pyIMD Documentation

Release 0.0.8

Andreas P. Cuny

Jun 29, 2020

Contents:

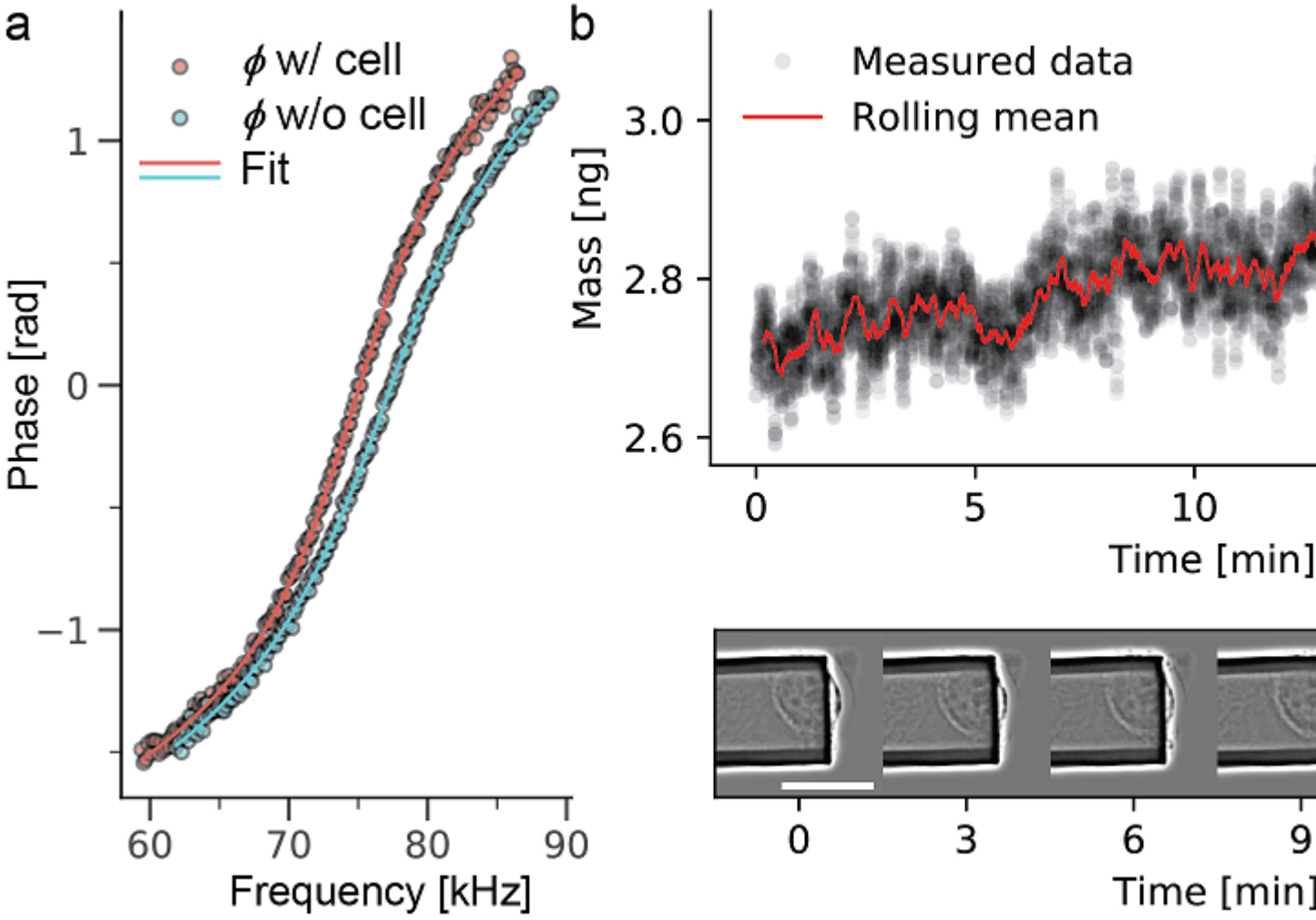


Fig. 1: Evolution of mass over time and the corresponding microscopy images are shown for a time span of 20min. The mass data was acquired every 10 ms (data shown in black), overlaid in red is the rolling mean with a window of 1000. Images taken every 3 min over the observed times span. The mammalian cell increases mass steadily..

The total mass of single cells can be accurately monitored in real time under physiological conditions with our recently developed picobalance. It is a powerful tool to investigate crucial processes in biophysics, cell biology or medicine, such as cell mass regulation. However, processing of the raw data can be challenging, as computation is needed to

extract the mass and long-term measurements can generate large amounts of data. Here, we introduce the software package **pyIMD** that automates raw data processing, particularly when investigating non-migrating cells. **pyIMD** stands for Python inertial mass determination and is implemented using Python 3.6 and can be used as a command line tool or as a stand-alone version including a graphical user interface.

This documentation of **pyIMD** describes the API and provides sample data sets as well as sample scripts to run **pyIMD** from Jupyter or the Python console. It also contains a tutorial about how **pyIMD** is used with the user interface.

CHAPTER 1

Installation

1.1 Stable release

1.1.1 As module

To install pyIMD, just run this command in your terminal:

```
$ pip install pyIMD
```

Installing pyIMD this way ensures that you get always the latest release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.1.2 As stand alone executable

If you want to install pyIMD on your system without installing Python yourself just download the pre-compiled executable matching your operating system:

pyIMD can then be used trough its graphical user interface (GUI) directly.

1.2 From sources

The latest sources for pyIMD can be downloaded from the Github repo.

You can clone the public repository:

```
$ git clone git://git.gitlab.com/csb.ethz/pyIMD.git
```

Once you have a copy of the source, navigate into the directory and run:

```
$ python setup.py install .
```

CHAPTER 2

Use and Examples

The examples show the basic usage of **pyIMD** to calculate the mass

2.1 pyIMD example script

This example script demonstrates the simplest interaction with **pyIMD**:

```
# /*****************************************************************************
# * Copyright © 2018-2019, ETH Zurich, D-BSSE, Andreas P. Cuny & Gotthold Fläschner
# * All rights reserved. This program and the accompanying materials
# * are made available under the terms of the GNU Public License v3.0
# * which accompanies this distribution, and is available at
# * http://www.gnu.org/licenses/gpl
# *
# * Contributors:
# *     Andreas P. Cuny - initial API and implementation
# *****************************************************************************/

from pyIMD.imd import InertialMassDetermination

# Create the inertial mass determination object
imd = InertialMassDetermination()

# Create a config file for the project / experiment to analyze using default values.␣
↪Note non default parameters can be
# added as optional arguments for e.g. spring_constant = 5.
file_path1 = "/pyIMD/examples/data/show_case/20170712_RSN_3_B"
file_path2 = "/pyIMD/examples/data/show_case/20170712_RSN_3_A"
file_path3 = "/pyIMD/examples/data/show_case/20170712_RSN_3_A_long_term.tdms"
imd.create_pyimd_project(file_path1, file_path2, file_path3, '\t', 23, 'PLL', figure_
↪width=5.4, figure_height=9.35,
                         initial_parameter_guess=[73.0, 5.2, 0.0, 0.0], upper_
↪parameter_bounds=[100.0, 7.0, 3.0, 3.0],
```

(continues on next page)

(continued from previous page)

```
                          spring_constant=8.0, cell_position=9.5, cantilever_
↪length=100.0, figure_format='pdf')

# Print the config file to the console to check if all parameters are set correctly␣
↪before starting the calculation.
imd.print_pyimd_project()

# If one needs to change a parameter on the fly just type: imd.settings.<parameter_
↪key> = value as eg.
# imd.settings.figure_resolution_dpi = 300. Note: Just hit imd.settings. + TAB to get␣
↪automatically a list of all
# available <parameter_keys>

# To enter all the parameters one can also start the settings user interface and␣
↪enter all the parameter values there.
# imd.show_settings_dialog()

# Run the inertial mass determination
imd.run_intertial_mass_determination()

# Save the config file for the project / experiment for documentation purpose or to␣
↪re-run with different /
# same parameter later
imd.save_pyimd_project("/pyIMD/examples/data/show_case/pyIMDProjectName.xml")

# To load an existing project type
imd.load_pyimd_project("/pyIMD/examples/data/show_case/pyIMDProjectName.xml")
# change a parameter i.e
imd.settings.figure_format = 'png'
# and run again
imd.run_intertial_mass_determination()
```

2.2 pyIMD example IPython/Jupyter notebook

[11]:
```
from pyIMD.imd import InertialMassDetermination
```

[12]:
```
imd = InertialMassDetermination()
```

```
2019-03-24 22:05:11 - pyIMD.imd - Object constructed successfully
```

[13]:
```
file_path1 = "../data/pll/20170712_RSN_3_B"
file_path2 = "../data/pll/20170712_RSN_3_A"
file_path3 = "../data/pll/20170712_RSN_3_A_long_term.tdms"
```

[14]:
```
imd.create_pyimd_project(file_path1, file_path2, file_path3, '\t', 23, 'PLL', figure_
↪height=40,
                         figure_unit='cm',initial_parameter_guess=[70.0, 2, 0.0, 0.0],
                         upper_parameter_bounds=[100.0, 5, 3.0, 3.0], spring_
↪constant=1.05,
                         cell_position=3.23, cantilever_length=59.84, figure_format=
↪'png',
                         correct_for_frequency_offset=True, frequency_offset_n_
↪measurements_used=15)
```

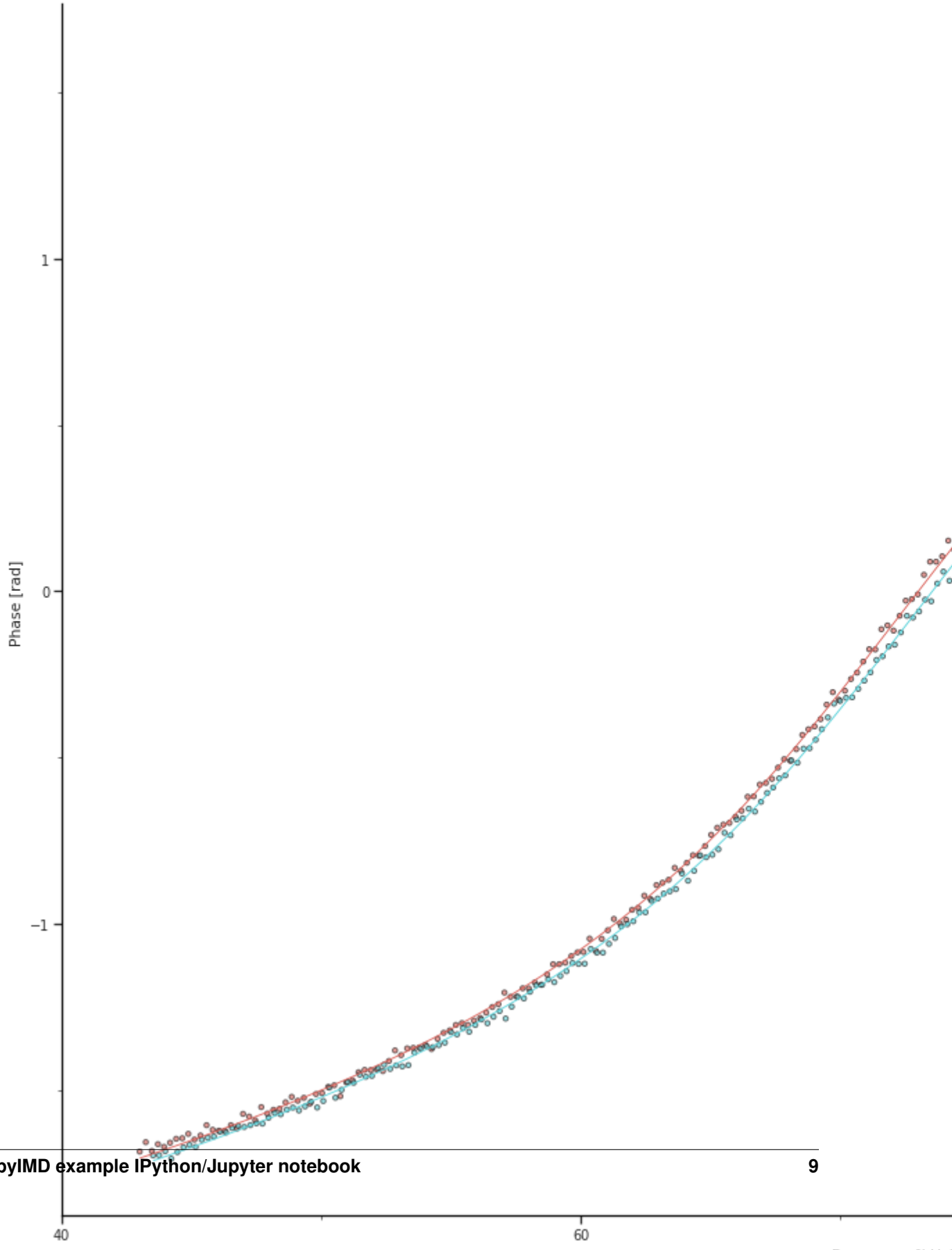
[15]:
```
imd.run_inertial_mass_determination()
```

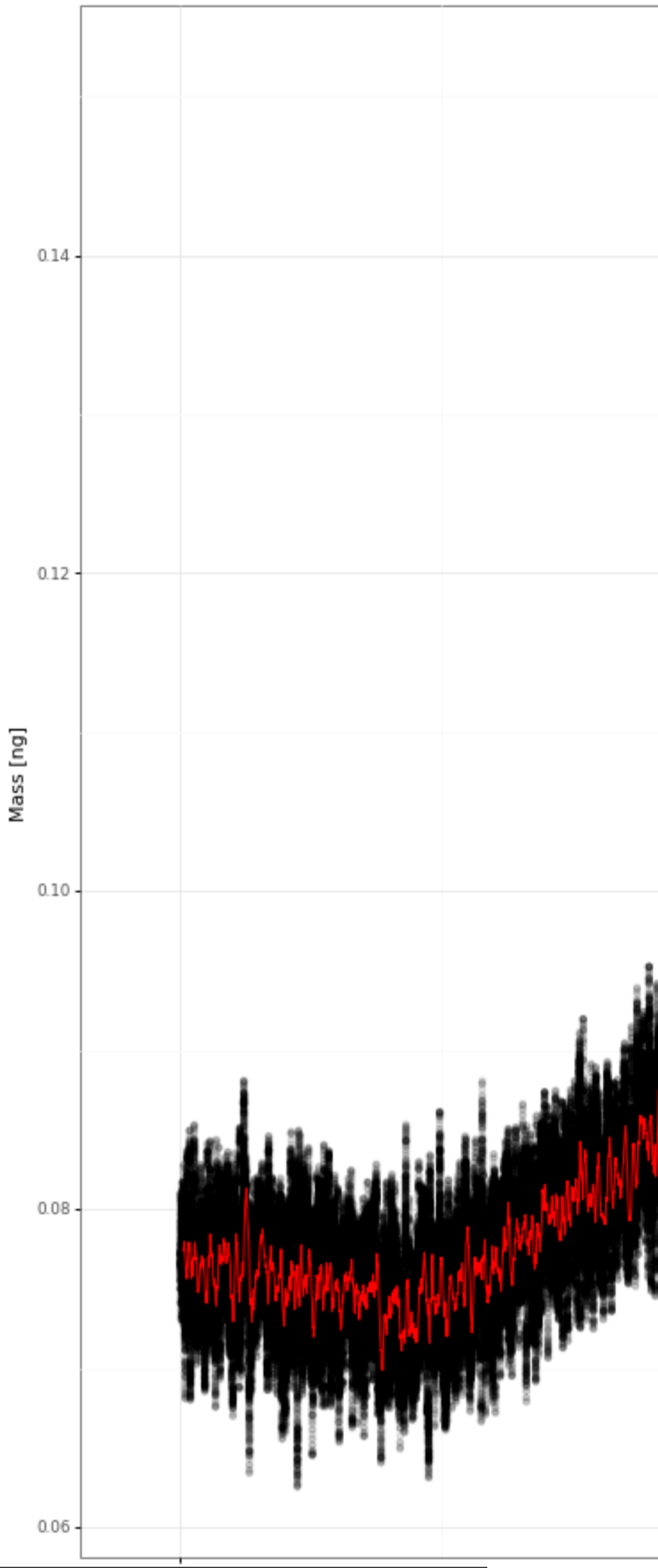
```
2019-03-24 22:05:12 - pyIMD.imd - Start reading all files
2019-03-24 22:06:32 - pyIMD.imd - Done reading all files
2019-03-24 22:06:32 - pyIMD.imd - Done converting units
2019-03-24 22:06:33 - pyIMD.imd - Done with pre start no cell resonance frequency␣
↪calculation
2019-03-24 22:06:34 - pyIMD.imd - Done with pre start with cell resonance frequency␣
↪calculation
2019-03-24 22:06:36 - pyIMD.imd - Done with pre start frequency shift figure␣
↪generation
2019-03-24 22:06:36 - pyIMD.imd - Offset calculation result: -0.10173519457150339
100%|| 692625/692625 [00:21<00:00, 32255.79it/s]
2019-03-24 22:06:57 - pyIMD.imd - Start writing figure to disk
2019-03-24 22:08:50 - pyIMD.imd - Done writing figure to disk
2019-03-24 22:08:50 - pyIMD.imd - Start writing data to disk
2019-03-24 22:08:54 - pyIMD.imd - Done writing data to disk
2019-03-24 22:08:54 - pyIMD.imd - Done with all calculations
```

The **run_intertial_mass_determination()** method generates automatically figures of the curve fitting for the per pre experiment data with and without a cell attached to the cantilever. A combined figure illustrating the shift in the phase response and the function fits and the resulting calculated cell mass of the long term measurement data.

2.2.1 Show resulting plots

- Shift in the phase response with and without cell attached to the cantilever and the corresponding function fit





- Final result of the mass calculation for the long term data set

[ ]:

2.3 pyIMD tutorial with user interface

Before starting, make sure pyIMD is *installed*

This tutorial provides a simple example with a test dataset, teaching step by step how to:

- create a pyIMD project
- calculate the mass form the measured data

The layout of the following windows and the paths are set for windows and might differ for Mac or Unix. First, lets have a look at the input data. The typical data set consists of 3 files: 1) a sweep file of the cantilever WITHOUT cell (text file with multi-line header) 2) a sweep file of the cantilever WITH cell (text file with multi-line header) and 3) the actual (long-term) measurement file, which is either a text file or TDMS file (lab-view specific file type). A typical time resolution is 10 ms for the data acquisition so these files can be quite large. **Fig. 1** visualizes the data input which can be found as example data set for download and testing.

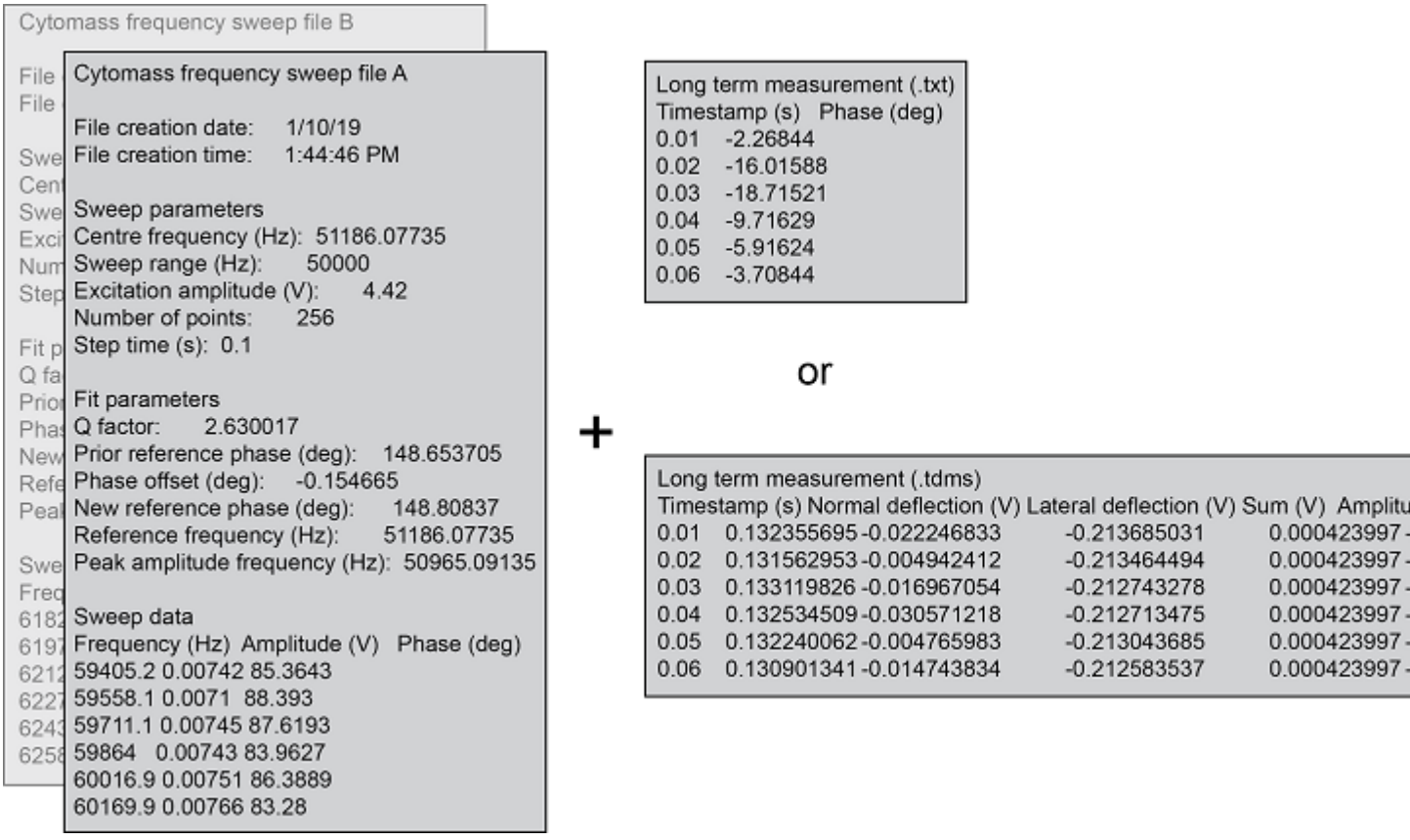


Fig. 1: **Figure 1**: Data format for pyIMD. pyIMD supports data from picobalance device controllers. (Cytomass and Nanonis)

The example pyIMD script section demonstrates how a pyIMD project is created on the console:

```
from pyIMD.imd import InertialMassDetermination

# Create the inertial mass determination object
imd = InertialMassDetermination()

# Create a config file for the project / experiment to analyze using default values.␣
↪Note non default parameters can be
# added as optional arguments for e.g. cell_position = 9.5.
file_path1 = "/pyIMD/examples/data/show_case/0190110_ShowCase_PLL_B.txt"
file_path2 = "/pyIMD/examples/data/show_case/20190110_ShowCase_PLL_A.txt"
file_path3 = "/pyIMD/examples/data/show_case/20190110_ShowCase_PLL_LongTerm.txt"
imd.create_pyimd_project(file_path1, file_path2, file_path3, '\t', 23, 'PLL', figure_
↪width=16.5, figure_height=20,
                         initial_parameter_guess=[60.0, 2.0, 0.0, 0.0], cell_
↪position=9.5, figure_format='pdf')
```

When using pyIMD through its user interface (UI) in the stand alone mode, the pyIMD project is created in exact the same way in the background. Yet, the user does not need to take care to type the paths or arguments correctly as all the input entered trough the UI will be validated automatically. **Fig. 2** shows the main window and the settings window of the pyIMD application. A new pyIMD project is created by selecting the three data files required for the calculation from a directory (3). Next, it needs to be declared which measurement each file contains and what the measurement mode is (5). Using the menu (1), opens the settings dialog and lets you determine all project related parameters such as the names of the output figures. After all settings are set, the mass calculation is started with (6).

The tools menu in **Fig. 2 (7)** allows for data concatenation from multiple files into a single one, in case the data was acquired with the Nanonis data logger. The resulting file can then be loaded as mentioned above along with the before and after cell attachment file.

```
# Run the inertial mass determination
imd.run_inertial_mass_determination()
```

The console (8) logs all actions performed with the UI and indicates when all calculations are done. The results can be viewed in the results tab (2), where as all the output figures are listed as well as the data can be inspected.

The first output created by pyIMD are control figures visualizing the fit of the cantilevers phase response is shown for the case with and without cell (**Fig. 3**). The shift towards lower frequencies can be clearly seen, when the cell is attached. Moreover, the Q-factor changes and therefore the slope of the response curve. If the fits are not fitting the raw data the parameter 'initial_parameter_guess', 'lower_parameter_bounds', 'upper_parameter_bounds' need to be adjusted in the settings dialog.

The analysis output by the software is shown in **Fig. 4**, the exemplary data for a mammalian cell is provided for download. The evolution of mass vs time is shown for a time span of 20 min. The mass data was acquired every 10 ms (data shown in black), overlaid in red is the rolling mean with a window of 1000 (adjustable parameter 'rolling_window_size'). Images taken every 3 min over the observed time span, we see on average a steady increase of the cell mass, the spring constant is 8 N/m (adjustable parameter 'sprint_constant'). The position of the cell projected along the long axis of the cantilever was 9.5 um (adjustable parameter, 'cell_position') and did not change, which is of importance for the current use of the software.

The project can either be re-run with different parameters, to i.e. improve the function fits or be saved using the menu (**Fig. 2**, (1)).

```
# save a pyIMD project
imd.save_pyimd_project("/pyIMD/examples/data/show_case/pyIMDShowCaseProject.xml")
```

A previously saved project can be loaded again at a later time from the menu (**Fig. 2**, (1))or also from the command line without the user interface:

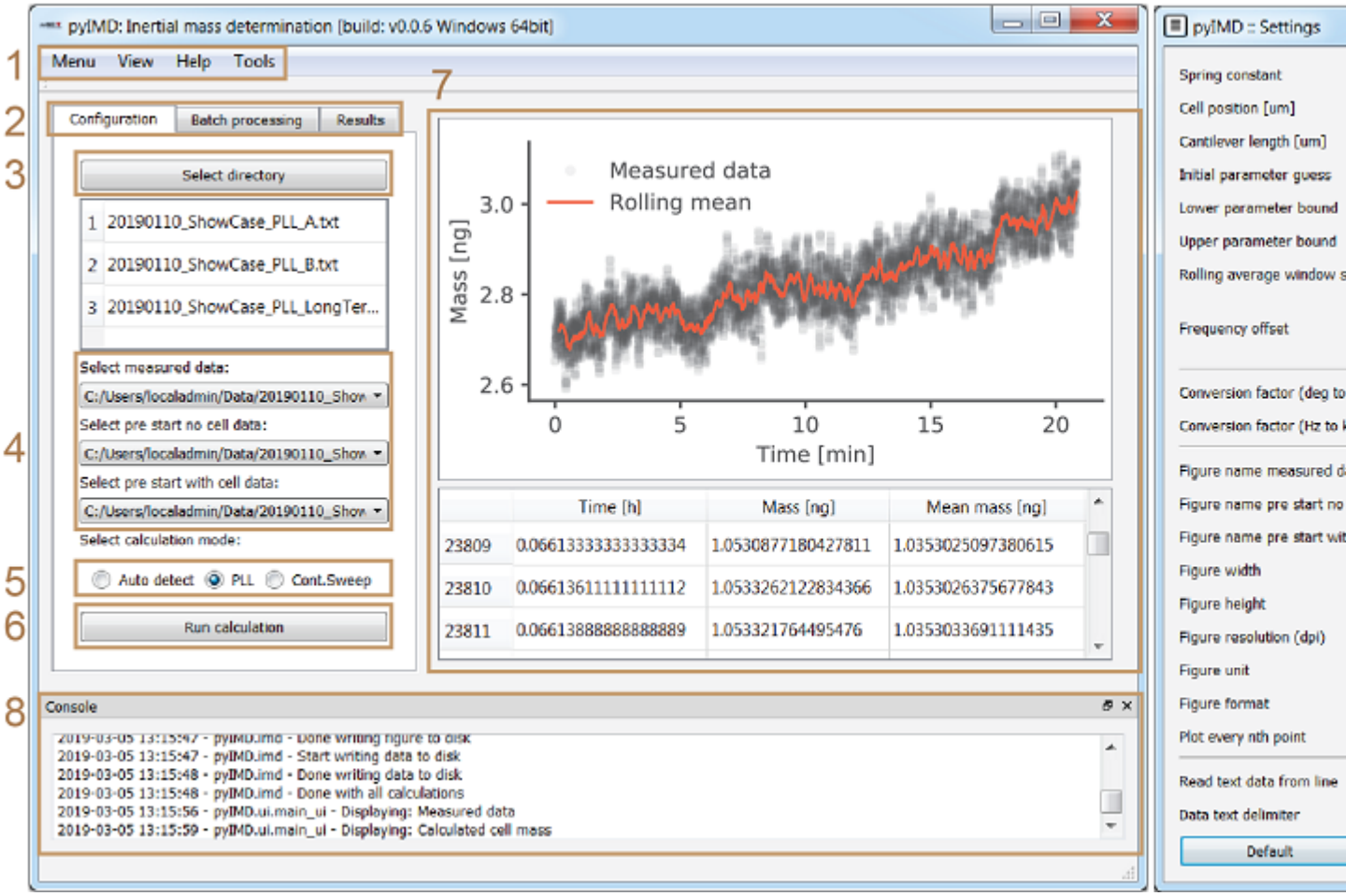


Fig. 2: **Figure 2**: . Through the menu bar (1) the pyIMD project can be loaded, saved, and the settings and parameter dialog opened (shown at the right-hand side). The help menu contains the software documentation, the quick help (also shown during startup), change log and information about the software dependencies and authors. The tabs (2) allow to switch between single calculation, batch calculation, and results. After all calculations are done the results tab is enabled and shows the latest result figures and data table in (7). (3) Creates a new pyIMD project while selecting at least three data files required for the calculation. After the files have been selected, it needs to be declared which type of data they contain, i.e. whether it is the single reference measurement of the cantilever without cell or the reference measurement with cell or the time resolved data (4). (5) Sets the acquisition mode that was used to collect the experimental long-term data. (6) Starts the mass calculation. If the batch processing is selected in (2) one or multiple pyIMD project files can be loaded, which will be run sequentially in different threads. With the settings dialog on the right, all the required parameters needed for the calculation as well as the output file formats or file names are set. The user input is validated live and if a parameter of a wrong type is entered, the input field turns yellow to notify the user of the mistake. When the user has inserted all necessary parameters correctly and started the calculation, a process is reported in the info window (8), and finally the result is shown in the main window.

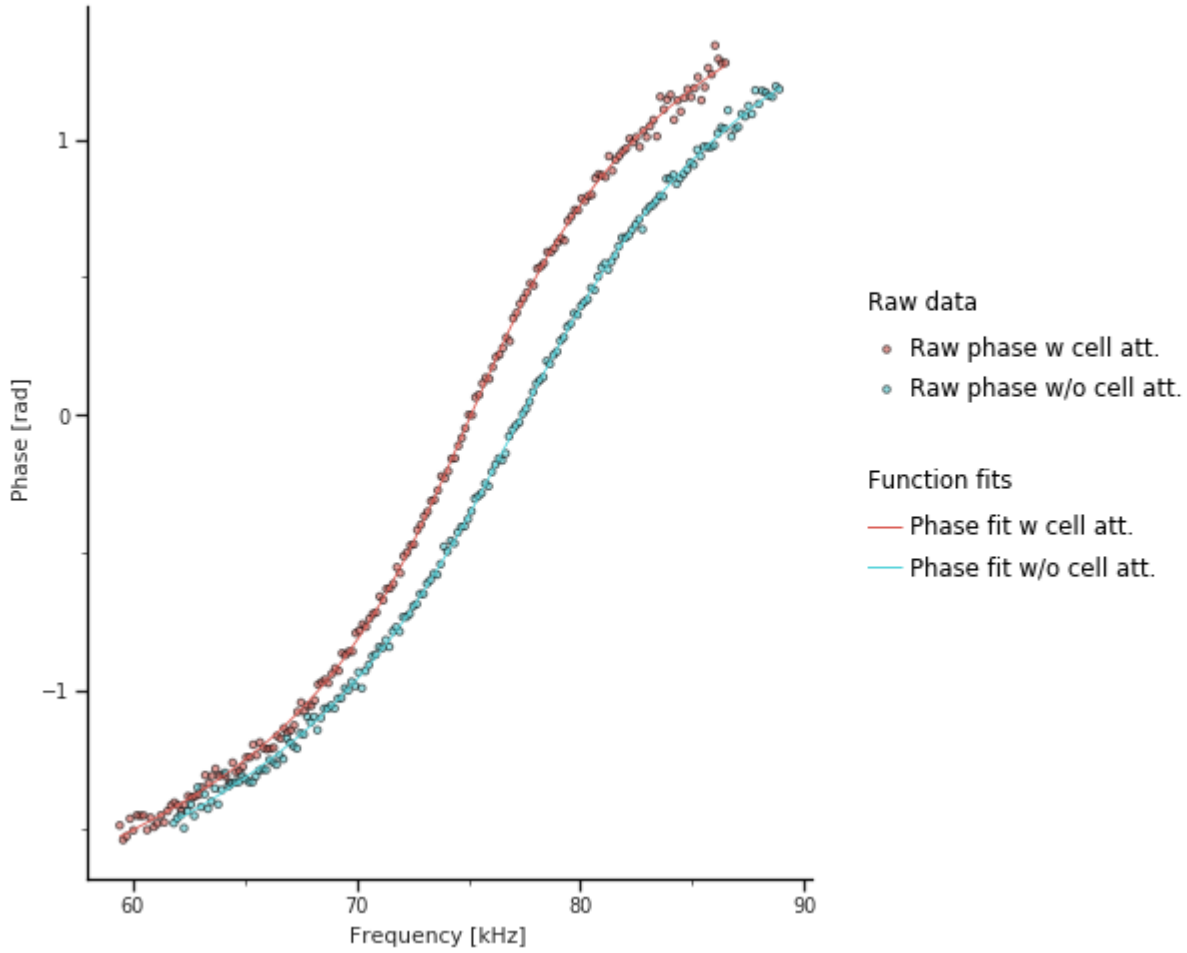


Fig. 3: **Figure 3**: Frequency vs cantilever phase response

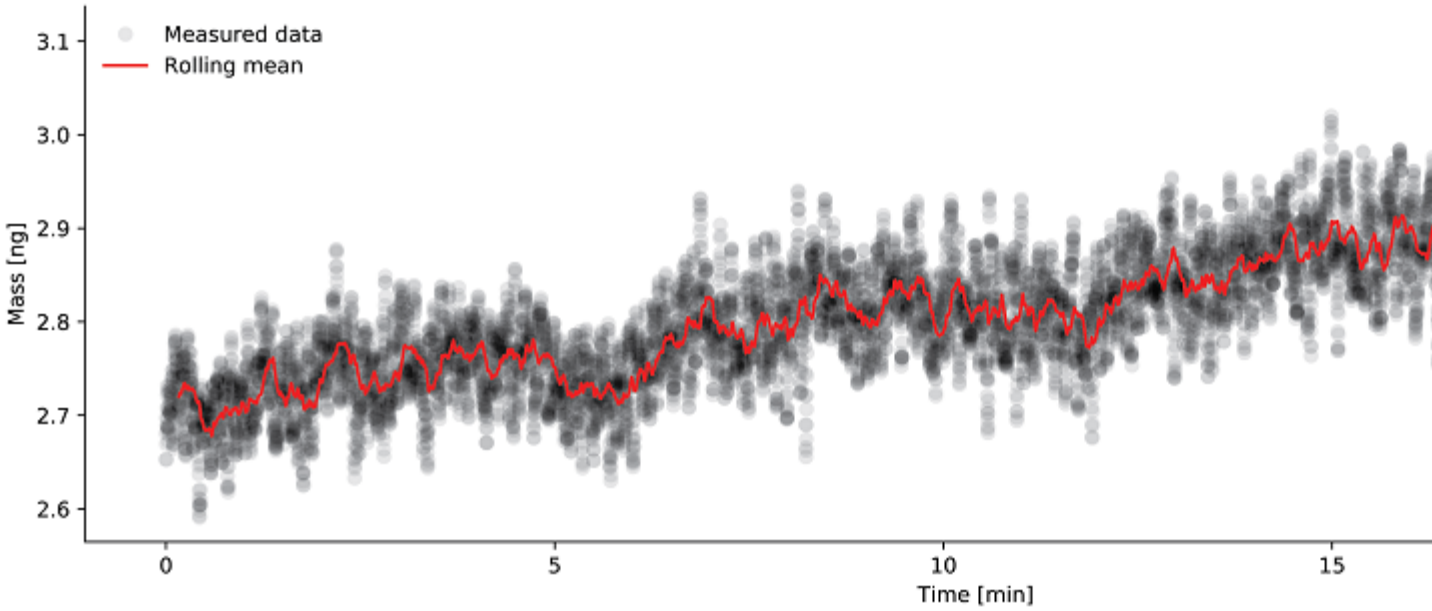


Fig. 4: **Figure 4**: Evolution of mass over time

```
# load a pyIMD project
imd.load_pyimd_project("/pyIMD/examples/data/show_case/pyIMDShowCaseProject.xml")
```

2.4 pyIMD example script Nanonis long term

This example script demonstrates the command line interface use with **pyIMD** and Nanonis long term type of data:

```
# /*******************************************************************************
# * Copyright © 2018-2019, ETH Zurich, D-BSSE, Andreas P. Cuny & Gotthold Fläschner
# * All rights reserved. This program and the accompanying materials
# * are made available under the terms of the GNU Public License v3.0
# * which accompanies this distribution, and is available at
# * http://www.gnu.org/licenses/gpl
# *
# * Contributors:
# *     Andreas P. Cuny - initial API and implementation
# *******************************************************************************/

from pyIMD.imd import InertialMassDetermination

# Create the inertial mass determination object
imd = InertialMassDetermination()

# Create a config file for the project / experiment to analyze using default values.␣
↪Note non default parameters can be
# added as optional arguments for e.g. spring_constant = 5.
file_path1 = "/pyIMD/examples/data/nanonis_long_term/20190510_LC_05_B001.dat"
file_path2 = "/pyIMD/examples/data/nanonis_long_term/20190510_LC_05_A001.dat"
file_path3 = "/pyIMD/examples/data/nanonis_long_term/20190510_LC_05_Longterm001.dat"
imd.create_pyimd_project(file_path1, file_path2, file_path3, '\t', 23, 'PLL', figure_
↪width=5.4, figure_height=9.35,
```

(continues on next page)

(continued from previous page)

```
                        initial_parameter_guess=[73.0, 5.2, 0.0, 0.0], upper_
↪parameter_bounds=[100.0, 8.0, 3.0, 3.0],
                        spring_constant=8.0, cell_position=10, cantilever_length=100.
↪0)

# Print the config file to the console to check if all parameters are set correctly␣
↪before starting the calculation.
imd.print_pyimd_project()

# Save the config file for the project / experiment for documentation purpose or to␣
↪re-run with different /
# same parameter later
imd.save_pyimd_project("/pyIMD/examples/data/nanonis_long_term/pyIMDProjectName.xml")

# Run the inertial mass determination
imd.run_intertial_mass_determination()
```

CHAPTER 3

API Reference

3.1 analysis

pyIMD.analysis.curve_fit.**fit_function**(*x*, *fn*, *q*, *a*, *b*)

Defines the phase response of a damped harmonic oscillator (i.e. the cantilever with or without cell). It is called from calculate_resonance_frequencies, to be fitted to the data primarily to extract the natural resonance frequency.

Parameters

- **x** (*float*) – Frequency (the independent variable of that function)
- **fn** (*float*) – Natural resonance frequency
- **q** (*float*) – Q factor (losses)
- **a** (*float*) – Linear factor accounting for a linear background
- **b** (*float*) – Constant Phase-Offset

Returns Returns the phase.

Return type phase (*float*)

pyIMD.analysis.calculations.**calculate_mass**(*spring_constant*, *res_freq_after_cell_load*, *res_freq_before_cell_load*)

Calculates the mass given the spring constant of the cantilever and the resonance frequency without and with cell attached to the cantilever.

Args:

spring_constant (*float*): Stiffness of the cantilever [in N/m] res_freq_after_cell_load (*float*): Resonance frequency of the cantilever AFTER the cell is picked up, at time point t [in kHz] res_freq_before_cell_load (*float*): Resonance frequency of the cantilever BEFORE the cell is picked up [in kHz]

Returns: mass (*float*): Returns data as float, which is the mass at time point t.

pyIMD.analysis.calculations.**calculate_position_correction**(*cell_position*, *cantilever_length*)

Calculates the correction factor with which the measured mass needs to be multiplied to get all the mass present on the cantilever. This is needed as the cantilever is differently sensitive to mass, depending on the location where this mass is attached. The measurements are performed with the first mode of vibration, which is described by the factor kL = 1.875. For higher modes, different would be used (4.694 for the second , 7.855 for the third etc.)

Parameters

- **cell_position** (*float*) – Cell position from the free end of the cantilever [in micrometer]
- **cantilever_length** (*float*) – Cantilever length [in micrometer]

Returns Returns a double which is the correction factor.

Return type correction_factor (*float*)

pyIMD.analysis.calculations.**calculate_resonance_frequencies**(*frequency_array*, *phase_array*, *initial_param_guess*, *lower_param_bounds*, *upper_param_bounds*)

Calculate_resonance_frequencies calculates the resonance frequency from input frequency and phase array. It does so via fitting the phase response of a harmonic oscillator (defined in pyIMD.analysis.curve_fit). The first fit parameter of the fit parameter array is the resonance frequency.

Parameters

- **frequency_array** (*float array*) – Array of frequencies [in kHz]
- **phase_array** (*float array*) – Array of phase [in Rad]
- **initial_param_guess** (*float*) – Initial parameter guess (1x4 array)
- **lower_param_bounds** (*float*) – Lower bounds (1x4 array)
- **upper_param_bounds** (*float*) – Upper bounds (1x4 array)

Returns Resonance frequency [in kHz]

Return type resonance_frequency (*float*)

Returns

Curve fit parameters curve_fit_parameter[0] := Q factor (losses)

curve_fit_parameter[1] := Linear factor accounting for a linear background

curve_fit_parameter[2] := Offset of the background

Return type curve_fit_parameter (*float array*)

3.2 configuration

class pyIMD.configuration.config.**Settings**

Bases: object

Settings initialization with default parameter values stored in pyIMD.configuration.defaults

calculation_mode
Parameter defining the calculation mode.

Parameters **mode** (*str*) – The calculation mode. PLL, Cont.Sweep or Auto.

cantilever_length
Parameter defining the cantilever length.

Parameters **cantilever_length** (*float*) – Cantilever length in microns.

cell_position
Parameter defining the cell position offset.

Parameters **cell_position** (*float*) – Cell position offset in microns from the free end of the cantilever.

conversion_factor_deg_to_rad
Parameter defining the data conversion factor from degrees to radians.

Parameters **factor** (*float*) – Data conversion factor from degrees to radians.

conversion_factor_hz_to_khz
Parameter defining the data conversion factor from Hz to kHz.

Parameters **factor** (*float*) – Data conversion factor from hertz to kilo hertz.

correct_for_frequency_offset
Parameter defining if PLL measurement data should be corrected for a potential frequency offset between the frequency pre start measured after a cell is attached to the cantilever and the actual measurement.

Parameters **correction** (*boolean*) – Boolean. False by default. True runs in the PLL case a frequency offset correction.

figure_format
Parameter defining the result figure(s) format.

Parameters **unit** (*str*) – Pdf and png ar currently supported.

figure_height
Parameter defining the result figure(s) height.

Parameters **height** (*float*) – Figure height in the unit specified.

figure_name_measured_data
Parameter defining the figure name for resulting calculated mass of the measured data.

Parameters **name** (*str*) – Figure name for resulting calculated mass of the measured data.

figure_name_pre_start_no_cell
Parameter defining the figure name for the function fit pre start no cell.

Parameters **name** (*str*) – Figure name for the function fit pre start no cell.

figure_name_pre_start_with_cell
Parameter defining the figure name for the function fit pre start with cell.

Parameters **name** (*str*) – Figure name for the function fit pre start with cell.

figure_plot_every_nth_point
Parameter defining how many data points are used for visualization. For very large data sets a number > 1 could increase the readability of the figure and lower the file size.

Parameters **nth_point** (*int*) – Pdf and png ar currently supported.

figure_resolution_dpi
Parameter defining the result figure(s) resolution in dpi.

Parameters **resolution** (*int*) – Figure resolution. E.g. 72, 150, 300.

figure_units
Parameter defining the result figure(s) unit.

Parameters **unit** (*str*) – Figure unit. E.g. mm, cm, in.

figure_width
Parameter defining the result figure(s) width.

Parameters **width** (*float*) – Figure width in the unit specified.

frequency_offset
Parameter defining the frequency offset.

Parameters **freq_offset** (*int*) – Frequency offset with which the measurement data will be corrected with.

frequency_offset_mode
Parameter defining the mode to be used to calculate the frequency offset. This applies only to in PLL recorded data.

Parameters **mode** (*str*) – Mode for frequency offset calculation. Either Auto or Manual.

frequency_offset_n_measurements_used
Parameter defining how many data points of the measurement should be used to calculate the average frequency offset.

Parameters **n_measurements** (*int*) – Number of measurement data points to be used to calculate the average freq. offset.

initial_parameter_guess
Parameter defining the initial parameter guess.

Parameters **array** (*list*) – Initial parameter guess.

lower_parameter_bounds
Parameter defining the lower parameter bounds.

Parameters **array** (*list*) – Lower parameter bounds.

measurements_path
Parameter defining the path to the measurement file.

Parameters **path** (*str*) – Path to the measurement file. (.tdms or .txt file).

new_pyimd_project(*pre_start_no_cell_path*, *pre_start_with_cell_path*, *measurements_path*, *text_data_delimiter*, *read_text_data_from_line*, *calculation_mode*, ***kwargs*)
Create a new pyIMD project with the following arguments. Two modes enable the analysis of different experimental setups. PLL mode and Cont.Sweep mode. For more information please read the documentation.

Parameters

- **pre_start_no_cell_path** (*str*) – File path + file name of initial frequency shift measurement before cell attachment (txt file).
- **pre_start_with_cell_path** (*str*) – File path + file name of initial frequency shift measurement after cell attachment (txt file).
- **measurements_path** (*str*) – File path + file name of the actual measurement (tdms file (default) or txt file).
- **text_data_delimiter** (*str*) – Text file data delimiter i.e ‘ ‘ for tab delimited or ‘,’ for comma separated data.

- **read_text_data_from_line** (*int*) – Line number from which data of pre start measurements should be read. Typically the first few lines contain header information and no data.
- **calculation_mode** (*str*) – PLL := phase lock loops mode Cont.Sweep := sweep mode Auto := Auto detection of the mode (experimental)

Keyword Arguments

- **figure_width** (*float*) – Width of result figures
- **figure_height** (*float*) – Height of result figures
- **figure_units** (*str*) – Figure units i.e cm, inch
- **figure_format** (*str*) – Figure format i.e png or pdf
- **figure_resolution_dpi** (*int*) – Resolution of result figures in dpi
- **figure_name_pre_start_no_cell** (*str*) – Figure name of function fit for pre start with no cell loaded data
- **figure_name_pre_start_with_cell** (*str*) – Figure name of function fit for pre start with cell loaded data
- **figure_name_measured_data** (*str*) – Figure name of the resulting mass of the measured data
- **figure_plot_every_nth_point** (*'int'*) – Parameter defining how many data points will be plotted. For large data stets to increase readability and reducing file size.
- **conversion_factor_hz_to_khz** (*float*) – Conversion factor to convert from hertz to kilo hertz
- **conversion_factor_deg_to_rad** (*float*) – Conversion factor to convert from degrees to radian
- **spring_constant** (*float*) – Spring constant value of the cantilever
- **initial_parameter_guess** (*list*) – Initial parameter guess
- **lower_parameter_bounds** (*list*) – Lower parameter bounds
- **upper_parameter_bounds** (*list*) – Upper parameter bounds
- **rolling_window_size** (*'int'*) – Window size for calculating the rolling average.
- **correct_for_frequency_offset** (*'bool'*) – Correct for potential frequency offset during PLL mode.
- **frequency_offset_mode** (*'str'*) – Frequency offset correction mode (Auto or Manual)
- **frequency_offset_n_measurements_used** (*'int'*) – Number of measurement data points to be used for automatic frequency offset correction
- **frequency_offset** (*'float'*) – Frequency offset either set manually or calculated automatically
- **cantilever_length** (*float*) – Cantilever length in microns
- **cell_position** (*float*) – Cell position offset from cantilever tip in microns
- **project_folder_path** (*str*) – Path to project data files. Also used to store pyIMD results such as data and figures.

pre_start_no_cell_path
Parameter defining the path to the pre start no cell file.

Parameters **path** (*str*) – Path to the pre start no cell file (.txt).

pre_start_with_cell_path
Parameter defining the path to the pre start with cell file.

Parameters **path** (*str*) – Path to the pre start with cell file (.txt).

project_folder_path
Parameter defining the path to the files.

Parameters **path** (*str*) – The path to the files.

read_pyimd_project (*file_path*)
Read a pre defined pyIMD project form a XML file from disk.

Parameters **file_path** (*str*) –

Returns String reporting the success of failure of loading a pyIMD project.

Return type status (*str*)

read_text_data_from_line
Parameter defining the length of the header inside the initial sweep files. From this line number the data will be imported.

Parameters **line_number** (*int*) – Line number from where on to read the data from.

rolling_window_size
Parameter defining the window size of the rolling window applied to the data for visualizing the trend.

Parameters **window_size** (*int*) – Rolling window size.

selected_files
Parameter defining the selected files for calculation.

Parameters **files** (*list*) – Selected files for calculation.

spring_constant
Parameter defining the spring constant of the cantilever.

Parameters **spring_constant** (*float*) – Spring constant of the cantilever.

text_data_delimiter
Parameter defining the text file data delimiter.

Parameters **delimiter** (*str*) – Text file data delimiter i.e ‘ ‘ for tab delimited or ‘,’ for comma separated data.

upper_parameter_bounds
Parameter defining the upper parameter bounds.

Parameters **array** (*list*) – Upper parameter bounds.

write_pyimd_project (*file_path*)
Write the current pyIMD project as XML file to disk. :param file_path: :type file_path: *str*

Returns String reporting the success of failure of loading a pyIMD project.

Return type status (*str*)

3.3 io

pyIMD.io.read_from_disk.**read_from_dat**(*file*, *delimiter*)
Method to read data from dat files (i.e from Nanonis software).

Parameters

- **file** (*str*) – File path + file name.
- **delimiter** (*str*) – Delimiter used in the data file to separate columns

Returns Returns data structured in a pandas data frame.

Return type data (*pandas data frame*)

pyIMD.io.read_from_disk.**read_from_file**(*file*, *delimiter*)
Method to read data from a file.

Parameters

- **file** (*str*) – File path + file name to a .TDMS or .txt file.
- **delimiter** (*str*) – Delimiter used in the data file to separate columns

Returns Returns data structured in a pandas data frame.

Return type data (*pandas data frame*)

pyIMD.io.read_from_disk.**read_from_tdms**(*file*)
Method to read data from National Instruments technical data management streaming files (TDMS).

Parameters **file** (*str*) – File path + file name string.

Returns Returns data structured in a pandas data frame.

Return type data (*pandas data frame*)

pyIMD.io.read_from_disk.**read_from_text**(*file*, *delimiter*, *read_from_row*)
Method to read data from text files.

Parameters

- **file** (*str*) – File path + file name.
- **delimiter** (*str*) – Delimiter used in the data file to separate columns
- **read_from_row** (*int*) – Row number from where to start reading data to be able to skip heading text rows. Make sure that you keep the Frequency, Amplitude and Phase headers.

Returns Returns data structured in a pandas data frame.

Return type data (*pandas data frame*)

pyIMD.io.write_to_disk.**write_concat_data**(*directory*, *delimiter*, *time_interval*)
Method to write concatenate data from single dat files (i.e data logger from Nanonis software).

Parameters

- **directory** (*str*) – Directory containing files to concatenate.
- **delimiter** (*str*) – Delimiter to be used in the data file to separate columns.
- **time_interval** (*int*) – Measurement time interval in milliseconds.

Returns Writes concatenated data to single .csv file.

Return type file (*void*)

pyIMD.io.write_to_disk.**write_to_disk_as**(*file_format*, *plot_object*, *file*, ***kwargs*)
Method to write figures in various file formats

Parameters

- **file_format** (*str*) – File format identifier i.e. png or pdf
- **plot_object** (*ggplot object*) – ggplot object
- **file** (*str*) – File path + file name of the figure to save

Keyword Arguments

- **width** (*int*) – Figure width (optional)
- **height** (*int*) – Figure height (optional)
- **units** (*'str`*) – Figure units (optional) 'in', 'mm' or 'cm'
- **resolution** (*int*) – Figure resolution in dots per inch [dpi] (optional)

Returns Writes figure to disk in the respective file format

Return type file (*void*)

pyIMD.io.write_to_disk.**write_to_pdf**(*plot_object*, *file*, ***kwargs*)
Method to write figures in pdf format to current directory

Parameters

- **plot_object** (*ggplot object*) – ggplot object
- **file** (*str*) – File path + file name of figure to save

Keyword Arguments

- **width** (*int*) – Figure width (optional)
- **height** (*int*) – Figure height (optional)
- **units** (*'str`*) – Figure units (optional) 'in', 'mm' or 'cm'
- **resolution** (*int*) – Figure resolution in dots per inch [dpi] (optional)

Returns Writes figure to disk as pdf

Return type pdf file (*void*)

pyIMD.io.write_to_disk.**write_to_png**(*plot_object*, *file*, ***kwargs*)
Method to write figures in png format to current directory

Parameters

- **plot_object** (*ggplot obj*) – ggplot object
- **file** (*str*) – File path + file name of the figure to save

Keyword Arguments

- **width** (*int*) – Figure width (optional)
- **height** (*int*) – Figure height (optional)
- **units** (*str*) – Figure units (optional) 'in', 'mm' or 'cm'
- **resolution** (*int*) – Figure resolution in dots per inch [dpi] (optional)

Returns Writes figure to disk as png

Return type png file (*void*)

3.4 plotting

pyIMD.plotting.figures.**create_montage_array**(*img_stack*, *size*)

Creates an image montage of a 3D numpy array with the shape [image frames, image row, image col] for the specified size.

Parameters

- **img_stack** (*3D numpy array*) – 3D numpy image array [image row, image col, image frames].
- **size** (*numpy array*) – Array specifying the amount of images displayed in the montage per row and column. If one argument is replaced with np.nan, the needed amount of rows or columns is calculated automatically. E. g. [5, np.nan]

Returns 2D numpy array with the image montage

Return type montage (*2D numpy array*)

pyIMD.plotting.figures.**get_montage_array_size**(*size*, *image_row_count*, *image_col_count*, *frame_count*)

Calculates the final size of a numpy array needed to hold a the number of specified image frames given the row and column count of the final array.

Parameters

- **size** (*numpy array*) – Array specifying the amount of images displayed in the montage per row and column. If one argument is replaced with np.nan, the needed amount of rows or columns is calculated automatically. E. g. [5, np.nan]
- **image_row_count** (*int*) – Number of rows per image
- **image_col_count** (*int*) – Number of columns per image
- **frame_count** (*int*) – Number of image frames in the stack

Returns Array with the number of rows and columns needed in the montage array for the images

Return type montage_size (*numpy array*)

pyIMD.plotting.figures.**plot_fitting**(*x*, *y*, *resonance_frequency*, *parameter*)

Plots the phase response and the corresponding fit of the harmonic damped oscillator.

Parameters

- **x** (*float array*) – X coordinates (frequency in kHz)
- **y** (*float array*) – Y coordinates (phase in radians)
- **resonance_frequency** (*float array*) – Resonance frequency given by the fit of x and y
- **parameter** (*float array*) – Others parameters of function fit (Q factor, offset, linear background)

Returns Returns a ggplot object

Return type p (*ggplot object*)

pyIMD.plotting.figures.**plot_mass**(*calculated_cell_mass*, *plot_every_nth_point*)

Plots the resulting mass

Parameters

- **calculated_cell_mass** (*pandas data frame*) – Pandas data frame [Nx3] with time and calculated cell mass and rolling mean averaged cell mass

- **plot_every_nth_point** (*int*) – If 1 all data points are plotted. Otherwise every nth data point is used for plotting.

Returns Returns a ggplot plot object

Return type p (*ggplot object*)

pyIMD.plotting.figures.**plot_response_shift**(*x*, *y*, *resonance_frequency_without*, *parameter_without*, *xx*, *yy*, *resonance_frequency_with*, *parameter*)

Plots the phase response of pre start data without and with cell attached to cantilever with the respective function fit.

Parameters

- **x** (*float array*) – X coordinates w/o cell (frequency in kHz)
- **y** (*float array*) – Y coordinates w/o cell (phase in radians)
- **xx** (*float array*) – X coordinates w/ cell(frequency in kHz)
- **yy** (*float array*) – Y coordinates w/ cell (phase in radians)
- **resonance_frequency_without** (*float array*) – Resonance frequency given by the fit of x and y w/o cell
- **resonance_frequency_with** (*float array*) – Resonance frequency given by the fit of x and y w/ cell
- **parameter** (*float array*) – Others parameters of function fit (Q factor, offset, linear background) w/o cell
- **parameter_without** (*float array*) – Others parameters of function fit (Q factor, offset, linear background) w/ cell

Returns Returns a ggplot object

Return type p (*ggplot object*)

3.5 ui

class pyIMD.ui.main_ui.**IMDWindow**

Bases: PyQt5.QtWidgets.QMainWindow

Implementation of the pyIMD main user interface window.

closeEvent(*event*)

Application close event override of QMainWindow closeEvent

Parameters **event** (*QCloseEvent*) – A QCloseEvent

Returns O when process finished correctly otherwise >0

Return type status_code (*int*)

close_application(*event*)

Opens a message box to handle program exit properly asking the user if the project should be saved first.

Parameters **event** (*QCloseEvent*) – A QCloseEvent

Returns 0 when process finished correctly, otherwise >0

Return type status_code (*int*)

static get_logger_object(*name*)
Gets a logger object to log messages of pyIMD status to the console in a standardized format.

Returns Returns a logger object with correct string formatting.

Return type logger (*object*)

handle_change_console_text(*text*)
Implementation of the handle_change_console_text slot.

Parameters **text** (*str*) – String received from Settings instance to print to the console.

on_about()
Displays the about window.

on_change_log()
Displays the change log window.

on_combo_box_changed(*index*)
Prints the selected item of the data drop down list to the console.

Parameters **index** (*int*) – Index of the selected item from the drop down list.

on_concatenation()
Opens concatenation dialog and starts file concatenation in new thread.

on_data_list_selection_changed()
Adds the selected data to the PandasDataFrameModel model to be displayed in the results table view.

on_quick_instructions()
Displays the quick instructions window.

static on_read_documentation()
Opens the documentation in the default web browser.

on_settings_changed(*changed_settings*)
Update settings from settings dialog to settings configuration as soon as user commits parameter changes.

Parameters **changed_settings** (*dict*) – Settings dictionary

Returns Updates the changed settings on the object directly

Return type Null (*void*)

on_task_finished(*task*)
Enable the Result tab after the inertial mass determination run is finished.

Args: task: ThreadPoolExecutor task

on_update_text(*text*)
Writes new text to the console at the last text cursor position

Parameters **text** (*str*) – Text to be shown on the console.

open_project()
Opens a pyIMD project file (.xml) using the IntertialMassDetermination.load_pyimd_project method

print_to_console(*text*)
Print text to console.

Parameters **text** (*str*) –

run_batch_calculation()
Implementation of the pyIMD calculation batch mode based on pyIMD project files.

run_calculation()
Implementation of the pyIMD calculation start as new thread.

save_project()
Saves a pyIMD project file as .xml using the IntertialMassDetermination.save_pyimd_project method

Returns Saves pyIMD project as xml file to disk

Return type Null (*void*)

select_batch_files()
Selection of .xml pyIMD project files for batch calculation.

select_data_files()
Select data files to create a new pyIMD project

send_to_console_signal
pyqtSignal used to send a text to the console.

Parameters **message** (*str*) –

setup_console_connection()
Set up the console connection between the settings and the main window.

show_console()
Show and hide the console with the program log.

show_data()
Display the selected file names om the file viewer.

show_settings_dialog()
Show the settings dialog.

sync_settings()
Synchronizes the settings of the UI with the pyIMD object settings object.

class pyIMD.ui.main_ui.**Stream**
Bases: PyQt5.QtCore.QObject

Implementation of a stream to handle logging messages

stream_signal
pyqtsignal to redirect sterr

write(*text: object*) → object
Emits text formatted as string. :param text: :type text: *str*

class pyIMD.ui.settings.**SettingsDialog**(*settings_dictionary*)
Bases: PyQt5.QtWidgets.QDialog

Settings QDialog user interface implementation.

check_state()
Live validation if parameters entered by user are valid.

Returns Returns color formatter validator state.

Return type sender (*obj*)

close_settings_dialog()
Close the settings UI dialog without saving changes made on parameters

Returns None.

Return type Null (*void*)

commit_parameters()
Saves changes on parameters.

Returns Returns the changed parameters as dictionary.

Return type Parameters (*dict*)

find_checked_radiobutton()
Find the checked radiobutton

Returns Returns the name of the selected radio button.

Return type selected radio (*str*)

on_frequency_offset_mode_auto(*checked*)
Enables the auto offset mode fields

Parameters **checked** (*bool*) – Boolean enabling or disabling the frequency offset spin

Returns None

Return type Null (*void*)

on_frequency_offset_mode_manual(*checked*)
Enables the manual offset mode fields

Parameters **checked** (*bool*) – Boolean enabling or disabling the frequency offset field

Returns None

Return type Null (*void*)

on_toggle_frequency_offset(*state*)
Enables or disables the frequency offset optional parameters

Parameters **state** (*int*) – State enabling or disabling the frequency offset correction

Returns None

Return type Null (*void*)

print_to_console(*text*)
Print changes to console

Parameters **text** (*str*) – Text to print to the console

Returns Prints message to console.

Return type Message (*str*)

send_to_console_signal
pyqtSignal sends message to console

Returns Status message to be send to console.

Return type message (*str*)

set_defaults()
Set parameters default values to user interface.

Returns None

Return type Null (*void*)

set_values()
Set parameter values to user interface.

Returns None

Return type Null (*void*)

settings_has_changed_signal
pyqtSignal sends dictionary with all settings

Returns Dictionary with settings.

Return type settings (*dict*)

3.6 imd

class pyIMD.imd.**InertialMassDetermination**
Bases: PyQt5.QtCore.QObject

Constructs a IntertialMassDetermination object

concatenate_files(*directory*, *time_interval*, ***kwargs*)
Method to write concatenate data from single dat files (i.e data logger files from Nanonis software).

Parameters

- **directory** (*str*) – Directory containing files to concatenate.
- **time_interval** (*int*) – Measurement time interval in milliseconds.

Keyword Arguments **delimiter** (*str*) – Delimiter to be used in the data file to separate columns.(i.e. , s) If empty it uses default given by settings.

Returns Writes concatenated data to single .csv file.

Return type file (*void*)

convert_data()
Converts imported data to correct units needed for further calculation.

create_pyimd_project(*pre_start_no_cell_path*, *pre_start_with_cell_path*, *measurements_path*, *text_data_delimiter*, *read_text_data_from_line*, *calculation_mode*, ***kwargs*)
Create a pyIMD project with the following arguments. Two modes enable the analysis of different experimental setups. PLL mode and Cont.Sweep mode. For more information please read the documentation.

Parameters

- **pre_start_no_cell_path** (*str*) – File path + file name of initial frequency shift measurement before cell attachment (txt file).
- **pre_start_with_cell_path** (*str*) – File path + file name of initial frequency shift measurement after cell attachment (txt file).
- **measurements_path** (*str*) – File path + file name of the actual measurement (tdms file (default) or txt file).
- **text_data_delimiter** (*str*) – Text file data delimiter i.e ‘ ‘ for tab delimited or ‘,’ for comma separated data.
- **read_text_data_from_line** (*int*) – Line number from which data of pre start measurements should be read Typically the first few lines contain header information and no data.
- **calculation_mode** (*str*) – PLL := phase lock loops mode Cont.Sweep := sweep mode Auto := Auto detection of the mode (experimental)

Keyword Arguments

- **figure_width** (*float*) – Width of result figures
- **figure_height** (*float*) – Height of result figures
- **figure_units** (*str*) – Figure units i.e cm, inch
- **figure_format** (*str*) – Figure format i.e png or pdf
- **figure_resolution_dpi** (*int*) – Resolution of result figures in dpi
- **figure_name_pre_start_no_cell** (*str*) – Figure name of function fit for pre start with no cell loaded data
- **figure_name_pre_start_with_cell** (*str*) – Figure name of function fit for pre start with cell loaded data
- **figure_name_measured_data** (*str*) – Figure name of the resulting mass of the measured data
- **figure_plot_every_nth_point** (*'int'*) – Parameter defining how many data points will be plotted. For large data stets to increase readability and reducing file size.
- **conversion_factor_hz_to_khz** (*float*) – Conversion factor to convert from hertz to kilo hertz
- **conversion_factor_deg_to_rad** (*float*) – Conversion factor to convert from degrees to radian
- **spring_constant** (*float*) – Spring constant value of the cantilever
- **initial_parameter_guess** (*list*) – Initial parameter guess
- **lower_parameter_bounds** (*list*) – Lower parameter bounds
- **upper_parameter_bounds** (*list*) – Upper parameter bounds
- **rolling_window_size** (*'int'*) – Window size for calculating the rolling average.
- **correct_for_frequency_offset** (*'bool'*) – Correct for potential frequency offset during PLL mode.
- **frequency_offset_mode** (*'str'*) – Frequency offset correction mode (Auto or Manual)
- **frequency_offset_n_measurements_used** (*'int'*) – Number of measurement data points to be used for automatic frequency offset correction
- **frequency_offset** (*'float'*) – Frequency offset either set manually or calculated automatically
- **cantilever_length** (*float*) – Cantilever length in microns
- **cell_position** (*float*) – Cell position offset from cantilever tip in microns
- **project_folder_path** (*str*) – Path to project data files. Also used to store pyIMD results such as data and figures.

establish_console_connection()
Establish console connection between the imd object and the settings dialog.

static get_logger_object(*name*)
Gets a logger object to log messages of pyIMD status to the console in a standardized format.

Returns Returns a logger object with correct string formatting.

Return type logger (*object*)

handle_change_console_text(*string*)

Parameters **string** (*str*) – String received from Settings instance to print to the console.

load_pyimd_project(*file_path*)
Loads a pre defined pyIMD project form a XML file.

Parameters **file_path** (*str*) – Full path + file name to the pyIMD project file.

Returns String reporting the success of failure of loading a pyIMD project.

Return type status (*str*)

on_settings_changed(*changed_settings*)
Update settings

print_pyimd_project()
Prints the current pyIMD settings and parameters to the console.

Returns pyIMD settings and parameter summary as formatted string.

Return type pyIMD project summary (*str*)

run_batch_inertial_mass_determination(**args*)
Runs the inertial mass determination calculation in batch mode. Specify one or multiple pyIMD project files which will be run sequentially. NOTE: In a future release this will be parallelized using multiple threads to perform the calculations in parallel to gain speed. Currently the focus is not on speed but the idea is to analyze many experiments conveniently over night for example.

Parameters **args** (*list*) – List of one or many file paths + file names to valid pyIMD project files.

Returns Returns result structured in a pandas data frame and saves function fit plots as pdf files.

Return type void

run_inertial_mass_determination()
Runs the inertial mass determination calculation

Returns Returns result structured in a pandas data frame and saves function fit plots as pdf or png files directly to the disk.

Return type void

save_pyimd_project(*file_path*)
Saves the current pyIMD project as XML file.

Parameters **file_path** (*str*) – Full path + file name to the pyIMD project file.

Returns String reporting the success of failure of loading a pyIMD project.

Return type status (*str*)

show_settings_dialog()
Shows the settings dialog in a pop up window.

CHAPTER 4

Authors

- Andreas P. Cuny <andreas.cuny@bsse.ethz.ch>
- Gotthold Fläschner <gotthold.flaeschner@bsse.ethz.ch>

CHAPTER 5

License

pyIMD is released under the **GPL v3** license:

5.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. 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 them 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 prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. 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.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential

component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey 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; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has

substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but

permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If 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 convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU 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 that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and

conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

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.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS 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.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

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 state 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 3 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, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program 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, your program’s commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <https://www.gnu.org/licenses/>.

The GNU 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 Lesser General Public License instead of this License. But first, please read <https://www.gnu.org/licenses/why-not-lgpl.html>.

CHAPTER 6

References

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

G

H

I

L

M

N

O

P

R

S

T

U

W