FatGHoL Documentation

Release 5.4

Riccardo Murri

January 12, 2016

Contents

Author Riccardo Murri <riccardo.murri@gmail.com>

Date 2012-02-08

Revision $Revision$

Contents:

CHAPTER 1

Installation of FatGHoL

Note: Although it is *theoretically* possible to install FatGHoL in Windows or MacOSX, I have never attempted that.

Therefore, **the following instructions are Linux-only.**

These instructions show how to install FatGHoL from its source repository into a separate directory. This has the advantage that all code is confined in a single directory, and can thus be easily replaced/removed. The instructions can be easily adapted to system-wide installation by anyone having a bit of familiarity with Linux system administration.

Although FatGHoL is a pure-Python module, it depends on the LinBox exact linear algebra library for computing the rank of homology matrices. Unfortunately, this complicates the installation procedure: LinBox depends on several other libraries, which must be downloaded and compiled. The sections below detail what should be installed in order to get a working FatGHoL installation.

1.1 Initial installation

1.1.1 0. The prerequisite of prerequisites: C++ compiler and SVN

Before you install anything else, you need to have a working C and C++ compiler on the system, and the Subversion (SVN) source control system. This is generally not a problem on Linux systems, which come with the GCC compiler preinstalled, and SVN is available as an optional package. To check if you have a C++ compiler installed, type the following commands at your shell prompt:

```
c++ --version; svn --version
```

If you get output similar to the following (the version number and copyright may vary), then everything is OK:

```
c++ (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

svn, version 1.6.12 (r955767)
 compiled Aug  5 2011, 17:07:24

Copyright (C) 2000-2009 CollabNet.
Subversion is open source software, see http://subversion.tigris.org/
...
```

If you instead get a "command not found" error, then you need to install the C/C++ compiler and SVN:

- On Debian and Ubuntu, install packages `subversion`, `gcc` and `g++`:

```
sudo apt-get install subversion gcc g++
```

- On Fedora, RHEL, or CentOS, install packages `subversion`, `gcc` and `gcc-c++`:

```
# run this command as "root" user
yum install subversion gcc gcc-c++
```

- On other Linux distributions, please refer to your distribution website for instructions.

1.1.2 1. Download the FatGHoL sources into the installation directory

Check-out the `fatghol` files into the installation directory:

```
svn checkout http://fatghol.googlecode.com/svn/trunk/ "$HOME/fatghol"
cd "$HOME/fatghol"
```

In this step and in the following ones, the directory `$HOME/fatghol` is be the installation folder of FatGHoL. You can change this to another directory path; any directory that's writable by your Linux account will be OK.

1.1.3 2. Install auxiliary libraries and software

This is the crucial step: we're now going to install all the software required by FatGHoL and LinBox in one go.

On Debian and Ubuntu, the required software is already available as optional system packages, so it can be installed with ease; an installation script is provided for other Linux systems, which should be able to install the required software without assistance.

- On Debian and Ubuntu systems, the following command should install all the required software:

```
sudo apt-get install python-dev swig liblinbox-dev
```

- On other systems, you may want to use the `prereq.sh` script that you can find in the FatGHoL installation directory. Invoking the script like this:

```
cd $HOME/fatghol
./prereq.sh
```

  will download and install all necessary dependencies into a directory `sw` (created inside the FatGHoL installation directory).

  You can edit the initial section of the `prereq.sh` script to select what software needs to be installed on your system. The default is to install all dependencies.

  Note: The `prereq.sh` script may take a very long time to complete; especially the compilation of the linear algebra library ATLAS can take hours!

  On the other hand, if you interrupt the script, just invoke it again with `./prereq.sh` and it should be able to pick up from whence it left.

- If the script does not work, you may want to attempt installation from source of the packages, according to the instructions given on the respective websites. These are the packages needed by FatGHoL:
  - Python, version (at least) 2.6
  - SWIG (version 1.3.40 is known to work well with FatGHoL)
  - LinBox, at least version 1.1.6

  Note that the `./mgn.sh` script for running graph homology computations expects the libraries to be available in the `sw` directory withing the FatGHoL installation directory.

1.1.4 3. Install FatGHoL

Last step: run the `setup.py` script to compile the glue code that links FatGHoL with LinBox:

```
cd $HOME/fatghol
python setup.py develop
```

Note: If you have installed Python with the installation script `prereq.py`, then you need to type the following command *before* you run `setup.py`, to be sure it is executed by the Python interpreter installed in step 2.:

```
PATH=$HOME/fatghol/sw/bin:$PATH
export PATH
```

1.1.5 4. Check your installation

Now you can check your FatGHoL installation; just type the command:

```
./mgn.sh selftest
```

and you should see the following output appear on your screen (elapsed times will of course be different):

```
Module 'rg' OK, passed all doctests.
Module 'homology' OK, passed all doctests.
Module 'graph_homology' OK, passed all doctests.
Module 'combinatorics' OK, passed all doctests.
Module 'iterators' OK, passed all doctests.
Module 'cyclicseq' OK, passed all doctests.
Checking homology algorithm (no checkpointing)
  Computation of M_{0,3} homology: OK (elapsed: 0.010s)
  Computation of M_{0,4} homology: OK (elapsed: 0.110s)
  Computation of M_{0,5} homology: OK (elapsed: 24.030s)
  Computation of M_{1,1} homology: OK (elapsed: 0.010s)
  Computation of M_{1,2} homology: OK (elapsed: 0.180s)
  Computation of M_{2,1} homology: OK (elapsed: 6.050s)
Checking homology algorithm (checkpointing)
  Computation of M_{0,3} homology: OK (elapsed: 0.000s)
  Computation of M_{0,4} homology: OK (elapsed: 0.120s)
  Computation of M_{0,5} homology: OK (elapsed: 24.520s)
  Computation of M_{1,1} homology: OK (elapsed: 0.000s)
  Computation of M_{1,2} homology: OK (elapsed: 0.120s)
  Computation of M_{2,1} homology: OK (elapsed: 6.020s)
Checking homology algorithm (restoring from checkpointed state)
  Computation of M_{0,3} homology: OK (elapsed: 0.000s)
  Computation of M_{0,4} homology: OK (elapsed: 0.040s)
  Computation of M_{0,5} homology: OK (elapsed: 0.350s)
  Computation of M_{1,1} homology: OK (elapsed: 0.000s)
  Computation of M_{1,2} homology: OK (elapsed: 0.020s)
  Computation of M_{2,1} homology: OK (elapsed: 0.090s)
```

If you get errors, do not despair! Feel free to write *me <mailto:riccardo.murri@gmail.com>* and I will do my best to help.

1.2 Upgrade

These instructions show how to upgrade the FatGHoL scripts to the latest version found in the source repository.

1. *cd* to the directory containing the FatGHoL virtualenv; assuming it's named `fatghol` as in the above installation instructions, you can issue the commands:

```
cd $HOME/fatghol # use '/opt/fatghol' if root
```

2. Upgrade the *fatghol* source and run the *setup.py* script again:

```
svn update
export PATH=$HOME/fatghol/sw/bin:$PATH
python setup.py develop
```

1.3 HTML Documentation

HTML documentation for the FatGHoL programming interface can be read online at:

http://fatghol.googlecode.com/svn/trunk/doc/html/index.html

You can also generate a local copy from the sources:

```
cd $HOME/fatghol # use '/opt/fatghol' if root
cd doc
make html
```

Note that you need the Python package *Sphinx <http://sphinx.pocoo.org>* in order to build the documentation locally.

CHAPTER 2

Using FatGHoL

FatGHoL comes with a front-end script to compute the (co)homology of the moduli space of marked smooth Riemann surfaces (using Kontsevich' "graph homology" complex).

The front-end script is called `mgn.sh`; you can invoke it with the `--help` command line option to get a recap of its functionality:

```
$ ./mgn.sh --help
usage: mgn [-h] [-D [DEBUG]] [-l LOGFILE] [-o OUTFILE] [-s CHECKPOINT_DIR]
           [-u] [-v] [-V]
           ACTION [ARG [ARG ...]]

    Actions:

      graphs G N
        Generate the graphs occurring in M_{g,n}.

      homology G N
        Print homology ranks of M_{g,n}.

      latex G N [-s DIR] [-o FILE]
        Read the listings of M_{g,n} fatgraphs (from directory DIR)
        and output a pretty-print catalogue of the graphs as LaTeX documents.

      valences G N
        Print the vertex valences occurring in M_{g,n} graphs.

      shell
        Start an interactive PyDB shell.

      selftest
        Run internal code tests and report failures.


positional arguments:
  ACTION                Action to perform, see above.
  ARG                   Arguments depend on the actual action, see above.

optional arguments:
  -h, --help            show this help message and exit
  -D [DEBUG], --debug [DEBUG]
                        Enable debug features:
                            * pydb -- run Python debugger if an error occurs
                            * profile -- dump profiler statistics in a .pf file.
                            Several features may be enabled by separating them
                            with a comma, as in '-D pydb,profile'.
  -l LOGFILE, --logfile LOGFILE
                        Redirect log messages to the named file
                            (by default log messages are output to STDERR).
```

```
  -o OUTFILE, --output OUTFILE
                        Save results into named file.
  -s CHECKPOINT_DIR, --checkpoint CHECKPOINT_DIR
                        Directory for saving computation state.
  -u, --afresh          Do NOT restart computation from the saved state in checkpoint directory.
  -v, --verbose         Print informational and status messages as the computation goes on.
  -V, --version         show program's version number and exit
```

That should be pretty much self-explanatory; however a bit more detail is given below.

2.1 The `graphs` action

With `./mgn.sh graphs 0 4` (for example) you can print a list of all the fatgraphs having genus 0 and 4 boundary cycles. If you want to save the list to a file, use the `-o` option, followed by the file name.

The list of fatgraphs is also saved in directory `M0,4.data/` in several `.list` files, depending on the number of vertices. For instance, the `M0,4-MgnGraphsIterator3.list` file is the one collecting fatgraphs with 3 vertices.

2.2 The `homology` action

With `./mgn.sh homology 0 4` (for example), you can print the Betti numbers of the moduli space of smooth marked Riemann surfaces having genus 0 and 4 marked points. If you want to save the result to a file, use the `-o` option, followed by the file name.

Internally, the `homology` function uses the `graphs` function, so it generates all by-products of that function. In particular, graphs lists are generated and saved in the *checkpoint directory* (`M0,4.data` in this example).

In addition to graph lists, the boundary operator matrices are computed and saved in the *checkpoint directory*. After their ranks have been computed, they are saved as well, so invoking the `homology` action with a fully-populated *checkpoint directory* gives the result almost istantaneously.

2.3 The `latex` action

This reads the contents of an existing checkpoint directory and generates a LaTeX report on all the graphs: what graphs are there, their automorphisms and markings, etc.

To save the report into a file, use the `-o` option followed by the file name, e.g.:

```
./mgn.sh latex 0 4 -o report.tex
```

The name of the *checkpoint directory* is automatically generated from the parameters G and N; use the `-s` option to use a different directory.

2.4 The `valence` action

This prints the valences of vertices of fatgraphs with given genus and number of boundary cycles.

As this is a very simple computation, nothing is saved to the *checkpoint directory*.

2.5 Checkpoint directory

This is a directory where FatGHoL saves result of computationally-expensive functions. When FatGHoL is invoked again at a later time, it loads the results from the checkpoint directory instead of calculating them again; this results in a substantial speedup. However, you can use the `-u` command-line option to tell FatGHoL to ignore the contents of a checkpoint directory.

The name of the *checkpoint directory* is automatically generated from the parameters G and N; use the `-s` option to use a different directory.

There is no way of avoiding that FatGHoL creates a checkpoint directory and populates it.

CHAPTER 3

FatGHoL programming API

This is the documentation of the functions and classes available in the `fatghol` Python module, for use in your own programs.

3.1 Fatgraph-related modules

This is the core of FatGHOL: the Python modules that implement an interface for generating fatgraphs and computing their homology complex.

3.1.1 *fatghol.rg*

3.1.2 *fatghol.homology*

3.1.3 *fatghol.graph_homology*

3.1.4 *fatghol.valences*

3.2 Utility modules

These are support modules, used by the core FatGHoL modules. They are more general in nature and most code from these modules could be packaged separately and re-used in other projects.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search