PyFRAP: Software for analysis of Fluorescence Recovery After Photoconversion (FRAP) experiments

User Guide

Alexander Bläßle and Patrick Müller

Friedrich Miescher Laboratory of the Max Planck Society
Spemannstraße 39
72076 Tübingen
Germany

E-Mail: alexander.blaessle@tuebingen.mpg.de
Website: `http://www.fml.tuebingen.mpg.de/mueller-group/`

1 Introduction

Fluorescence Recovery After Photobleaching (FRAP) is a microscopy-based technique for measuring protein mobility.
Here, we provide a user guide for the PyFRAP software.

2 Installation

PyFRAP was developed as an open source graphical user interface (GUI) in Python with PyQT and SciPy in order to make it accessible and extendable across the most frequently used operating systems Ubuntu Linux, Mac OS X, and Microsoft Windows. Over the past two decades, Python has become a widely used scientific programming language and provides PyFRAP users with enormous resources and easily addable software packages (**?**).

All software packages needed to run PyFRAP are freely available. PyFRAP can be installed using stand-alone executables (see Section 2.1). Alternatively, users can run the PyFRAP packages from source (see Section 2.2), which offers the possibility to edit the PyFRAP code and to import new modules.

2.1 Running PyFRAP using stand-alone executables

Download the executable that fits your system from http://people.tuebingen.mpg.de/mueller-lab/. This is suitable for users who want to analyze FDAP experiments and do not need to customize the PyFRAP code. A list of currently available binary files and systems on which the binaries have been tested can be found in Table 1. If there is no executable available for your system, we recommend using the Anaconda installation approach explained in Section 2.2.1.

OS	Version	32-bit	64-bit	Executable	Test System
Linux	3.13.0-36-generic		×		Thinkpad x230

Table 1: List of systems on which the currently available PyFRAP executables have been tested. The executables might also run on systems not listed here.

2.2 Running PyFRAP from source

In order to be able to edit the PyFRAP code and to import new modules, it is necessary to download and install all necessary Python packages and to run PyFRAP from source. There are two ways to do this:

1. Download and install the Anaconda Python distribution (see Section 2.2.1).
2. Download and install all Python packages manually (see Section 2.2.2).

2.2.1 Running PyFRAP using the Anaconda distribution

Anaconda is a bundle of Python packages and includes most of the packages needed to run PyFRAP. To install Anaconda, follow these steps:

- Go to http://continuum.io/downloads and download the current Python 2.7.x release of Anaconda for your operating system
- Launch the installer by double-clicking (Mac OS X and Windows) or
  - Open a Terminal
  - Go to the directory containing the installer by typing

    ```
    cd path/to/installer
    ```

    and execute the installer with

    ```
    ./installer
    ```
- Follow the instructions of the installer
- Download the current version of Gmsh from `http://geuz.org/gmsh/`. Unpack the zip file and copy it to your *Programs* under Microsoft Windows or mount the .dmg file and copy Gmsh.app to the *Applications* folder under Mac OSX.
- Both for Microsoft Windows and Mac OSX the Gmsh executable needs to be added to the `PATH` environment variable. Instructions how to add an executable to `PATH` can be found in Sections 2.6 and **??** for Mac OS X and Microsoft Windows respectively. The standard paths for the Gmsh executable are `/Applications/Gmsh.app/MacOS` for Mac OS X and `????` for Microsoft Windows.
- Installing the pysparse package will speed FRAP simulations substantially.
- Launch PyFRAP by double-clicking `pyfrap_app.py` in the PyFRAP source directory (Windows) or
  - Open a Terminal
  - Go to the directory containing the PyFRAP source files

    ```
    cd path/to/PyFRAP
    ```
  - Launch PyFRAP by typing

    ```
    python pyfrap_app.py
    ```

2.2.2 Running PyFRAP using a manual Python installation

In this section, we explain how to manually install all necessary Python packages on Linux, Mac OS X, and Windows in order to run PyFRAP. The manual installation allows for customizability as well as debugging options. The instructions provided here describe the installation process for computers that currently do not have Python installed. For computers on which Python is already installed, the installation of PyFRAP will differ from the instructions provided below. We recommend running PyFRAP using a Debian-based Linux distribution such as Ubuntu since installing Python packages is more straightforward using such operating systems.

2.3 Manual installation under Linux

Here we explain how to manually install and run PyFRAP on Linux operating systems. The following instructions are only suitable for Debian-based Linux distributions and have been tested on Ubuntu Linux 12.04, 13.10, and 14.04 (64-bit). Installation steps may vary between different versions and distributions of Linux (e.g. RedHat-based Linux distributions such as Fedora or Suse). .

- Open a Terminal.
- In your Terminal, type (you will need sudo rights):

  ```
  sudo apt-get install python-numpy
  sudo apt-get install python-scipy
  sudo apt-get install python-matplotlib
  sudo apt-get install python-qt4
  sudo apt-get install python-skimage
  sudo apt-get install gmsh
  sudo apt-get install python-fipy
  ```

  Note: On Ubuntu versions older than 12.10, python-skimage needs to be installed from here: `http://neuro.debian.net/pkgs/python-skimage.html`.

- Go to your PyFRAP folder by typing

  ```
  cd path/to/PyFRAP/
  ```

  and launch PyFRAP by typing

  ```
  python pyfrp_app.py
  ```

  If PyFRAP does not launch, open a Python Terminal and try to import all necessary packages by typing

  ```
  import numpy
  import scipy
  import matplotlib
  import matplotlib.image
  import PyQt4
  import code
  import fipy
  ```

  This should run without any problems. If you receive an error message while importing any of these modules, please try to reinstall the package or visit the development website of the problematic package.

2.4 Manual installation under Mac OS X

Here we explain how to manually install and run PyFDAP on Mac OS X. The following instructions have only been tested on Mac OS X 10.9.4, 10.9.5 (64-bit). Installation steps may vary between different versions of OS X.

- Installing Python packages requires the C++ compiler gcc. gcc can be obtained by downloading XCode from the Apple AppStore.
- Launch a bash terminal under Applications → Utilities → Terminal.
- Type

  ```
  gcc
  ```

  You should see a popup window asking you to install Command Line Tools. Follow the instructions in the popup window.
- Homebrew is a package manager for MacOSX that facilitates installing packages under OSX. Download homebrew by typing:

  ```
  ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/
      go/install)"
  ```

- Check the homebrew installation by typing:

  ```
  brew update
  brew doctor
  ```

  If the ouput returns any problems, visit the homebrew website (`http://brew.sh/`) for further instructions.
- Install Python by typing into the bash terminal:

  ```
  brew install python
  ```

  Note that MacOSX comes with a native Python installation. If you want to use the native Python installation, you could either install all packages separately by using the python package index (pip) or use homebrew to install all packages and then link them using the `site` package `https://docs.python.org/2/library/site.html`. However, we recommend using the Python installation of homebrew.
- Link the new homebrew installation by typing into the bash terminal

  ```
  brew link python
  brew linkapps
  ```

- Link the new python installation into .bash_profile by launching the text editor `nano`:

```
nano ~/.bash_profile
```

and add the following lines:

```
PATH="/usr/local/bin:${PATH}"
export PATH
export PYTHONPATH=/usr/local/lib/python2.7/site-packages/:
```

Press `Ctrl+O` and `Ctrl+X` to save the new .bash_profile and exit. Restart the terminal and type:

```
which python
```

The output should be

```
/usr/local/bin/python
```

If not, ensure that you set the Python path properly or don't have a different homebrew installation prefix. If everything went right, you will now use the homebrew Python installation when you call `python` in the terminal.

- Download and install PyQT4 and SIP by typing into the terminal

```
brew install sip
brew install pyqt
brew linkapps
```

- Download and install nose and Numpy by typing into the terminal

```
pip install nose
brew install numpy
brew link numpy
```

Sometimes numpy can also be found by typing into the terminal

```
brew install homebrew/python/numpy
brew link numpy
```

- Download and install Scipy by typing into the terminal

```
pip install scipy
```

or

```
brew install scipy
```

- Download and install scikit-image by typing into the terminal

```
pip install cython
pip install scikit-image
```

- Download and install matplotlib by typing into the terminal

```
pip install python-dateutil
pip install pyparsing
brew install matplotlib
```

- Download and install PIL by typing into the terminal

```
brew install Homebrew/python/pillow
```

- Download and install Gmsh by typing into the terminal

```
brew install gmsh
```

- Download and install pyparse by typing into the terminal

```
pip install pyparse
```

- Download and install fipy by typing into the terminal

```
pip install fipy
```

- For a better performance during the FRAP simulations we recommend using the pysparse package. Currently it seems that this package is corrupted in pip, so building it from source is necessary. To do so, go to `http://sourceforge.net/projects/pysparse/` and download the latest version of pysparse. Unpack the downloaded file and go to the folder containing the unpacked files by typing

```
cd path/to/unpacked files
```

  Build the pyparse package by typing

```
python setup.py install
```

- Go to your PyFRAP folder by typing into the terminal

```
cd path/to/PyFRAP/
```

  and launch pyFRAP by typing into the terminal

```
python pyfrp_app.py
```

If PyFRAP does not launch, open a Python terminal and try to import all necessary packages by typing into the terminal:

```
import numpy
import scipy
import matplotlib
import matplotlib.image
import PyQt4
import code
import fipy
```

If you receive an error message while importing any of these modules, try to reinstall the package or visit the development website of the problematic package.

2.5 Manual installation under Microsoft Windows

Here we explain how to manually install and start PyFDAP on Microsoft Windows. The following instructions have only been tested for Microsoft Windows 8 (64-bit) and may differ for other versions.

- Download and install the current version of Python 2.7x from `https://www.python.org/download/releases/`.
- Download and install the current version of PyQt4 from `http://www.riverbankcomputing.co.uk/software/pyqt/download`. The Windows installer will also install the required package SIP an all necessary QT libraries.
- Download and install the current version of scipy-stack from `http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy-stack`. scipy-stack includes important Python packages such as nose, numpy, scipy and matplotlib. If you want to install the packages separately because there is no suitable installation binary of scipy-stack available, you can use the following links. However, we recommend using scipy-stack:
  - Numpy: `http://sourceforge.net/projects/numpy/files/NumPy/` if you are running a 32bit system, on a 64 bit system go to `http://www.kfd.uci.edu/~gohlke/pythonlibs/`
  - Scipy: `http://sourceforge.net/projects/scipy/files/scipy/`
  - Matplotlib: `http://matplotlib.org/downloads.html`
  - Nose: `https://nose.readthedocs.org/en/latest/`
  - Ipython: `https://github.com/ipython/ipython/releases`

  All packages on `http://www.lfd.uci.edu/~gohlke/pythonlibs` are provided in the wheel (.whl) format. An instruction on how to install wheel packages can be found in Section 2.8.

- Download and install the current version of scikit-image from `http://www.lfd.uci.edu/~gohlke/pythonlibs/#scikit-image`.
- Download and install the current version of pyparse from `http://www.lfd.uci.edu/~gohlke/pythonlibs/#pysparse`. More details about pysparse can be found in Section 2.9.
- Download the current version of Gmsh from `http://geuz.org/gmsh/`. Unpack the zip file and copy it to your *Programs* folder.
- Download and unpack the current version of PyFRAP from `http://people.tuebingen.mpg.de/mueller-lab/`.
- Go to your PyFRAP folder and launch `pyfrp_app.py`. If PyFRAP does not launch, open a Python terminal and try to import all necessary packages by typing:

```
import numpy
import scipy
import matplotlib
import matplotlib.image
import PyQt4
import code
```

  This should run without any problems. If you receive an error message while importing any of these modules, please try to reinstall the package or visit the development website of the problematic package.
- NOTE: The PyFRAP video export has not been tested under Microsoft Windows. The required software Mencoder can be downloaded from `http://mplayerwin.sourceforge.net/downloads.html`.

2.6 Adding a program to the PATH environment variable in Microsoft Windows

The `PATH` environment variable is a variable containing folders where the Mac OSX's terminal looks for executables. Here we explain how to add a folder to this list such that entering the path to the executable is not necessary anymore.

- Start the Terminal. The Terminal can be found in *Applications* → *Utilities* → *Terminal.*
- To test if the program is already accesible through the `PATH`, type

```
name_of_program
```

- If you receive the error message `name_of_program: command not found`, the program needs to be added to you `PATH`.

- To add `program` to `PATH`, edit .bash_profile by launching the text editor `nano`:

  ```
  nano ~/.bash_profile
  ```

  and add the following lines:

  ```
  export PATH="path/to/executable:$PATH"
  ```

  Press `Ctrl+O` and `Ctrl+X` to save the new .bash_profile and exit.

- After adding `program` to the `PATH` environment variable, restart the Terminal such that the Terminal updates `PATH`.

2.7 Adding a program to the PATH environment variable in Microsoft Windows

The `PATH` environment variable is a variable containing folders where the Microsoft Windows command prompt looks for executables. Here we explain how to add a folder to this list such that entering the path to the executable is not necessary anymore.

- Start the command promt by pressing `Win+R`, type `cmd` and hit `Enter`.
- To test if the program is already accesible through the `PATH`, type

  ```
  name_of_program
  ```

- If you receive the error message `'name_of_program' is not recognized as an internal or externa` the program needs to be added to you `PATH`.
- To add `program` to `PATH`

  ```
  set PATH=%PATH%;path/to/program
  ```

- After adding `program` to the `PATH` environment variable, restart the command prompt such that the command prompt updates `PATH`.

2.8 Installing wheel packages in Microsoft Windows

Wheel packages are a common way to distribute system specific precompiled python packages. You can install wheel packages using pip.

- Start the command promt by pressing `Win+R`, type `cmd` and hit `Enter`.
- Make sure that pip was added to your systems `PATH` by typing

  ```
    pip
  ```

If this is not the case, follow the instructions in Section 2.7 to add pip to the command prompt's `PATH` environment variable. Note that for standard Python installations, the path to pip should be `C:\Python27\Scripts`.

- To install wheel packages, enter into the command prompt

```
cd path\to\wheelpackage
pip install wheelpackage.whl
```

2.9 Installing pysparse in Microsoft Windows

Pysparse is a python package for handling sparse matrices. Having pysparse installed will substantially speed up FRAP simulations in PyFRAP. However, Anaconda does not provide pysparse and the it seems that currently this package is corrupted in pip, so building it from a wheel is necessary. To do so,

- Download the current version of pyparse from `http://www.lfd.uci.edu/~gohlke/pythonlibs/#pysparse`.
- Follow the instructions in Section 2.8 to install the downloaded wheel package.
- In case of a 64bit system, there is a mistake in the file pysparse package that is easy to fix. The code of the function `addAt` in the file `pysparseMatrix.py` needs to be changes to:

```
def addAt(self, vector, id1, id2):
    import numpy as np
    id1=id1.astype(np.int32)
    id2=id2.astype(np.int32)

    self.matrix.update_add_at(vector, id1, id2)
```

  If you use the Anaconda Python distribution according to Section 2.2.1, you should find this file in
  `C:\Users\your_user_name\Anaconda\Lib\site-packages\fipy\matrices\pysparseMatrix.py`.
  For a standard Python installation, this file should be in
  `C:\Python27\Lib\site-packages\fipy\matrices\pysparseMatrix.py`.