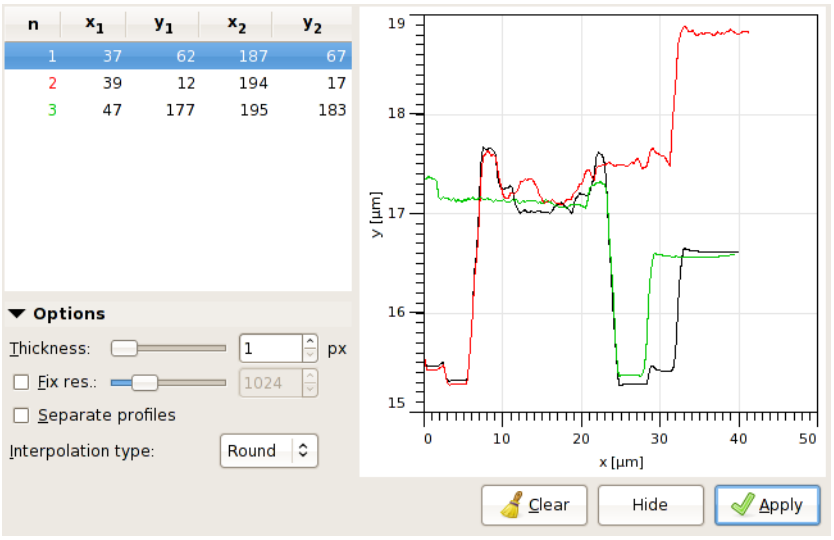


Distance tool with three selected lines.

Profile Extraction

The profile extraction tool can be accessed from the toolbox. You can use mouse to draw several profiles in the image and they can be further moved and adjusted. The dialog includes a live profile graph preview. Profiles can be of different "thickness" which means that more neighbour data perpendicular to profile direction are used for evaluation of one profile point for thicker profiles. This can be very useful for noise suppression while measuring regular objects.

After profiles are chosen, they can be extracted to graphs (separate or grouped in one Graph window) that can be further analysed using Graph functions.



Profile tool with three extracted profiles and expanded options.

The profile curve is constructed from data sampled at regular intervals along the selected line. Values in points that do not lie exactly at pixel centres (which normally occurs for oblique lines) are interpolated using the chosen interpolation mode. Unless an explicit number of samples to take is set using the *Fix res.* option, the number of samples corresponds to the line length in pixels. This means that for purely horizontal or purely vertical lines no interpolation occurs.

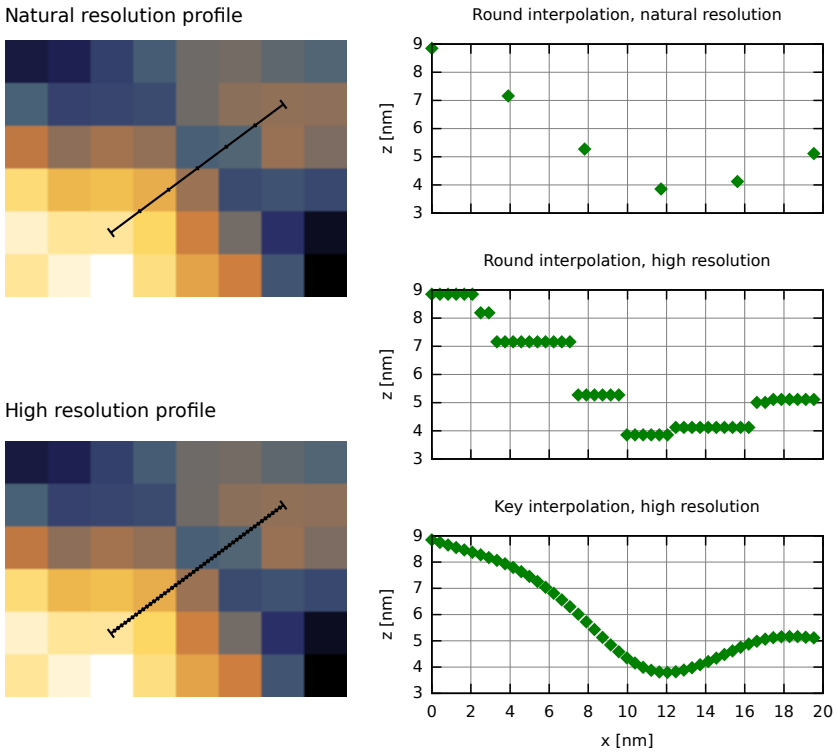


Illustration of data sampling in profile extraction for oblique lines. The figures on the left show the points along the line where the values are read for natural and very high resolution. The graphs on the right show the extracted values. Comparison of the natural and high resolution profiles taken with Round interpolation reveals that indeed natural-resolution curve points form a subset of the high-resolution points. The influence of the interpolation method on values taken in non-grid positions is demonstrated by the lower two graphs, comparing Round and Key interpolation at high resolution.

Radial profiles

It is also possible to extract radial profiles, i.e. angularly averaged shapes of symmetrical surface features, by selecting the *Radial profiles* check box. In this case the abscissa of the extracted graph is the distance from the centre instead of the distance along the line. The origin is in the centre of the selected line.

Although the line can be adjusted manually, finding the best centre for the radial profile manually may be difficult. Therefore, the tool can perform the precise location of the best centre itself. You only need to select the line approximately and then press *Symmetrize* to adjust the currently edited line or *Symmetrize All* to adjust all lines. The lines will be shifted slightly to minimise the differences between line profiles taken in different directions from the centre.

4.2 Interpolation

Most geometrical transformations, such as rotation, scaling or drift compensation utilize or depend on data interpolation. Also some other operations, e.g. profile extraction, can work with values between individual pixels and hence involve interpolation. Since SPM data are relatively coarsely sampled compared to measured details (full images are typically only a few hundred pixels in width and height), the interpolation method used can become critical for proper quantitative analysis of data properties. Gwyddion implements several interpolation methods [1] and the user can choose which method to use for most of the modules using interpolation.

Here, we describe the principles and properties of one-dimensional interpolation methods. All implemented two-dimensional interpolation methods are separable and thus simply composed of the corresponding one-dimensional methods. The following interpolation method are currently available:

Round Round interpolation (also called nearest neighbourhood interpolation) is the simplest method – it just takes rounded value of the expected position and finds therefore the closest data value at integer position. Its polynomial degree is 0, regularity C^{-1} and order 1.

Linear Linear interpolation is a linear interpolation between the two closest data values. The value z at point of relative position x is obtained as

$$z = (1-x)z_0 + xz_1$$

where z_0 and z_1 are values at the preceding and following points, respectively. Its polynomial degree is 1, regularity C^0 and order 2. It is identical to the second-order B-spline.

Key Key interpolation (more precisely Key's interpolation with $a = -1/2$ which has the highest interpolation order) makes use also of values in the before-preceding and after-following points z_{-1} and z_2, respectively. In other words it has support of length 4. The value is then obtained as

$$z = w_{-1}z_{-1} + w_0z_0 + w_1z_1 + w_2z_2$$

where

$$\begin{aligned} w_{-1} &= \left(-\tfrac{1}{2} + (1 - \tfrac{x}{2})x\right)x \\ w_0 &= 1 + (-\tfrac{5}{2} + \tfrac{3}{2}x)x^2 \\ w_1 &= \left(\tfrac{1}{2} + (2 - \tfrac{3}{2}x)x\right)x \\ w_2 &= (-\tfrac{1}{2} + \tfrac{x}{2})x^2 \end{aligned}$$

are the interpolation weights. Key's interpolation degree is 3, regularity C^1 and order 3.

Schaum Schaum interpolation (more precisely fourth-order Schaum) has also support of length 4. The interpolation weights are

$$\begin{aligned} w_{-1} &= -\tfrac{1}{6}x(x-1)(x-2) \\ w_0 &= \tfrac{1}{2}(x^2-1)(x-2) \\ w_1 &= -\tfrac{1}{2}x(x+1)(x-2) \\ w_2 &= \tfrac{1}{6}x(x^2-1) \end{aligned}$$

Its polynomial degree is 3, regularity C^0 and order 4.

NNA Nearest neighbour approximation is again calculated from the closest four data values but unlike all others it is not piecewise-polynomial. The interpolation weights are

$$w_k = \frac{\dfrac{1}{r_k^4}}{\displaystyle\sum_{j=-1}^{2} \frac{1}{r_j^4}},$$

for $k = -1, 0, 1, 2$, where $r_{-1} = 1 + x$, $r_0 = x$, $r_1 = 1 - x$, $r_2 = 2 - x$. Its order is 1.

B-spline The weights are

$$\begin{aligned} w_{-1} &= \tfrac{1}{6}(1-x)^3 \\ w_0 &= \tfrac{2}{3} - x^2(1 - \tfrac{x}{2}) \\ w_1 &= \tfrac{1}{6} + \tfrac{1}{2}x\big(1 + x(1-x)\big) \\ w_2 &= \tfrac{1}{6}x^3 \end{aligned}$$

However, they are not used with directly function values as above, but with interpolation coefficients calculated from function values [1]. Its polynomial degree is 3, regularity C^2 and order 4.

O-MOMS The weights are

$$\begin{aligned} w_{-1} &= \tfrac{4}{21} + \left(-\tfrac{11}{21} + (\tfrac{1}{2} - \tfrac{x}{6})x\right)x \\ w_0 &= \tfrac{13}{21} + \left(\tfrac{1}{14} + (-1 + \tfrac{x}{2})x\right)x \\ w_1 &= \tfrac{4}{21} + \left(\tfrac{3}{7} + (\tfrac{1}{2} - \tfrac{x}{2})x\right)x \\ w_2 &= (\tfrac{1}{42} + \tfrac{1}{6}x^2)x \end{aligned}$$

However, they are not used directly with function values as above, but with interpolation coefficients calculated from function values [1]. Its polynomial degree is 3, regularity C^0 and order 4.

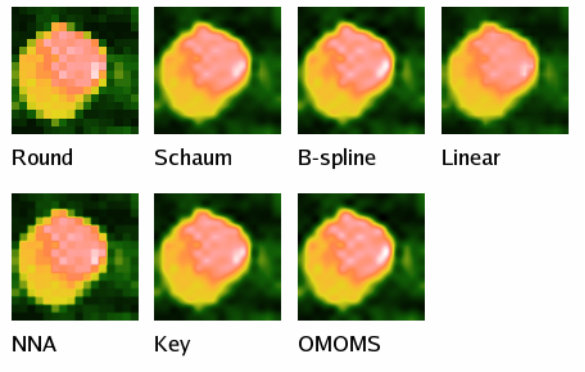


Illustration of the available interpolation types (the original pixels are obvious on the result of Round interpolation). All images have identical false color map ranges.

References

[1] P. Thévenaz, T. Blu, M. Unser: Interpolation revisited. IEEE Transactions on medical imaging, Volume 10, Number 7, July 2000, 739

4.3 Data Leveling and Background Subtraction

Leveling

The data obtained from SPM microscopes are very often not leveled at all; the microscope directly outputs raw data values computed from piezoscanner voltage, strain gauge, interferometer or other detection system values. This way of exporting data enables the user to choose his/her own method of leveling data.

The choice of leveling method should be based on your SPM system configuration. Basically, for systems with independent scanner(s) for each axis, plane leveling should be sufficient. For systems with scanner(s) moving in all three axes (tube scanners) 2nd order polynomial leveling should be used.

Of course, you can use higher order leveling for any data, however, this can supress real features on the surface (namely waviness of the surface) and therefore alter the statistical functions and quantities evaluated from the surface.

Fix Zero and Zero Mean Value

Data Process → *Level* → *Fix Zero*

Data Process → *Level* → *Zero Mean Value*

The simplest modules that are connected with data leveling are Fix Zero and Zero Mean Value that simply set the average height of the data to put the minimum to zero (Fix Zero) or mean value to zero (Zero Mean Value).

Plane Level 

Data Process → *Level* → *Plane Level*

Plane leveling is usually one of the first functions applied to raw SPM data. The plane is computed from all the image points and is subtracted from the data.

If a mask is present plane leveling offers to use the data under mask for the plane fitting, exclude the data under mask or ignore the mask and use the entire data.

Tip You can quickly apply plane leveling by simply right-clicking on the image window and selecting *Level*.

Three Point Leveling Tool 

The *Three Point Leveling* tool can be used for leveling very complicated surface structures. The user can simply mark three points in the image that should be at the same level, and then click *Apply*. The plane is computed from these three points and is subtracted from the data.

Facet Level

Data Process → Level → Facet Level

Facet Level levels data by subtracting a plane similarly to the standard Plane Level function. However, the plane is determined differently: it makes facets of the surface as horizontal as possible. Thus for surfaces with flat horizontal areas it leads to much better results than the standard Plane Level especially if large objects are present.

On the other hand, it is not suitable for some types of surface. These includes random surfaces, data with considerable fine noise and non-topographic images as the method does not work well if the typical lateral dimensions and "heights" differ by many orders.

Similarly to Plane Level, Facet Level can include or exclude the data under mask. This choice is offered only if a mask is present.

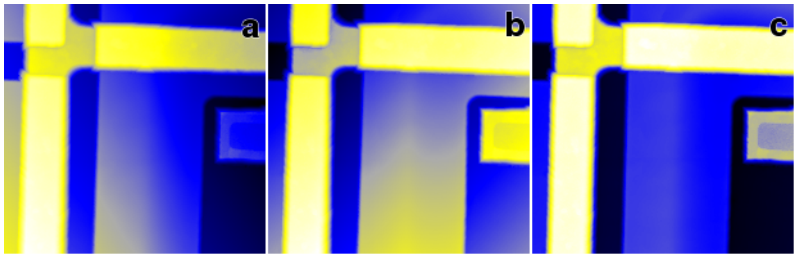
Finding the orientation of the facets is an iterative process that works as follows. First, the variation of local normals is determined:

$$\beta^2 = \frac{1}{N}\sum_{i=1}^{N} \mathbf{n}_i^2$$

where $\mathbf{n}_i$ is the vector of local facet normal (see inclination coordinates) in the i-th pixel. Then the prevalent normal is estimated as

$$\mathbf{n} = \frac{\sum_{i=1}^{N} \mathbf{n}_i \exp\left(-c\frac{\mathbf{n}_i^2}{\beta^2}\right)}{\sum_{i=1}^{N} \exp\left(-c\frac{\mathbf{n}_i^2}{\beta^2}\right)}$$

where $c = 1/20$ is a constant. Subsequently, the plane corresponding to the prevalent normal $\mathbf{n}$ is subtracted and these three steps are repeated until the process converges. The gaussian weighting factors serve to pick a single set of similar local facet normals and converge to their mean direction. Without these factors, the procedure would obviously converge in one step to the overall mean normal – and hence would be completely equivalent to plain plane leveling.



Facet Level example: (a) uncorrected, sloping data; (b) data leveled by standard plane fitting (Plane Level); (c) data leveled by Facet Level.

Level Rotate

Data Process → Level → Level Rotate

Level Rotate behaves similarly to Plane Level, however it does not simply subtract the fitted plane from the data. Instead, this module takes the fitted plane parameters and rotates the image data by a calculated amount to make it lie in a plane. So unlike Plane Level, this module should therefore preserve angle data in the image.

Background Subtraction

Gwyddion has several special modules for background subtraction. All allow you to extract the subtracted background to a separate data window.

Tip For finer control, you can use any of Gwyddion's filtering tools on an image, and then use the Data Arithmetic module to subtract the results from your original image.

Polynomial Background

Data Process → Level → Polynomial Background

Fits data by a polynomial of the given order and subtracts this polynomial. In the *Independent degree* mode the horizontal and vertical polynomial orders can be generally set separately, i.e. the fitted polynomial is

$$\sum_{j=0}^{m}\sum_{k=0}^{n} a_{j,k}x^j y^k$$

where m and n are the selected horizontal and vertical polynomial degrees, respectively. In the *Limited total degree* mode the fitted polynomial is

$$\sum_{j+k\leq n} a_{j,k}x^j y^k$$

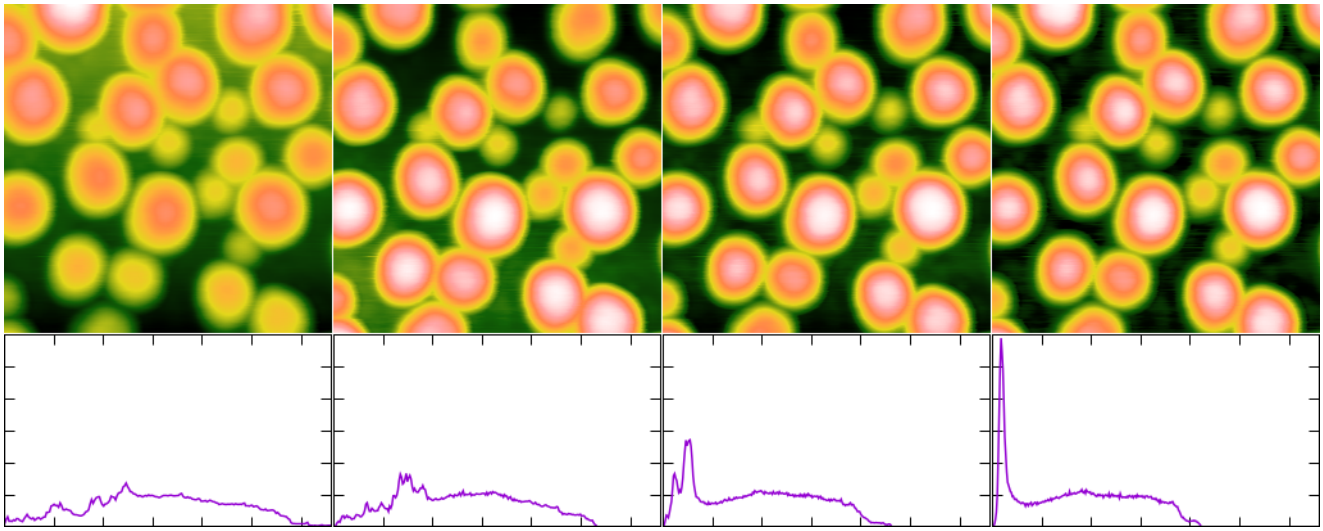
where n is the selected total polynomial degree.

Similarly to Plane Level, polynomial background subtraction can include or exclude the data under mask. This choice is offered only if a mask is present.

Flatten base

Data Process → Level → Flatten Base

When a number of large features are present on a flat base surface the combination of masking, plane, facet and/or polynomial levelling can be used to level the flat base. It can require, however, several steps and trial and error parameter adjustment. *Flatten Base* attempts to perform this levelling automatically using a combination of facet and polynomial levelling with automated masking. It attempts to maximise the sharpness of the height distribution peak corresponding to the flat base surface.



Flatten Base example: original image, levelled using Facet Level, levelled using Plane Level, levelled using Flatten Base. The original image is shown in linear colour scale, the levelled images are shown in adaptive colour scale. The graph below each image shows the corresponding height distribution (with the same axis ranges).

Revolve Arc

Data Process → Level → Revolve Arc

Revolves virtual "arc" of given radius horizontally or vertically over (or under) the data. The envelope of this arc is treated as a background, resulting in removal of features larger than the arc radius (approximately).

Median Level

Data Process → Level → Median Level

Filters data with a median filter using a large kernel and treats the result as background. Only features smaller than approximately the kernel size will be kept.

Note This method can be very slow.

Fit Sphere

Data Process → *Level* → *Fit sphere*

Fits part of sphere surface on the data. Sphere orientation (i.e. centre position) and initial fit values can be preset before fitting. Marquardt-Levenberg fitting routine is used to calculate the result.

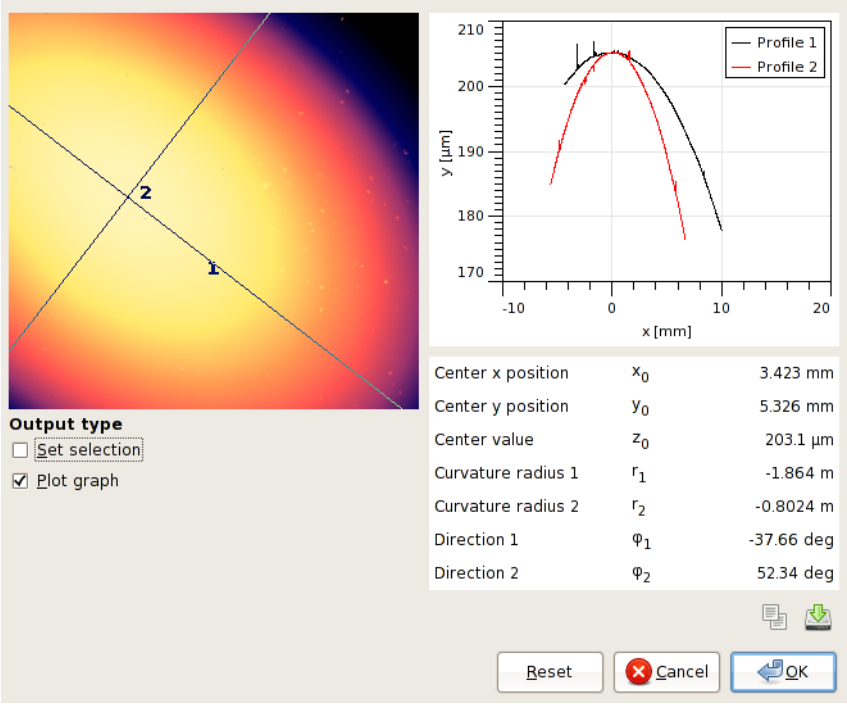
Curvature

Data Process → *Level* → *Curvature*

The global surface curvature parameters are calculated by fitting a quadratic polynomial and finding its main axes. Positive signs of the curvature radii correspond to a concave (cup-like) surface, whereas negative signs to convex (cap-like) surface, mixed signs mean a saddle-like surface.

Beside the parameter table, it is possible to set the line selection on the data to the fitted quadratic surface axes and/or directly read profiles along them. The zero of the abscissa is placed to the intersection of the axes.

Similarly to the background subtraction functions, if a mask is present on the data the module offers to include or exclude the data under mask.



Curvature dialog screenshot showing the strong deflection of a glass plate with a thin film with compressive internal stress.

4.4 Filters

Basic Filters Tool

The Basic Filters tool lets you apply several simple filters to your image. This can be very useful for data denoising; however, the real measured data will get altered in the process, so great care should be taken not to destroy important features of the image.

- Mean filter – takes the mean value of neighbourhood of the filtered value as the value.
- Median filter – takes the median value of neighbourhood of the filtered value as the value.
- Conservative denoise filter – checks whether the value is not extreme within the neighbourhood. If yes, filter substitutes the value by of the next highest (lowest) value.
- Kuwahara filter – is an edge-preserving smoothing filter.

- Minimum filter – also known as erode filter, replaces values by minimum found in neighbourhood.
- Maximum filter – also known as dilate filter, replaces values by maximum found in neighbourhood.
- Dechecker filter – a smoothing filter specially designed to remove checker pattern from the image while preserving other details. It is a convolution filter with kernel

$$w_{\text{dechecker}} = \begin{pmatrix} 0 & 1/144 & -1/72 & 1/144 & 0 \\ 1/144 & -1/18 & 1/9 & -1/18 & 1/144 \\ -1/72 & 1/9 & 7/9 & 1/9 & -1/72 \\ 1/144 & -1/18 & 1/9 & -1/18 & 1/144 \\ 0 & 1/144 & -1/72 & 1/144 & 0 \end{pmatrix}$$

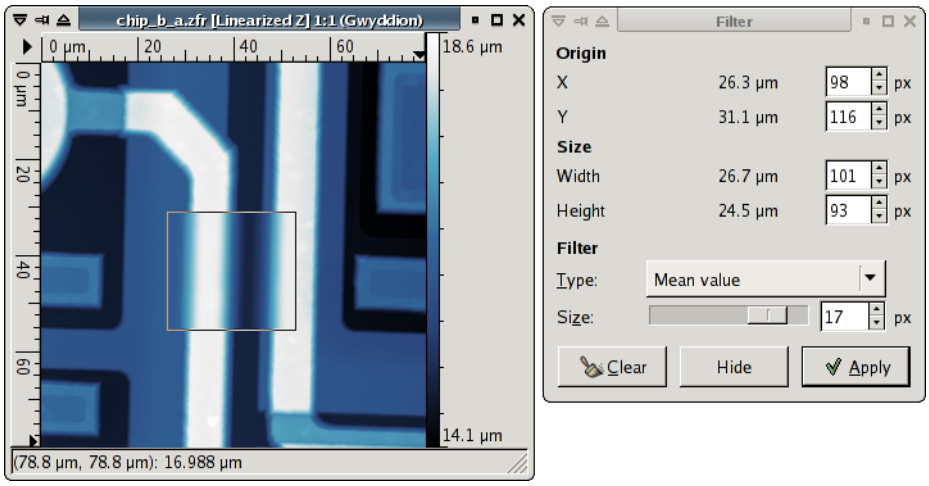
- Gaussian filter – a smoothing filter, the size parameter determines the FWHM (full width at half maximum) of the Gaussian. The relation between FWHM and σ is

$$\text{FWHM} = 2\sqrt{2\ln 2}\,\sigma \approx 2.35482\sigma$$

Tip By default, these filters will be applied to the entire image. However, you can apply a filter to a specific region within your image by selecting it with the mouse. This can be useful for correcting badly measured areas within a good image. To apply a filter to the entire image again, just click once anywhere within the image window.

Moreover, there are more denoising functions in Gwyddion, for example DWT denoising and FFT filtering. For details see section Extended Data Edit.

If you need to only suppress some values in the SPM data that are obviously wrong, you can also try the Mask of Outliers module and the Remove Data Under Mask module. For details see section Data Edit.



Screenshot of filter tool with median filter applied to a rectangular selection

Convolution

Data Process → Integral Transforms → Convolution Filter

Convolutions with arbitrary kernels up to 9×9 can be performed with the Convolution Filter module.

The *Divisor* entry represents a common factor all the coefficients are divided before applying the filter. This allows to use denormalized coefficients that are often nicer numbers. The normalization can be also calculated automatically when *automatic* is checked. When the sum of the coefficients is nonzero, it makes the filter sum-preserving, i.e. it the factor normalizes the sum of coefficients to unity. When the sum of the coefficients is zero, the automatic factor is simply let equal to 1.

Since many filters used in practice exhibit various types of symmetry, the coefficients can be automatically completed according to the selected symmetry type (odd, even). Note the completion is performed on pressing **Enter** in the coefficient entry.

In a fresh installation only a sample Identity filter is present (which is not particularly useful as it does nothing). This filter cannot be modified, to create a new filter use the *New* button on the *Presets* page.

4.5 Presentations

Presentation modules do not modify the data, instead, they output their results into a separate layer displayed on top of the original data. The other data processing modules and tools will still operate on the underlying data. To remove a presentation, right-click on the data window, and select *Remove Presentation*.

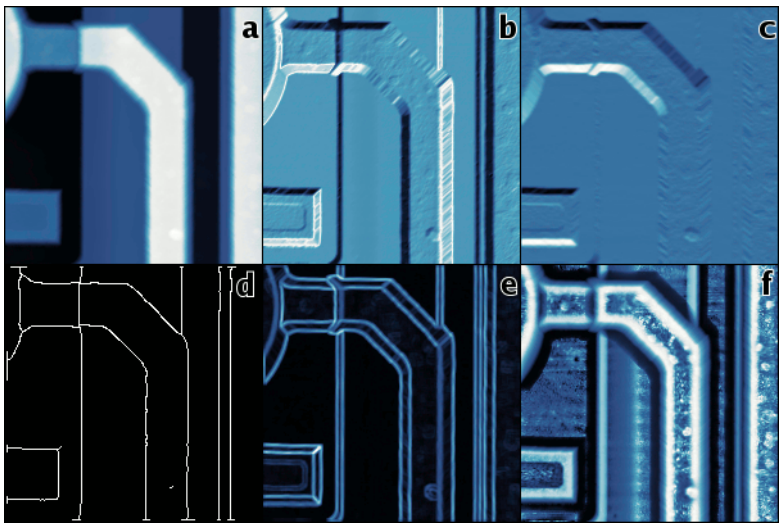
Basic Operations

The *Data Process* → *Presentation* menu contains a few basic presentation operations:

Attach Presentation Attaches another data field as a presentation to the current data. Note that this useful option can be particularly confusing while evaluating anything from the data as all the computed values are evaluated from the underlying data (not from the presentation, even if it looks like the data).

Remove Presentation Removes presentation from the current data window. This is an alternative to the right-click data window menu.

Extract Presentation Extracts presentation from the current data window to a new channel in the same file. In this way one can get presentation data for further processing. Note, however, the extracted data have no absolute scale information as presentation often help to visualize certain features, but the produced values are hard or impossible to assign any physical meaning to. Hence the value range of the new channel is always $[0, 1]$.



Presentation examples: (a) original data, (b) shading, (c) vertical Prewitt gradient, (d) Canny edge detection, (e) local non-linearity edge detection, (f) local contrast improvement.

Shading Presentation

Data Process → *Presentation* → *Shading*

Simple and very useful way of seeing data as illuminated from some direction. The direction can be set by user. It is also possible to mix the shaded and original images for presentational purposes. Of course, the resulting image is meaningless from the physical point of view.

Gradient Detection Presentations

Data Process → *Presentation* → *Gradient*

Sobel horizontal and vertical gradient filter and Prewitt horizontal and vertical gradient filter create similar images as shading, however, they output data as a result of convolution of data with relatively standardized kernel. Thus, they can be used for further presentation processing for example. The kernels for horizontal filters are listed below, vertical kernels differ only by reflection about main diagonal.

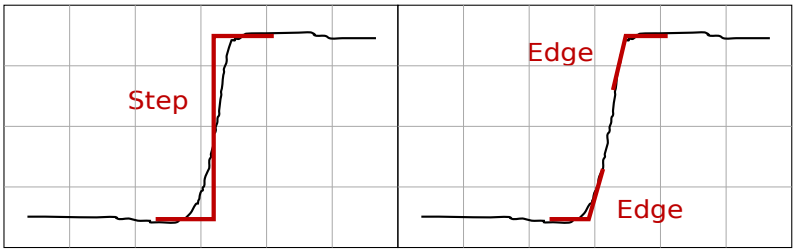
$$w_{\text{Prewitt}} = \begin{pmatrix} 1/3 & 0 & -1/3 \\ 1/3 & 0 & -1/3 \\ 1/3 & 0 & -1/3 \end{pmatrix}, \quad w_{\text{Sobel}} = \begin{pmatrix} 1/4 & 0 & -1/4 \\ 1/2 & 0 & -1/2 \\ 1/4 & 0 & -1/4 \end{pmatrix}$$

Edge Detection Presentations

Data Process → Presentation → Edge Detection

One is often interested in the visualization of the discontinuities present in the image, particularly in discontinuities in the value (zeroth order) and discontinuities in the derivative (first order). Although the methods of location of both are commonly referred to as "edge detection" methods, these are actually quite different, therefore we will refer to the former as to step detection and to the latter as to edge detection. Methods for the detection of more specific features, e.g. corners, are commonly used too, these methods usually are of order zero.

The order of a discontinuity detection method can be easily seen on its output as edge detection methods produce typical double edge lines at value discontinuities as is illustrated in the following figure. While the positions of the upper and lower edge in an ideal step coincide, real-world data tend to actually contain two distinct edges as is illustrated in the picture. In addition, finding two edges on a value step, even an ideally sharp one, is often an inherent feature of edge detection methods.



Step versus edge in one dimension.

The following step and edge detection functions are available in Gwyddion (the later ones are somewhat experimental, on the other hand they usually give better results than the well-known algorithms):

Canny Canny edge detector is a well-known step detector that can be used to extract the image of sharp value discontinuities in the data as thin single-pixel lines.

Laplacian of Gaussians Laplacian presents a simple convolution with the following kernel (that is the limit of discrete Laplacian of Gaussians filter for $\sigma \to 0$):

$$w_{\text{laplace}} = \begin{pmatrix} 0 & 1/4 & 0 \\ 1/4 & -1 & 1/4 \\ 0 & 1/4 & 0 \end{pmatrix}$$

Zero Crossing Zero crossing step detection marks lines where the result of Laplacian of Gaussians filter changes sign, i.e. crosses zero. The FWHM (full width half maximum) of the Gaussians determines the level of details covered. Threshold enables to exclude sign changes with too small absolute value of the neighbour pixels, filtering out fine noise. Note, however, that for non-zero threshold the edge lines may become discontinuous.

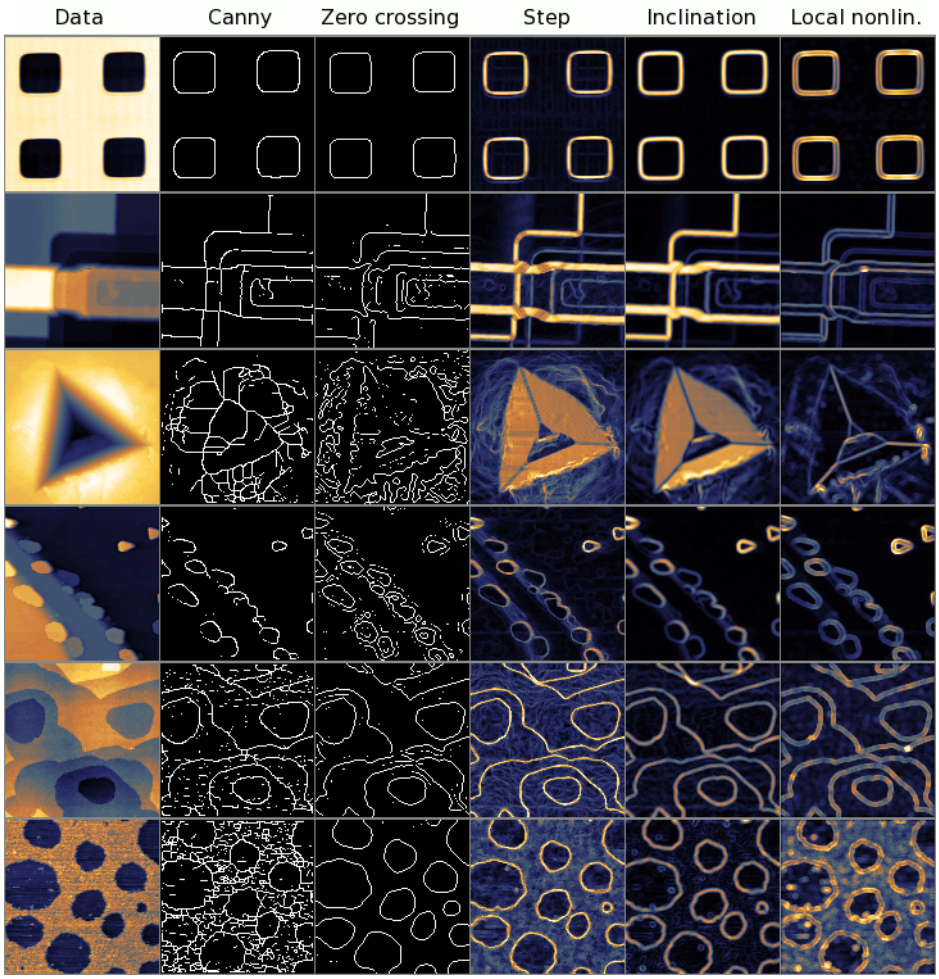
Step A step detection algorithm providing a good resolution, i.e. sharp discontinuity lines, and a good dynamic range while being relatively insensitive to noise. The principle is quite simple: it visualizes the square root of the difference between the 2/3 and 1/3 quantiles of the data values in a circular neighbourhood of radius 2.5 pixels centered around the sample.

RMS This step detector visualizes areas with high local value variation. The root mean square of deviations from the mean value of a circular neighbourhood of radius 2.5 pixels centered around each sample is calculated and displayed.

RMS Edge This function essentially postprocesses RMS output with a filter similar to Laplacian to emphasize boundaries of areas with high local value variation. Despite the name it is still a step detector.

Local Non-Linearity An edge detector which visualizes areas that are locally very non-planar. It fits a plane through a circular neighbourhood of radius 2.5 pixels centered around each sample and then it calculates residual sum of squares of this fit reduced to plane slope, i.e. divided by $1 + b_x^2 + b_y^2$ where b_x and b_y are the plane coefficients in x and y directions, respectively. The square root is then displayed.

Inclination Visualizes the angle ϑ of local plane inclination. Technically this function belongs among step detectors, however, the accentuation of steps in its output is not very strong and it is more intended for easy visual comparison of different slopes present in the image.



Comparison of step and edge detection methods on several interesting, or typical example data. Canny and Zero crossing are step detectors that produce one pixel wide edge lines, Step and Inclination are step detectors with continuous output, Local nonlinearity is an edge detector – the edge detection can be easily observed on the second and third row. Note zero crossing is tunable, it parameters were chosen to produce reasonable output in each example.

Local Contrast

Data Process → *Presentation* → *Local Contrast*

A method to visualize features in areas with low and high value variation at the same time. This is achieved by calculation of local value range, or variation, around each data sample and stretching it to equalize this variation over all data.

Rank

Data Process → *Presentation* → *Rank*

An alternative local contrast enhancement method. It is an equalising high-pass filter, somewhat complementary to the median filter. Each pixel value is transformed to its rank among all values from a certain neihbourhood. The neighbourhood radius can be specified as *Kernel size*.

The net effect is that all local maxima are equalised to the same maximum value, all local minima to the same minimum value, and values that are neither maxima nor minima are transformed to the range between based on their rank. Since the output of the filter with radius r can contain at most $\pi(r+1/2)^2$ different values (approximately), the filter also leads to value discretisation, especially for small kernel sizes.

Logscale

Data Process → *Presentation* → *Logscale*

Logarithmic scale is used for false colors data presentation.

SEM Image

Data Process → *Presentation* → *SEM Image*

The function renders as SEM image-like presentation from a topographical image using the simplest possible Monte Carlo method. For each surface pixel, a number of lines originating from this pixel are chosen with random directions and Gaussian length distribution. The standard deviation of the Gaussian distribution is controlled with the *Integration radius* parameters. If the other end of the line hits free space, the lightness in the origin pixel is increased. If it hits material, i.e. height value below the surface at the endpoint, the lightness is decreased. More precisely, this describes the *Monte Carlo* method. The number of lines can be controlled with the *Quality* parameter. Equivalently, the same intensity can be calculated by direct integration over all pixels within a circular neighbourhood. This corresponds to the *Integration* method.

Since even this simple computation can take a long time, it is useful to consider how its speed depends on the settings. The computation time for *Integration* depends only on the integration radius. The computation time for *Monte Carlo*, on the other hand, depends essentially only on *Quality* (there is also some dependence on the local topography).

4.6 Data Edit and Correction

There are several modules that enable direct or indirect editing of the SPM data. In principal, most of the data processing modules change the data in one way or another. However, in this section we would like to describe the modules and tools that are specifically designed to correct local defects in an image. The functions below remove "bad" data from an image, and then fill it in using an interpolation algorithm.

Align Rows 

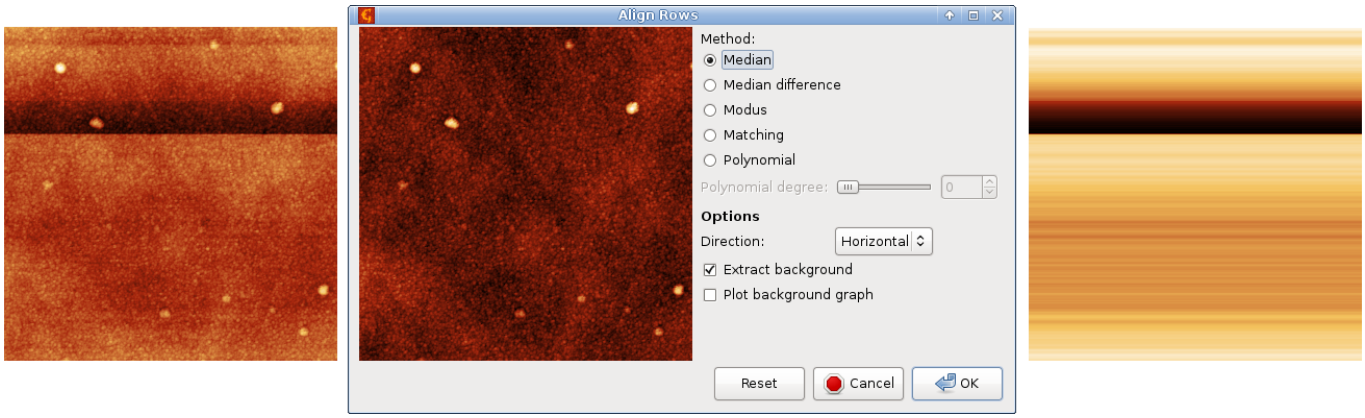
Profiles taken in the fast scanning axis (usually x-axis) can be mutually shifted by some amount or have slightly different slopes. The basic line correction function *Data Process* → *Correct Data* → *Align Rows* deals with this type of discrepancy using several different correction algorithms:

- The most basic correction methods are based on finding a representative height of each scan line and subtracting it, thus moving the lines to the same height. These include *Median* and *Modus* that subtract the corresponding quantities from each row.
- The *Polynomial* method is similar, except polynomials is subtracted in general, not just constant values. For polynomial degree of 0 the mean value of each row is subtracted. Degree 1 means removal of linear slopes, degree 2 bow removal, etc.
- *Median difference* shifts the lines so that the median of differences (between vertical neighbour pixels) becomes zero, instead of the difference of medians. Therefore it better preserves large features while it is more sensitive to completely bogus lines.
- Finally, *Matching* minimizes a certain line difference function that gives more weight to flat areas and less weight to areas with large slopes. This algorithm is somewhat experimental but it may be useful sometimes.

Similarly as in the two-dimensional polynomial levelling, the background, i.e. values subtracted from individual rows can be extracted to another image. Or plotted in a graph since the value is the same for the entire row.

The line correction function support masking, allowing the exclusion of large features that could distract the corection algorithms. The masking options are offered only if a mask is present though. Note the Path level tools described below offers a different method of choosing the image parts important for alignment. It can be more convenient in some cases.

Tip Use **Ctrl-F** (*Repeat Last*) to run the line correction with the same settings on several images without going through the dialogue.



Line correction example: an image with defects (left), the row alignment dialogue previewing the median correction (centre), and extracted row background (right). Note the false colour scales are not the same in the three images.

Step Line Correction

Function *Step Line Correction* attempts to deal with shifts that may occur in the middle of a scan line. It tries to identify misaligned segments within the rows and correct the height of each such segment individually. Therefore it is often able to correct data with discontinuities in the middle of a row. This function is somewhat experimental and the exact way it works can be subject to futher changes.

Remove Spots Tool

The Remove Spots tool can be used for removing very small parts of the image that are considered a scanning error, dust particle or anything else that should not be present in the data. Note that doing so can dramatically alter the resulting statistical parameters of the surface, so be sure not to remove things that are really present on the surface.

While using this tool you can pick up position of the spot to magnify its neighbourhood in the tool window. Then, in the tool window, select a rectangle around the area that should be removed. You can then select one of several interpolation methods for creating data in place of the former "spot":

- Hyperbolic flatten - uses information from selected area boundaries to interpolate the information inside area.
- Pseudo-Laplace and Laplace solver - solves Laplace equation to calculate data inside area; the boundary is treated as virtual source.
- Fractal correction - uses whole image to determine fractal dimension. Then tries to create randomly rough data that have the same fractal dimension and put them into the area.

Clicking *Apply* will execute the selected algorithm.

Note Spot removal will only work for regions of size 64 × 64 pixels or smaller. To remove larger regions, create a mask using the Mask Editor tool, then use *Data Process* → *Correct Data* → *Remove Data Under Mask*.

Remove Grains Tool

This simple tool removes manually selected connected parts of mask or interpolates the data under them, or possibly both. The part of mask to remove is selected by clicking on it with left mouse button.

Remove Scars

Data Process → *Correct Data* → *Remove Scars*

Scars (or stripes, strokes) are parts of the image that are corrupted by a very common scanning error: local fault of the closed loop. Line defects are usually parallel to the fast scanning axis in the image. This function will automatically find and remove these scars, using neighbourhood lines to "fill-in" the gaps. The method is run with the last settings used in Mark Scars.

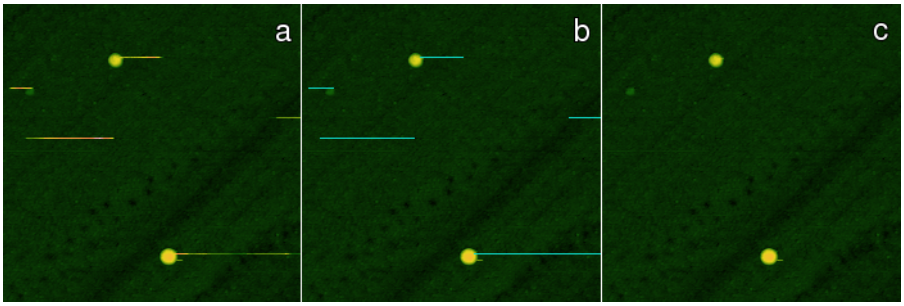
Mark Scars

Data Process → Correct Data → Mark Scars

Similarly, the `Mark Scars` module can create a mask of the points treated as scars. Unlike Remove Scars which directly interpolates the located defects, this module lets you interactively set several parameters which can fine-tune the scar selection process:

- Maximum width – only scars that are as thin or thinner than this value (in pixels) will be marked.
- Minimum length – only scars that are as long or longer than this value (in pixels) will be marked.
- Hard threshold – the minimum difference of the value from the neighbouring upper and lower lines to be considered a defect. The units are relative to image RMS.
- Soft threshold – values differing at least this much do not form defects themselves, but they are attached to defects obtained from the hard threshold if they touch one.
- Positive, Negative, Both – the type of defects to remove. Positive means defects with outlying values above the normal values (peaks), negative means defects with outlying values below the normal values (holes).

After clicking *Ok* the new scar mask will be applied to the image. Other modules or tools can then be run to edit this data.



Scars marking and removal example: (a) original data with defects, (b) data with marked deffects, (c) corrected data.

Remove Data Under Mask

Data Process → Correct Data → Remove Data Under Mask

This function substitutes the data under the mask by the solution of solving the Laplacian equation. The data values around the masked areas define the boundary conditions. The solution is calculated iteratively and it can take some time to converge.

Fractal Correction

Data Process → Correct Data → Fractal Correction

The Fractal Correction module, like the Remove Data Under Mask module, replaces data under the mask. However, it uses a different algorithm to come up with the new data: The fractal dimension of the whole image is first computed, and then the areas under the mask are substituted by a randomly rough surface having the same fractal dimension. The root mean square value of the height irregularities (roughness) is not changed by using this module.

Note This calculation can take some time, so please be patient.

Warning Running this module on data that do not have fractal properties can cause really unrealistic results and is strictly not recommended.

Mask of Outliers

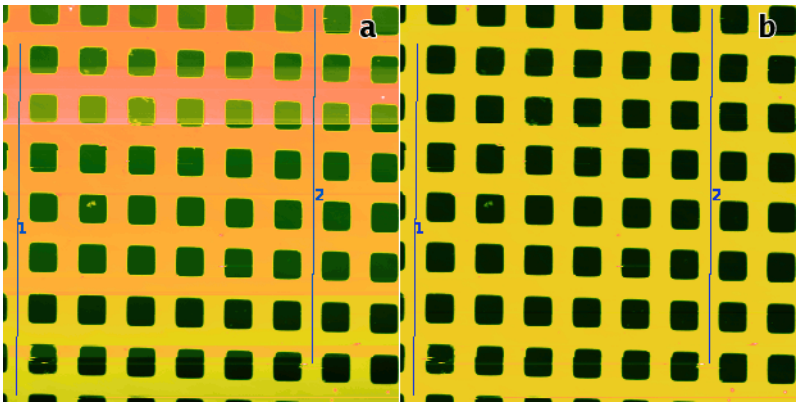
Data Process → *Correct Data* → *Mask of Outliers*

This module creates mask of areas in the data that not pass the 3σ criterion. All the values above and below this confidence interval are marked in mask and can be edited or processed by other modules afterwards.

Path Leveling Tool

The *Path Leveling* tool can be used to correct the heights in an arbitrary subset of rows in complicated images.

First, one selects a number of straight lines on the data. The intersections of these lines with the rows then form a set of points in each row that is used for leveling. The rows are moved up or down to minimize the difference between the heights of the points of adjacent rows. Rows that are not intersected by any line are not moved (relatively to neighbouring rows).



Path Level example: (a) uncorrected data with steps that the automated method may fail to correct, two suitable leveling lines are selected; (b) the result of Path Level application with line width 5.

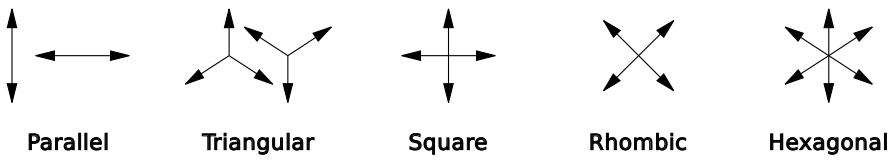
Unrotate

Data Process → *Correct Data* → *Unrotate*

Unrotate can automatically make principal directions in an image parallel with horizontal and/or vertical image edges. For that to work, the data need to have some principal directions, therefore it is most useful for scans of artifical and possibly crystallic structures.

The rotation necessary to straighten the image – displayed as *Correction* – is calculated from peaks in angular slope distribution assuming a prevalent type of structure, or symmetry. The symmetry can be estimated automatically too, but it is possible to select a particular symmetry type manually and let the module calculate only corresponding rotation correction. Note if you assume a structure type that does not match the actual structure, the calculated rotation is rarely meaningful.

It is recommended to level (or facet-level) the data first as overall slope can skew the calculated rotations.



Orientations of prevalent directions corresponding to Unrotate symmetry types.

The assumed structure type can be set with *Assume* selector. Following choices are possible:

Detected Automatically detected symmetry type, displayed above as *Detected*.

Parallel Parallel lines, one prevalent direction.

Triangular Triangular symmetry, three prevalent directions (unilateral) by 120 degrees.

Square Square symmetry, two prevalent directions oriented approximately along image sides.

Rhombic Rhombic symmetry, two prevalent directions oriented approximately along diagonals. The only difference from Square is the preferred diagonal orientation (as opposed to parallel with sides).

Hexagonal Hexagonal symmetry, three prevalent directions (bilateral) by 120 degrees.

4.7 Extended Data Edit

This section presents extended modules designed for editing (correcting) SPM data. Using simple data editing tools presented in chapter Data Edit and Correction it is possible to correct many local scanning defects that can be found on SPM images. There are also many error sources within SPM methods that lead to global errors, like low frequencies modulated on the data or data drift in the slow scanning axis.

Drift Compensation

Data Process → Correct Data → Compensate Drift

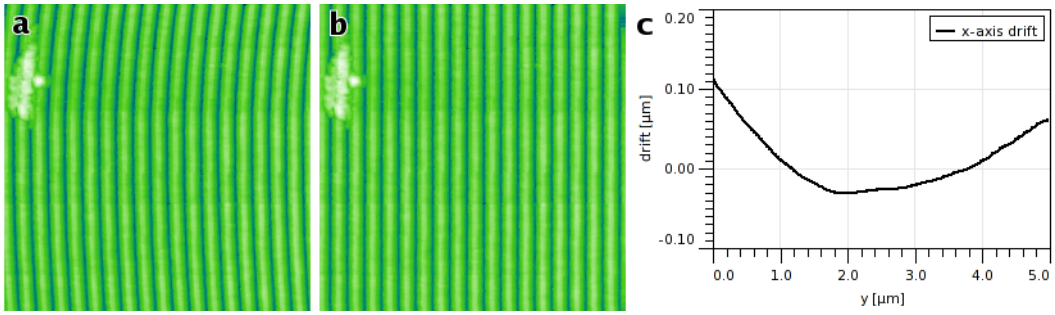
Compensate Drift calculates and/or corrects drift in the fast scanning axis (horizontal). This adverse effect can be caused by thermal effects or insufficient mechanical rigidity of the measuring device.

The drift graph, which is one of possible outputs, represents the horizontal shift of individual rows compared to a reference row (which could be in principle chosen arbitrarily, in practice the zero shift is chosen to minimize the amount of data sticking out of the image after compensation), with the row y-coordinate on the abscissa.

The drift is determined in two steps:

1. A mutual horizontal offset is estimated for each couple of rows not more distant than *Search range*. It is estimated as the offset value giving the maximum mutual correlation of the two rows. Thus a set of local row drift estimations is obtained (together with the maximum correlation scores providing an estimate of their actual similarity).
2. Global offsets are calculated from the local ones. At present the method is very simple as it seems sufficient in most cases: local drift derivatives are fitted for each row onto the local drift estimations and the global drift is then obtained by integration (i.e. summing the local drifts).

Option *Exclude linear skew* subtracts the linear term from the calculated drift, it can be useful when the image is anisotropic and its features are supposed to be oriented in a direction not paralled to the image sides.



Drift correction example: (a) original data exhibiting strong drift in the fast scan axis, (b) corrected data, (c) calculated drift graph.

1D FFT Filter

Data Process → Correct Data → 1D FFT Filtering

One excellent way of removing frequency based of noise from an image is to use Fourier filtering. First, the Fourier transform of the image is calculated. Next, a filter is applied to this transform. Finally, the inverse transform is applied to obtain a filtered image. Gwyddion uses the Fast Fourier Transform (or FFT) to make this intensive calculation much faster.

Within the 1D FFT filter the frequencies that should be removed from spectrum (suppress type: null) or suppressed to value of neighbouring frequencies (suppress type: suppress) can be selected by marking appropriate areas in the power spectrum graph. The selection can be inverted easily using the Filter type choice. 1D FFT filter can be used both for horizontal and vertical direction.

2D FFT Filter

Data Process → Correct Data → 2D FFT Filtering

2D FFT filter acts similarly as the 1D variant (see above) but using 2D FFT transform. Therefore, the spatial frequencies that should be filtered must be selected in 2D using mask editor. As the frequencies are related to center of the image (corresponding to zero frequency), the mask can be snapped to the center (coordinate system origin) while being edited. There are also different display and output modes that are self-explanatory – image or FFT coefficients can be outputted by module (or both).

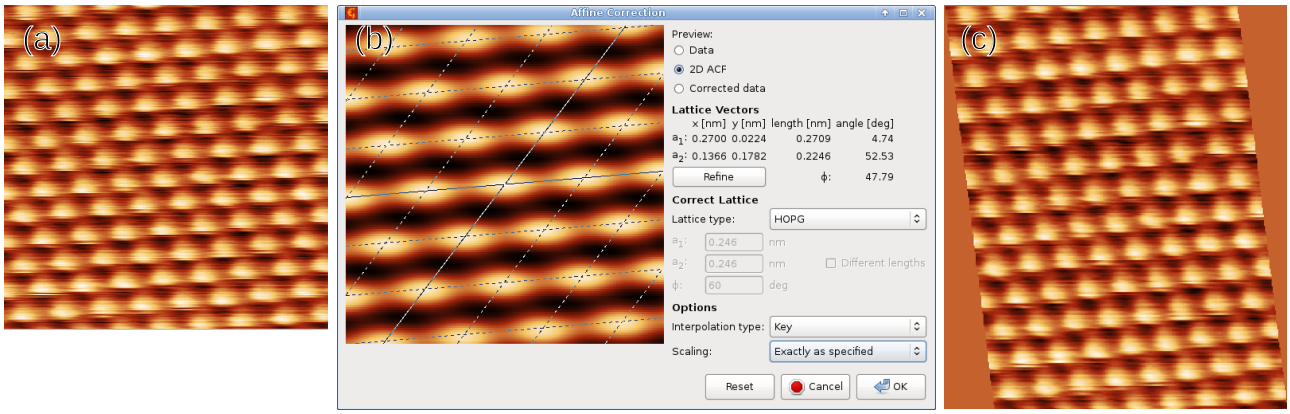
Affine Distortion

Data Process → Correct Data → Affine Distortion

Affine distortion in the horizontal plane caused by thermal drift is common for instance in STM. If the image contains a regular structure, for instance an atomic lattice of known parameters, the distortion can be easily corrected using this function.

The affine distortion correction requires to first select the distorted lattice in the image. This is done by moving the lattice selection on the preview with mouse until it matches the regular features present in the image. For images of periodic lattices, it is usually easier to select the lattice in the autocorrelation function image (*2D ACF*). Also, only a rough match needs to be found manually in this case. Button *Refine* refines the selected lattice vectors to the nearest maxima in autocorrelation function with subpixel precision.

The correct lengths of the lattice vectors a_1 and a_2 and the angle φ between them, entered to the dialog, determine the affine transformation to perform. A few common lattice types (such as HOPG surface) are offered predefined, but it is possible to enter arbitrary lengths and angle.



Affine correction example: (a) original image exhibiting an affine distortion, (b) correction dialog with the lattice selected on the two-dimensional autocorrelation, (c) corrected image.

It should be noted that the correction method described above causes all lateral scale information in the image to be lost because the new lateral scale is fully determined by the correct lattice vectors. This is usually the best option for STM images of known atomic lattices, however, for a general skew or affine correction it can be impractical. Therefore, the dialog offers three different scaling choices:

Exactly as specified Lattice vectors in the corrected image will have the specified lengths and angle between them. Scale information of the original image is discarded completely.

Preserve area Lattice vectors in the corrected image will have the specified ratio of lengths and angle between them. However, the overall scale is calculated as to make the affine transformation area-preserving.

Preserve X scale Lattice vectors in the corrected image will have the specified ratio of lengths and angle between them. However, the overall scale is calculated as to make the affine transformation preserve the original x-axis scale. This is somewhat analogous to the scale treatment in Drift compensation.

Polynomial Distortion

Data Process → Correct Data → Polynomial Distortion

General distortion in the horizontal plane can be compensated, or created, with Polynomial distortion. It performs transforms that can be expressed as

$$x_{\text{old}} = P_x(x_{\text{new}}, y_{\text{new}}),$$
$$y_{\text{old}} = P_y(x_{\text{new}}, y_{\text{new}}),$$

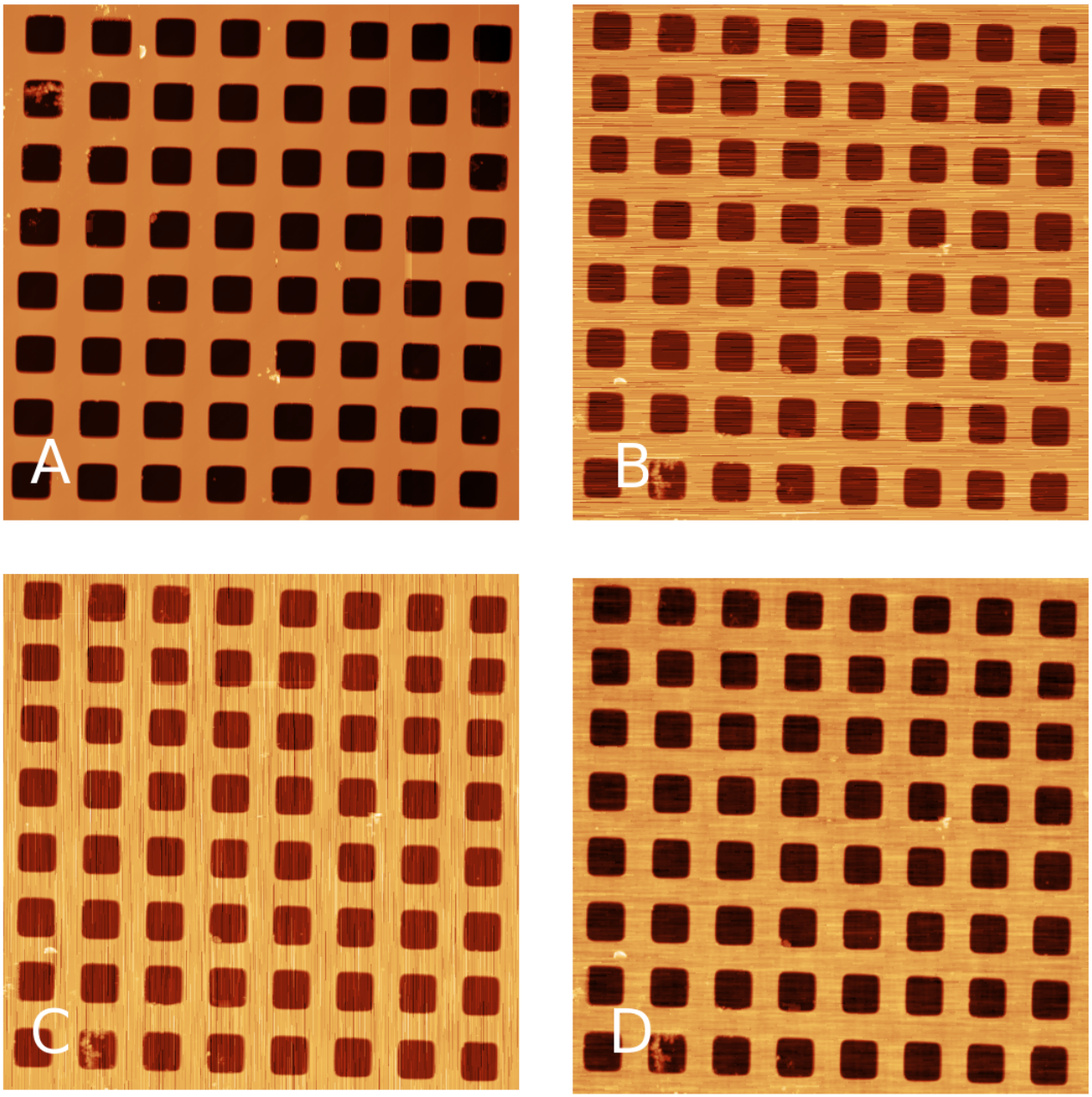
where P_x and P_y are polynomials up to the third total order with user-defined coefficients. Note the direction of the coordinate transform – the reverse direction would not guarantee an unambiguous mapping.

The polynomial coefficients are entered as scale-free, i.e. as if the coordinate ranges were always [0, 1]. If *Instant updates* are enabled, pressing Enter in a coefficient entry (or just leaving moving keyboard focus elsewhere) updates the preview.

XY denoising

Data Process → Multidata → XY denoise

Calculates denoised image on the basis of two measurements of the same area – one performed in x direction and one in y direction (and rotated back to be aligned the same way as the x direction one). It is based on work of E. Anguiano and M. Aguilar (see [1]). Module performs FFT of both images, combines information from both images in reciprocal space, and then performs backward FFT in order to get denoised image. It is useful namely for large scars and fast scanning axis stripes removal.



XY denoise procedure simulation: A) original data, B) simulated measurement in x axis, C) simulated measurement in y axis, D) denoised image.

References

[1] E. Anguiano and M. Aguilar, Ultramicroscopy, 76 (1999) 47

4.8 Statistical Analysis

While analyzing randomly rough surfaces we often need a statistical approach to determine some set of representative quantities. Within Gwyddion, there are several ways of doing this. In this section we will explain the various statistical tools and modules offered in Gwyddion, and also present the basic equations which were used to develop the algorithms they utilize.

Scanning probe microscopy data are usually represented as a two-dimensional data field of size $N \times M$, where N and M are the number of rows and columns of the data field, respectively. The real area of the field is denoted as $L_x \times L_y$ where L_x and L_y are the dimensions along the respective axes. The sampling interval (distance between two adjacent points within the scan) is denoted

Δ. We assume that the sampling interval is the same in both the x and y direction and that the surface height at a given point (x,y) can be described by a random function $\xi(x,y)$ that has given statistical properties.

Note that the AFM data are usually collected as line scans along the x axis that are concatenated together to form the two-dimensional image. Therefore, the scanning speed in the x direction is considerably higher than the scanning speed in the y direction. As a result, the statistical properties of AFM data are usually collected along the x profiles as these are less affected by low frequency noise and thermal drift of the sample.

Statistical Quantities Tool

Statistical quantities include basic properties of the height values distribution, including its variance, skewness and kurtosis. The quantities accessible within Gwyddion by means of the *Statistical Quantities* tool are as follows:

- Mean value, minimum, maximum and median.
- RMS value of the height irregularities: this quantity is computed from data variance.
- Grain-wise RMS value which differs from the ordinary RMS only if masking is used. The mean value is then determined for each grain (contiguous part of mask or inverted mask, depending on masking type) separately and the variance is then calculated from these per-grain mean values.
- R_a value of the height irregularities: this quantity is similar to RMS value with the only difference in exponent (power) within the data variance sum. As for the RMS this exponent is $q = 2$, the R_a value is computed with exponent $q = 1$ and absolute values of the data (zero mean).
- Height distribution skewness: computed from 3rd central moment of data values.
- Height distribution kurtosis: computed from 4th central moment of data values.
- Projected surface area and surface area: computed by simple triangulation.
- Mean inclination of facets in area: computed by averaging normalized facet direction vectors.
- Variation, which is calculated as the integral of the absolute value of the local gradient.

Tip By default, the Statistical Quantities tool will display figures based on the entire image. If you would like to analyze a certain region within the image, simply click and drag a rectangle around it. The tool window will update with new numbers based on this new region. If you want you see the stats for the entire image again, just click once within the data window and the tool will reset.

More precisely, RMS (σ), skewness (γ_1), and kurtosis (γ_2) are computed from central moments of i-th order μ_i according to the following formulas:

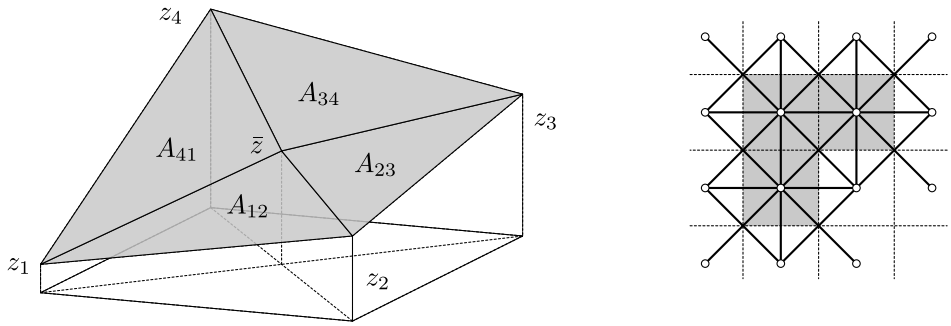
$$\sigma = \mu_2^{1/2}\,, \quad \gamma_1 = \frac{\mu_3}{\mu_2^{3/2}}\,, \quad \gamma_2 = \frac{\mu_4}{\mu_2^2} - 3$$

The surface area is estimated by the following method. Let z_i for $i = 1,2,3,4$ denote values in four neighbour points (pixel centres), and h_x and h_y pixel dimensions along corresponding axes. If an additional point is placed in the centre of the rectangle which corresponds to the common corner of the four pixels (using the mean value of the pixels), four triangles are formed and the surface area can be approximated by summing their areas. This leads to the following formulas for the area of one triangle (top) and the surface area of one pixel (bottom):

$$A_{12} = \frac{h_x h_y}{4}\sqrt{1 + \left(\frac{z_1 - z_2}{h_x}\right)^2 + \left(\frac{z_1 + z_2 - 2\bar{z}}{h_y}\right)^2}$$

$$A = A_{12} + A_{23} + A_{34} + A_{41}$$

The method is now well-defined for inner pixels of the region. Each value participates on eight triangles, two with each of the four neighbour values. Half of each of these triangles lies in one pixel, the other half in the other pixel. By counting in the area that lies inside each pixel, the total area is defined also for grains and masked areas. It remains to define it for boundary pixels of the whole data field. We do this by virtually extending the data field with a copy of the border row of pixels on each side for the purpose of surface area calculation, thus making all pixels of interest inner.



Surface area calculation triangulation scheme (left). Application of the triangulation scheme to a three-pixel masked area (right), e.g. a grain. The small circles represent pixel-center vertices z_i, thin dashed lines stand for pixel boundaries while thick lines symbolize the triangulation. The surface area estimate equals to the area covered by the mask (grey) in this scheme.

Statistical Functions Tool

One-dimensional statistical functions can be accessed by using the *Statistical Functions* tool. Within the tool window, you can select which function to evaluate using the selection box on the left labeled *Output Type*. The graph preview will update automatically. You can select in which direction to evaluate (horizontal or vertical), but as stated above, we recommend using the fast scanning axis direction. You can also select which interpolation method to use. When you are finished, click *Apply* to close the tool window and output a new graph window containing the statistical data.

Tip Similar to the Statistical Quantities tool, this tool evaluates for the entire image by default, but you can select a sub-region to analyze if you wish.

Height and Angle Distribution Functions

The simplest statistical functions are the height and slope distribution functions. These can be computed as non-cumulative (i.e. densities) or cumulative. These functions are computed as normalized histograms of the height or slope (obtained as derivatives in the selected direction – horizontal or vertical) values. In other words, the quantity on the abscissa in "angle distribution" is the tangent of the angle, not the angle itself.

The normalization of the densities $\rho(p)$ (where p is the corresponding quantity, height or slope) is such that

$$\int_{-\infty}^{\infty} \rho(p)\,\mathrm{d}p = 1$$

Evidently, the scale of the values is then independent on the number of data points and the number of histogram buckets. The cumulative distributions are integrals of the densities and they have values from interval $[0,1]$.

First-Order vs. Second-Order Quantities

The height and slope distribution quantities belong to the first-order statistical quantities, describing only the statistical properties of the individual points. However, for the complete description of the surface properties it is necessary to study higher order functions. Usually, second-order statistical quantities observing mutual relationship of two points on the surface are employed. These functions are namely the autocorrelation function, the height-height correlation function, and the power spectral density function. A description of each of these follows:

Autocorrelation Function

The autocorrelation function is given by

$$\begin{aligned} G(\tau_x,\tau_y) &= \iint_{-\infty}^{\infty} z_1 z_2 w(z_1,z_2,\tau_x,\tau_y)\,\mathrm{d}z_1\,\mathrm{d}z_2 \\ &= \lim_{S\to\infty} \frac{1}{S} \iint_S \xi(x_1,y_1)\,\xi(x_1+\tau_x,y_1+\tau_y)\,\mathrm{d}x_1\,\mathrm{d}y_1 \end{aligned}$$

where z_1 and z_2 are the values of heights at points (x_1,y_1), (x_2,y_2); furthermore, $\tau_x = x_1 - x_2$ and $\tau_y = y_1 - y_2$. The function $w(z_1,z_2,\tau_x,\tau_y)$ denotes the two-dimensional probability density of the random function $\xi(x,y)$ corresponding to points (x_1,y_1), (x_2,y_2), and the distance between these points τ.

From the discrete AFM data one can evaluate this function as

$$G(m,n) = \frac{1}{(N-n)(M-m)} \sum_{l=1}^{N-n} \sum_{k=1}^{M-m} z_{k+m,l+n} z_{k,l}$$

where $m = \tau_x/\Delta x$, $n = \tau_y/\Delta y$. The function can thus be evaluated in a discrete set of values of τ_x and τ_y separated by the sampling intervals Δx and Δy, respectively. The two-dimensional autocorrelation function can be calculated with *Data Process* → *Statistics* → *2D Autocorrelation*.

For AFM measurements, we usually evaluate the one-dimensional autocorrelation function based only on profiles along the fast scanning axis. It can therefore be evaluated from the discrete AFM data values as

$$G_x(m) = G(m,0) = \frac{1}{N(M-m)} \sum_{l=1}^{N} \sum_{k=1}^{M-m} z_{k+m,l} z_{k,l}$$

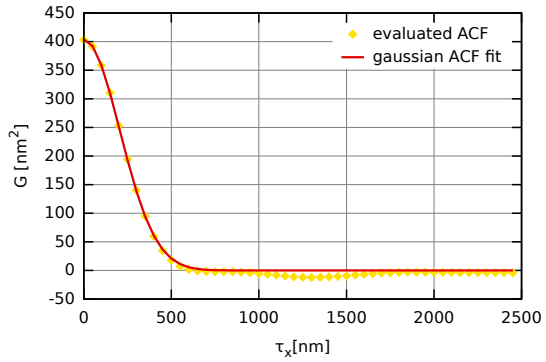
The one-dimensional autocorrelation function is often assumed to have the form of a Gaussian, i.e. it can be given by the following relation

$$G_x(\tau_x) = \sigma^2 \exp(-\tau_x^2/T^2)$$

where σ denotes the root mean square deviation of the heights and T denotes the autocorrelation length.

For the exponential autocorrelation function we have the following relation

$$G_x(\tau_x) = \sigma^2 \exp(-\tau_x/T)$$



Autocorrelation function obtained for simulated Gaussian randomly rough surface (i.e. with a Gaussian autocorrelation function) with $\sigma \approx 20\,\mathrm{nm}$ *and* $T \approx 300\,\mathrm{nm}$.

We can also introduce the radial ACF $G_r(\tau)$, i.e. angularly averaged two-dimensional ACF, which of course contains the same information as the one-dimensional ACF for isotropic surfaces:

$$G_r(\tau) = \int_0^{2\pi} W(\tau\cos\varphi, \tau\sin\varphi)\,\mathrm{d}\varphi$$

Note For optical measurements (e. g. spectroscopic reflectometry, ellipsometry) the Gaussian autocorrelation function is usually expected to be in good agreement with the surface properties. However, some articles related with surface growth and oxidation usually assume that the exponential form is closer to the reality.

Height-Height Correlation Function

The difference between the height-height correlation function and the autocorrelation function is very small. As with the autocorrelation function, we sum the multiplication of two different values. For the autocorrelation function, these values represented the different distances between points. For the height-height correlation function, we instead use the power of difference between the points.

For AFM measurements, we usually evaluate the one-dimensional height-height correlation function based only on profiles along the fast scanning axis. It can therefore be evaluated from the discrete AFM data values as

$$H_x(\tau_x) = \frac{1}{N(M-m)} \sum_{l=1}^{N} \sum_{n=1}^{M-m} (z_{n+m,l} - z_{n,l})^2$$

where $m = \tau_x/\Delta x$. The function thus can be evaluated in a discrete set of values of τ_x separated by the sampling interval Δx.

The one-dimensional height-height correlation function is often assumed to be Gaussian, i.e. given by the following relation

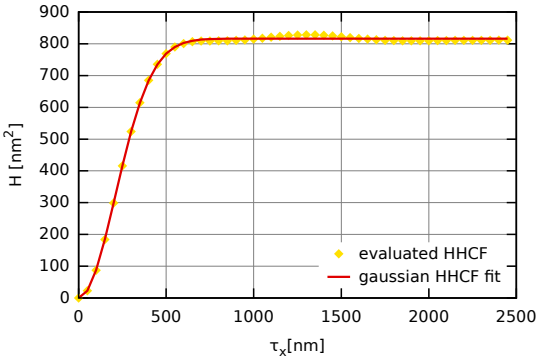
$$H_x(\tau_x) = 2\sigma^2 \left[1 - \exp\left(-\frac{\tau_x^2}{T^2}\right)\right]$$

where σ denotes the root mean square deviation of the heights and T denotes the autocorrelation length.

For the exponential height-height correlation function we have the following relation

$$H_x(\tau_x) = 2\sigma^2 \left[1 - \exp\left(-\frac{\tau_x}{T}\right)\right]$$

In the following figure the height-height correlation function obtained for a simulated Gaussian surface is plotted. It is fitted using the formula shown above. The resulting values of σ and T obtained by fitting the HHCF are practically the same as for the ACF.



Height-height correlation function obtained for simulated Gaussian randomly rough surface with $\sigma \approx 20\,\mathrm{nm}$ *and* $T \approx 300\,\mathrm{nm}$.

Power Spectral Density Function

The two-dimensional power spectral density function can be written in terms of the Fourier transform of the autocorrelation function

$$W(K_x, K_y) = \frac{1}{4\pi} \iint_{-\infty}^{\infty} G(\tau_x, \tau_y)\, \mathrm{e}^{-\mathrm{i}(K_x\tau_x + K_y\tau_y)}\, \mathrm{d}\tau_x\, \mathrm{d}\tau_y$$

Similarly to the autocorrelation function, we also usually evaluate the one-dimensional power spectral density function which is given by the equation

$$W_1(K_x) = \int_{-\infty}^{\infty} W(K_x, K_y)\, \mathrm{d}K_y$$

This function can be evaluated by means of the Fast Fourier Transform as follows:

$$W_1(K_x) = \frac{2\pi}{NMh} \sum_{j=0}^{N} |\hat{P}_j(K_x)|^2$$

where $P_j(K_x)$ is the Fourier coefficient of the j-th row, i.e.

$$\hat{P}_j(K_x) = \frac{h}{2\pi} \sum_{k=0}^{N} z_{kj} \exp(-\mathrm{i}K_x kh)$$

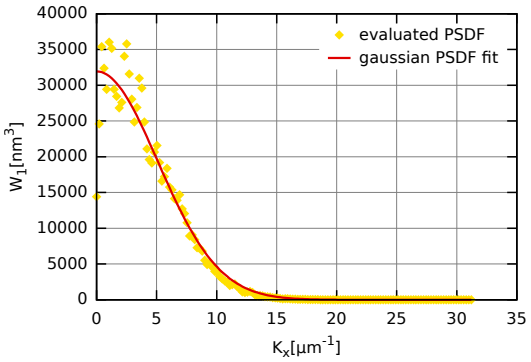
If we choose the Gaussian ACF, the corresponding Gaussian relation for the PSDF is

$$W_1(K_x) = \frac{\sigma^2 T}{2\sqrt{\pi}} \exp(-K_x^2 T^2/4)$$

For the surface with exponential ACF we have

$$W_1(K_x) = \frac{\sigma^2 T}{\pi} \frac{1}{1+K_x^2 T^2}$$

In the following figure the resulting PSDF and its fit for the same surface as used in the ACF and HHCF fitting are plotted. We can see that the function can be again fitted by Gaussian PSDF. The resulting values of σ and T were practically same as those from the HHCF and ACF fit.



PSDF obtained for data simulated with Gaussian autocorrelation function.

We can also introduce the radial PSDF $W_r(K)$, i.e. angularly integrated two-dimensional PSDF, which of course contains the same information as the one-dimensional PSDF for isotropic rough surfaces:

$$W_r(K) = \int_0^{2\pi} W(K\cos\varphi, K\sin\varphi)\, K\, \mathrm{d}\varphi$$

For a surface with Gaussian ACF this function is expressed as

$$W_r(K) = \frac{\sigma^2 T}{2} KT \exp(-K^2T^2/4)$$

while for exponential-ACF surface as

$$W_r(K) = \sigma^2 T \frac{KT}{(1+K^2T^2)^{3/2}}$$

Tip Within Gwyddion you can fit all statistical functions presented here by their Gaussian and exponential forms. To do this, fist click *Apply* within the Statistical Functions tool window. This will create a new graph window. With this new window selected, click on *Graph* → *Fit Graph*.

Minkowski Functionals

The Minkowski functionals are used to describe global geometric characteristics of structures. Two-dimensional discrete variants of volume V, surface S and connectivity (Euler-Poincaré Characteristic) χ are calculated according to the following formulas:

$$V = \frac{N_{\text{white}}}{N}, \quad S = \frac{N_{\text{bound}}}{N}, \quad \chi = \frac{C_{\text{white}} - C_{\text{black}}}{N}$$

Here N denotes the total number of pixels, N_{white} denotes the number of "white" pixels, that is pixels above the threshold. Pixels below the threshold are referred to as "black". Symbol N_{bound} denotes the number of white-black pixel boundaries. Finally, C_{white} and C_{black} denote the number of continuous sets of white and black pixels respectively.

For an image with continuous set of values the functionals are parametrized by the height threshold value ϑ that divides white pixels from black, that is they can be viewed as functions of this parameter. And these functions $V(\vartheta)$, $S(\vartheta)$ and $\chi(\vartheta)$ are plotted.

Row/Column Statistics Tool

This tool calculates numeric characteristics of each row or column and plots them as a function of its position. This makes it kind of complementary to Statistical Functions tool. Available quantities include:

- Mean value, minimum, maximum and median.
- RMS value of the height irregularities computed from data variance R_q.
- Skewness and kurtosis of the height distribution.
- Surface line length. It is estimated as the total length of the straight segments joining data values in the row (column).
- Overall slope, i.e. the tangent of the mean line fitted through the row (column).
- Tangent of β_0. This is a characteristics of the steepnes of local slopes, closely related to the behaviour of autocorrelation and height-height correlation functions at zero. For discrete values it is calculated as follows:

$$\tan^2 \beta_0 = \frac{1}{(N-1)h^2} \sum_{i=1}^{N-1} (z_i - z_{i-1})^2$$

- Standard roughness parameters Ra, Rz, Rt.

In addition to the graph displaying the values for individual rows/columns, the mean value and standard deviation of the selected quantity is calculated from the set of individual row/column values. It must be emphasised that the standard deviation of the selected quantity represents the dispersion of values for individual rows/columns and cannot be interpreted as an error of the analogous two-dimensional quantity.

Two-Dimensional Slope Statistics

Several functions in *Data Process* → *Statistics* operate on two-dimensional slope (derivative) statistics.

Slope Distribution calculates a plain two-dimensional distribution of derivatives, that is the horizontal and vertical coordinate on the resulting data field is the horizontal and vertical derivative, respectively. The slopes can be calculated as central derivatives (one-side on the borders of the image) or, if *Use local plane fitting* is enabled, by fitting a local plane through the neighbourhood of each point and using its gradient.

Slope Distribution can also plot summary graphs representing one-dimensional distributions of quantities related to the local slopes and facet inclination angles given by the following formula:

$$\mathbf{v} = \left(\frac{\mathrm{d}z}{\mathrm{d}x}, \frac{\mathrm{d}z}{\mathrm{d}y}\right), \quad v = |\mathbf{v}|, \quad \vartheta = \operatorname{atan} v, \quad \varphi = \operatorname{atan}_2(v_y, -v_x)$$

Three different plots are available:

- *Inclination (ϑ)*, the distribution of the inclination angle ϑ from the horizontal plane. Of course, representing the slope as an angle requires the value and the dimensions to be the same physical quantity.
- *Inclination (gradient)* is similar to the ϑ plot, except the distribution of the derivative v is plotted instead of the angle.
- *Inclination (φ)* visualises the integral of v^2 for each direction φ in the horizontal plane. This means it is not a plain distribution of φ because areas with larger slopes contribute more significantly than flat areas.

Angle Distribution function is a visualization tool that does not calculate a distribution in the strict sense. For each derivative $\mathbf{v}$ the circle of points satisfying

$$2\mathbf{r} \cdot \mathbf{v} = r^2$$

is drawn. The number of points on the circle is given by *Number of steps*.

Facet Analysis

Data Process → Statistics → Facet Analysis

Facet analysis enables to interactively study orientations of facets occuring in the data and mark facets of specific orientations on the image. The left view displays data with preview of marked facets. The right smaller view, called facet view below, displays two-dimensional slope distribution.

The centre of facet view always correspond to zero inclination (horizontal facets), slope in x-direction increases towards left and right border and slope in y-direction increases towards top and bottom borders. The exact coordinate system is a bit complex and it adapts to the range of slopes in the particular data displayed.

Facet plane size controls the size (radius) of plane locally fitted in each point to determine the local inclination. The special value 0 stands for no plane fitting, the local inclination is determined from symmetric x and y derivatives in each point. The choice of neighbourhood size is crucial for meaningful results: it must be smaller than the features one is interested in to avoid their smoothing, on the other hand it has to be large enough to suppress noise present in the image.

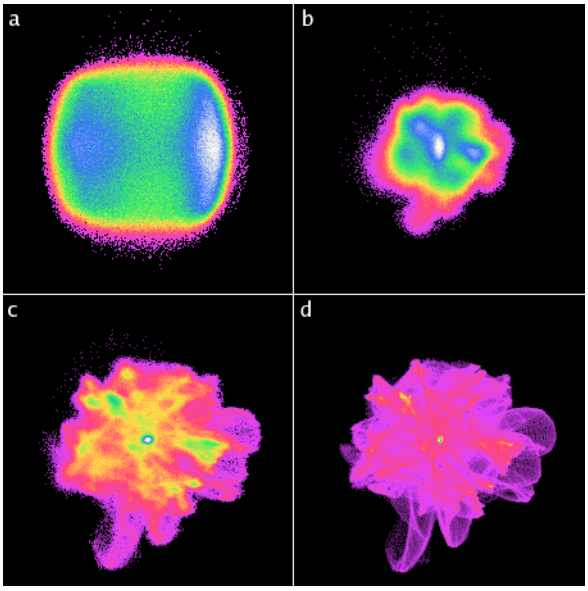


Illustration of the influence of fitted plane size on the distribution of a scan of a delaminated DLC surface with considerable fine noise. One can see the distribution is completely obscured by the noise at small plane sizes. The neighbourhood sizes are: (a) 0, (b) 2, (c) 4, (d) 7. The angle and false color mappings are full-scale for each particular image, i.e. they vary among them.

Both facet view and data view allow to select a point with mouse and read corresponding facet normal inclination value ϑ and direction φ under *Normal*. When you select a point on data view, the facet view selection is updated to show inclination in this point.

Button *Find Maximum* sets facet view selection to slope distribution maximum (the initial selection position).

Button *Mark* updates the mask of areas with slope similar to the selected slope. More precisely, of areas with slope within *Tolerance* from the selected slope. The facet view then displays the set of slopes corresponding to marked points (note the set of selected slopes may not look circular on facet view, but this is only due to selected projection). Average inclination of all points in selected range of slopes is displayed under *Mean Normal*.

4.9 One-Dimensional Roughness Parameters

Standardized one-dimensional roughness parameters can be evaluated with the roughness tool.

The one-dimensional texture is split into waviness (the low-frequency components defining the overall shape) and roughness (the high-frequency components) at the cut-off frequency. This frequency is specified in the units of the Nyquist frequency, that is value 1.0 corresponds to the Nyquist frequency.

In the following formulas we assume the mean value of r_j is zero, i.e. it holds

$$r_j = z_j - \bar{z}$$

Roughness Amplitude Parameters

Roughness Average R_a Standards: ASME B46.1-1995, ASME B46.1-1985, ISO 4287-1997, ISO 4287/1-1997.

Arithmetical mean deviation. The average deviation of all points roughness profile from a mean line over the evaluation length

$$R_\mathrm{a} = \frac{1}{N}\sum_{j=1}^{N} |r_j|$$

An older means of specifying a range for R_a is RHR. This is a symbol on a drawing specifying a minimum and maximum value for R_a.

Root Mean Square Roughness R_q Standards: ASME B46.1-1995, ISO 4287-1997, ISO 4287/1-1997.

The average of the measured height deviations taken within the evaluation length and measured from the mean line

$$R_\mathrm{q} = \sqrt{\frac{1}{N}\sum_{j=1}^{N} r_j^2}$$

Maximum Height of the Profile R_t Standards: ASME B46.1-1995, ISO 4287-1997.

Maximum peak-to-peak-valley height. The absolute value between the highest and lowest peaks

$$R_\mathrm{t} = \left|\min_{1\le j\le N} r_j\right| + \left|\max_{1\le j\le N} r_j\right|$$

Maximum Profile Valley Depth R_v, R_m Standards: ASME B46.1-1995, ASME B46.1-1985, ISO 4287-1997, ISO 4287/1-1997.

Lowest valley. There is the depth of the deepest valley in the roughness profile over the evaluation length

$$R_\mathrm{v} = \left|\min_{1\le j\le N} r_j\right|$$

Maximum Profile Peak Height R_p Standards: ASME B46.1-1995, ASME B46.1-1985, ISO 4287-1997, ISO 4287/1-1997.

Highest peak. There is the height of the highest peak in the roughness profile over the evaluation length

$$R_\mathrm{p} = \left|\max_{1\le j\le N} r_j\right|$$

Average Maximum Height of the Profile $R_{\mathrm{t}m}$ Standards: ASME B46.1-1995, ISO 4287-1997.

Mean peak-to-valley roughness. It is determined by the difference between the highest peak ant the lowest valley within multiple samples in the evaluation length

$$R_{\mathrm{t}m} = R_{\mathrm{v}m} + R_{\mathrm{p}m}$$

where $R_{\mathrm{v}m}$ and $R_{\mathrm{p}m}$ are defined below.

For profile data it is based on five sample lengths ($m = 5$). The number of samples corresponds with the ISO standard.

Average Maximum Profile Valley Depth $R_{\mathrm{v}m}$ Standards: ISO 4287-1997.

The mean valley depth based on one peak per sampling length. The single deepest valley is found in five sampling lengths ($m = 5$) and then averaged

$$R_{\mathrm{v}m} = \frac{1}{m}\sum_{i=1}^{m} R_{\mathrm{v}i}$$

where

$$R_{\mathrm{v}i} = \left|\min r_j\right| \quad \text{for} \quad (i-1)\frac{N}{m} < j < i\frac{N}{m}$$

Average Maximum Profile Peak Height R_{pm} Standards: ISO 4287-1997.

The mean peak height based on one peak per sampling length. The single highest peak is found in five sampling lengths ($m = 5$) and then averaged

$$R_{pm} = \frac{1}{m}\sum_{i=1}^{m} R_{pi}$$

where

$$R_{pi} = \left|\max r_j\right| \quad \text{for} \quad (i-1)\frac{N}{m} < j < i\frac{N}{m}$$

Base roughness depth R_{3z} Standards: ISO 4287-1997.

The distance between the third highest peak and the third lowest valley. A peak is a portion of the surface above the mean line crossings.

Base roughness profile depth R_{3zISO} Standards: ISO 4287-1997.

The height of the third highest peak from the third lowest valley per sampling length. The base roughness depth is found in five sampling lengths and then averaged.

Ten-point height R_z Standards: ISO 4287-1997

The average absolute value of the five highest peaks and the five lowest valleys over the evaluation length.

Average peak-to-valley profile roughness R_{zISO} Standards: ISO 4287-1997.

The average peak-to-valley roughness based on one peak and one valley per sampling length. The single largest deviation is found in five sampling lengths and then averaged. It is identical to R_{tm}.

The Amplitude Distribution Function ADF Standards: ISO 4287-1997.

The amplitude distribution function is a probability function that gives the probability that a profile of the surface has a certain height z at any position x.

The Bearing Ratio Curve BRC Standards: ISO 4287-1997.

The Bearing Ratio Curve is related to the ADF, it is the corresponding cumulative probability distribution and sees much greater use in surface finish. The bearing ratio curve is the integral (from the top down) of the ADF.

Skewness R_{sk} Standards: ISO 4287-1997.

Skewness is a parameter that describes the shape of the ADF. Skewness is a simple measure of the asymmetry of the ADF, or, equivalently, it measures the symmetry of the variation of a profile about its mean line

$$R_{sk} = \frac{1}{NR_q^3}\sum_{j=1}^{N} r_j^3$$

Kurtosis R_{ku} Standards: ISO 4287-1997.

Kurtosis is the ADF shape parameter considered. Kurtosis relates to the uniformity of the ADF or, equivalently, to the spikiness of the profile.

$$R_{ku} = \frac{1}{NR_q^4}\sum_{j=1}^{N} r_j^4$$

4.10 Grain Analysis

There are several grain-related algorithms implemented in Gwyddion. First of all, simple thresholding algorithms can be used (height, slope or curvature thresholding). These procedures can be very efficient namely within particle analysis (to mark particles located on flat surface).

Thresholding methods can be accessed within Gwyddion as *Data Process* → *Grains* → *Mark by Threshold*. Height, slope and curvature thresholding is implemented within this module. The results of each individual thresholding methods can be merged together using several operators.

The automated Otsu's thresholding method is available as *Data Process* → *Grains* → *Mark by Otsu's*. This method classifies the data values into two classes, minimising the intra-class variances within both. It is most suitable for images that contain two relatively well defined value levels.

Another grain marking function, *Data Process* → *Grains* → *Mark by Edge Detection*, is based on edge detection (local curvature). The image is processed with a difference-of-Gaussians filter of a given size and thresholding is then performed on this filtered image instead of the original.

Grains that touch image eges can be removed using *Data Process* → *Grains* → *Remove Edge-Touching* menu choice. This is useful if such grains are considered incomplete and must be excluded from analysis. Several other other functions that may be useful for modification of grain shapes after marking are performed by the Mask Editor tool.

Watershed 

Data Process → *Grains* → *Mark by Watershed*

For more complicated data structures the effectiveness of thresholding algorithms can be very poor. For these data a *watershed algorithm* can be used more effectively for grain or particle marking.

The watershed algorithm is usually employed for local minima determination and image segmentation in image processing. As the problem of determining the grain positions can be understood as the problem of finding local extremes on the surface this algorithm can be used also for purposes of grain segmentation or marking. For convenience in the following we will treat the data inverted in the z direction while describing the algorithm (i.e. the grain tops are forming local minima in the following text). We applied two stages of the grain analysis (see [1]):

1. Grain location phase: At each point of the inverted surface the virtual water drop was placed (amount of water is controlled by parameter *Drop size*). In the case that the drop was not already in a local minimum it followed the steepest descent path to minimize its potential energy. As soon as the drop reached any local minimum it stopped here and rested on the surface. In this way it filled the local minimum partially by its volume (see figure below and its caption). This process was repeated several times (parameter *Number of steps*). As the result a system of lakes of different sizes filling the inverted surface depressions was obtained. Then the area of each of the lakes was evaluated and the smallest lakes are removed under assumption that they were formed in the local minima originated by noise (all lakes smaller than parameter *Threshold* are removed). The larger lakes were used to identify the positions of the grains for segmentation in the next step. In this way the noise in the AFM data was eliminated. As a result
2. Segmentation phase: The grains found in the step 1 were marked (each one by a different number). The water drops continued in falling to the surface and filling the local minima (amount of water is controlled by parameter *Drop size*). Total number of steps of sphlashing a drop at every surface position is controlled by parameter *Number of steps*. As the grains were already identified and marked after the first step, the next five situations could happen as soon as the drop reached a local minimum.
   (a) The drop reached the place previously marked as a concrete grain. In this case the drop was merged with the grain, i. e. it was marked as a part of the same grain.
   (b) The drop reached the place where no grain was found but a concrete grain was found in the closest neighbourhood of the drop. In this case the drop was merged with the grain again.
   (c) The drop reached the place where no grain was found and no grain was found even in the closest neighbourhood of the drop. In that case the drop was not marked at all.
   (d) The drop reached the place where no grain was found but more than one concrete grain was found in the closest neighbourhood (e. g. two different grains were found in the neighbourhood). In this case the drop was marked as the grain boundary.
   (e) The drop reached the place marked as grain boundary. In this case the drop was marked as the grain boundary too.

In this way we can identify the grain positions and then determine the volume occupied by each grain separately. If features of interest are valleys rather than grains (hills), parameter *Invert height* can be used.

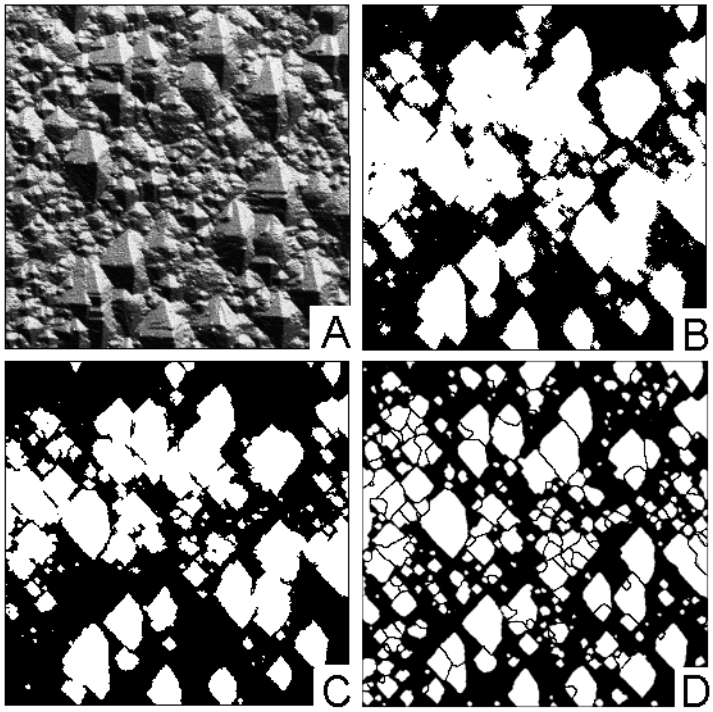


Image of grain-like surface structure (a) and corresponding results of height thresholding (b), curvature thresholding (c), and watershed (d) algorithm. Within watershed algorithm it is possible to segment image even further.

Segmentation

Data Process → Grains → Mark by Segmentation

This function a different approach based on a *watershed algorithm*, in this case the classical Vincent algorithm for watershed in digital spaces [2], which is applied to a preprocessed image. Generally, the result is an image fully segmented to motifs, each pixel belonging to one or separating two of them. By default, the algorithm marks *valleys*. To mark upward grains, which is more common in AFM, use the opton *Invert height*.

The preprocessing has the following parameters:

Gaussian smoothing Dispersion of Gaussian smoothing filter applied to the data. A zero value means no smoothing.

Add gradient Relative weight of local gradient added to the data. Large values mean areas with large local slope tend to become grain boundaries.

Add curvature Relative weight of local gradient added to the data. Large values mean locally concave areas with tend to become grain boundaries.

Barrier level Relative height level above which pixels are never assigned to any grain. If not 100%, this creates an exception to the full-segmentation property.

Prefill level Relative height level up to which the surface is prefilled, obliterating any details at the bottoms of deep valleys.

Prefill from minima Relative height level up to which the surface is prefilled from each local minimum, obliterating any details at the bottoms of valleys.

Statistics

Grain properties can be studied using several functions. The simplest of them is Grain Statistics

Grain Statistics

Data Process → *Grains* → *Statistics*

This function calculates the total number of marked grains, their total projected area, both as an absolute value and as a fraction of total data field area, total grain volumes, total length of grain boundaries and the mean area and equivalent square size of one grain. The mean size is calculated by averaging the equivalent square sides so its square is not, in general, equal to the mean area.

Overall characteristics of the marked area can be also obtained with Statistical Quantities tool when its *Use mask* option is switched on. By inverting the mask the same information can be obtained also for the non-grain area.

Grain Distributions 

Data Process → *Grains* → *Distributions*

Grain Distributions is the most powerful and complex tool. It has two basic modes of operation: graph plotting and raw data export. In graph plotting mode selected characteristics of individual grains are calculated, gathered and summary graphs showing their distributions are plotted.

Raw data export is useful for experts who need for example to correlate properties of individual grains. In this mode selected grain characteristics are calculated and dumped to a text file table where each row corresponds to one grain and columns correspond to requested quantities. The order of the colums is the same as the relative order of the quantities in the dialog; all values are written in base SI units, as is usual in Gwyddion.

Grain Property Correlation

Data Process → *Grains* → *Correlate*

Grain correlation plots a graph of one selected graph quantity as the function of another grain quantity, visualizing correlations between them.

Grain Measurement Tool 

The grain measurement tool is the interactive method to obtain the same information about individual grains as Grain Distributions in raw mode. After selecting a grain on the data window with mouse, all the available quantities are displayed in the tool window.

Beside physical characteristics this tool also displays the grain number. Grain numbers corresponds to row numbers (counting from 1) in files exported by Grain Distributions.

Grain Properties

Grain Distributions and Grain measurement tool can calculate the following grain properties:

Value-related properties

- *Minimum*, the minimum value (height) occuring inside the grain.
- *Maximum*, the maximum value (height) occuring inside the grain.
- *Mean*, the mean of all values occuring inside the grain, that is the mean grain height.
- *Median* the median of all values occuring inside the grain, that is the median grain height.
- *Minimum on boundary*, the minimum value (height) occuring on the inner grain boundary. This means within the set of pixels that lie inside the grain but at least one of their neighbours lies outside.
- *Maximum on boundary*, the maximum value (height) occuring on the inner grain boundary, defined similarly to the minimum.

Area-related properties

- *Projected area*, the projected (flat) area of the grain.
- *Equivalent square side*, the side of the square with the same projected area as the grain.
- *Equivalent disc radius*, the radius of the disc with the same projected area as the grain.

- *Surface area*, the surface area of the grain, see statistical quantities section for description of the surface area estimation method.
- *Area of convex hull*, the projected area of grain convex hull. The convex hull area is slightly larger than the grain area even for grains that appear to be fairly convex due to pixelization of the mask. The only grains with exactly the same area as their convex hulls are perfectly rectangular grains.

Boundary-related properties

- *Projected boundary length*, the length of the grain boundary projected to the horizontal plane (that is not taken on the real three-dimensional surface). The method of boundary length estimation is described below.
- *Minimum bounding size*, the minimum dimension of the grain in the horizontal plane. It can be visualized as the minimum width of a gap in the horizontal plane the grain could pass through.
- *Minimum bounding direction*, the direction of the gap from the previous item. If the grain exhibits a symmetry that makes several directions to qualify, an arbitrary direction is chosen.
- *Maximum bounding size*, the maximum dimension of the grain in the horizontal plane. It can be visualized as the maximum width of a gap in the horizontal plane the grain could fill up.
- *Maximum bounding direction*, the direction of the gap from the previous item. If the grain exhibits a symmetry that makes several directions to qualify, an arbitrary direction is chosen.
- *Maximum inscribed disc radius*, the radius of maximum disc that fits inside the grain. The entire full disc must fit, not just its boundary circle, which matters for grains with voids within. You can use Mask Editor tool to fill voids in grains to get rid of voids.
- *Maximum inscribed disc center x position*, the horizontal coordinate of center of the maximum inscribed disc. More precisely, of one such disc if it is not unique.
- *Maximum inscribed disc center y position*, the vertical coordinate of center of the maximum inscribed disc. More precisely, of one such disc if it is not unique but of the same as in the previous item.
- *Minimum circumcircle radius*, the radius of minimum circle that contains the grain.
- *Minimum circumcircle center x position*, the horizontal coordinate of center of the minimum circumcircle.
- *Minimum circumcircle center y position*, the vertical coordinate of center of the minimum circumcircle.
- *Mean radius*, the mean distance from grain center of mass to its boundary. This quantity is mostly meaningful only for convex or nearly-convex grains.

Volume-related properties

- *Zero basis*, the volume between grain surface and the plane $z = 0$. Values below zero form negative volumes. The zero level must be set to a reasonable value (often Fix Zero is sufficient) for the results to make sense, which is also the advantage of this method: one can use basis plane of his choice.
- *Grain minimum basis*, the volume between grain surface and the plane $z = z_{\min}$, where $z_{\min}$ is the minimum value (height) occuring in the grain. This method accounts for grain surrounding but it typically underestimates the volume, especially for small grains.
- *Laplacian backround basis*, the volume between grain surface and the basis surface formed by laplacian interpolation of surrounding values. In other words, this is the volume that would disappear after using Remove Data Under Mask or Grain Remover tool with Laplacian interpolation on the grain. This is the most sophisticated method, on the other hand it is the hardest to develop intuition for.

Position-related properties

- *Center x position*, the horizontal coordinate of the grain center. Since the grain area is defined as the area covered by the corresponding mask pixels, the center of a single-pixel grain has half-integer coordinates, not integer ones. Data field origin offset (if any) is taken into account.
- *Center y position*, the vertical coordinate of the grain center. See above for the interpretation.

Slope-related properties

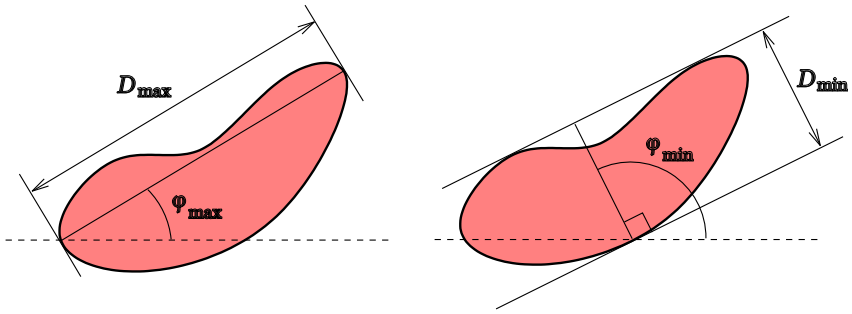
- *Inclination* ϑ, the deviation of the normal to the mean plane from the z-axis, see inclinations for details.
- *Inclination* φ, the azimuth of the slope, as defined in inclinations.

Curvature-related properties

- *Curvature center x*, the horizontal position of the center of the quadratic surface fitted to the grain surface.
- *Curvature center y*, the vertical position of the center of the quadratic surface fitted to the grain surface.
- *Curvature center z*, the value at the center of the quadratic surface fitted to the grain surface. Note this is the value at the fitted surface, not at the real grain surface.
- *Curvature 1*, the smaller curvature (i.e. the inverse of the curvature radius) at the center.
- *Curvature 2*, the larger curvature (i.e. the inverse of the curvature radius) at the center.
- *Curvature angle 1*, the direction corresponding to *Curvature 1*.
- *Curvature angle 2*, the direction corresponding to *Curvature 2*.

Moment-related properties

- *Major semiaxis of equivalent ellipse*, the length of the major semiaxis of an ellipse which has the same second order moments in the horizontal plane.
- *Minor semiaxis of equivalent ellipse*, the length of the minor semiaxis of an ellipse which has the same second order moments in the horizontal plane.
- *Orientation of equivalent ellipse*, the direction of the major semiaxis of an ellipse which has the same second order moments in the horizontal plane. For a circular grain, the angle is set to zero by definition.

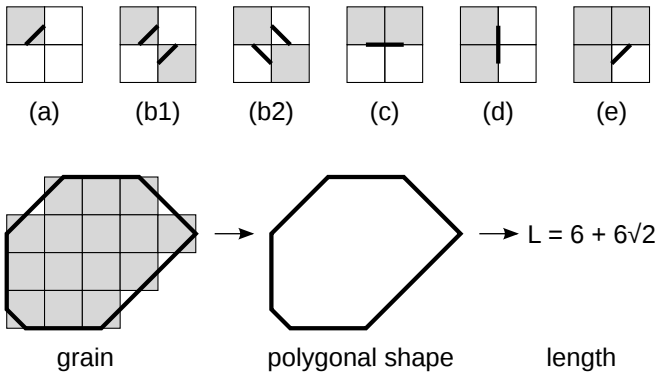


Maximum and minimum bounding dimensions and angles of a grain.

The grain boundary length is estimated by summing estimated contributions of each four-pixel configuration on the boundary. The contributions are displayed on the following figure for each type of configuration, where h_x and h_y are pixel dimension along corresponding axes and h is the length of the pixel diagonal:

$$h = \sqrt{h_x^2 + h_y^2}$$

The contributions correspond one-to-one to lengths of segments of the boundary of a polygon approximating the grain shape. The construction of the equivalent polygonal shape can also be seen in the figure.



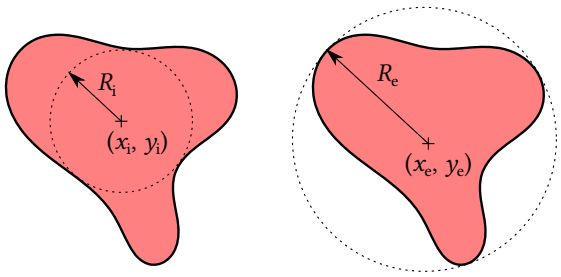
Contributions of pixel configurations to the estimated boundary length (top). Grey squares represent pixels inside the grain, white squares represent outside pixels. The estimated contribution of each configuration is: (a) $h/2$, (b1), (b2) h, (c) h_y, (d) h_x, (e) $h/2$. Cases (b1) and (b2) differ only in the visualization of the polygonal shape segments, the estimated boundary lengths are identical. The bottom part of the figure illustrates how the segments join to form the polygon.

The grain volume is, after subtracting the basis, estimated as the volume of exactly the same body whose upper surface is used for surface area calculation. Note for the volume between vertices this is equivalent to the classic two-dimensional trapezoid integration method. However, we calculate the volume under a mask centered on vertices, therefore their contribution to the integral is distributed differently as shown in the following figure.

$\frac{1}{96}$	$\frac{5}{48}$	$\frac{1}{96}$
$\frac{5}{48}$	$\frac{13}{24}$	$\frac{5}{48}$
$\frac{1}{96}$	$\frac{5}{48}$	$\frac{1}{96}$

Contributions of individual pixels to the volume of single pixel (grey).

Curvature-related properties of individual grains are calculated identically to the global curvature calculated by Curvature. See its description for some discussion.



Maximum inscribed disc and minimum circumcircle of a grain.

Inscribed discs and circumcircles of grains can be visualized using *Data Process* → *Grains* → *Select Inscribed Discs* and *Data Process* → *Grains* → *Select Exscribed Circles*. These functions create circular selections representing the corresponding disc or circle for each grain that can be subsequently displayed using Selection Manager tool.

Grain Filtering

Marked grains can be filtered by thresholding by any of the available grain quantities using *Data Process* → *Grains* → *Filter* menu choice. The module can be used for basic operations, such as removal of tiny grains using a pixel area threshold, as well as complex filtering using logical expressions involving several grain quantitities.

The filter retains grains that satisfy the condition specified as *Keep grains satisfying* and removes all other grains. The condition is expressed as a logical expression of one to three individual thresholding conditions, denoted `A`, `B` and `C`. The simplest expression is just `A`, stating that quantity `A` must lie within given thresholds.

Each condition consits of lower and upper thresholds for one grain quantity, for instance pixel area or minimum value. The values must lie within the interval [lower,upper] to satisfy the condition and thus retain the grains. Note it is possible to choose the lower threshold larger than the upper threshold. In this case the condition is inverted, i.e. the grain is retained if the value lies outside [upper,lower].

Individual grain quantities are assigned to `A`, `B` and `C` by selecting the quantity in the list and clicking on the corresponding button in *Set selected as*. The currently selected set of quantities is displayed in the *Condition A*, *Condition B* and *Condition C* headers.

Grain Leveling

Grains can be aligned vertically using *Data Process* → *Grains* → *Level Grains*. This function vertically shifts each grain to make a certain height-related quantity of all grains equal. Typically, the grain minimum values are aligned but other choices are possible.

Data between grains are also vertically shifted. The shifts are interpolated from the grain shifts using the Laplace equation, leading to a smooth transition of the shifts between the grains (though with no regard to other possible surface features).

Distance Transform

The distance transform assigns to each pixel its distance to the grain (mask) boundary. It is, in a certain sense, complementary to the watershed. The distance transform can perform the transform using true Euclidean distance

$$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

as well as classical simple distances such as the city-block (4-neigbourhood)

$$|x_1-x_2|+|y_1-y_2|$$

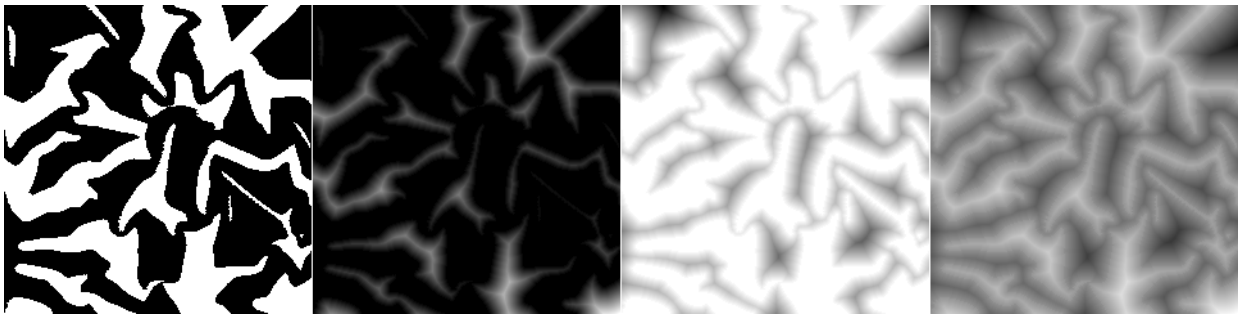
and chessboard (8-neigbourhood)

$$\max(|x_1-x_2|,|y_1-y_2|)$$

distances that are useful mainly for didactic purposes. In addition, octagonal distances are available that are calculated by taking the cityblock and chessboard distance steps alternatively, obtaining '48' and '84' variants depending on which step is taken first. Finally, an octagonal distance without any specifier is available that is the average of the two variants.

If the transform is applied to grain interiors the distance is zero outside grains and increases towards the grain 'centres'. Conversely, it can be applied to the exteriors and it is then highest for pixels farthest from any grain. It is also possible to calculate signed two-side transform which is the difference of the two transforms, i.e. it is positive inside grains and negative outside.

Option *Shrink from border* controls the handling of borders. When it is enabled, image boundaries are considerd to be also grain (or non-grain) boundaries. Pixels on the image edge thus cannot receive large values. When it is disabled, the grains (or non-grains) are effectively infinitely large outside the image. So pixels close to the boundary can receive large distance values.



Euclidean distance transform example: the soruce mask shown as white on black, interior distance transform, exterior transform (with inverted greyscale for comparison) and the signed transform.

References

[1] Petr Klapetek, Ivan Ohlídal, Daniel Franta, Alberto Montaigne-Ramil, Alberta Bonanni, David Stifter and Helmut Sitter: Acta Physica Slovaca, 3 (223-230) 2003

[2] Luc Vincent and Pierre Soille: IEEE Transactions on Pattern Analysis and Machine Intelligence, 13 (583–598) 1991

4.11 Fourier Transform

Two-dimensional Fourier transform can be accessed using *Data Process* → *Integral Transforms* → *2D FFT* which implements the Fast Fourier Transform (FFT). Fourier transform decomposes signal into its harmonic components, it is therefore useful while studying spectral frequencies present in the SPM data.

The *2D FFT* module provides several types of output:

- Modulus – absolute value of the complex Fourier coefficient, proportional to the square root of the power spectrum density function (PSDF).
- Phase – phase of the complex coefficient (rarely used).
- Real – real part of the complex coefficient.
- Imaginary – imaginary part of the complex coefficient.

and some of their combinations for convenience.

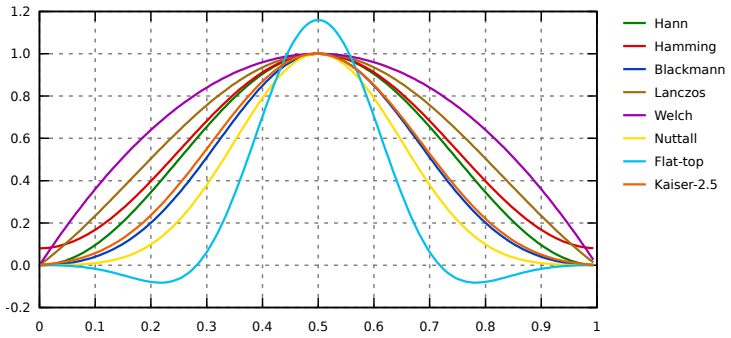
Radial sections of the two-dimensional PSDF can be conveniently obtained with *Data Process* → *Statistics* → *PSDF Section*. Several other functions producing spectral densities are described in section Statistical Analysis. It is also possible to filter images in the frequency domain using one-dimensional or two-dimensional FFT filters.

For scale and rotation-independent texture comparison it is useful to transform the PSDF from Cartesian frequency coordinates to coordinates consisting of logarithm of the spatial frequency and its direction. Scaling and rotation then become mere shifts in the new coordinates. The function *Data Process* → *Statistics* → *Log-Phi PSDF* calculates directly the transformed PSDF. The dimensionless horizontal coordinate is the angle (from 0 to 2π) the vertical is the frequency logarithm. It is possible to smooth the PSDF using a Gaussian filter of given width before the transformation.

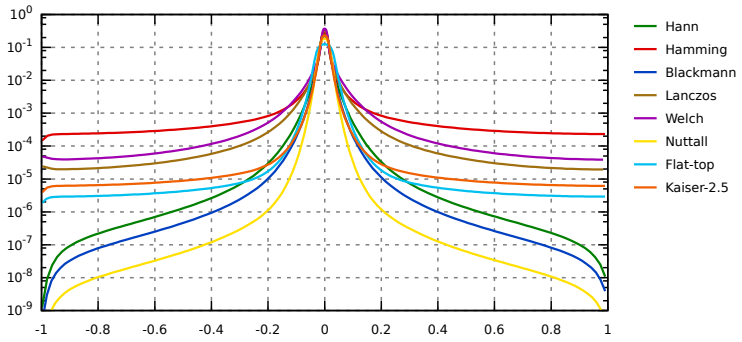
Note that the Fourier transform treats data as being infinite, thus implying some cyclic boundary conditions. As the real data do not have these properties, it is necessary to use some windowing function to suppress the data at the edgest of the image. If you do not do this, FFT treats data as being windowed by rectangular windowing function which has really bad Fourier image thus leading to corruption of the Fourier spectrum.

Gwyddion offers several windowing functions. Most of them are formed by some sine and cosine functions that damp data correctly at the edges. In the following windowing formula table the independent variable x is from interval $[0,1]$ which corresponds to the normalized abscissa; for simplicity variable $\xi = 2\pi x$ is used in some formulas. The available windowing types include:

Name	**Formula**
None	1
Rect	0.5 at edge points, 1 everywhere else
Hann	$w_{\text{Hann}}(x) = 0.5 - 0.5\cos\xi$
Hamming	$w_{\text{Hamming}}(x) = 0.54 - 0.46\cos\xi$
Blackmann	$w_{\text{Blackmann}}(x) = 0.42 - 0.5\cos\xi + 0.08\cos 2\xi$
Lanczos	$w_{\text{Lanczos}}(x) = \operatorname{sinc}\pi(2x-1)$
Welch	$w_{\text{Welch}}(x) = 4x(1-x)$
Nutall	$w_{\text{Nutall}}(x) = 0.355768 - 0.487396\cos\xi + 0.144232\cos 2\xi - 0.012604\cos 3\xi$
Flat-top	$w_{\text{flattop}}(x) = 0.25 - 0.4825\cos\xi + 0.3225\cos 2\xi - 0.097\cos 3\xi + 0.008\cos 4\xi$
Kaiser α	$w_{\text{Kaiser},\alpha}(x) = \dfrac{\mathrm{I}_0\left(\pi\alpha\sqrt{4x(1-x)}\right)}{\mathrm{I}_0(\pi\alpha)}$, where I_0 is the modified Bessel function of zeroth order and α is a parameter



Windowing functions: Hann, Hamming, Blackmann, Lanczos, Welch, Nutall, Flat-top, Kaiser 2.5.



Envelopes of windowing functions frequency responses: Hann, Hamming, Blackmann, Lanczos, Welch, Nutall, Flat-top, Kaiser 2.5.

Fourier transforms of data with sizes that are not factorable into small prime factors can be very slow – and many programs only implement FFT of arrays with dimensions that are powers of two.

In Gwyddion, however, the Fourier transform can be applied to data fields and lines of arbitrary dimensions, with no data resampling involved (at least since version 2.8). Fourier transforms are calculated either using the famous FFTW library or, if it is not available, using Gwyddion built-in routines that are slower but can also handle transforms of arbitrary size.

Nevertheless, if the data size is not factorable into small prime factors the transform is still considerably slower. Hence it is preferable to transform data fields of "nice" sizes.

4.12 Wavelet Transform

The wavelet transform is similar to the Fourier transform (or much more to the windowed Fourier transform) with a completely different merit function. The main difference is this: Fourier transform decomposes the signal into sines and cosines, i.e. the functions localized in Fourier space; in contrary the wavelet transform uses functions that are localized in both the real and Fourier space. Generally, the wavelet transform can be expressed by the following equation:

$$F(a,b) = \int_{-\infty}^{\infty} f(x)\, \psi^*_{(a,b)}(x)\, \mathrm{d}x$$

where the * is the complex conjugate symbol and function ψ is some function. This function can be chosen arbitrarily provided that obeys certain rules.

As it is seen, the Wavelet transform is in fact an infinite set of various transforms, depending on the merit function used for its computation. This is the main reason, why we can hear the term "wavelet transform" in very different situations and applications. There are also many ways how to sort the types of the wavelet transforms. Here we show only the division based on the wavelet orthogonality. We can use *orthogonal wavelets* for discrete wavelet transform development and *non-orthogonal wavelets* for continuous wavelet transform development. These two transforms have the following properties:

1. The discrete wavelet transform returns a data vector of the same length as the input is. Usually, even in this vector many data are almost zero. This corresponds to the fact that it decomposes into a set of wavelets (functions) that are orthogonal to its translations and scaling. Therefore we decompose such a signal to a same or lower number of the wavelet coefficient spectrum as is the number of signal data points. Such a wavelet spectrum is very good for signal processing and compression, for example, as we get no redundant information here.

2. The continuous wavelet transform in contrary returns an array one dimension larger thatn the input data. For a 1D data we obtain an image of the time-frequency plane. We can easily see the signal frequencies evolution during the duration of the signal and compare the spectrum with other signals spectra. As here is used the non-orthogonal set of wavelets, data are correlated highly, so big redundancy is seen here. This helps to see the results in a more humane form.

For more details on wavelet transform see any of the thousands of wavelet resources on the Web, or for example [1].

Within Gwyddion data processing library, both these transforms are implemented and the modules using wavelet transforms can be accessed within *Data Process* → *Integral Transforms* menu.

Discrete Wavelet Transform

The discrete wavelet transform (DWT) is an implementation of the wavelet transform using a discrete set of the wavelet scales and translations obeying some defined rules. In other words, this transform decomposes the signal into mutually orthogonal set of wavelets, which is the main difference from the continuous wavelet transform (CWT), or its implementation for the discrete time series sometimes called discrete-time continuous wavelet transform (DT-CWT).

The wavelet can be constructed from a scaling function which describes its scaling properties. The restriction that the scaling functions must be orthogonal to its discrete translations implies some mathematical conditions on them which are mentioned everywhere, e.g. the dilation equation

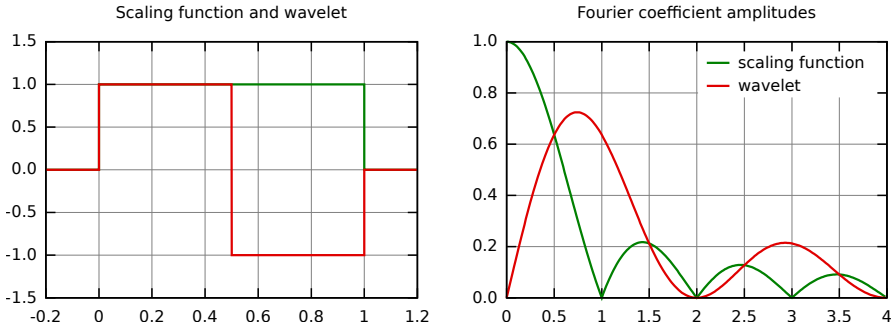
$$\phi(x) = \sum_{k=-\infty}^{\infty} a_k \phi(Sx - k)$$

where S is a scaling factor (usually chosen as 2). Moreover, the area between the function must be normalized and scaling function must be ortogonal to its integer translations, i.e.

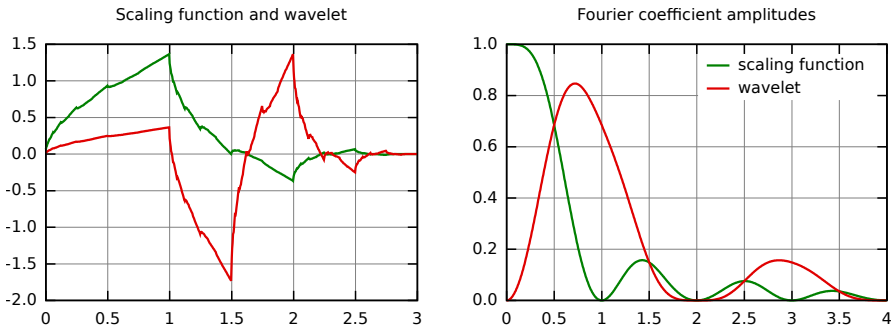
$$\int_{-\infty}^{\infty} \phi(x)\,\phi(x+l)\,\mathrm{d}x = \delta_{0,l}$$

After introducing some more conditions (as the restrictions above does not produce unique solution) we can obtain results of all these equations, i.e. the finite set of coefficients a_k that define the scaling function and also the wavelet. The wavelet is obtained from the scaling function as N where N is an even integer. The set of wavelets then forms an orthonormal basis which we use to decompose the signal. Note that usually only few of the coefficients a_k are nonzero, which simplifies the calculations.

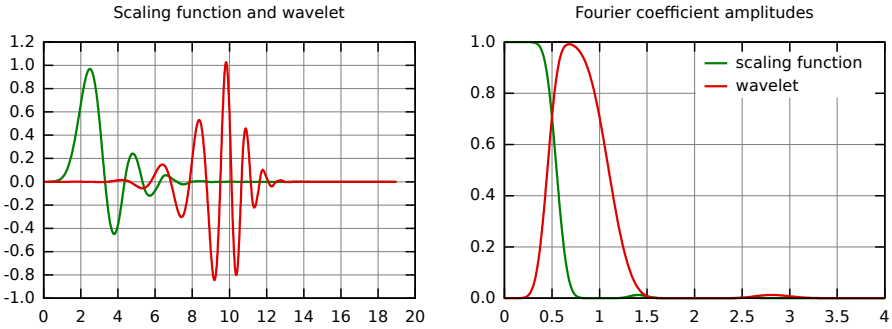
In the following figure, some wavelet scaling functions and wavelets are plotted. The most known family of orthonormal wavelets is the family of Daubechies. Her wavelets are usually denominated by the number of nonzero coefficients a_k, so we usually talk about Daubechies 4, Daubechies 6, etc. wavelets. Roughly said, with the increasing number of wavelet coeficients the functions become smoother. See the comparison of wavelets Daubechies 4 and 20 below. Another mentioned wavelet is the simplest one, the Haar wavelet, which uses a box function as the scaling function.



Haar scaling function and wavelet (left) and their frequency content (right).



Daubechies 4 scaling function and wavelet (left) and their frequency content (right).



Daubechies 20 scaling function and wavelet (left) and their frequency content (right).

There are several types of implementation of the DWT algorithm. The oldest and most known one is the Mallat (pyramidal) algorithm. In this algorithm two filters – smoothing and non-smoothing one – are constructed from the wavelet coefficients and those filters are recurrently used to obtain data for all the scales. If the total number of data $D = 2^N$ is used and the signal length is L, first $D/2$ data at scale $L/2^{N-1}$ are computed, then $(D/2)/2$ data at scale $L/2^{N-2}$, ... up to finally obtaining 2 data at scale $L/2$. The result of this algorithm is an array of the same length as the input one, where the data are usually sorted from the largest scales to the smallest ones.

Within Gwyddion the pyramidal algorithm is used for computing the discrete wavelet transform. Discrete wavelet transform in 2D can be accessed using DWT module.

Discrete wavelet transform can be used for easy and fast denoising of a noisy signal. If we take only a limited number of highest coefficients of the discrete wavelet transform spectrum, and we perform an inverse transform (with the same wavelet basis) we can obtain more or less denoised signal. There are several ways how to choose the coefficients that will be kept. Within Gwyddion, the universal thresholding, scale adaptive thresholding [2] and scale and space adaptive thresholding [3] is implemented. For threshold determination within these methods we first determine the noise variance guess given by

$$\hat{\sigma} = \frac{\text{Median}\,|Y_{ij}|}{0.6745}$$

where Y_{ij} corresponds to all the coefficients of the highest scale subband of the decomposition (where most of the noise is assumed to be present). Alternatively, the noise variance can be obtained in an independent way, for example from the AFM signal variance while not scanning. For the highest frequency subband (universal thresholding) or for each subband (for scale adaptive thresholding) or for each pixel neighbourhood within subband (for scale and space adaptive thresholding) the variance is computed as

$$\hat{\sigma}_Y^2 = \frac{1}{n^2}\sum_{i,j=1}^{n} Y_{ij}^2$$

Treshold value is finally computed as

$$T(\hat{\sigma}_X) = \hat{\sigma}^2/\hat{\sigma}_X$$

where

$$\hat{\sigma}_X = \sqrt{\max(\hat{\sigma}_Y^2 - \hat{\sigma}^2, 0)}$$

When threshold for given scale is known, we can remove all the coefficients smaller than threshold value (hard thresholding) or we can lower the absolute value of these coefficients by threshold value (soft thresholding).

DWT denoising can be accessed with *Data Process* → *Integral Transforms* → *DWT Denoise*.

Continuous Wavelet Transform

Continuous wavelet transform (CWT) is an implementation of the wavelet transform using arbitrary scales and almost arbitrary wavelets. The wavelets used are not orthogonal and the data obtained by this transform are highly correlated. For the discrete time series we can use this transform as well, with the limitation that the smallest wavelet translations must be equal to the data sampling. This is sometimes called Discrete Time Continuous Wavelet Transform (DT-CWT) and it is the most used way of computing CWT in real applications.

In principle the continuous wavelet transform works by using directly the definition of the wavelet transform, i.e. we are computing a convolution of the signal with the scaled wavelet. For each scale we obtain by this way an array of the same length N as the signal has. By using M arbitrarily chosen scales we obtain a field $N \times M$ that represents the time-frequency plane directly.

The algorithm used for this computation can be based on a direct convolution or on a convolution by means of multiplication in Fourier space (this is sometimes called Fast Wavelet Transform).

The choice of the wavelet that is used for time-frequency decomposition is the most important thing. By this choice we can influence the time and frequency resolution of the result. We cannot change the main features of WT by this way (low frequencies have good frequency and bad time resolution; high frequencies have good time and bad frequency resolution), but we can somehow increase the total frequency of total time resolution. This is directly proportional to the width of the used wavelet in real and Fourier space. If we use the Morlet wavelet for example (real part – damped cosine function) we can expect high frequency resolution as such a wavelet is very well localized in frequencies. In contrary, using Derivative of Gaussian (DOG) wavelet will result in good time localization, but poor one in frequencies.

CWT is implemented in the CWT module that can be accessed with *Data Process* → *Integral Transforms* → *CWT*.

References

[1] A. Bultheel: Bull. Belg. Math. Soc.: (1995) 2

[2] S. G. Chang, B. Yu, M. Vetterli: IEEE Trans. Image Processing, (2000) 9 p. 1532

[3] S. G. Chang, B. Yu, M. Vetterli: IEEE Trans. Image Processing, (2000) 9 p. 1522

4.13 Fractal Analysis

In practice objects exhibiting random properties are encountered. It is often assumed that these objects exhibit the self-affine properties in a certain range of scales. Self-affinity is a generalization of self-similarity which is the basic property of most of the deterministic fractals. A part of self-affine object is similar to whole object after anisotropic scaling. Many randomly rough surfaces are assumed to belong to the random objects that exhibit the self-affine properties and they are treated self-affine statistical fractals. Of course, these surfaces can be studied using atomic force microscopy (AFM). The results of the fractal analysis of the self-affine random surfaces using AFM are often used to classify these surfaces prepared by various technological procedures [1,2,3,4].

Within Gwyddion, there are different methods of fractal analysis implemented within *Data Process* → *Statistics* → *Fractal analysis*.

Cube counting method [1,2] is derived directly from a definition of box-counting fractal dimension. The algorithm is based on the following steps: a cubic lattice with lattice constant l is superimposed on the z-expanded surface. Initially l is set at $X/2$ (where X is length of edge of the surface), resulting in a lattice of $2 \times 2 \times 2 = 8$ cubes. Then $N(l)$ is the number of all cubes that contain at least one pixel of the image. The lattice constant l is then reduced stepwise by factor of 2 and the process repeated until l equals to the distance between two adjacent pixels. The slope of a plot of $\log N(l)$ versus $\log 1/l$ gives the fractal dimension D_f directly.

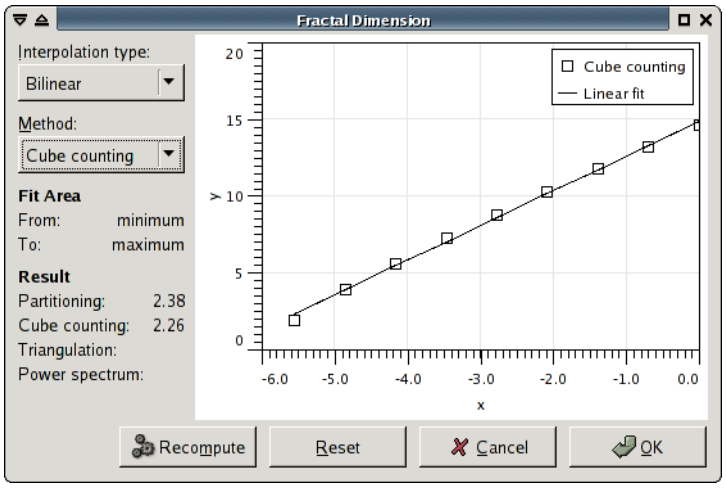
Triangulation method [1] is very similar to cube counting method and is also based directly on the box-counting fractal dimension definition. The method works as follows: a grid of unit dimension l is placed on the surface. This defines the location of the vertices of a number of triangles. When, for example, $l = X/4$, the surface is covered by 32 triangles of different areas inclined at various angles with respect to the xy plane. The areas of all triangles are calculated and summed to obtain an approximation of the surface area $S(l)$ corresponding to l. The grid size is then decreased by successive factor of 2, as before, and the process continues until l corresponds to distance between two adjacent pixel points. The slope of a plot of $S(l)$ versus $\log 1/l$ then corresponds to $D_\mathrm{f} - 2$.

Variance method [3,4] is based on the scale dependence of the variance of fractional Brownian motion. In practice, in the variance method one divides the full surface into equal-sized squared boxes, and the variance (power of RMS value of heights), is calculated for a particular box size. Fractal dimension is evaluated from the slope β of a least-square regression line fit to the data points in log-log plot of variance as $D_\mathrm{f} = 3 - \beta/2$.

Power spectrum method [3,4,5] is based on the power spectrum dependence of fractional Brownian motion. In the power spectrum method, every line height profiles that forms the image is Fourier transformed and the power spectrum evaluated and then all these power spectra are averaged. Fractal dimension is evaluated from the slope β of a least-square regression line fit to the data points in log-log plot of power spectrum as $D_\mathrm{f} = 7/2 - \beta/2$.

The axes in Fractal Dimension graphs always show already logarithmed quantities, therefore the linear dependencies mentioned above correspond to straight lines there. The measure of the axes should be treated as arbitrary.

Note, that results of different methods differ. This fact is caused by systematic error of different fractal analysis approaches.



Fractal Dimension dialog.

Moreover, the results of the fractal analysis can be influenced strongly by the tip convolution. We recommend therefore to check the certainty map before fractal analysis. In cases when the surface is influenced a lot by tip imaging, the results of the fractal analysis can be misrepresented strongly.

Note, that algorithms that can be used within the fractal analysis module are also used in Fractal Correction module and Fractal Correction option of Remove Spots tool.

References

[1] C. Douketis, Z. Wang, T. L. Haslett, M. Moskovits: Fractal character of cold-deposited silver films determined by low-temperature scanning tunneling microscopy. Physical Review B, Volume 51, Number 16, 15 April 1995, 51

[2] W. Zahn, A. Zösch: The dependence of fractal dimension on measuring conditions of scanning probe microscopy. Fresenius J Analen Chem (1999) 365: 168-172

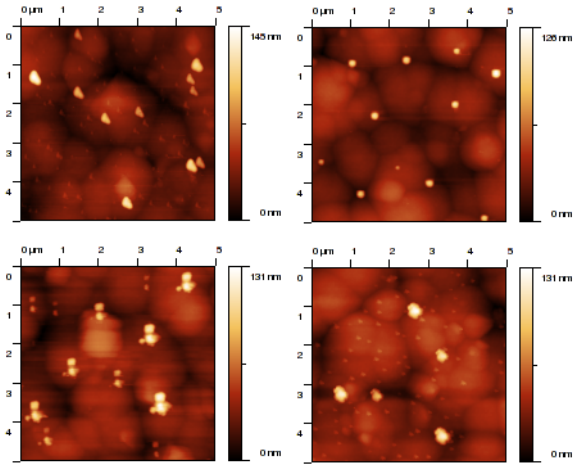
[3] A. Van Put, A. Vertes, D. Wegrzynek, B. Treiger, R. Van Grieken: Quantitative characterization of individual particle surfaces by fractal analysis of scanning electron microscope images. Fresenius J Analen Chem (1994) 350: 440-447

[4] A. Mannelquist, N. Almquist, S. Fredriksson: Influence of tip geometry on fractal analysis of atomic force microscopy images. Appl. Phys. A 66,1998, 891-895

[5] W. Zahn, A. Zösch: Characterization of thin film surfaces by fractal geometry. Fresenius J Anal Chem (1997) 358: 119-121

4.14 Tip Convolution Artefacts

Tip convolution artefact is one of the most important error sources in SPM. As the SPM tip is never ideal (like delta function) we often observe a certain degree of image distortion due to this effect. We can even see some SPM tips imaged on the surface scan while sharp features are present on the surface.



Images of ZnSe surface measured with four different SPM tips (more or less broken ones).

We can fortunately simulate and/or correct the tip effects using algorithms of dilation and/or erosion, respectively. These algorithms were published by Villarubia (see [1]).

Obtaining the Tip Geometry

For studying the tip influence on the data we need to know tip geometry first. In general, the geometry of the SPM tip can be determined in these ways:

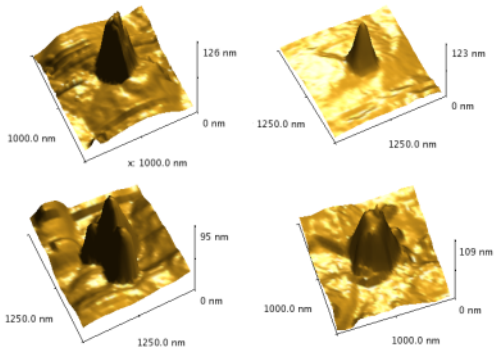
1. use manufacturer's specifications (tip geometry, apex radius and angle)
2. use scanning electron microscope of other independent technique to determine tip properties
3. use known tip characterizer sample (with steep edges)
4. use blind tip estimation algorithm together with tip characterizers or other suitable samples

Within Gwyddion, we can use the first and the last approach of the mentioned ones. Using tip modelling (*Data Process* → *Tip* → *Model Tip*) most of the tips with simple geometries can be simulated. This way of tip geometry specification can be very efficient namely when we need to check only certainty map of perform tip convolution simulation.

To obtain more detailed (and more realistic) tip structure blind tip estimation algorithm can be used (*Data Process* → *Tip* → *Blind Estimation*).

Blind tip estimation algorithm is an extension of the well-known fact that on some surface data we can see images of certain parts of tip directly. The algorithm iterates over all the surface data and at each point tries to refine each tip point according to steepest slope in the direction between concrete tip point and tip apex.

We can use two modification of this algorithm within Gwyddion: *partial* tip estimation that uses only limited number of highest points on the image and *full* tip estimation that uses full image (and is much slower therefore). Within Gwyddion tip estimation module we can use also partial tip estimation results as starting point for full estimation. This should improve the full tip estimation algorithm speed.

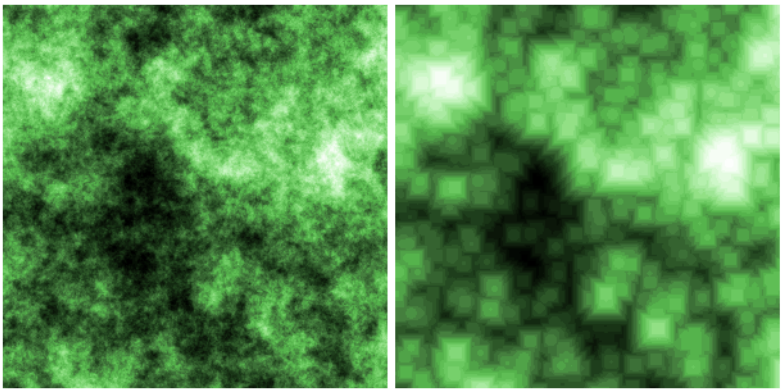


SPM tips obtained from data of previous figure using blind estimation algorithm.

Tip Convolution and Surface Reconstruction

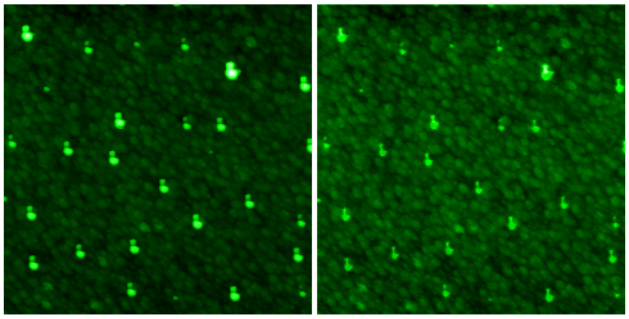
When we know tip geometry, we can use tip convolution (dilation) algorithm to simulate data acquisition process. For doing this use Dilation module (*Data Process* → *Tip* → *Dilation*). This can be in particular useful when working with data being result of some numerical modelling (see e.g. [2]).

Note this algorithm (as well as the following two) requires compatible scan and tip data, i.e. the physical dimensions of a scan pixel and of a tip image pixels have to be equal. This relation is automatically guaranteed for tips obtained by blind estimate when used on the same data (or data with an identical measure). If you obtained the tip image by other means, you may need to resample it.



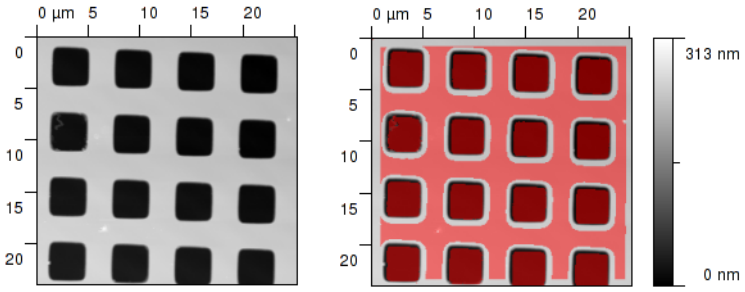
Simulated fractal surface before (left) and after (right) tip convolution.

The opposite of the tip convolution is surface reconstruction (erosion) that can be used to correct partially the tip influence on image data. For doing this, use Surface Reconstruction function (*Data Process* → *Tip* → *Surface Reconstruction*). Of course, the data corresponding to points in image not touched by tip (e. g. pores) cannot be reconstructed as there is no information about these points.



Original and reconstructed image of ZnSe imaged by broken SPM tip.

As it can be seen, the most problematic parts of SPM image are data points where tip did not touch the surface in a single point, but in multiple points. There is a loss of information in these points. Certainty map algorithm can mark points where surface was probably touched in a single point.



Certainty map obtained from standard grating. Note that the modelled tip parameters were taken from datasheet here for illustration purposes. (left) – sample, (right) – sample with marked certainty map.

Certainty map algorithm can be therefore used to mark data in the SPM image that are corrupted by tip convolution in an irreversible way. For SPM data analysis on surfaces with large slopes it is important to check always presence of these points. Within Gwyddion you can use Certainty Map function for creating these maps (*Data Process* → *Tip* → *Certainty Map*).

References

[1] J. S. Villarubia, J. Res. Natl. Inst. Stand. Technol. 102 (1997) 425.

[2] P. Klapetek, I. Ohlídal, Ultramicroscopy, 94 (19-29), 2003

4.15 Multiple Data

Arithmetic

Data Process → *Multidata* → *Arithmetic*

Data Arithmetic module enables to perform arbitrary point-wise operations on a single data field or on the corresponding points of several data fields (currently up to eight). And although it is not its primary function it can be also used as a calculator with immediate expression evaluation if a plain numerical expression is entered. The expression syntax is described in section Expressions.

The expression can contain the following variables representing values from the individual input data fields:

Variable	Description
`d1`, ..., `d8`	Data value at the pixel. The value is in base physical units, e.g. for height of 233 nm, the value of `d1` is 2.33e-7.
`m1`, ..., `m8`	Mask value at the pixel. The mask value is either 0 (for unmasked pixels) or 1 (for masked pixels). The mask variables can be used also if no mask is present; the value is then 0 for all pixels.
`bx1`, ..., `bx8`	Horizontal derivative at the pixel. Again, the value is in physical units. The derivative is calculated as standard symmetrical derivative, except in edge pixels where one-side derivative is taken.
`by1`, ..., `by8`	Vertical derivative at the pixel, defined similarly to the horizontal derivative.
`x`	Horizontal coordinate of the pixel (in real units). It is the same in all fields due to the compatibility requirement (see below).
`y`	Vertical coordinate of the pixel (in real units). It is the same in all fields due to the compatibility requirement (see below).

In addition, the constant π is available and can be typed either as π or **`pi`**.

All data fields that appear in the expression have to be compatible. This means their dimensions (both pixel and physical) have to

be identical. Other data fields, i.e. those not actually entering the expression, are irrelevant. The result is always put into a newly created data field in the current file (which may be different from the files of all operands).

Since the evaluator does not automatically infer the correct physical units of the result the units have to be explicitly specified. This can be done by two means: either by selecting a data field that has the same value units as the result should have, or by choosing option *Specify units* and typing the units manually.

The following table lists several simple expression examples:

Expression	Meaning
`-d1`	Value inversion. The result is very similar to Invert Value, except that Invert Value reflects about the mean value while here we simply change all values to negative.
`(d1 - d2)^2`	Squared difference between two data fields.
`d1 + m1*1e-8`	Modification of values under mask. Specifically, the value 10^{-8} is added to all masked pixels.
`d1*m3 + d2*(1-m3)`	Combination of two data fields. Pixels are taken either from data field 1 or 2, depending on the mask on field 3.

In the calculator mode the expression is immediately evaluated as it is typed and the result is displayed below Expression entry. No special action is necessary to switch between data field expressions and calculator: expressions containing only numeric quantities are immediately evaluated, expressions referring to data fields are used to calculate a new data field. The preview showing the result of an operation with fields is not immediately updated as you type; you can update it either by clicking *Update* or just pressing **Enter** in the expression entry.

Detail Immersion

Data Process → *Multidata* → *Immerse*

Immerse insets a detailed, high-resolution image into a larger image. The image the function was run on forms the large, base image.

The detail can be positioned manually on the large image with mouse. Button *Improve* can then be used to find the exact coordinates in the neighbourhood of the current position that give the maximum correlation between the detail and the large image. Or the best-match position can be searched through the whole image with *Locate*.

It should be noted that correlation search is insensitive to value scales and offsets, therefore the automated matching is based solely on data features, absolute heights play no role.

Result Sampling controls the size and resolution of the result image:

Upsample large image The resolution of the result is determined by the resolution of the inset detail. Therefore the large image is scaled up.

Downsample detail The resolution of the result is determined by the resolution of the large image. The detail is downsampled.

Detail Leveling selects the transform of the z values of the detail:

None No z value adjustment is performed.

Mean value All values of the detail image are shifted by a constant to make its mean value match the mean value of the corresponding area of the large image.

Merging

Data Process → *Multidata* → *Merge*

Images that form parts of a larger image can be merged together with Merge. The image the function was run on corresponds to the base image, the image selected with *Merge with* represents the second operand. The side of the base image the second one will be attached to is controlled with *Put second operand* selector.

If the images match perfectly, they can be simply placed side by side with no adjustments. This behaviour is selected by option *None* of alignment control *Align second operand*.

However, usually adjustments are necessary. If the images are of the same size and aligned in direction pependicular to the merging direction the only degree of freedom is possible overlap. The *Join* aligment method can be used in this case. Unlike in the correlation search described below, the absolute data values are matched. This makes this option suitable for merging even very slowly varying images provided they absolute height values are well defined.

Option *Correlation* selects automated alignment by correlation-based search of the best match. The search is performed both in the direction parallel to the attaching side and in the perpendicular direction. If a parallel shift is present, the result is expanded to contain both images fully (with undefined data filled with a background value).

Option *Boundary treatment* is useful only for the latter case of imperfectly aligned images. It controls the treatment of overlapping areas in the source images:

First operand Values in overlapping areas are taken from the first, base image.

Second operand Values in overlapping areas are taken from the second image.

Smooth A smooth transition between the first and the second image is made through the overlapping area by using a weighted average with a suitable weighting function.

Mutual Crop

Data Process → Multidata → Mutual Crop

Two slightly different images of the same area (for example, before and after some treatment) can be croped to intersecting area (or non-intersecting parts can be removed) with this module.

Intersecting part is determined by correlation of larger image with center area of smaller image. Images resolution (pixels per linear unit) should be equal.

The only parameter now is *Select second operand* - correlation between it and current image will be calculated and both data fields will be cropped to remove non-intersecting near-border parts.

Cross-Correlation

Data Process → Multidata → Cross-correlation

This module finds local correlations between details on two different images. As an ideal output, the shift of every pixel on the first image as seen on the second image is returned. This can be used for determining local changes on the surface while imaged twice (shifts can be for example due to some sample deformation or microscope malfunction).

For every pixel on the first operand (actual window), the module takes its neighbourhood and searches for the best correlation in the second operand within defined area. The position of the correlation maximum is used to set up the value of shift for the mentioned pixel on the first operand.

Second operand Image to be used for comparison with the first operand - base image.

Search size Used to set the area whera algorithm will search for the local neighbourhood (on the second operand). Should be larger than window size. Increase this size if there are big differences between the compared images.

Window size Used to set the local neighbourhood size (on the first operand). Should be smaller than search size. Increasing this value can improve the module functionality, but it will surely slow down the computation.

Output type Determines the output (pixel shift) format.

Add low score threshold mask For some pixels (with not very pronounced neighbourhood) the correlation scores can be small everywhere, but the algorithm anyway picks some maximum value from the scores. To see these pixels and possibly remove them from any further considerations you can let the module to set mask of low-score pixel shifts that have larger probability to be not accurately determined.

Mask by Correlation

Data Process → *Multidata* → *Mask by Correlation*

This module searches for a given correlation pattern within the actual image. The resulting pattern position is marked as a mask in the data window.

Correlation kernel Image to be found on the base image.

Output type There are several possibilities what to output: local correlation maxima (single points), masks of kernel size for each correlation maximum (good for presentation purposes), or simply the correlation score.

Correlation method Algorithm for computing correlation can be selected here.

Threshold Threshold for determining whether the local maximum will be treated as "correlation kernel found here".

Neural network processing

Neural network processing can be used to calculate one kind of data from another even if the formula or relation between them is not explicitly known. The relation is built into the network implicitly by a process called training which employs pairs of known input and output data, usually called model and signal. In this process, the network is optimised to reproduce as well as possible the signal from model. A trained network can then be used to process model data for which the output signal is not available and obtain – usually somewhat approximately – what the signal would look like. Another possible application is the approximation of data processing methods that are exact but very time-consuming. In this case the signal is the output of the exact method and the network is trained to reproduce that.

Since training and application are two disparate steps they are present as two different functions in Gwyddion.

Training

Data Process → *Multidata* → *Neural Network Training*

The main functions that control the training process are contained in tab *Training*:

Model Model data, i.e. input for training. Multiple models can be chosen sequentially for training (with corresponding signals).

Signal Signal data for training, i.e. the output the trained network should produce. The signal data field must be compatible with model field, i.e. it must have the same pixel and physical dimensions.

Training steps Number of training steps to perform when *Train* is pressed. Each step consists of one pass through the entire signal data. It is possible to set the number of training steps to zero; no training pass is performed then but the model is still evaluated with the network and you can observe the result.

Train Starts training. This is a relatively slow process, especially if the data and/or window size are large.

Reinitialize Reinitializes the neural network to an untrained state. More precisely, this means neuron weights are set to random numbers.

Masking Mode It is possible to train the network only on a subset of the signal, specified by a mask on the signal data. (Masking of model would be meaningless due to the window size.)

Neural network parameters can be modified in tab *Parameters*. Changing either the window dimensions or the number of hidden nodes means the network is reinitialized (as if you pressed *Reinitialize*).

Window width Horizontal size of the window. The input data for the network consists of an area around the model pixel, called window. The window is centered on the pixel so, odd sizes are usually preferred.

Window height Vertical size of the window.

Hidden nodes Number of nodes in the "hidden" layer of the neural network. More nodes can lead to more capable network, on the other hand, it means slower training and application. Typically, this number is small compared to the number of pixels in the window.

Power of source XY The power in which the model lateral dimensions units should appear in the signal. This is only used when the network is applied.

Power of source Z The power in which the model "height" units should appear in the signal. This is only used when the network is applied.

Fixed units Fixed units of the result. They are combined with the other units parameters so if you want result units that are independent of input units you have to set both powers to 0. This is only used when the network is applied.

Trained neural network can be saved, loaded to be retrained on different data, etc. The network list management is similar to raw file presets.

In addition to the networks in the list, there is one more unnamed network and that of the network currently in training. When you load a network the network in training becomes a copy of the loaded network. Training then does not change the named networks; to save the network after training (under existing or new name) you must explicitly use *Store*.

Application

Data Process → *Multidata* → *Apply Neural Network*

Application of a trained neural network is simple: just choose one from the list and press *OK*. The unnamed network currently in training is also present in the list under the label "In training".

Since neural networks process and produce normalised data, it does not perserve proportionality well, especially if the scale of training model differs considerably from the scale of real inputs. If the output is expected to scale with input you can enable option *Scale proportionally to input* that scales the output with the inverse ratio of actual and training input data ranges.

4.16 Graph Processing

Many of the Gwyddion data processing modules produce graph as an output. Graphs can be exported into text files or further analyzed within Gwyddion by several graph processing modules. These modules can be found in the Graph menu in the Gwyddion main window. Note that the number of graph modules is quite limited now and consists of basic modules for doing things that are very frequent within SPM data analysis. For more analytical tools you can use your favorite graph processing program.

In this section the graph modules present in Gwyddion are briefly presented.

Basic Operations

First of all zooming and data reading functions are available directly in the graph window:

- Logarithmic axes – horizontal and vertical axes can be switched between linear and logarithmic using the logscale buttons. Switching to logarithmic scale is possible only for positive values (either on abscissa or ordinate).
- Zoom in and zoom out – after selecting zoom in simply draw the area that should be zoomed by mouse. Zoom out restores the state where all data can be seen.
- Measure distances – enables user to select several points within the graph and displays their distances and angles between them.

Graph Level

Graph level is a very simple module that currently performs linear fit of each graph curve and subtracts the fitted linear functions from them.

Graph Align

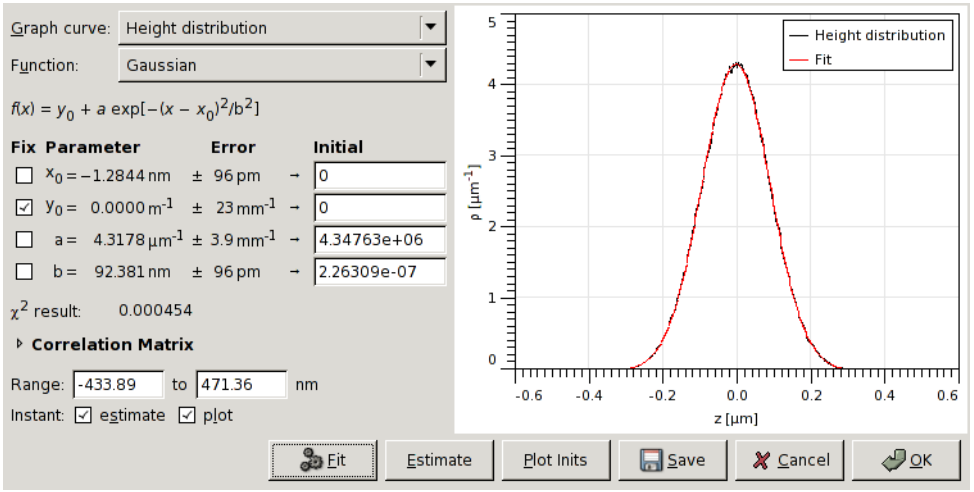
Graph align shifts the graphs curves horizontally to maximise their mutual correlations, i.e. match common features in the curves. This is useful in particular for comparison of profiles taken in different locations.

Function Fitting

The curve fitting is designed namely for fitting of statistical functions used in roughness parameters evaluation. Therefore most of the available functions are currently various statistical functions of surfaces with Gaussian or exponential autocorrelation functions. Nevertheless it also offers a handful of common general-purpose functions.

Within the fitting module you can select the area that should be fitted (with mouse or numerically), try some initial parameters, or let the module to guess them, and then fit the data using Marquardt-Levenberg algorithm.

As the result you obtain the fitted curve and the set of its parameters. The fit report can be saved into a file using *Save* button. Pressing *OK* button adds the fitted curve to the graph, if this is not desirable, quit the dialog with *Cancel*.



Curve fitting module dialog

Force-Distance Curve Fitting

The module for fitting of force-distance curves is very similar to the general curve fitting module, it is just specialized for force-distance curves. Currently, the module serves for fitting jump-in part of force-distance curve (representing attractive forces) using different models:

- van der Waals force between semisphere and half-space
- van der Waals force between pyramide and half-space
- van der Waals force between truncated pyramide and half-space
- van der Waals force between sphere and half-space
- van der Waals force between two spheres
- van der Waals force between cone and half-space
- van der Waals force between cylinder and half-space
- van der Waals force between paraboloid and half-space

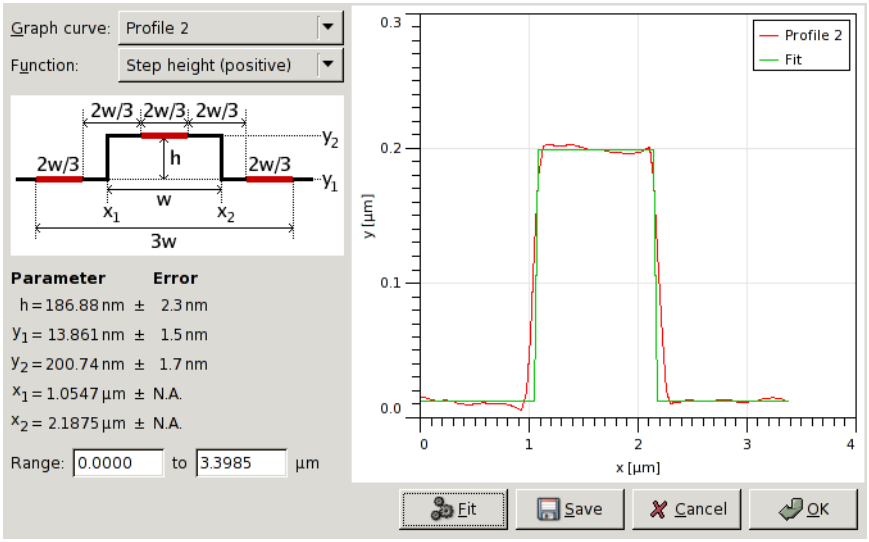
Note that the curve being fitted must be a real force-distance curve, not a displacement-distance or sensor-distance curve. Recalculation of cantilever deflection into force should be done before calling this module.

Also note that for small cantilever spring constants the amount of usable data in attractive region is limited by effect of jumping into contact.

Critical Dimension

Critical dimension module can be used to fit some “typical” objects that are often found while analyzing profiles extracted from microchips and related surfaces. These objects are located in the graph and their properties are evaluated.

The user interface of this module is practically the same as of the graph fit module.



Critical dimension module dialog.

DOS spectrum

DOS spectrum module intended to obtain Density-of-States spectra from I-V STM spectroscopy. It calculates

$$\left| \frac{\frac{\mathrm{d}I}{\mathrm{d}U}}{\frac{I}{U}} \right| (U)$$

and plots it as graph.

4.17 Synthetic Surfaces

Beside functions for analysis of measured data, Gwyddion provides several generators of artificial surfaces that can be used for testing or simulations also outside Gwyddion.

All the surface generators share a certain set of parameters, determining the dimensions and scales of the created surface and the random number generator controls. These parameters are described below, the parameters specific to each generator are described in the corresponding subsections.

Image parameters:

Horizontal, Vertical size The horizontal and vertical resolution of the generated surface in pixels.

Square image This option, when enabled, forces the horizontal and vertical resolution to be identical.

Width, Height The horizontal and vertical physical dimensions of the generated surface in selected units. Note square pixels are assumed so, changing one causes the other to be recalculated.

Dimension, Value units Units of the lateral dimensions (*Width*, *Height*) and of the values (heights). The units chosen here also determine the units of non-dimensionless parameters of the individual generators.

Take Dimensions from Current Channel Clicking this button fills all the above parameters according to the current channel.

Note that while the units of values are updated, the value scale is defined by generator-specific parameters that might not be directly derivable from the statistical properties of the current channel. Hence these parameters are not recalculated.

Replace the current channel This option has two effects. First, it causes the dimensions and scales to be automatically set to those of the current channel. Second, it makes the generated surface replace the current channel instead of creating a new channel.

Start from the current channel This option has two effects. First, it causes the dimensions and scales to be automatically set to those of the current channel. Second, it makes the generator to start from the surface contained in the current channel and modify it instead of starting from a flat surface. Note this does not affect whether the result actually goes to the current channel or a new channel is created.

Random generator controls:

Random seed The random number generator seed. Choosing the same parameters and resolutions and the same random seed leads to the same surface, even on different computers. Different seeds lead to different surfaces with the same overall characteristics given by the generator parameters.

New Replaces the seed with a random number.

Randomize Enabling this option makes the seed to be chosen randomly every time the generator is run. This permits to conveniently re-run the generator with a new seed simply by pressing **Ctrl-F** (see keyboard shortcuts).

Spectral

The spectral synthesis module creates randomly rough surfaces by constructing the Fourier transform of the surface according to specified parameters and then performing the inverse Fourier transform to obtain the real surface. The generated surfaces are periodic (i.e. perfectly tilable).

The Fourier image parameters define the shape of the PSDF, i.e. the Fourier coefficient modulus, the phases are chosen randomly. At present, all generated surfaces are isotropic, i.e. the PSDF is radially symmetric.

RMS The root mean square value of the heights (or of the differences from the mean plane which, however, always is the $z = 0$ plane). Button *Like Current Channel* sets the RMS value to that of the current channel.

Minimum, maximum frequency The minimum and maximum spatial frequency. Increasing the minimum frequency leads to "flattening" of the image, i.e. to removal of large features. Decreasing the maximum frequency limits the sharpness of the features.

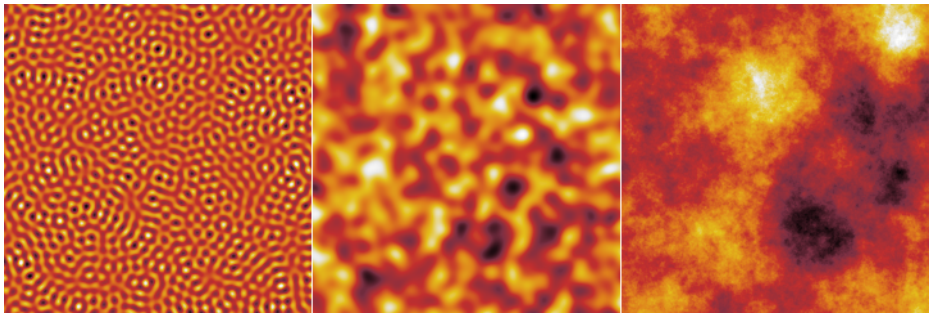
Enable Gaussian multiplier Enables the multiplication of the Fourier coefficients by a Gaussian function that in the real space corresponds to the convolution with a Gaussian.

Enable Lorentzian multiplier Enables the multiplication of the Fourier coefficients by a function proportional to $1/(1 + k^2T^2)^{3/4}$, where T is the autocorrelation length. So, the factor itself is not actually Lorentzian but it corresponds to Lorentzian one-dimensional power spectrum density which in turn corresponds to exponential autocorrelation function (see section Statistical Analysis for the discussion of autocorrelation functions). This factor decreases relatively slowly so the finite resolution plays usually a larger role than in the case of Gaussian.

Autocorrelation length The autocorrelation length of the Gaussian or Lorentzian factors (see section Statistical Analysis for the discussion of autocorrelation functions).

Enable power multiplier Enables multiplication of Fourier coefficients by factor proportional to $1/k^p$, where k is the spatial frequency and p is the power. This permits to generate various fractal surfaces.

Power The power p.

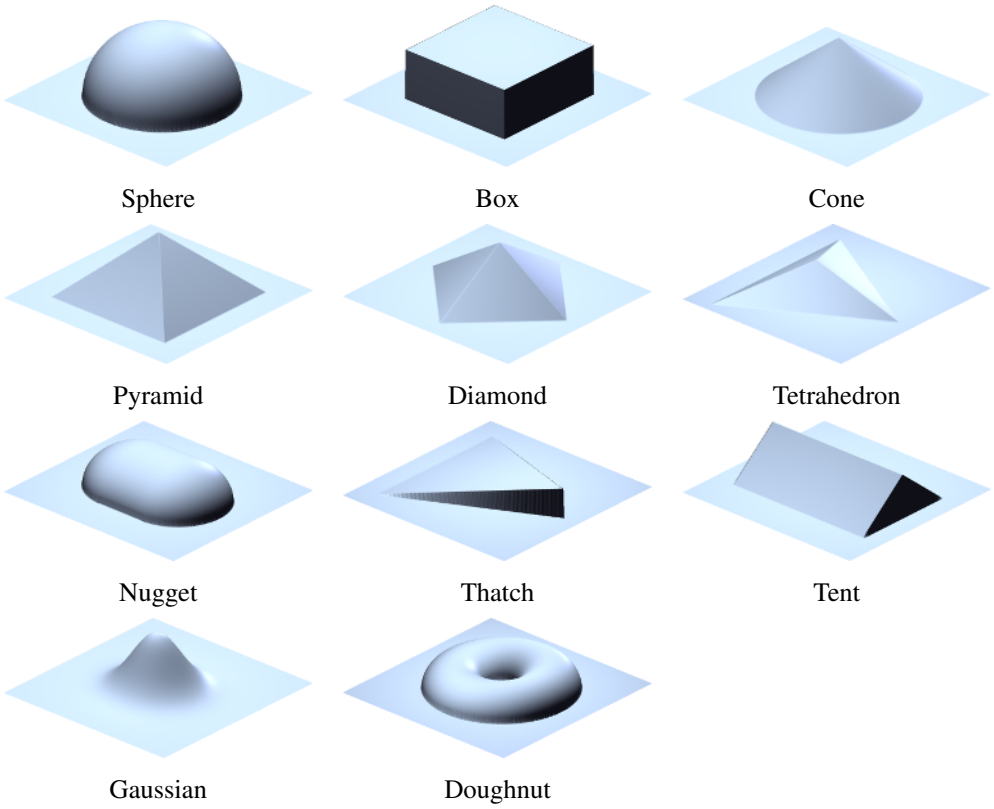


Artificial surfaces generated by spectral synthesis: a narrow range of spatial frequencies (left), Gaussian random surface (centre) and a fractal surface generated with power multiplier and p equal to 1.5 (right).

Objects

The object placement method permits to create random surfaces composed of features of a specific shape. The algorithm is simple: the given number of objects is placed on random positions at the surface. For each object placed, the new heights are changed to $\max(z, z_0 + h)$, where z is the current height at a specific pixel, h is the height of the object at this pixel (assuming a zero basis) and z_0 is the current minimum height over the basis of the object being placed. The algorithm considers the horizontal plane to be filled with identical copies of the surface, hence, the generated surfaces are also periodic (i.e. perfectly tilable).

Shape The shape (type) of placed objects. At present the possibilities include half-spheres, boxes, pyramids, tetrahedrons and some more weird shapes.



Coverage The average number of times an object covers a pixel on the image. Coverage value of 1 means the surface would be exactly once covered by the objects assuming that they covered it uniformly. Larger values mean more layers of objects – and slower image generation.

Size The lateral object size, usually the side of a containing square.

Aspect Ratio The ratio between the x and y dimensions of an object – with respect to some default proportions.

Changing the aspect ratio does not always imply mere geometrical scaling, e.g. objects called nuggets change between half-spheres and rods when the ratio is changed.

Height A quantity proportional to the height of the object, normally the height of the highest point.

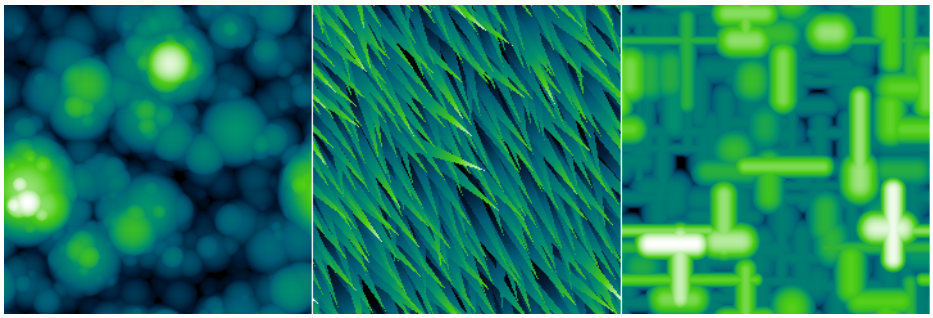
Checking *Scales with size* makes unperturbed heights to scale proportionally with object size. Otherwise the height is independent on size.

Button *Like Current Channel* sets the height value to a value based on the RMS of the current channel.

Orientation The rotation of objects with respect to some base orientation, measured counterclockwise.

Truncate The shapes can be truncated at a certain height, enabling creation of truncated cones, pyramids, etc. The truncation height is given as a proportion to the total object height. Unity means the shape is not truncated, zero would mean complete removal of the object.

Each parameter can be randomized for individual objects, this is controlled by *Variance*. For multiplicative quantities (all except orientation and truncation), the distribution is log-normal with the RMS value of the logarithmed quantity given by *Variance*.



Artificial surfaces generated by object placement: spheres of varied size (left), narrow thatches of varied direction (centre), nuggets of varied aspect ratio (right).

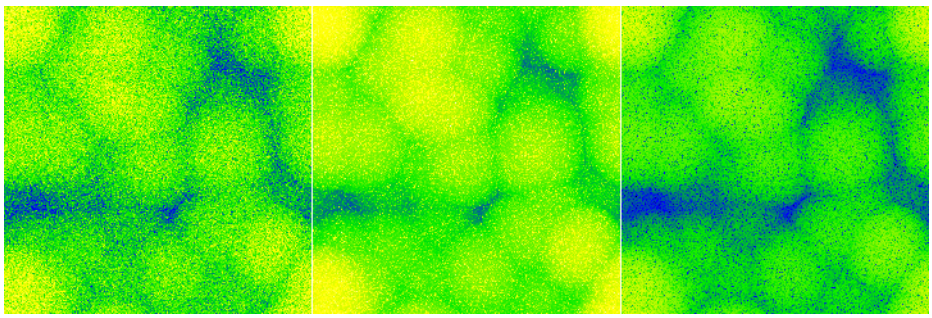
Noise

Random uncorrelated point noise is generated independently in each pixel. Several distributions are available.

Distribution The distribution of the noise value. The possibilities include Gaussian, exponential, uniform and triangular distributions.

Direction The noise can be generated as symmetrical or one-sided. The mean value of the distribution of a symmetrical noise is zero, i.e. the mean value of data does not change when a symmetrical noise is added. One-sided noise only increases (if positive) or decreases (if negative) the data values.

RMS Root mean square value of the noise distribution. More precisely, it is the RMS of the corresponding symmetrical distribution in the case the distribution is one-sided.



Different artificial noise applied to the same surface: symmetrical Gaussian noise (left); positive exponential noise (centre); negative exponential noise (right). All images have the same false colour scale and all noises have the same RMS.

Line Noise

Line noise represents noise with non-negligible duration that leads to typical steps or scars (also called strokes) in the direction of the fast scanning axis. Parameters *Distribution*, *Direction* and *RMS* have the same meaning as in Point noise. Other parameters control the lateral characteristics of the noise.

Two basic line defect types are available: steps and scars. Steps represent abrupt changes in the value that continue to the end of the scan (or until another step occurs). Scars are local changes of the value with a finite duration, i.e. the values return to the original level after some time. Ridges are similar but on a larger scale: they can span several scan lines.

Steps have the following parameters:

Density Average number of defects per scan line, including any dead time (as determined by parameter *Within line*).

Within line Fraction of the time to scan one line that corresponds to actual data acquisition. The rest of time is a dead time. Value 1 means there is no dead time, i.e. all steps occur within the image. Value 0 means the data acquisition time is negligible to the total line scan time, consequently, steps only occur between lines.

Cumulative For cumulative steps the random step value is always added to the current value offset; for non-cumulative steps the new value offset is directly equal to the random step value.

Scars have the following parameters:

Coverage The fraction of the the image covered by defect if they did not overlap. Since the defect may overlap coverage value of 1.0 does not mean the image is covered completely.

Length Scar length in pixels.

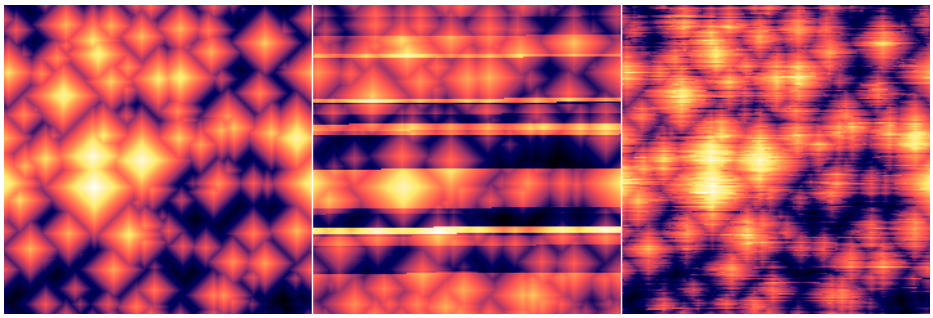
Variance Variance of the scar length, see Objects for description of variances.

Ridges have the following parameters:

Density Average number of defects per scan line, including any dead time (as determined by parameter *Within line*).

Within line Fraction of the time to scan one line that corresponds to actual data acquisition. The rest of time is a dead time. Value 1 means there is no dead time, i.e. all value changes occur within the image. Value 0 means the data acquisition time is negligible to the total line scan time, consequently, value changes only occur between lines.

Width Mean duration of the defect, measured in image size. Value 1 means the mean duration will be the entire image scanning time. Small values mean the defects will mostly occupy just one scan line.



Different types of line noise added to an artificial pyramidal surface: unmodified surface (left); with relatively unfrequent non-cumulative steps (centre); with scars of mean length of 16 px and high coverage (right).

Pattern

Regular geometrical patterns represent surfaces often encountered in microscopy as standards or testing samples such as ridges, steps or holes. Each type of pattern has its own set of geometrical parameters determining the shape and dimensions of various part of the pattern. Each parameter has a variance control, similar to Object synthesis, that permits to make the pattern irregular in some aspects.

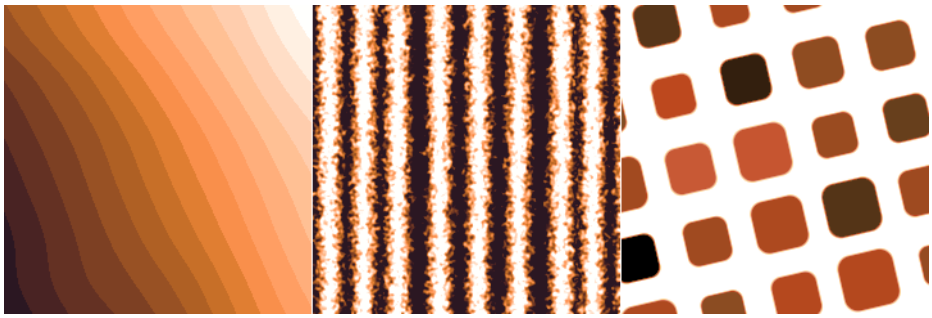
The placement of the pattern in the horizontal plane is controlled by parameters in tab *Placement*, common to all pattern types:

Orientation The rotation of the pattern with respect to some base orientation, measured counterclockwise.

This tab also contains the deformation parameters. While enabling the variation of geometrical parameters makes the generated surface somewhat irregular the shape of its features is maintained. *Deformation* is a complementary method to introduce irregularity, specifically by distorting the pattern in the *xy* plane. It has two parameters:

Amplitude The magnitude of the lateral deformation. It is a relative numerical quantity essentially determining how far the deformation can reach.

Lateral scale The characteristic size of the deformations. It describes not how far the features are moved but how sharply or slowly the deformation itself changes within the horizontal plane.



Artificial pattern surfaces: sharp steps oriented at 30 deg, deformed with a deformation with a large autocorrelation length (left); non-uniformly spaced ridges with moderate slopes between the top and bottom planes, deformed with a deformation with a small autocorrelation length (centre); unperturbed pattern of rounded holes with a large variation in size and depth (right).

Columnar films

The columnar film growth simulation utilises a simple Monte Carlo deposition algorithm in which small particles are incident on the surface from directions generated according to given paramerters, and they stick to the surface around the point where they hit it, increasing the height there locally. The shadowing effect then causes more particles to stick to higher parts of the surface and less particles to the lower parts. This positive local height feedback leads to the formation of columns. The algorithm considers the horizontal plane to be filled with identical copies of the surface, hence, the generated surfaces are also periodic (i.e. perfectly tilable). The surface generator has the following parameters:

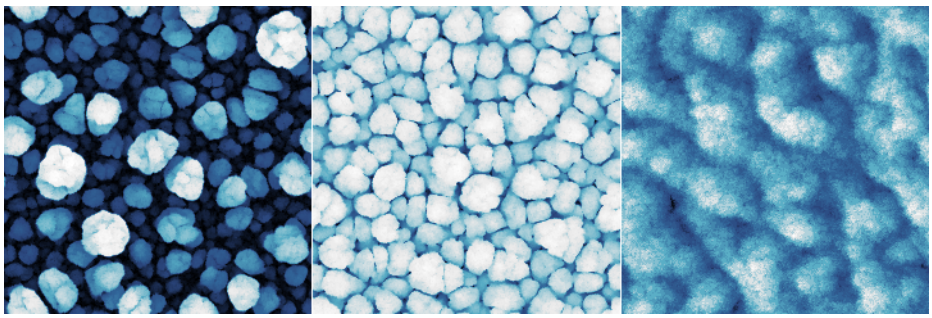
Coverage The average number of times a particle is generated above each surface pixel.

Height Local height increase occurring when the particle sticks to a pixel. Since the horizontal size of the particle is always one pixel the height is measured in pixels. A height of one pixel essentially means cubical particles, as far as growth is concerned. From the point of view of collision detection the particles are considered infinitely small.

Inclination Central inclination angle with which the particles are generated (angle of incidence). The value of zero means very small angles of incidence have the highest probability. Large values mean that the particles are more likely to impinge at large angles than at small angles. But for large direction variance, the distribution is still isotropic in the horizontal plane.

Direction Central direction in the horizontal plane with which the particles are generated. Large variance means the distribution is isotropic in the horizontal plane; for small variances the growth is anisotopic.

Relaxation Method for determination of the pixel the particle will finally stick to. Two options exist at this moment. Weak relaxation, in which only the two pixels just immediately before collision and after collision are considered and the particle sticks to the lower of them. In the case of strong relaxation, a 3×3 neighbourhood is considered in addition. The particle can move to a lower neighbour pixel with a certain probability before sticking permanently.



Artificial columnar surfaces: loosely packed columns (left); tightly packed columns (centre); directional growth with strong relaxation (right).

Ballistic deposition

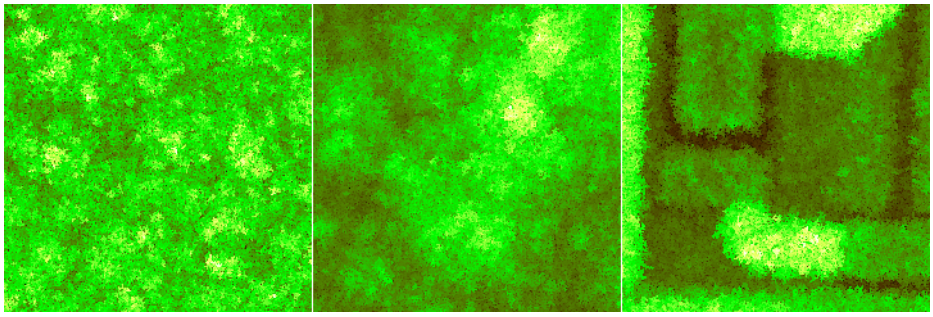
Vertical ballistic deposition is one of the simplest fundamental film growth models. Particles fall vertically onto randomly chosen sites (pixels). The height in the site is incremented by the particle height. However, if the new height would be smaller than the height in any of the four neigbour sites, the particle is assumed to stick to the neigbour columns. Thus height at the impact site

then becomes the maximum of the neigbour heights instead. This is the only mechanism introducing lateral correlations in the resulting roughness.

The simulation has only a few parameters:

Coverage The average number of times a particle is generated above each surface pixel.

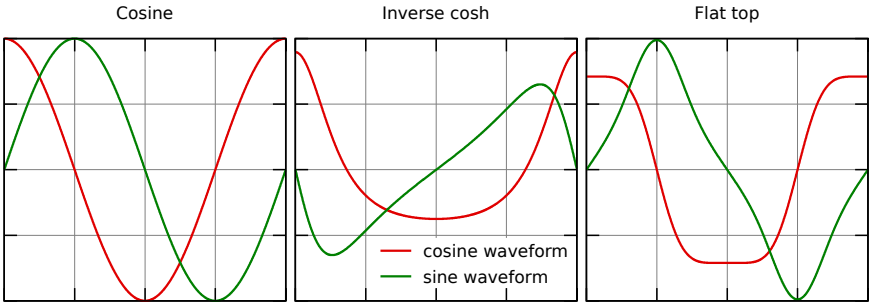
Height Local height increase occurring when the particle falls onto a pixel.



Artificial surfaces created by ballistic deposition: initial stage with relatively small correlation length (left); after correlations reached image size (centre); deposition starting from a microchip surface (right).

Waves

The wave synthesis method composes the image using interference of waves from a number of point sources. Beside the normal cosine wave, a few other wave types are available, each having a cosine and sine form that differ by phase shift $\pi/2$ in all frequency components. When the wave is treated as complex, the cosine form is its real part and its sine form is its imaginary part.



Sine and cosine wave forms for the available wave types. Only the cosine form is used for displacement images; the entire complex wave is used for intensity and phase images.

The generator has the following options:

Quantity Quantity to display in the image. Displacement is the sum of the real parts. Amplitude is the absolute value of the complex wave. Phase is the phase angle of the complex wave.

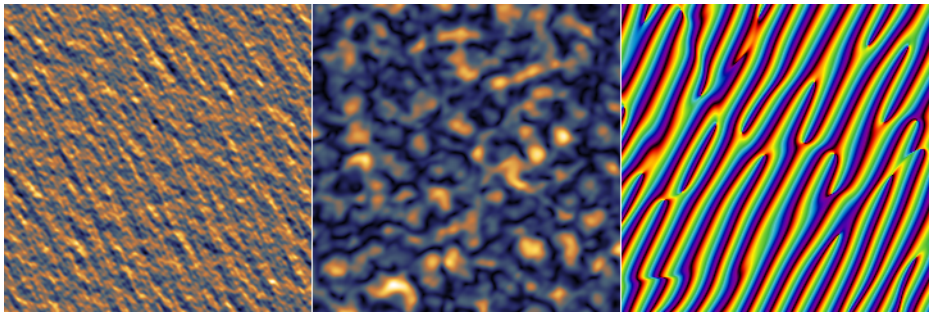
Number of waves The number of point sources the waves propagate from.

Wave form One of the wave forms described above.

Amplitude Approximate amplitude (rms) of heights in the generated image. Note it differs from the amplitude of individual waves: the amplitude of heights in the generated image would grow with the square root of the number of waves in such case, whereas it in fact stays approxiately constant unless the amplitude changes.

Frequency Spatial frequency of the waves. It is relative to the image size, i.e. the value of 1.0 means wavelength equal to the lenght of the image side.

X center*, *Y center Locations of the point sources. Zero corresponds to the image centre. The positions are also measured in image sizes. Generally, at least one of the corresponding variances should be non-zero; otherwise all the point sources coincide (some interesting patterns can be still generated with varying frequency though).



Artificial wave surfaces: displacement image generated with fairly large frequency variation (left); amplitude image for inverse cosh wave form (centre); phase image for a small cluster of far-away sources (right).

Domains

The simulation implements a hybrid non-equilibrium Ising model [1], combining a discrete short-scale discrete Ising model with continuous slow inhibitor.

The discrete variable u has two possible values, between which it can flip probabilistically with probability

$$p = \min\left\{\frac{1}{2}\exp(-\Delta E/T), 1\right\}$$

where ΔE is the change of energy resulting from flipping and T is the temperature. The energy is given by the number of neighbours in the opposite state in the directions along the coordinate axes n_o, the number of neighbours in the opposite state in the directions along diagonals n_d and also by bias caused by the inhibitor field v:

$$E = n_\mathrm{o} + \frac{1}{2}n_\mathrm{d} + Buv$$

The continuous inhibitor field v is governed by a local differential equation coupled to the variable u:

$$\frac{\mathrm{d}v}{\mathrm{d}t} = -v - \nu - \mu u$$

where μ is the inhibitor coupling and ν is the bias parameter. The boundary conditions of the simulation are periodic, hence, the generated images are also periodic (i.e. perfectly tilable).

The calculation has the following parameters:

Number of iterations One iteration consists of four steps: A Monte Carlo update of u, a time step of solution of the differential equation for v, another Monte Carlo step and another differential euqation time step. The values of v shown correspond to the second update. The quantity shown as u is the average from the two values surrounding in time the corresponding value of v. Hence the images of u are three-valued, not two-valued.

***Temperature* T** Temperature determines the probability with which the two-state variable u can flip to the other value if it results in a configurations with higher energy or similar (flips that lower the energy a lot occur unconditonally). A larger temperature means less separation between the two u domains.

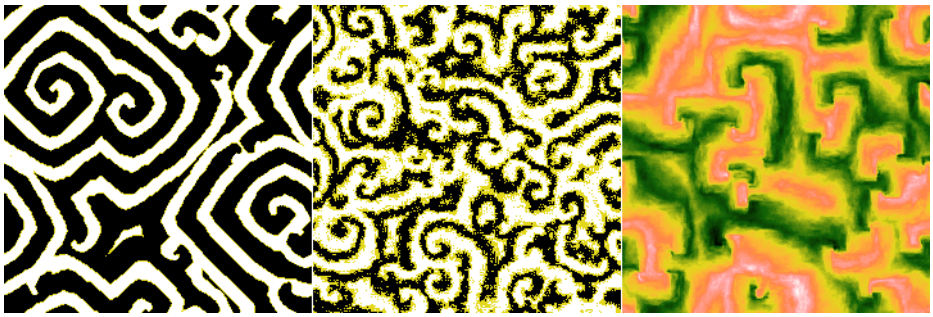
***Inhibitor strength* B** Strength with which the continuous variable biases the energy of each pixel. For large values the inhibitor has larger influence compared to the surface tension.

***Inhibitor coupling* μ** Coupling factor between u and v in the differential equation for v.

***Bias* ν** Bias in the differential equation for v towards larger or smaller values.

Monte Carlo time step Time step in the differential equation corresponding to one Monte Carlo step. So this parameter determines the relative speed of the two processes.

Height Range of values of the created images.



Artificial domain surfaces: spiral waves with long-range order (left); short-range ordered spiral waves at a higher temperature (centre); the continous variable for a low-temperature situation with small strength and coupling (right).

Diffusion

The simulation implements a Monte Carlo discrete diffusion limited aggregation model. Pixel-shaped particles fall onto the surface and move around it until they stick to a cluster, slowly forming a monolayer. The flux of incoming particles is low so that only a few particles are free to move at any given time and the particles can travel a considerable distance on the surface before sticking. For typical parameter values, this leads to the formation of "fractal flake" clusters.

The calculation has the following parameters:

Coverage The average number of times a particle falls onto any given pixel. Coverage value of 1 means the surface would be covered by a monolayer, assuming the particles cover it uniformly.

Flux The average number particles falling onto any given pixel per simulation step. Smaller values of flux lead to larger structures as the particles diffuse for longer time before they meet another particle. The value is given as a decimal logarithm, indicated by the units log_{10}.

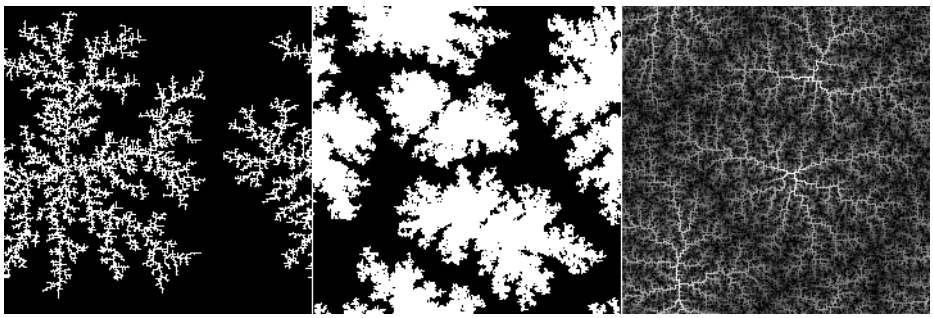
Height The height of one particle which gives the height of the steps in the resulting image.

Sticking probability The probability that a free particle will finally stick and stop moving when it touches another single particle. Probabilities that the particle will stick if it touches two or three particles increase progressively depending on this value. The sticking probability is always zero for a particle with no neighbours and always one for a particle with all four neighbours.

Activation probability The probability a particle that has not stuck will move if it touches another particle. The probability for more touching particles decreases as a power of the single-particle probability. Particles without any neighbours can always move freely.

Passing Schwoebel probability A particle that has fallen on the top of an already formed cluster can have reduced probability to descend to the lower layer due to so called Schwoebel barrier. If this parameter is 1 there is no such barrier, i.e. particles can descend freely. Conversely, probability 0 means particles can never descend to the lower layer. The value is given as a decimal logarithm, indicated by the units log_{10}.

Note that some parameter combinations, namely very small flux combined with very small Schwoebel barrier passing probability, can lead to excessively long simulation times.



Artificial diffusion surfaces: sparse clusters with high sticking probability (left); dense clusters with low sticking probability (centre); mutltilayer fractal structures with high Schwoebel barrier (right).

Lattice

The lattice synthesis module creates surfaces based on randomised two-dimensional lattices. It has two major parts. The first part, controlled by parameters in tab *Lattice*, is the creation of a set of points in plane that are organised to a more or less randomised lattice. The second part, controlled by parameters in tab *Surface*, is the actual construction of a surface based on quantities calculated from Voronoi tessellation and/or Delaunay triangulation of the set of points.

The creation of the lattice has the following parameters:

Lattice The base lattice type. The random lattice corresponds to completely randomly placed points. Other types (square, hexagonal, triangular) correspond to regular organisations of points.

Size The average cell size. More precisely, this parameter describes the mean point density. It is equal to the side of the square if the same number of points was organised to a square lattice.

Lattice relaxation Amount to which the lattice is allowed to relax. The relaxation process pushes points away from very close neighbours and towards large empty space. The net result is that cell sizes become more uniform. Of course, relaxation has no effect on regular lattices. It should be noted that relaxation requires progressive retessellation and large relaxation parameter values can slow down the surface generation considerably.

Height relaxation Amount to which the random values assigned to each point (see below) are allowed to relax. The relaxation process is similar to diffusion and leads to overal smoothing of the random values.

Orientation The rotation of the lattice with respect to some base orientation, measured counterclockwise. It is only available for regular lattices as the random lattice is isotropic.

Amplitude, ***Lateral scale*** Parameters that control the lattice deformation. They have the same meaning as in Pattern synthesis.

The final surface is constructed as a weighted sum of a subset of basic quantities derived from the Voronoi tessellation or Delaunay triangulation. Each quantity can be enabled and disabled. When it is selected in the list, its weight in the sum and its thresholding parameters can be modified using the sliders. The lower and upper threshold cut the value range (which is always normalised to $[0, 1]$) by changing all values larger than the upper threshold equal to the thresold value and similarly for the lower thresold.

Some of the basic quantities are calculated from lateral coordinates only. Some, however, are calculated from random values ("heights") assigned to each point in the set. The available quantities include:

Random constant Rounding interpolation between the random values assigned to each point of the set. This means each Voronoi cell is filled with a constant random value.

Random linear Linear interpolation between the random values assigned to each point of the set. Thus the surface is continuous, with each Delaunay triangle corresponding to a facet.

Random bumpy An interpolation similar to the previous one, but non-linear, creating relatively level areas around each point of the set.

Radial distance Distance to the closest point of the set.

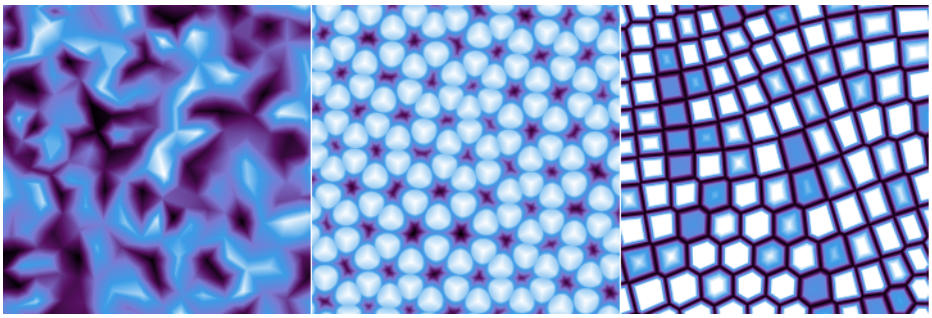
Segmented distance Distance to the closest Voronoi cell border, scaled in each cell segment so that the set point is in the same distance from all borders.

Segmented random The same quantity as segmented distance, but multiplied by the radndom value assigned to the point of the set.

Border distance Distance to the closest Voronoi cell border.

Border random The same quantity as border distance, but multiplied by the radndom value assigned to the point of the set.

Second nearest distance Distance to the second nearest point of the set.



Artificial lattice surfaces: faceted surface created by random linear interpolation (left); bumps in a distorted triangular pattern (centre); distorted rectangular pattern with segments separated by trenches (right).

Brownian

The module generates, among other things, surfaces with profiles similar to samples of fractional Brownian motion. The construction method is, however, not very sophisticated. Starting from corners of the image, interior points are recursively linearly interpolated along the horizontal and vertical axes, adding noise that scales with distance according to the given Hurst exponent. Some of the surfaces are essentially the same as those generated using spectral synthesis, however, constructing them in the direct space instead of frequency space allows different modifications of their properties.

The generator has the following parameters:

Hurst exponent The design Hurst exponent H. For the normal values between 0 and 1, the square root of height-height correlation function grows as H-th power of the distance. The construction algorithm permits formally even negative values beause it stops at the finite resolution of one pixel.

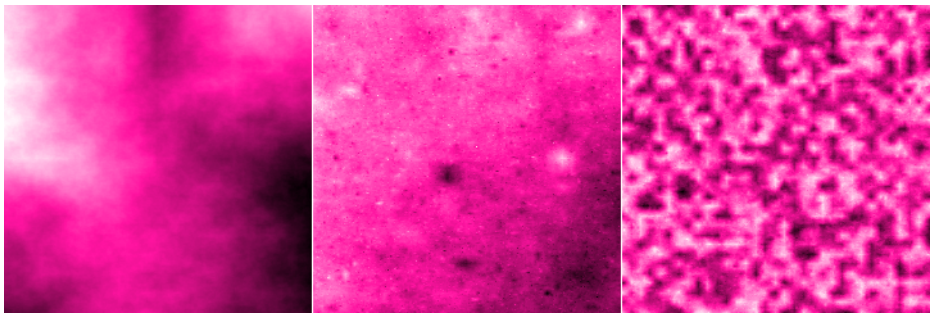
Stationarity scale Scale at which stationarity is enforced (note Fractional Brownian motion is not a stationary). When this scale is comparable to the image size or larger, it has little effect. However, when it is small the image becomes "homogeneised" instead of self-similar above this scale.

Distribution Distribution of the noise added during the generation. Uniform and Gaussian result in essentially the same surfaces (statistically); the former is faster though. More heavy-tailed distributions, i.e. exponential and especially power, lead to prominent peaks and valleys.

Power Power α for the power distribution. The probability density function is proportional to

$$\frac{\alpha/2}{(|z|+1)^{\alpha+1}}$$

RMS The root mean square value of the heights (or of the differences from the mean plane which, however, always is the $z = 0$ plane). Note this value applies to the specific generated image, not the process as such, which does not have finite RMS. Button *Like Current Channel* sets the RMS value to that of the current channel.



Artificial Brownian surfaces: classic surface generated with H about 0.7 (left); nebula-like image generated with a heavy-tailed power distribution (centre); the effect of small stationarity scale (right).

References

[1] L. M. Pismen, M. I. Monine, G. V. Tchernikov: Physica D, (2004) 199 p 82

4.18 Calibration and uncertainties

Calibration data

Calibration data can be used to provide correction of measured data or perform uncertainty calculations. Generally, calibration data can be of different types and different levels of complexity. For most of the cases user acquires error in each axis, e. g. using a calibrated standard. This value can be used for data correction. Similarly, the value of uncertainty is mostly determined for each axis from calibrated standard certificate and from measurement process uncertainty budget.

In more complex cases, calibration data can be determined locally. Scanner error cannot always be described by only three parameters (one for each axis) and its uncertainty is not necessarily the same in whole range. For precise measurements it is therefore practical to determine local errors and namely local uncertainties that can be used for further calculations. By "local" we mean here uncertainties that are related to certain location in the complete volume that can be reached by the scanner.

To obtain local errors and uncertainties, one can use a calibration standard again or use a more complex instrument, like interferometer for scanning stage calibration. This is usually done in metrology institutes.

In Gwyddion, there is a set of tools helping local uncertainty processing. Primary calibration data, related to a scanning stage, can be determined or loaded. They can be assigned to a certain SPM measurement data creating a set of calibrations. These are used automatically in tools and modules where uncertainty propagation calculation can be performed in order to provide measurement uncertainty.

Calibration data acquisition

Data Process → Calibration → 3D calibration

Calibration data can be acquired in the following ways:

The module *3D calibration → Create...* can be used for creating simplest primary calibration data - based only on knowledge of xyz errors and uncertainties and on scanner volume geometry. We also need to specify calibration name. Primary calibration data will be accessible under this name module for its data application to SPM measurements.

Using *3D calibration → Get simple error map...* module, we can determine primary xyz calibration data from a set of calibration grating measurements. Here we need to provide several measurements of calibration grating for different z-levels of the scanner. This forms several cross-sections of the full scanner volume. Providing single grating detail, nominal grating pitch and assuming that the grating planarity and orthogonality is much higher than that of scanning stage we can determine primary calibration data on the basis of correlation. Note that this way of calibrating a microscope is only very approximate and its use for metrology purposes is very limited. However, it can provide a lot of useful information about our scanner properties if we are unable to perform more complex analysis.

Finally, using *3D calibration → Load from file...* we can load any primary 3D calibration data determined by an external device, like set of interferometers. Data should be a plain text file containing number of calibration sets and sets (x, y, z, x_err, y_err, z_err, x_unc, y_unc, z_unc).

Calibration data application and further use

Data Process → Calibration → 3D calibration

Primary calibration data obtained in previous steps are related to a scanning stage, not to concrete SPM measurements. We can use primary calibration data for different measurements. To use primary calibration data for our measured data processing, we need to apply them first. Using module *3D calibration → Apply to data...* we can review and select calibration data applied on our height field measurements. After applying calibration data a set of calibration datafields is created and attached to selected data. A ‘C’ character appears in data browser as a sign of data with attached calibration. Note that the calibration attached to SPM measurement is no more related with primary calibration data (that were used for its creation).

When there is calibration attached to data, data reading and statistical quantities evaluation modules and tools recognize it automatically. Measurement uncertainty is then added to the measurement results. Uncertainty is calculated using uncertainty propagation law.

Chapter 5

Summaries and Tables

This chapter provides the reference for various formats, syntaxes and command options that you might find helpful. Most are related to advanced use of Gwyddion, except perhaps the table of common keyboard shortcuts and table of supported file formats.

5.1 gwyddion

gwyddion — SPM data visualization and analysis

Synopsis

`gwyddion` [*OPTION*...] [*FILE*...]

Description

Gwyddion is a graphical SPM (Scanning Probe Microscope) data visualization and analysis program, using Gtk+.

Options

The program accepts all standard Gtk+, Gdk, and GtkGLExt options like `--display` or `--sync`. Please see documentation of these packages for description of toolkit options.

The behaviour of the remote-control options `--remote-*` is undefined when more than one instance of Gwyddion is running on the display. They can choose an arbitrary instance to communicate to.

If a directory is given as *FILE* argument the program opens a file chooser in this directory.

Gwyddion options:

`--help` Prints a brief help and terminates.

`--version` Prints version information and terminates.

`--no-splash` Disables splash screen on program startup.

`--remote-new` Opens files given on the command line in an already running instance of Gwyddion on the display. Runs a new instance if none is running.

This is probably the most useful remote control option. File type associations are usually installed to run Gwyddion with this option.

`--remote-existing` Opens files given on the command line in an already running instance of Gwyddion on the display. Fails if none is running.

This is useful if you want to handle the case of Gwyddion not running differently than by starting it.

`--remote-query` Succeeds if an instance of Gwyddion is already running on the display and prints its instance identifier. Fails if none is running.

The instance identifier depends on the remote control backend in use. In some cases it is useful as a global window identifier, in some it is not. With libXmu this option prints the X11 Window, on Win32 HWND is printed, while with LibUnique the startup id is printed.

`--check` Instead of running the user interface and opening *FILE*s, it loads the files, performs a sanity check on them (printing errors to standard error output) and terminates.

--disable-gl Disables OpenGL entirely, including any checks whether it is available. This option, of course, has any effect only if Gwyddion was built with OpenGL support and one of the most visible effects is that 3D view becomes unavailable. However, you may find it useful if you encounter a system so broken that even checking for OpenGL capabilities leads to X server errors. It can also help when you run Gwyddion remotely using X11 forwarding and the start-up time seems excessively long.

--log-to-file Redirects messages from GLib, Gtk+, Gwyddion, etc. to `~/.gwyddion/gwyddion.log` or file given in GWYDDION_LOGFILE environment variable. This option is most useful on Unix as on Win32 messages are redirected to a file by default.

--no-log-to-file Prevents redirection of messages from GLib, Gtk+, Gwyddion, etc. to a file. This is most useful on Win32 (where messages are redirected to a file by default) provided that stdout and stderr go somewhere you can see them.

--debug-objects Prints list of objects created during run time, with creation and desctruction times or reference counts on program exit. Useful only for developers.

--startup-time Prints wall-clock time taken by various startup (and shutdown) tasks. Useful only for developers and people going to complain about too slow startup.

Environment

On Linux/Unix, following environment variables can be used to override compiled-in installation paths (MS Windows version always looks to directories relative to path where it was installed). Note they are intended to override system installation paths therefore they are not path lists, they can contain only a single path.

GWYDDION_DATADIR Base data directory where resources (color gradients, OpenGL materials, ...) were installed. Gwyddion looks into its `gwyddion` subdirectory for resources.

When it is unset, it defaults to compiled-in value of `${datadir}` which is usually `/usr/local/share`.

GWYDDION_LIBDIR Base library directory where modules were installed. Gwyddion looks into its `gwyddion/modules` subdirectory for modules.

When it is unset, it defaults to compiled-in value of `${libdir}` which is usually `/usr/local/lib` or `/usr/local/lib64`.

GWYDDION_LIBEXECDIR Base lib-exec directory where plug-ins were installed. Gwyddion looks into its `gwyddion/plugins` subdirectory for plug-ins.

When it is unset, it defaults to compiled-in value of `${libexecdir}` which is usually `/usr/local/libexec`.

GWYDDION_LOCALEDIR Locale data directory where message catalogs (translations) were installed.

When it is unset, it defaults to compiled-in value of `${datadir}/locale` which is usually `/usr/local/share/locale`.

Other variables that influence Gwyddion run-time behaviour include GLib+ variables and Gtk+ variables and some Gwyddion-specific variables:

GWYDDION_LOGFILE Name of file to redirect log messages to. On MS Windows, messages are always sent to a file as working with the terminal is cumbersome there. The default log file location, `gwyddion.log` in user's Documents and Settings, can be overridden with `GWYDDION_LOGFILE`. On Unix, messages go to the terminal by default and this environment variable has effect only if `--log-to-file` is given.

Files

~/.gwyddion/settings Saved user settings and tool states. Do not edit while Gwyddion is running, it will overwrite it at exit.

~/.gwyddion/glmaterials, ~/.gwyddion/gradients, ... User directories with various resources (OpenGL materials, color gradients, ...).

$GWYDDION_DATADIR/gwyddion/glmaterials, $GWYDDION_DATADIR/gwyddion/gradients ... The same for system-wide resources.

~/.gwyddion/pixmaps Directory to place user icons to. This is mainly useful for installation of modules to home.

$GWYDDION_DATADIR/gwyddion/pixmaps, The same for system-wide icons.

~/.gwyddion/modules Directory to place user modules to. They should be placed into `file`, `graph`, `process`, `layer`, and `tools` subdirectories according to their kind, though this is more a convention than anything else.

$GWYDDION_LIBDIR/gwyddion/modules, The same for system-wide modules.

~/.gwyddion/plugins Directory to place user plug-ins to. They should be placed into `file` and `process` subdirectories according to their kind.

$GWYDDION_LIBEXECDIR/gwyddion/plugins, The same for system-wide plug-ins.

~/.gwyddion/pygwy Directory to place user python modules or scripts to.

See also

gwyddion-thumbnailer(1), gxsm(1)

5.2 gwyddion-thumbnailer

gwyddion-thumbnailer — Creates thumbnails of SPM data files

Synopsis

`gwyddion-thumbnailer` --version | --help

`gwyddion-thumbnailer` [*OPTION*...] *MODE* [*ARGUMENT*...]

Description

Gwyddion-thumbnailer creates thumbnails of SPM (Scanning Probe Microscope) image files. Depending on the mode of operation, described below, the thumbnails are written to conform to various desktop standards so that they can be displayed in nautilus(1), thunar(1) and similar file managers.

Gwyddion-thumbnailer loads and renders files using gwyddion(1), libraries and modules, therefore, it can create thumbnails of all file formats supported by your Gwyddion installation. This also means it inherits Gwyddion settings, e.g. the default false color gradient, and that it is influenced by the same environment variables as Gwyddion.

Informative Options

--help Prints a brief help and terminates.

--version Prints version information and terminates.

Thumbnailing Options

--update Writes the thumbnail only if it does not exist yet or does not seem to be up-to-date. By default, gwyddion-thumbnailer overwrites existing thumbnails with fresh ones even if they seem up to date.

Mode

Three thumbnailing modes are available: `gnome2`, `tms` and `kde4`; and one special mode: `check`. They are described below.

Gnome 2

`gwyddion-thumbnailer` [*OPTION*...] gnome2 *MAX-SIZE INPUT-FILE OUTPUT-FILE*

In `gnome2` mode, gwyddion-thumbnailer creates PNG thumbnails according to the Gnome thumbnailer specification. Using the convention from this specification, it should be run

```
gwyddion-thumbnailer gnome2 %s %i %o
```

Gwyddion installs the corresponding GConf schemas and enables thumbnailers for all file types it supports by default, so usually this should Just Work and should not need to be set up manually.

The thumbnails created in `gnome2` more are identical as in `tms` mode, including all the PNG auxiliary chunks (provided that the same *MAX-SIZE* as in `tms` mode is specified, of course).

TMS

`gwyddion-thumbnailer` [*OPTION*...] tms *MAX-SIZE INPUT-FILE*

In `tms` mode, gwyddion-thumbnailer creates PNG thumbnails according to the Thumbnail Managing Standard. Argument *MAX-SIZE* must be `128` or `normal` (both meaning 128 pixels) or `256` or `large` (both meaning 256 pixels).

Output file name is not given as it is prescribed by the TMS. The thumbnail is placed to the directory for normal or large thumbnails according to given *MAX-SIZE*.

This mode can also be useful for manual batch-creation of thumbnails. For instance, to create them for all `*.afm` files in directory `scans` and its subdirectories, you can run

```
find scans -type f -name '*.afm' -print0 \\
    | xargs -0 -n 1 gwyddion-thumbnailer --update tms normal
```

And then go make yourself a coffee because this will take some time.

KDE 4

`gwyddion-thumbnailer` kde4 *MAX-SIZE INPUT-FILE*

In `kde4` mode, gwyddion-thumbnailer creates PNG thumbnails that are intended to be consumed by gwythumbcreator KDE module. The thumbnail, again identical as in the other modes, is written to the standard output.

Do *not* use this mode from the command line. It is documented for completness, however, the protocol between gwythumbcreator and gwyddion-thumbnailer must be considered private and it can change at any time.

Check

`gwyddion-thumbnailer` check *INPUT-FILE*

The `check` mode does not serve for thumbnail creation. Instead, gwyddion-thumbnailer prints information about available thumbnails of *INPUT-FILE* and cached failures to produce a thumbnail by individual applications, as described by the TMS.

If the normal-sized thumbnail exists and is up to date, the large version does not exist and there is one cached failure from gnome-thumbnail-factory, the output can be for instance:

```
File:   INPUT-FILE
URI:    file:///home/me/Pictures/naughty/broken-tip3/INPUT-FILE
Normal: /home/me/.thumbnails/normal/MD5.png
        status: OK
Large:  /home/me/.thumbnails/large/MD5.png
        status: Thumbnail does not exist or stat() fails on it.
Failed: /home/me/.thumbnails/fail/gnome-thumbnail-factory/MD5.png
```

URI is the canonical URI of the input file, *MD5* stands for the hex representation of MD5 sum of the URI, as described by the TMS. If there are no cached failures, no Failed lines are printed.

This function can be used to check thumbnails of any kind, not necessarily created by gwyddion or gwyddion-thumbnailer. In future, it might be reported as an error if the thumbnail does not contain Gwyddion-specific information though.

See also

gwyddion(1),

5.3 Keyboard Shortcuts

Shortcut	Menu equivalent	Context	Action
F1		almost all windows	Open a relevant section of this user guide in a web browser.
Ctrl-Q	*File* → *Quit*	toolbox, data window, 3D window, graph window, tool window	Quit Gwyddion.
Ctrl-O	*File* → *Open*	toolbox, data window, 3D window, graph window, tool window	Open a data file.
Ctrl-S	*File* → *Save*	toolbox, data window, 3D window, graph window, tool window	Save current data (you will be prompted for a file name if none is associated yet).
Ctrl-Shift-S	*File* → *Save As*	toolbox, data window, 3D window, graph window, tool window	Save current data under a different name. The file name associated with the data changes to the new name.
Ctrl-Shift-M	*File* → *Merge*	toolbox, data window, 3D window, graph window, tool window	Merge data from a file to the current file.
Ctrl-H	*File* → *Open Recent* → *Document History*	toolbox, data window, 3D window, graph window, tool window	Open the document history browser (or bring it forward if it is already displayed).
Ctrl-Z	*Edit* → *Undo*	toolbox, data window, 3D window, graph window, tool window	Undo the last processing step applied on current data.
Ctrl-Y	*Edit* → *Redo*	toolbox, data window, 3D window, graph window, tool window	Redo the last processing step applied on current data.
Ctrl-K	*Edit* → *Remove Mask*	toolbox, data window, 3D window, graph window, tool windows	Remove mask from current data window.
Ctrl-Shift-K	*Edit* → *Remove Presentation*	toolbox, data window, 3D window, graph window, tool windows	Remove presentation from current data window.
Ctrl-Shift-B	*Metadata Browser* (in the context menu)	image or volume data window	Show metadata browser for the data.
+		image or volume data window	Zoom current data window in.
=		image or volume data window	Zoom current data window in.
-		image or volume data window	Zoom current data window out.
Z		image or volume data window	Zoom current data window 1:1.

Shortcut	Menu equivalent	Context	Action
Ctrl-F	*Data Process* → *Repeat Last*	toolbox, data window, 3D window, graph window, tool windows	Repeat last data processing function with the last used parameters, on current data. Normally the operation is repeated silently, but if the processing step cannot be carried out without a human interaction, a dialog is shown.
Ctrl-Shift-F	*Data Process* → *Re-Show Last*	toolbox, data window, 3D window, graph window, tool windows	Re-show parameter dialog of the last data processing function. If the operation has no parameters to set, it is performed immediately.

You can assign your own keyboard shortcuts to all functions in the menus and it is also possible to invoke tools with keyboard shortcuts.

To change the keyboard shortcut of a menu item simply select the item using the mouse or arrow keys, press the key combination you want to assing to it and it will be immediately assigned. The shortcut must be either a special key, e.g. **F3**, or a key combination including modifiers, e.g. **Ctrl-Shift-D**. It is not possible to assign bare keys such as **Q**.

To prevent inadvertent modification of shortcuts, they can be changed only if *Edit* → *Keyboard Shortcuts* is enabled. Modifications are disabled by default which is also the recommended setting during normal use.

All keyboard shortucts are stored in file `ui/accel_map` in the user's directory, which usually means `~/.gwyddion` (Unix) or `Documents and Settings\gwyddion` (MS Windows). Assigning shortcuts to tools can be only done by editing this file. Each line corresponds to an action that can be invoked with a shortcut. For instance the Mask Editor tool's line is by default:

```
; (gtk_accel_path "<tool>/GwyToolMaskEditor" "")
```

Semicolons represents comments, i.e. lines starting with a semicolon are inactive. Hence, to assign the combo **Ctrl-Shift-E** to the Mask Editor tool, remove the semicolon to make the line active and fill the desired shortcut in the empty quotes:

```
(gtk_accel_path "<tool>/GwyToolMaskEditor" "<Control><Shift>e")
```

5.4 Supported File Formats

File Format	Extensions	Module	Read	Write	SPS	Volume
Accurex II text data	.txt	accurexii-txt	Yes	—	—	—
AIST-NT	.aist	aistfile	Yes	—	—	—
Alicona Imaging AL3D data	.al3d	alicona	Yes	—	—	—
Ambios AMB	.amb	ambfile	Yes	—	—	—
Anfatec	.par, .int	anfatec	Yes	—	—	—
A.P.E. Research DAX	.dax	apedaxfile	Yes	—	—	—
A.P.E. Research APDT	.apdt	apedaxfile	Yes	—	—	—
A.P.E. Research DAT	.dat	apefile	Yes	—	—	—
Text matrix of data values	.txt	asciiexport	—	Yes	—	—
Assing AFM	.afm	assing-afm	Yes	Yes	—	—
Attocube Systems ASC	.asc	attocube	Yes	—	—	—
Image Metrology BCR, BCRF	.bcr, .bcrf	bcrfile	Yes	—	—	—
Burleigh BII	.bii	burleigh_bii	Yes	—	—	—
Burleigh IMG v2.1	.img	burleigh	Yes	—	—	—

File Format	Extensions	Module	Read	Write	SPS	Volume
Burleigh exported data	.txt, .bin	burleigh_exp	Yes	—	—	—
Code V interferogram data	.int	codevfile	Yes	—	—	—
Createc DAT	.dat	createc	Yes	—	—	—
Benyuan CSM	.csm	csmfile	Yes	—	—	—
Digital Micrograph DM3 TEM data	.dm3	dm3file	Yes	—	—	—
DME Rasterscope	.img	dmefile	Yes	—	—	—
Gwyddion dumb dump data	.dump	dumbfile	Yes	—	—	—
ECS	.img	ecsfile	Yes	—	—	—
Nanosurf EZD, NID	.ezd, .nid	ezdfile	Yes	—	—	—
FemtoScan SPM	,spm.	femtoscan	Yes	—	—	—
FemtoScan text data	.txt	femtoscan-txt	Yes[a]	—	—	—
Flexible Image Transport System (FITS)	.fits,, .fit	fitsfile	Yes	—	—	—
DME GDEF	.gdf	gdeffile	Yes	—	—	—
Gwyddion Simple Field	.gsf	gsffile	Yes	Yes	—	—
Gwyddion native data	.gwy	gwyfile	Yes	Yes	Yes	Yes
Gwyddion XYZ data	.gxyzf	gxyzffile	Yes[b]	Yes	—	—
Psi HDF4	.hdf	hdf4file	Yes	—	—	—
Hitachi AFM	.afm	hitachi-afm	Yes	—	—	—
Hitachi S-3700 and S-4800 SEM data	.txt, +, image	hitachi-sem	Yes	—	—	—
WaveMetrics IGOR binary wave v5	.ibw	igorfile	Yes	—	—	—
Intematix SDF	.sdf	intematix	Yes	—	—	—
ISO 28600:2011 SPM data transfer format	.spm	iso28600	Yes	Yes	Limited[c]	—
JEOL	.tif	jeol	Yes	—	—	—
JPK Instruments	.jpk	jpkscan	Yes	—	—	—
JEOL JSPM	.tif	jspmfile	Yes	—	—	—
Olympus LEXT 4000	.lext	lextfile	Yes	—	—	—
FEI Magellan SEM images	.tif	magellan	Yes	—	—	—
MapVue	.map	mapvue	Yes	—	—	—
Zygo MetroPro DAT	.dat	metropro	Yes	—	—	—
MicroProf TXT	.txt	microprof	Yes	—	—	—
MicroProf FRT	.frt	microprof	Yes	—	—	—
DME MIF	.mif	miffile	Yes	—	—	—
Molecular Imaging MI	.mi	mifile	Yes	—	Limited[c]	—
Aarhus MUL	.mul	mulfile	Yes	—	—	—
Nanoeducator	.mspm, .stm, .spm	nanoeducator	Yes	—	Yes	—
Nanomagnetics NMI	.nmi	nanomagnetics	Yes	—	—	—
Nanonics NAN	.nan	nanonics	Yes	—	—	—
Nanonis SXM	.sxm	nanonis	Yes	—	—	—
NanoObserver	.nao	nanoobserver	Yes	—	—	—
Nanoscan XML	.xml	nanoscan	Yes	—	—	—
NanoScanTech	.nstdat	nanoscantech	Yes	—	Yes	Yes
Veeco Nanoscope III	.001, .002, etc.	nanoscope	Yes	—	Limited[c]	Yes

File Format	Extensions	Module	Read	Write	SPS	Volume
Veeco Nanoscope II	.001, .002, etc.	nanoscope-ii	Yes	—	—	—
Nanotop SPM	.spm	nanotop	Yes	—	—	—
GSXM NetCDF	.nc	netcdf	Yes	—	—	—
Nearly raw raster data (NRRD)	.nrrd	nrrdfile	Yes[d]	Yes[e]	—	—
NT-MDT	.mdt	nt-mdt	Yes	—	Yes	Yes
EM4SYS NX II	.bmp	nxiifile	Yes	—	—	—
NT-MDT old MDA	.sxml, .dat	oldmda	—	—	—	Yes
Olympus LEXT 3000	.ols	ols	Yes	—	—	—
Open Microscopy OME TIFF	.ome.tiff,, .ome.tif	ometiff	Yes	—	—	—
Omicron SCALA	.par, .tf*, .tb*, .sf*, .sb*	omicron	Yes	—	Yes	—
Omicron flat format	.*_flat	omicronflat	Yes	—	—	—
Omicron MATRIX	.mtrx	omicronmatrix	Yes	—	—	—
Wyko OPD	.opd	opdfile	Yes	—	—	—
Wyko ASCII	.asc	opdfile	Yes	—	—	—
OpenGPS X3P (ISO 5436-2)	.x3p	opengps	Yes	—	—	—
Pixmap images	.png, .jpeg, .tiff, .tga, .pnm, .bmp	pixmap	Yes[f]	Yes[g]	—	—
Nanosurf PLT	.plt	pltfile	Yes	—	—	—
Pacific Nanotechnology PNI	.pni	pnifile	Yes	—	—	—
Park Systems	.tiff,, .tif	psia	Yes	—	—	—
SymPhoTime TTTR v2.0 data	.pt3	pt3file	Yes	—	—	—
Quesant AFM	.afm	quesant	Yes	—	—	—
Raw text files	any	rawfile	Yes	—	—	—
Raw binary files	any	rawfile	Yes	—	—	—
Graph text data (raw)	any	rawgraph	Yes[h]	—	—	—
XYZ data	.xyz, .dat	rawxyz	Yes[b]	Yes	—	—
Renishaw WiRE Data File	.wdf	renishaw	Yes	—	Yes	Yes
RHK Instruments SM3	.sm3	rhk-sm3	Yes	—	Limited[c]	—
RHK Instruments SM4	.sm4	rhk-sm4	Yes	—	Limited[c]	—
RHK Instruments SM2	.sm2	rhk-spm32	Yes	—	Limited[c]	—
Automation and Robotics Dual Lens Mapper	.mcr, .mct, .mce	robotics	Yes	—	—	—
S94 STM files	.s94	s94file	Yes	—	—	—
Surfstand Surface Data File	.sdf	sdfile	Yes	Yes	—	—
Micromap SDFA	.sdfa	sdfile	Yes	—	—	—
Seiko SII	.xqb, .xqd, .xqt, .xqp, .xqj, .xqi	seiko	Yes	—	—	—
Sensofar PLu v2000	.plu	sensofar	Yes	—	—	—
Sensolytics DAT	.dat	sensolytics	Yes	—	—	—
Shimadzu	.sph, .spp, .001, .002, etc.	shimadzu	Yes	—	—	—
Shimadzu ASCII	.txt	shimadzu	Yes	—	—	—
IonScope SICM	.img	sicmfile	Yes	—	—	—
Surface Imaging Systems	.sis	sis	Yes	—	—	—
SPIP ASCII	.asc	spip-asc	Yes	—	—	—
Thermicroscopes SPMLab R4-R7	.tfr, .ffr, etc.	spmlab	Yes	—	—	—
Thermicroscopes SPMLab floating point	.flt	spmlabf	Yes	—	—	—

File Format	Extensions	Module	Read	Write	SPS	Volume
SPML (Scanning Probe Microscopy Markup Language)	.xml	spml	Yes	—	—	—
ATC SPMxFormat data	.spm	spmxfile	Yes	—	—	—
Omicron STMPRG	tp*, ta*	stmprg	Yes	—	—	—
Molecular Imaging STP	.stp	stpfile	Yes	—	—	—
Surf	.sur	surffile	Yes	—	—	—
Tescan MIRA SEM images	.tif	tescan	Yes	—	—	—
FEI Tecnai imaging and analysis (former Emispec) data	.ser	tiaser	Yes	—	Yes	Yes
Unisoku	.hdr, .dat	unisoku	Yes	—	—	—
VTK structured grid file	.vtk	vtkfile	—	Yes	—	—
WinSTM data	.stm	win_stm	Yes	—	—	—
WITec Project data	.wip	wipfile	Yes	—	Yes	Yes
WITec ASCII export	.dat	witec-asc	Yes	—	—	—
WITec	.wit	witfile	Yes	—	—	—
WSF ASCII data	.wsf	wsffile	Yes	—	—	—
Nanotec WSxM	.tom,, .stp	wsxmfile	Yes	—	—	—
Carl Zeiss SEM scans	.tif	zeiss	Yes	—	—	—
Zemax grid sag data	.dat	zemax	Yes	—	—	—
OpenEXR images	.exr	hdrimage.cc	Yes	Yes	—	—

[a] Regular sampling in both X and Y direction is assumed.

[b] XYZ data are interpolated to a regular grid upon import.

[c] Spectra curves are imported as graphs, positional information is lost.

[d] Not all variants are implemented.

[e] Data are exported in a fixed attached native-endian float point format.

[f] Import support relies on Gdk-Pixbuf and hence may vary among systems.

[g] Usually lossy, intended for presentational purposes. 16bit grayscale export is possible to PNG, TIFF and PNM.

[h] At present, only simple two-column data, imported as graph curves, are supported.

5.5 High-Depth Image Formats

Gwyddion can export data to 16bit greyscale PNG, PNM and TIFF images and to OpenEXR images with half, float and 32bit data types. In case of 16bit images the full data range is always stretched to the full greyscale range; OpenEXR export permits to specify the value scaling factor.

When data are exported to a high-depth image additional information is stored to the file to enable automated loading back to Gwyddion without having to specify the dimensions and scales manually. By storing this additional information to image files you create in other programs, you can also make them directly loadable to Gwyddion with correct dimensions and scales. The information is organised as key-value pairs, stored using individual format-specific means for each format, described in the following table.

Format	Method
PNG	`tEXt` chunks
OpenEXR	named attributes
PNM	header comments of the form `#key:value`

Most keys are identical to those used in Gwyddion Simple Fields, except for the added `Gwy::` prefix, so see also GSF description for more details. Floating point values are stored directly if the format permits it (OpenEXR), otherwise a text representation of the number is used (in the C format). The keys are listed below.

Key	Type	Meaning
`Gwy::XReal`	floating point	Horizontal size in physical units (given by `XYUnits`), a positive floating point number.
`Gwy::YReal`	floating point	Vertical size in physical units (given by `XYUnits`), a positive floating point number.
`Gwy::XOffset`	floating point	Horizontal offset in physical units (given by `XYUnits`).
`Gwy::YOffset`	floating point	Vertical offset in physical units (given by `XYUnits`).
`Gwy::ZScale`	floating point	Value scaling factor. Image data are to be multiplied by this factor to obtain physical values. This parameter is usually used with limited-range floating point formats such as half. For integer data, `Gwy::ZMin` and `Gwy::ZMax` is usually used.
`Gwy::ZMin`	floating point	Value in physical units corresponding to the minimum value representable in the image (normally 0).
`Gwy::ZMax`	floating point	Value in physical units corresponding to the maximum value representable in the image.
`Gwy::XYUnits`	string	Lateral units, i.e. units of physical sizes and offsets.
`Gwy::ZUnits`	string	Value units, i.e. units of data values.
`Gwy::Title`	string	Data/channel title.

In case of PNG, the scaling information is also stored in the standard `sCAL` and `pCAL` chunks (with linear scaling formula). Conversely, if these chunks are present (and the Gwyddion-specific are absent) the information from them is used in import. See the PNG specifiation for the chunk description.

5.6 Expressions

Expressions used in Data Arithmetic module, grain quantity formulas and in graph function fitting have syntax similar to common programming languages.

All numbers are real (floating point), number literals use standard notation. Examples of valid numbers: `1`, `.707`, `2.661`, `8.2e-34`.

Function, constant, and variable names start with a letter and continue with zero or more letters, numbers, or underscores. Examples of valid identifiers: `pow10` (a function), `Pi` (a constant), `d2_2` (a variable).

The precedence of operations is summarized in following table.

Operation	Associativity	Examples
parentheses	N.A.	`(x)`
function call and unary operators	right to left	`-sqrt 3`
power operator	right to left	`2^16`
multiplication, division, and modulo operators	left to right	`9/2 * 8`
addition and subtraction operators	left to right	`3 -4 + 5`

Note `-3^2` is 9, that is `(-3)^2`, like in **bc**, but unlike in Perl or Python.

Available operators and functions are listed in following table.

Operator	Meaning
+ (unary)	no op
− (unary)	negative value
~	negative value (equivalent to −)
+ (binary)	addition
− (binary)	subtraction
*	multiplication
/	division
%	floating point modulo
^	power
abs	absolute value
floor	rounding down to nearest integer
ceil	rounding up to nearest integer
sqrt	square root
cbrt	cubic root
sin	sine function
cos	cosine function
tan	tangent function
asin	arc sine function
acos	arc cosine function
atan	arc tangent function
exp	base-e exponential function
ln	base-e logarithm function
log	base-e logarithm function
pow10	base-10 exponential function
log10	base-10 logarithm function
sinh	hyperbolic sine function
cosh	hyperbolic cosine function
tanh	hyperbolic tangent function
asinh	inverse hyperbolic sine function
acosh	inverse hyperbolic cosine function
atanh	inverse hyperbolic tangent function
pow	power function, `pow(x,y)` equals to `x^y`
min	minimum of two values
max	maximum of two values
mod	floating point modulo, `mod(x,y)` equals to `x % y`
hypot	Euclidean distance function, `hypot(x,y)` equals to `sqrt(x^2+y^2)`
atan2	arc tangent function of two variables

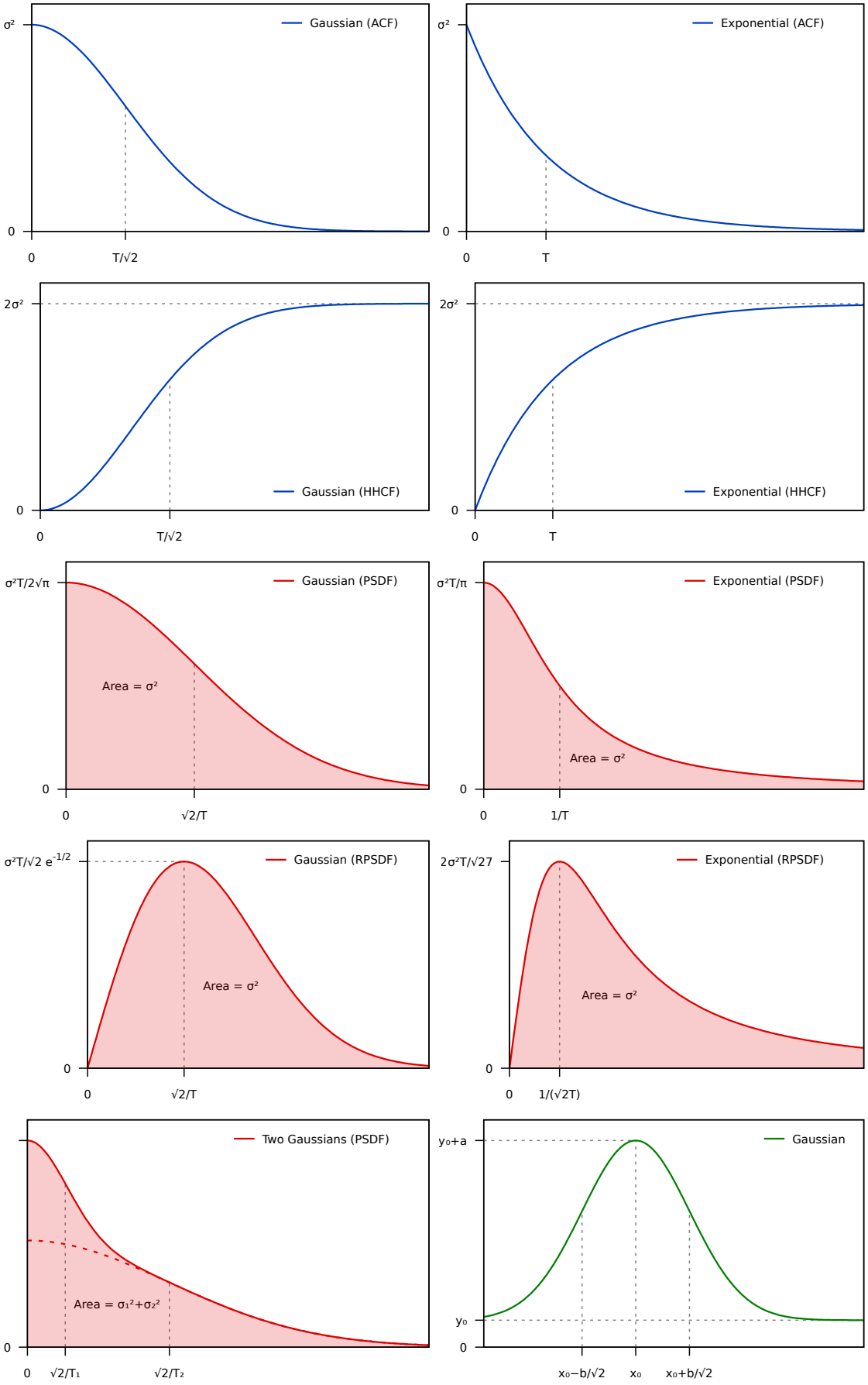
Beside that, there are a few peculiarities that may make typing simple expression easier:

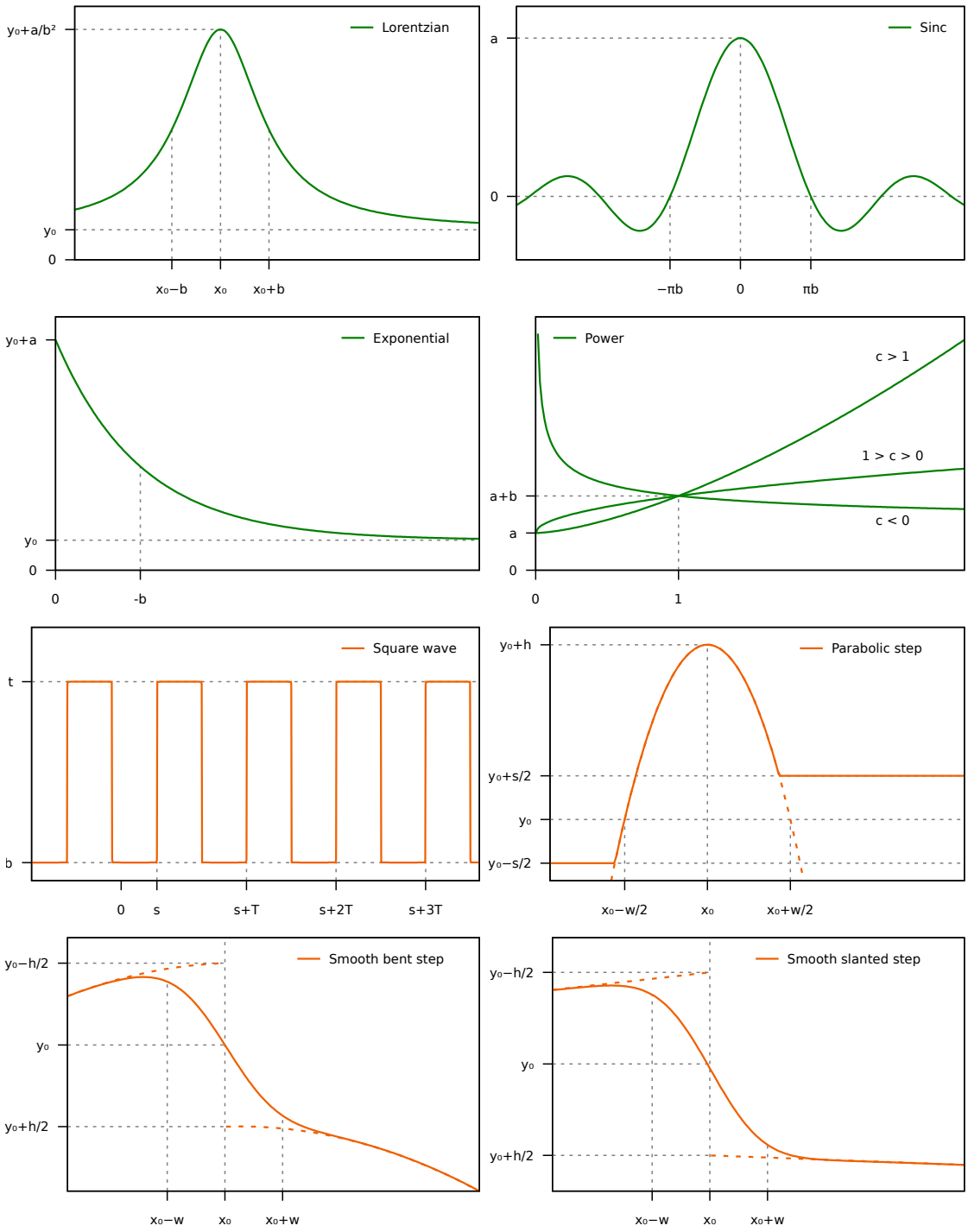
- Multiplication signs are optional, you can use spaces instead (or nothing, in some cases). E.g., `3/4 Pi` and `5(4+3)(2+1)` are valid expressions. However, `3a` is not a valid expression, `3e-4` always means `0.0003`, not `3*e -4`.
- There is no difference between function calls and unary operators, so parentheses can be often omitted. E.g, `sqrt 5` and `hypot hypot 3,4,5` are valid expression. The latter can be parenthesized as follows: `hypot(hypot(3,4),5)`.

  Note however, function calls have higher priority than any other operator, thus `sin Pi/2` is the same as `(sin Pi)/2`, not as `sin(Pi/2)`.

If in doubt, write out expressions in full form.

5.7 Fitting Functions





5.8 Resources

Various bits of data, e.g. false color maps or raw file import presets, are stored in standalone files that are collectively called resource files. Gwyddion looks for resources in two different locations: system and user-specific.

System resources are installed along with the program and they are not modifiable. Typically, they are located under a directory such as `/usr/share/gwyddion` (Unix), `Program Files\Gwyddion` (MS Windows) or other directory determined by GWYDDION_DATADIR.

User resources are located in a user's directory, this usually means under `~/.gwyddion` (Unix) or `Documents and Settings\gwyddion` (MS Windows).

All resource files are simple text files that can be easily examined and modified by text editors or sent to other users (if they are copied or created manually Gwyddion needs to be restarted to notice them). In most cases only characters of the ASCII can

appear in the files. If international text can appear there it must be in the UTF-8 encoding. Numerical values are represented in the standard POSIX format, i.e. with decimal point, independently on what decimal separator is usual in the user's language.

Resources are organized in subdirectories according to their kind, e.g. color gradients reside in the subdirectory `gradients`. The name of the file determines the resource name – gradient Gray is found in file `gradients/Gray`. Modules can define their own resource types; the types described here are the most important types but the list may not be comprehensive.

Every resource file has the same structure. It starts with a line identifying the resource type:

```
Gwyddion resource GwyGradient
```

where GwyGradient is the type name in the type system (which is quite a low-level detail but so it is), followed by named parameters in the form

```
name value
```

and resource data. Some resource types may contain only named parameters, other may contain only data.

Gradients

Gradients, i.e. false color maps, reside in directory `gradients`, they are identified by GwyGradient and contain only data. They can be edited in the application using the gradient editor.

The gradient data consists of rows corresponding to individual points in the gradient:

```
position red green blue alpha
```

The position determines where the color defined by *red*, *green*, *blue* and *alpha* components is placed in the interval $[0, 1]$ where 0 corresponds to the gradient start, 1 corresponds to the end. The color is interpolated linearly between the specified points.

The positions must form an increasing sequence from 0 to 1 (i.e. the minimum number of color points is two). The range of the color components is also $[0, 1]$. Note the alpha value, corresponding to opacity, is unused and must be given as 1 (fully opaque).

For instance, the standard gradient Red going from black (0 0 0) to red (1 0 0) to white (1 1 1) is defined as follows:

```
Gwyddion resource GwyGradient
0.0 0 0 0 1
0.5 1 0 0 1
1.0 1 1 1 1
```

OpenGL Materials

OpenGL materials reside in directory `glmaterials`, they are identified by GwyGLMaterial and contain only data. They can be edited in the application using the OpenGL material editor.

The material data consists of four RGBA lines, similar to gradients that correspond in to the four OpenGL material components in the following order:

1. ambient,
2. diffuse,
3. specular,
4. emission.

See section OpenGL Material Editor for explanation of the components. They are followed by a line containing the shininess, again as a number from the interval $[0, 1]$.

Note the emission component, while read and written by Gwyddion, is presently unused by the 3D view. It is recommended to set it to 0 0 0 1, i.e. black.

For instance, the standard material Red-Rubber with very dark red ambient color, grayish diffuse reflection, red specular reflection and low shininess is defined as follows:

```
Gwyddion resource GwyGLMaterial
0.05 0.0  0.0  1.0
0.5  0.4  0.4  1.0
0.7  0.04 0.04 1.0
0.0  0.0  0.0  1.0
.078125
```

Grain Values

Grain values reside in directory `grainvalues`, they are identified by GwyGrainValue and contain only named parameters. They can be used to define additional grain quantities, derived from the built-in quantities, that appear under *User* group in grain analysis functions. At the time of writing this, there is no editor in the application, new quantities must be created manually.

The named parameters are summarized in the following table:

Parameter	Required	Type	Description
`symbol`	required	identifier	Identifier to use in other expressions (but see below). It must be a valid identifier of ASCII letters, numbers and underscores, starting with a letter.
`expression`	required	free-form	Formula for calculation of this quantity from other grain quantities. The general expression syntax is described in section Expressions.
`symbol_markup`	optional	free-form	Fancy symbol that can include Greek letters or subscripts and superscripts expressed with the Pango markup language. It is used for presentation in the application so, while it is optional, it is recommended to at least define it identically to `symbol`.
`power_xy`	optional	integer	The power in which the lateral dimensions appear in the quantity. For instance, this is 1 for grain dimensions, 2 for areas and volumes. The default value is 0.
`power_z`	optional	integer	The power in which the "height" dimension appears in the quantity. For instance, this is 1 for values and volumes, 0 for dimensions and areas. The default value is 0.
`same_units`	optional	0 or 1	Given as 1 if the quantity makes sense only for lateral and "height" dimensions being the same physical quantities. For instance, this is required for the surface area. The default is 0.
`is_angle`	optional	0 or 1	Given as 1 if the quantity is an angle. The expression should calculate angles in radians. However, if `is_angle` is set Gwyddion knowns the value can be converted to degrees for presentation. The default is 0.

At present, user-defined grain quantities cannot depend on other user-defined grain quantities to avoid circular dependencies. The built-in grain quantities are listed below:

Symbol	Group	Name
x_c	Position	Center x position
y_c	Position	Center y position
z_min	Value	Minimum value
z_max	Value	Maximum value
z_m	Value	Mean value
z_med	Value	Median value
b_min	Value	Minimum value on boundary
b_max	Value	Maximum value on boundary
A_0	Area	Projected area
A_s	Area	Surface area
a_eq	Area	Equivalent square side
r_eq	Area	Equivalent disc radius
A_h	Area	Area above half-height
A_c	Area	Area of convex hull
V_0	Volume	Zero basis volume
V_min	Volume	Grain minimum basis volume
V_L	Volume	Laplacian background basis volume
L_b0	Boundary	Projected boundary length
D_min	Boundary	Minimum bounding size
phi_min	Boundary	Minimum bounding direction
D_max	Boundary	Maximum bounding size
phi_max	Boundary	Maximum bounding direction
R_i	Boundary	Maximum inscribed disc radius
x_i	Boundary	Maximum inscribed disc center x position
y_i	Boundary	Maximum inscribed disc center y position
R_e	Boundary	Minimum circumcircle radius
x_e	Boundary	Minimum circumcircle center x position
y_e	Boundary	Minimum circumcircle center y position
R_m	Boundary	Mean radius
theta	Slope	Inclination ϑ
phi	Slope	Inclination φ
x_0	Curvature	Curvature center x position
y_0	Curvature	Curvature center y position
z_0	Curvature	Curvature center z value
kappa_1	Curvature	Curvature 1
kappa_2	Curvature	Curvature 2
phi_1	Curvature	Curvature angle 1
phi_2	Curvature	Curvature angle 2
a_e1	Moment	Major semiaxis of equivalent ellipse
a_e2	Moment	Minor semiaxis of equivalent ellipse
phi_e1	Moment	Orientation of equivalent ellipse

For instance, a new grain value Height, measuring the grain height as the difference between the maximum and minimum value, can be defined as follows:

```
Gwyddion resource GwyGrainValue
symbol dz
symbol_markup Δz
```

```
power_xy 0
power_z 1
expression z_max - z_min
```

Raw File Presets

Raw file presents reside in directory `rawfile`, they are identified by GwyRawFilePreset and contain only named parameters. They are normally created and edited by the preset editor in the raw file import module.

The named parameters in the resource files correspond closely to the parameters in the user interface explained in detail in section Raw Data File Import. Hence, they will be described only briefly here.

Parameter	Type	Description
`xres`, `yres`	integer	horizontal and vertical size
`xreal`, `yreal`	number	physical dimensions, in units given by `xyexponent` and `xyunit`
`xyexponent`	multiple of 3	power of 10 to multiply `xreal` and `yreal` with
`xyunit`	string	base units of `xreal` and `yreal`, e.g. `"m"`
`zscale`	number	unit step in values
`zexponent`	multiple of 3	power of 10 to multiply `zscale` with
`zunit`	string	base units of `zscale`
`format`	0 or 1	0 means binary, 1 means text
`builtin` (binary)	integer	built-in data format id, see below
`offset` (binary)	integer	data offset in file, in bytes
`size` (binary)	integer	data value size, in bits
`skip` (binary)	integer	number of bits to skip after each value
`rowskip` (binary)	integer	number of additional bits to skip after each row
`sign` (binary)	0 or 1	0 means unsigned, 1 means signed
`revsample` (binary)	0 or 1	1 means reverse bits in values
`revbyte` (binary)	0 or 1	1 means reverse bits in bytes
`byteswap` (binary)	integer	byte swap pattern
`lineoffset` (text)	integer	lines to skip before starting to read the data
`skipfields` (text)	integer	fields to skip at the start of each line
`delimiter` (text)	string	field delimiter, empty string means arbitrary whitespace
`decomma` (text)	0 or 1	1 if decimal separator is comma, 0 for dot

Note the choice of a built-in binary format, i.e. nonzero `builtin`, implies the binary format to some extent. This means the options `size`, `revbyte` and `sign` are ignored as they are used only for detailed specification of user formats. The available formats are listed in the following table:

Type	Description
0	user-specified
1	signed 8bit integer
2	unsigned 8bit integer
3	signed 16bit integer
4	unsigned 16bit integer
5	signed 32bit integer
6	unsigned 32bit integer
7	IEEE float
8	IEEE double
9	signed 64bit integer
10	unsigned 64bit integer

5.9 Settings

Gwyddion module functions remember parameters values between invocations and also between individual sessions. The place where all the values are stored is called settings. The settings include a few program-wide parameters as well.

The permanent storage for the settings is the file `settings` in a user's directory, this usually means under `~/.gwyddion` (Unix) or `Documents and Settings\gwyddion` (MS Windows). The file is only read when Gwyddion starts and written when it terminates. You should keep this in mind if you want to do some manual modifications. Unknown entries in the settings are ignored but preserved.

The settings file starts with a magic header line

```
Gwyddion Settings 1.0
```

followed by lines with individual parameters and values (that form, technically, a serialised GwyContainer). Gwyddion writes the entries in the alphabetical order but this is not a requirement and you do not have to keep the order when modifying the file.

Each parameter line has the form

```
"key" type value
```

Typical module settings keys start with `/module/`*`modulename`*, although in a few cases the module name part is not actually the module name, either because several modules share settings or for historical reasons. Program-wide setting keys start with `/app/`. All possible value types are listed in the following table.

Type	Description
boolean	Logical value that can be either `True` or `False`.
char	Single character. Normal characters are represented directly using a single character. Special characters are represented using the hexadecimal notation as `0x`*`XX`*. This parameter type is not actually used much by modules.
int32	Signed 32bit integer. Gwyddion writes them in decimal notation but reads also other notations such as hexadecimal.
int64	Signed 64bit integer. Gwyddion writes them in decimal notation but reads also other notations such as hexadecimal.
double	Floating point number. They can be in the scientific format, e.g. `1.23e-4`. They are represented in the standard C/POSIX locale, i.e. decimal dot is used (not comma or other separators).
string	String of characters in double quotes, generally UTF-8 encoded. Special characters, including contained double quotes, are escaped using the standard backslash notation.

Some potentially useful program-wide options that can be modified by editing the settings file:

Key	Type	Description
/app/restore-tool-position	boolean	If set to `True`, Gwyddion restores not only size of tool dialogs but also their positions (if possible). For well-behaved window managers this is more likely to be annoying than helpful but in MS Windows you might want to try enabling it.
/app/3d/axes/disable	boolean	If set to `True`, axis labels will never be drawn on OpenGL 3D views, even if enabled. This can help with certain troublesome 3D driver/card/GtkGLExt combinations in which Gwyddion is likely to crash when it tries to draw the axes.
/app/help/user-guide-base	string	Base location of the user guide for help. If not set, the default on-line location is used, i.e. something like `"http://gwyddion.net/documentation/user-guide-en"`, depending on the language. If you want to use a local copy of the HTML guide, set this setting to directory name, for instance `"/home/yeti/docs/gwyddion-user-guide-xhtml-en-2014-09-17"`.

5.10 Toolbox Configuration

The lower part of the toolbox containing the buttons for functions and tools can be customised by editing file `ui/toolbox.xml`. Similarly to custom keyboard shortcuts, the file should be placed in the user's directory, which usually means `~/.gwyd`

dion (Unix) or Documents and Settings\gwyddion (MS Windows). A good starting point for customisation is the default ui/toolbox.xml file installed with Gwyddion under share/gwyddion.

The number buttons in a row is controlled by the width attribute of the top level element toolbox. To change it to five just change the begining of the file to

```
<toolbox width='5'>
```

Expandable and collapsable groups of buttons such as *Data Process* or *Tools* are created with tag group. You can create as many or as few groups as you want. Functions of different kinds can placed in one group if you wish. Each group must be identified with a unique id attribute which is, among other things, used to remember which group was collapsed and which expanded. The attribute title determines the title:

```
<group id='proc' title='Data Process'>
```

Individual buttons are created with elements item:

```
<item type='proc' function='edge_step' icon='gwy_edge'/>
```

Each item must have the type attribute, defining the function type. Unless the type is 'empty' it must also have a function attribute defining the specific function. Function names can be located in the module browser (*Info* → *Module Browser*), where they are listed in *Registered functions* for each module; or in the on-line module browser. The available function types are listed in the following table:

Type name	Function kind
empty	Empty placeholder that can be used for separation or row alignment.
builtin	A built-in function, which includes zooming in and out and 3D view activation. There are exacly four: 'display_3d', 'zoom_in', 'zoom_out' and 'zoom_1_1'.
proc	A two-dimensional data (image) processing function. It has prefix proc:: in the module browser.
graph	A graph function. It has prefix graph:: in the module browser.
volume	A volume data function. It has prefix volume:: in the module browser.
tool	A tool. It has prefix tool:: in the module browser.

The button icon is specified using the icon attribute. Some module functions have icons predefined (so you do not have to specify it) but not all have because the number of available functions is huge. A Gwyddion stock icon can be used or possibly a Gtk+ stock icon. Note Gwyddion icon names have words separated with underscores while Gtk+ icon names use dashes.

If you cannot choose from the provided set of icons it is also possible to draw your own icon and put it to ~/.gwyddion/pixmaps (or its equivalent on other systems), using the same naming convention as Gwyddion icons. It may be useful to start from the GIMP XCF source images for the icons since they contain individual pieces that can be mixed and matched. If you draw a nice icon you are of course encouraged to submit it for inclusion in Gwyddion.

Since tools are accessible only from the toolbox, not listing a tool in ui/toolbox.xml renders it unavailable. Therefore, a special empty item

```
<item type='tool'/>
```

can be used to place all tools that have not been explicitly placed yet to the corresponding position (in a pretty much arbitrary order).

5.11 Format of Gwyddion Files

A Gwyddion native data file consists of a tree-like structure of serialized objects. Generally, these objects can be of various kind and contain other embedded objects (hence the tree-like structure). It can be instructive to play with **gwydump**, a simple file structure visualizer available in on the project's web, for a while and examine the contents of various files.

First of all, we will describe physical file structure without regard to possible interpretation of contained data.

Byte Order

All data is stored in little-endian (also known as LSB or Intel) byte order.

File Header

The file header consists of four bytes (magic number) with the values of ASCII characters `GWYP`.

This is the new file format, an older version of file format with magic header `GWYO` also exists. It will not be discussed here.

File Data

The rest of the file consists of a serialized GwyContainer object that contains all the data. It is stored exactly the same way as any other object, that is as described in the next section.

Object Layout

An object consists of three parts (in the following order):

- Type name, stored as a `NUL`-terminated string of ASCII characters. This is the type name in GObject type system.
- Serialized data size, stored as an unsigned 32bit integer. It does not include the size of the type name and the size of self.
- Component list. Components are named parts of object data, each of particular data type: an atomic type, an array of atomic types, or again an object. They are stored in no particular order.

Components

Each component consists of three parts (in the following order):

- Name, stored as a `NUL`-terminated string.
- Type, stored as a single unsigned byte (character). The table of possible component types is presented below.
- Data, stored as whatever is appropriate for a particular type.

Data Types

Available atomic data types are listed in following table:

Type	Character	Note
boolean	`b`	Stored as a byte, zero is false, nonzero (normally 1) is true
character	`c`	
32bit integer	`i`	
64bit integer	`q`	
double	`d`	finite IEEE 754 double precision floating point number, i.e. files must not contain infinities and not-a-numbers
string	`s`	`NUL`-terminated and UTF-8 encoded
object	`o`	Serialized object as described above

Each atomic type except boolean has its array counterpart. The type character of array types is the same as of the corresponding atomic type, except it is uppercase. Arrays are stored as unsigned 32bit array length (the number of items), followed by the item values. Array data types are listed in following table:

	Type	Character	Note
	array of characters	`C`	Not `NUL`-terminated

	Type	Character	Note
	array of 32bit integers	`I`	
	array of 64bit integers	`Q`	
	array of doubles	`D`	
	array of strings	`S`	
	array of objects	`O`	Uppercase Oh, not zero

Top-Level GwyContainer

The names (keys) of data objects in a GwyContainer representing a Gwyddion file strongly resemble UNIX file names, i.e. they have the form of `/`-separated paths and form a sort of tree-like structure. For instance the title of the first channel, numbered 0, is stored under the key `/0/data/title`. Note some data or information is found under keys that do not seem logical; the reason is usually historical.

The following sections describe the organisation of interesting data and information in the GwyContainer. The list is not necessarily complete. However, since all data items in the file specify consistently their name, type and size in bytes it is always possible to skip unknown data types or data you are not interested in and extract only the desired data items.

Channels

The following table summarises the common keys of channel-related data in the top-level container for channel number 0. For other channels, the number 0 has to be replaced with the corresponding channel number. Note that channels are often numbered sequentially, starting from 0, however, they can have any numbers and the set of channels numbers does not have to be contiguous.

Key	Type	Meaning
`/0/data`	GwyDataField	Channel data.
`/0/data/title`	string	Channel title, as shown in the data browser.
`/0/data/visible`	boolean	Whether the channel should be displayed in a window when the file is loaded.
`/0/data/realsquare`	boolean	Whether the channel should be displayed as *Physically square* (as opposed to *Pixelwise square*).
`/0/base/palette`	string	Name of the false color gradient used to display the channel.
`/0/base/range-type`	32bit integer	False color mapping type (as set by the Color range tool), the value is from GwyLayerBasicRangeType enum.
`/0/base/min`	double	Minimum value for user-set display range.
`/0/base/max`	double	Maximum value for user-set display range.
`/0/mask`	GwyDataField	Mask data. The pixel dimensions of this data field must match those of the channel data.
`/0/mask/red`	double	Red component of the mask color.
`/0/mask/green`	double	Green component of the mask color.
`/0/mask/blue`	double	Blue component of the mask color.
`/0/mask/alpha`	double	Alpha (opacity) component of the mask color.
`/0/show`	GwyDataField	Presentation data. The pixel dimensions of this data field must match those of the channel data.
`/0/meta`	GwyContainer	Channel metadata. The keys are directly the names as displayed in the metadata browser and the string values are the values.
`/0/data/log`	GwyStringList	Channel log as a list of string log entries. They have the format *type*`::`*function*`(`*param*`=`*value*`, ...)@`*time*.
`/0/select/`*foo*	a GwySelection subclass	Selection data. Each kind of selection has (usually) a different object type and is stored under a different name; the specific name *foo* is the same as shown in the selection manager.

Channels are represented as GwyDataField objects. The components of a GwyDataField are summarised in the following table:

Component	Type	Meaning
`xres`	32bit integer	Horizontal size in pixels.
`yres`	32bit integer	Vertical size in pixels.
`xreal`	double	Horizontal dimension in physical units.
`yreal`	double	Vertical dimension in physical units.
`xoff`	double	Horizontal offset of the top-left corner in physical units. It usually occurs only if non-zero.
`yoff`	double	Vertical offset of the top-left corner in physical units. It usually occurs only if non-zero.
`si_unit_xy`	GwySIUnit	Unit of lateral dimensions.
`si_unit_z`	GwySIUnit	Unit of data values.
`data`	array of doubles	Field data, stored as a flat array of size `xres`×`yres`, from top to bottom and from left to right.

Graphs

The following table summarises the common keys of graph-related data in the top-level container for graph number 1. For other graphs, the number 1 has to be replaced with the corresponding graph number. Note that graphs are often numbered sequentially, starting from 1, however, they can have any numbers positive and the set of graph numbers does not have to be contiguous. The number 0 in the prefix of graph keys is a historical relic that does not mean anything and it is always 0.

Key	Type	Meaning
`/0/graph/graph/1`	GwyGraphModel	Graph model object data.
`/0/graph/graph/1/visible`	boolean	Whether the graph should be displayed in a window when the file is loaded.

Graphs are represented as GwyGraphModel objects. The components of a GwyGraphModel are summarised in the following table:

Component	Type	Meaning
`curves`	array of GwyGraphCurveModels	Individual graph curves.
`title`	string	Graph title as displayed in the data browser.
`x_unit`	GwySIUnit	Unit of the abscissa.
`y_unit`	GwySIUnit	Unit of the ordinate.
`top_label`	string	Label on the top axis.
`bottom_label`	string	Label on the bottom axis.
`left_label`	string	Label on the left axis.
`right_label`	string	Label on the right axis.
`x_is_logarithmic`	boolean	Whether the abscissa has a logarithmic scale.
`y_is_logarithmic`	boolean	Whether the ordinate has a logarithmic scale.
`x_min`	double	User-set minimum value of the abscissa.
`x_min_set`	boolean	Whether user-set minimum value of the abscissa should be used (otherwise the range is determined automatically).
`x_max`	double	User-set maximum value of the abscissa.
`x_max_set`	boolean	Whether user-set maximum value of the abscissa should be used (otherwise the range is determined automatically).

Component	Type	Meaning
y_min	double	User-set minimum value of the ordinate.
y_min_set	boolean	Whether user-set minimum value of the ordinate should be used (otherwise the range is determined automatically).
y_max	double	User-set maximum value of the ordinate.
y_max_set	boolean	Whether user-set maximum value of the ordinate should be used (otherwise the range is determined automatically).
grid-type	32bit integer	Type of grid shown. The value is from GwyGraphGridType enum.
label.has_frame	boolean	Whether the graph key has a frame.
label.frame_thickness	32bit integer	Width of graph key frame.
label.reverse	boolean	Whether to reverse the graph key.
label.visible	boolean	Whether the graph key is visible.
label.position	32bit integer	The position (corner) where the graph key is places. The value is from GwyGraphLabelPosition enum.

Graph curves are represented as GwyGraphCurveModel objects. The components of a GwyGraphCurveModel are summarised in the following table:

Component	Type	Meaning
xdata	array of doubles	Abscissa points. The number of points must match ydata.
ydata	array of doubles	Ordinate points. The number of points must match xdata.
description	string	Curve description (name).
type	32bit integer	Curve mode (points, lines, etc.) The value is from GwyGraphCurveType enum.
color.red	double	Red component of the curve color.
color.green	double	Green component of the curve color.
color.blue	double	Blue component of the curve color.
point_type	32bit integer	Type of symbols representing data points. The value is from GwyGraphPointType enum.
point_size	32bit integer	Size of symbols representing data points.
line_type	32bit integer	Type of lines connecting data points. The value is from GwyGraphLineType enum.
line_size	32bit integer	Width of lines connecting data points.

Spectra

The following table summarises the common keys of spectra-related data in the top-level container for spectra set number 0. For other spectra, the number 0 has to be replaced with the corresponding spectra set number. Note that spectra sets are often numbered sequentially, starting from 0, however, they can have any numbers and the set of spectra set numbers does not have to be contiguous.

	Key	Type	Meaning
	/sps/0	GwySpectra	Spectra data.

Sets of spectra of one kind are represented as GwySpectra objects. The components of a GwySpectra are summarised in the following table:

Component	Type	Meaning
title	string	Spectra title as displayed in the data browser.
si_unit_xy	GwySIUnit	Unit of spectrum position coordinates.
coords	array of doubles	Coordinates of points where the spectra were taken, in physical units. Each spectrum takes two items: for the horizontal and vertical coordinate. The number of coordinates must match the number of curves in data.
data	array of GwyDataLines	Individual spectra curves.
selected	array of 32bit integers	Indices of selected spectra curves.

Individual curves in spectra are represented as GwyDataLine objects. The components of a GwyDataLine are summarised in the following table:

Component	Type	Meaning
res	32bit integer	Number of data points.
real	double	Length in physical units.
off	double	Offset of the begining in physical units. It usually occurs only if non-zero.
si_unit_x	GwySIUnit	Unit of abscissa.
si_unit_y	GwySIUnit	Unit of data values.
data	array of doubles	Line data, stored as an array of res, from left to right.

Volume data

The following table summarises the common keys of volume-related data in the top-level container for volume data number 0. For other volume data, the number 0 has to be replaced with the corresponding volume data number. Note that volume data are often numbered sequentially, starting from 0, however, they can have any numbers and the set of volume data numbers does not have to be contiguous.

Key	Type	Meaning
/brick/0	GwyBrick	Volume data.
/brick/0/preview	GwyDataField	Two-dimensional data shown when the volume data is displayed in a window.
/brick/0/title	string	Volume data title, as shown in the data browser.
/brick/0/visible	boolean	Whether the volume data should be displayed in a window when the file is loaded.
/brick/0/preview/palette	string	Name of the false color gradient used to display the preview data.
/brick/0/meta	GwyContainer	Volume data metadata. The keys are directly the names as displayed in the metadata browser and the string values are the values.
/brick/0/log	GwyStringList	Volume data log as a list of string log entries. They have the format *type*::*function*(*param*=*value*, ...)@*time*.

Volume data are represented as GwyBrick objects. The components of a GwyBrick are summarised in the following table:

Component	Type	Meaning
xres	32bit integer	Horizontal size in pixels.
yres	32bit integer	Vertical size in pixels.
zres	32bit integer	Depth (number of levels) in pixels.
xreal	double	Horizontal dimension in physical units.
yreal	double	Vertical dimension in physical units.

Component	Type	Meaning
zreal	double	Depthwise dimension in physical units.
xoff	double	Horizontal offset of the top-left corner in physical units. It usually occurs only if non-zero.
yoff	double	Vertical offset of the top-left corner in physical units. It usually occurs only if non-zero.
zoff	double	Depthwise offset of the top-left corner in physical units. It usually occurs only if non-zero.
si_unit_x	GwySIUnit	Unit of horizontal lateral dimensions.
si_unit_y	GwySIUnit	Unit of vertical lateral dimensions.
si_unit_z	GwySIUnit	Unit of depthwise dimensions.
si_unit_w	GwySIUnit	Unit of data values.
data	array of doubles	Field data, stored as a flat array of size xres×yres×zres, from the zeroth to the last plane, top to bottom and from left to right.
calibration	GwyDataLine	Calibration of the z axis to represent non-linear sampling in this dimension. The number of points must be equal to zres. This component is present only if non-linear sampling is used.

Auxiliary Objects

The components of a GwySIUnit are summarised in the following table:

Component	Type	Meaning
unitstr	string	Textual representation of the unit, e.g. "A" or "m^-1" (as base SI unit, prefixes are ignored).

The components of a GwySelection are summarised in the following table. Some selection types can have other data members; refer to the documentation of specific selection classes for how to interpret the data.

Component	Type	Meaning
max	32bit integer	Maximum number of objects the selection can hold (this is the number set by gwy_selection_set_max_objects()).
data	array of doubles	Selection data. The number of items that form one selection object is determined by the selection type.

The components of a GwyStringList are summarised in the following table. Note if GwyStringLists are used to represent logs, the strings have a specific structure described above.

Component	Type	Meaning
strings	array of strings	List of string items.

5.12 Simple Field Files

The Gwyddion native file format captures all the information and state Gwyddion needs to save and consequently it is quite complex. Often it is not practical to save files in .gwy format in custom programs and scripts creating input for Gwyddion.

The Gwyddion Simple Field file format (.gsf) can be used in these situations instead. It is a single-channel format for 2D data that was designed to be easy and efficient to read and write, with human-readable header, reasonably expressive, and avoiding instrument or application specific fields (though it can optionally bear them).

GSF can be read and written by Gwyddion version 2.20 or later.

Overall structure

A GSF file consists of four parts, in the following order:

magic line Files begin with a "magic line" identifying the file type.

text header The header consists of lines of the form

```
name = value
```

defining individual parameters.

NUL padding The header is terminated by one to four NUL bytes, aligning the data start to a multiple of 4.

binary data Binary data is in 32bit floating-point format.

Magic line

GSF files start with the line

```
Gwyddion Simple Field 1.0
```

terminated by a linefeed character (`\n`, ASCII 0x0a).

Text header

Each header line has the form

```
name = value
```

where any whitespace before the name, around the equal sign and at the end of value is ignored. Field names are case-sensitive and follow the usual rules for identifiers in programming languages.

Similarly to the magic line, the lines in the text header are terminated by a linefeed character as is usual on Unix. This means the header must be read and written in binary mode to ensure preservation of end-of-line characters on other systems (and not changing the header size e.g. by LF → CRLF transformation).

Any non-ASCII characters, that can occur for example in the channel title, are represented in UTF-8 encoding. The NUL character may not occur in the header.

Header fields:

Name	Type	Value
`XRes`	Mandatory	The horizontal size in pixels, a positive integer.
`YRes`	Mandatory	The vertical size in pixels, a positive integer.
`XReal`	Optional	Horizontal size in physical units (given by `XYUnits`), a positive floating point number. It defaults to 1.0 if not given.
`YReal`	Optional	Vertical size in physical units (given by `XYUnits`), a positive floating point number. It defaults to 1.0 if not given.
`XOffset`	Optional	Horizontal offset in physical units (given by `XYUnits`), a floating point number. It defaults to 0.0 if not given.
`YOffset`	Optional	Vertical offset in physical units (given by `XYUnits`), a floating point number. It defaults to 0.0 if not given.
`Title`	Optional	Data/channel title. It has no default, applications might display 'Unknown' or something similar if not given.
`XYUnits`	Optional	Lateral units, i.e. units of physical sizes and offsets. They must be given as base units, that is `m` or `A` with no power-of-10 prefix (Gwyddion could deal with it but it might present a problem for other software). The default is no units. This means in SPM data, you normally wish to specify `XYUnits` as `m` because the lateral dimensions are in metres.
`ZUnits`	Optional	Value units, i.e. units of data values. See `XYUnits` above for details.

Floating point numbers can be in the scientific format, e.g. `1.23e-4`. They are represented in the standard C/POSIX locale, i.e. decimal dot is used (not comma or other separators).

The header may contain other fields beside those listed above. Gwyddion will load them into metadata. Common informational fields can include `Comment`, `Date` or `Direction`.

Fields may occur in any order, nevertheless, it is recommended to start with mandatory fields, continue with optional fields and put custom fields last.

A simple header example (also including the magic line):

```
Gwyddion Simple Field 1.0
XRes = 400
YRes = 400
XReal = 5e-05
YReal = 5e-05
XYUnits = m
ZUnits = V
Title = ADC2
```

NUL padding

The text header is followed by one to four NUL (`\0`, ASCII 0x00) bytes that (a) terminate it and (b) align the data start to an offset from the begining of file that is a multiple of 4. More precisely, denoting N the total length of the magic line and the text header, the data starts at the nearest multiple of 4 larger than N.

This padding to a multiple of 4 ensures aligned memory access when mapping the file directly to memory. The number of NUL bytes is uniquely determined by the remainder of the length modulo four (N mod 4):

Remainder	Number of padding NULs
0	4
1	3
2	2
3	1

Binary data

Data values are stored as IEEE 32bit single-precision floating point numbers, in little-endian (LSB, or Intel) byte order. Values are stored by row, from top to bottom, and in each row from left to right.

The physical units of these values are `ZUnits`.

The size of the image data is exactly `4*XRes*YRes` bytes and there is no data after it in the file.

5.13 Simple XYZ Files

Although Gwyddion does work with general XYZ data and raw XYZ data are interpolated to a grid upon import, need has arisen for a file format similar in spirit to Gwyddion simple field (.gsf) but representing the data in XYZ format. Such file format, called Gwyddion XYZ Field (`.gxyzf`), is described in this section.

It should be noted that Z simply stands for the ordinate here. Z values in the file may be actual Z coordinates (heights) but they may also be currents, voltages, etc.

GXYZF can be written by Gwyddion version 2.31 or later. They can also be read since this version, although the regularisation to a grid may be somewhat crude.

Overall structure

A GXYZF file consists of four parts, in the following order:

magic line Files begin with a “magic line” identifying the file type.

text header The header consists of lines of the form

```
name = value
```

defining individual parameters.

NUL padding The header is terminated by one to eight NUL bytes, aligning the data start to a multiple of 8.

binary data Binary data is in 64bit floating-point format.

Magic line

gxyzf files start with the line

```
Gwyddion XYZ Field 1.0
```

terminated by a linefeed character (`\n`, ASCII 0x0a).

Text header

Each header line has the form

```
name = value
```

where any whitespace before the name, around the equal sign and at the end of value is ignored. Field names are case-sensitive and follow the usual rules for identifiers in programming languages.

Similarly to the magic line, the lines in the text header are terminated by a linefeed character as is usual on Unix. This means the header must be read and written in binary mode to ensure preservation of end-of-line characters on other systems (and not changing the header size e.g. by LF → CRLF transformation).

Any non-ASCII characters, that can occur for example in channel titles, are represented in UTF-8 encoding. The NUL character may not occur in the header.

Header fields:

Name	Type	Value
`NChannels`	Mandatory	Number of value (Z) channels, a positive integer. The values stored for each point include also coordinates X and Y but they are not counted into `NChannels`.
`NPoints`	Mandatory	Number data points in the file.
`XYUnits`	Optional	Lateral units, i.e. units of X and Y values. They must be given as base units, that is `m` or `A` with no power-of-10 prefix (Gwyddion could deal with it but it might present a problem for other software). The default is no units. This means in SPM data, you normally wish to specify `XYUnits` as `m` because the lateral dimensions are in metres.
`ZUnits1`, `ZUnits2`, ...	Optional	Value units, i.e. units of data values for individual channels. Channels are numbered from 1 to `NChannels`. See `XYUnits` above for details.
`Title1`, `Title2`, ...	Optional	Titles of individual channels. Channels are numbered from 1 to `NChannels`. Titles have no default, applications might display 'Unknown' or something similar if not given.
`XRes`	Optional	Hint specifying the preferred horizontal size in pixels if the data are regularised to a grid, a positive integer. Readers are not required to honour it and may interpolate data to grids of different dimensions.
`YRes`	Optional	Hint specifying the preferred vertical size in pixels if the data are regularised to a grid, a positive integer. Readers are not required to honour it and may interpolate data to grids of different dimensions.

The header may contain other fields beside those listed above. Gwyddion will load them into metadata. Common informational fields can include `Comment`, `Date` or `Direction`.

Fields may occur in any order, nevertheless, it is recommended to start with mandatory fields, continue with optional fields and put custom fields last.

A simple header example of a two-channel file (also including the magic line):

```
Gwyddion XYZ Field 1.0
NChannels = 2
NPoints = 457884
XYUnits = m
ZUnits1 = m
ZUnits2 = V
Title1 = Height
Title2 = ADC2
```

NUL padding

The text header is followed by one to eight NUL (`\0`, ASCII 0x00) bytes that (a) terminate it and (b) align the data start to an offset from the begining of file that is a multiple of 8. More precisely, denoting N the total length of the magic line and the text header, the data starts at the nearest multiple of 8 larger than N.

This padding to a multiple of 8 ensures aligned memory access when mapping the file directly to memory. The number of NUL bytes is uniquely determined by the remainder of the length modulo eight (N mod 8):

Remainder	Number of padding NULs
0	8
1	7
2	6
3	5
4	4
5	3
6	2
7	1

Binary data

Data values are stored as IEEE 64bit double-precision floating point numbers, in little-endian (LSB, or Intel) byte order. Points are stored in arbitrary order. Each point is stored as a block of `NChannels+2` values: X, Y and then all ordinate values, in the channel order.

The physical units of the values are given by `XYUnits` for X and Y and then `ZUnits1`, `ZUnits2`, ... for the ordinate values.

The size of the data is exactly `8*NPoints*(NChannels+2)` bytes and there is no data after it in the file.

Chapter 6

Developing Gwyddion

You are encouraged to become a developer of Gwyddion.

If you want to become developer, we recommend you to start with some simple modules (see module tutorial), to see how the application works. If you write a module or plug-in, you are encouraged to share it here with other Gwyddion users. Let us know, so that we can link your modules or plug-ins to these pages or even include in it Gwyddion. You don't have to limit yourself to modules or plug-ins of course, but they should be easier to start with.

API References

There are many functions that can help you while developing your module. See API reference at Documentation section of the project web.

Bug reports

We will be very happy if you send us bug reports if you find errors in Gwyddion. For doing this, please, specify as much as possible the situation that led to error, operating system and Gwyddion version used. You can also send us the SPM data that were being processed when the problem was found, this is necessary namely for reporting bugs related to file loading and saving.

The preferred bug reporting method is to send an e-mail to klapetek@gwyddion.net.

Appendix A

GNU General Public License

A.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

A.2 Terms And Conditions For Copying, Distribution And Modification

Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License.

Exception:
If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed

need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

A.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>    <name of author>
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.,
51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author Gnomovision
comes with ABSOLUTELY NO WARRANTY; for details type "show w".
This is free software, and you are welcome to redistribute it under
certain conditions; type "show c" for details.
```

The hypothetical commands “show w” and “show c” should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than “show w” and “show c”; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
program “Gnomovision” (which makes passes at compilers) written
by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B

GNU Free Documentation License

B.1 Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

B.2 Applicability And Definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing

tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

B.3 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

B.4 Copying In Quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

B.5 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

GNU FDL MODIFICATION CONDITIONS

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

B.6 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

B.7 Collections Of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

B.8 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

B.9 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

B.10 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

B.11 Future Revisions Of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

B.12 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> SAMPLE INVARIANT SECTIONS LIST
>
> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> SAMPLE INVARIANT SECTIONS LIST
>
> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Chapter 7

Index