
DryMass Documentation

Release 0.3.0

Paul Müller

Jun 18, 2018

Contents:

1	Introduction	3
1.1	What is DryMass?	3
1.2	Quantitative phase imaging	3
2	Theory Notes	5
2.1	Computation of cell dry mass	5
2.1.1	Definition	5
2.1.2	Relative and absolute dry mass	6
2.1.3	Notes and gotchas	6
3	Getting started	7
3.1	Installing DryMass	7
3.1.1	Upgrade	7
3.1.2	Known issues	8
3.2	Command line interface	8
3.2.1	dm_convert	8
3.2.2	dm_extract_roi	9
3.2.3	dm_analyze_sphere	9
3.3	Configuration file	9
3.3.1	[bg] Background correction	10
3.3.2	[holo] Hologram analysis	10
3.3.3	[meta] Image meta data	10
3.3.4	[output] Supplementary data output	11
3.3.5	[roi] Extraction of regions of interest	11
3.3.6	[specimen] Specimen parameters	11
3.3.7	[sphere] Sphere-based image analysis	11
3.4	Troubleshooting	12
3.4.1	Freezing / no output	12
3.4.2	None of the above	12
4	Tutorials	13
4.1	T1: Bead analysis (CLI)	13
4.1.1	Introduction	13
4.1.2	Prerequisites	13
4.1.3	Execute dm_analyze_sphere	13
4.1.4	Examine the results	14
4.1.5	Post-processing	15

4.2	T2: HL60 cell analysis (CLI)	16
4.2.1	Introduction	16
4.2.2	Prerequisites	16
4.2.3	Find regions of interest	16
4.2.4	Set 2nd order polynomial background correction	17
4.2.5	Perform sphere analysis	17
4.2.6	Plot the results	20
4.2.7	Discussion	20
5	Code examples	23
5.1	Dry mass computation with radial inclusion factor	23
5.2	Comparison of relative and absolute dry mass	25
6	Code reference	29
6.1	cli.parse_funcs	29
7	Changelog	31
7.1	version 0.3.0	31
7.2	version 0.2.0	31
7.3	version 0.1.4	32
7.4	version 0.1.3	32
7.5	version 0.1.2	32
7.6	version 0.1.1	32
7.7	version 0.1.0	32
8	Bibliography	33
9	Indices and tables	35
	Bibliography	37
	Python Module Index	39

DryMass is a user-friendly quantitative phase imaging analysis software. This is the documentation of DryMass version 0.3.0.

1.1 What is DryMass?

DryMass is a software for quantitative phase imaging (QPI) analysis with functionalities such as

- extraction of meta data (e.g. wavelength, acquisition time) from experimental data files,
- quantitative phase image background correction,
- determination of dry mass for biological cells, or
- extraction of refractive index and radius for spherical phase objects such as liquid droplets, microgel beads, or cells.

1.2 Quantitative phase imaging

Quantitative phase imaging (QPI) is a 2D imaging technique that quantifies the phase retardation of a wave traveling through a specimen. For instance, digital holographic microscopy (DHM) [KvB07] can be used to record the quantitative phase image of biological cells, yielding the optical density from which the *dry mass* or the refractive index (RI) can be computed. Another example is electron holography [LL02] which can be used to visualize *p-n junctions* due to the different electronic potentials in the doped semiconductors. DryMass was designed for the analysis of single cells (typical units for distance [μm] and wavelength [nm]), but the concepts used apply to both methods.

2.1 Computation of cell dry mass

2.1.1 Definition

The concept of cell dry mass computation was first introduced by Barer [Bar52]. The dry mass m of a biological cell is defined by its non-aqueous fraction $f(x, y, z)$ (concentration or density in g/L), i.e. the number of grams of protein and DNA within the cell volume (excluding salts).

$$m = \iiint f(x, y, z) dx dy dz$$

The assumption of dry mass computation in QPI is that $f(x, y, z)$ is proportional to the RI of the cell $n(x, y, z)$ with a proportionality constant called the refraction increment α (units [mL/g])

$$n(x, y, z) = n_{\text{intra}} + \alpha f(x, y, z)$$

with the RI of the intracellular fluid n_{intra} , a dilute salt solution. These two equations can be combined to

$$m = \frac{1}{\alpha} \cdot \iiint (n(x, y, z) - n_{\text{intra}}) dx dy dz. \quad (2.1)$$

In QPI, the RI is measured indirectly as a projected quantitative phase retardation image $\phi(x, y)$.

$$\phi(x, y) = \frac{2\pi}{\lambda} \int (n(x, y, z) - n_{\text{med}}) dz$$

with the vacuum wavelength λ of the imaging light and the refractive index of the cell-embedding medium n_{med} . Integrating the above equation over the detector area (x, y) yields

$$\iint \phi(x, y) dx dy = \frac{2\pi}{\lambda} \iiint (n(x, y, z) - n_{\text{med}}) dx dy dz \quad (2.2)$$

If the embedding medium has the same refractive index as the intracellular solute ($n_{\text{med}} = n_{\text{intra}}$), then equations (2.1) and (2.2) can be combined to

$$m_{\text{med=intra}} = \frac{\lambda}{2\pi\alpha} \cdot \iint \phi(x, y) dx dy.$$

For a discrete image, this formula simplifies to

$$m_{\text{med=intra}} = \frac{\lambda}{2\pi\alpha} \cdot \Delta A \cdot \sum_{i,j} \phi(x_i, y_j) \quad (2.3)$$

with the pixel area ΔA and a pixel-wise summation of the phase data.

2.1.2 Relative and absolute dry mass

If however the medium surrounding the cell has a different refractive index ($n_{\text{med}} \neq n_{\text{intra}}$), then the phase ϕ is measured relative to the RI of the medium n_{med} which causes an underestimation of the dry mass if $n_{\text{med}} > n_{\text{intra}}$. For instance, a cell could be immersed in a protein solution or embedded in a hydrogel with a refractive index of $n_{\text{med}} = n_{\text{intra}} + 0.002$. For a spherical cell with a radius of 10 μm , the resulting dry mass is underestimated by 46pg. Therefore, it is called “relative dry mass” m_{rel} .

$$m_{\text{rel}} = \frac{\lambda}{2\pi\alpha} \cdot \iint \phi(x, y) dx dy,$$

If the imaged phase object is spherical with the radius R , then the “absolute dry mass” m_{abs} can be computed by splitting equation (2.1) into relative mass and suppressed spherical mass.

$$\begin{aligned} m_{\text{abs}} &= \frac{1}{\alpha} \cdot \iiint (n(x, y, z) - n_{\text{med}} + n_{\text{med}} - n_{\text{intra}}) dx dy dz \\ &= m_{\text{rel}} + \frac{4\pi}{3\alpha} R^3 (n_{\text{med}} - n_{\text{intra}}) \end{aligned}$$

For a visualization of the deviation of the relative dry mass from the actual dry mass for spherical objects, please have a look at the [relative vs. absolute dry mass example](#).

2.1.3 Notes and gotchas

- **The default refraction increment in DryMass** is $\alpha = 0.18\text{mL/g}$, as suggested for cells based on the refraction increment of cellular constituents by references [BJ54] and [Bar53]. The refraction increment can be manually set using the [configuration](#) key “refraction increment” in the “sphere” section.
- **Variations in the refraction increment** may occur and thus the above considerations are not always valid. The refraction increment is little dependent on pH and temperature, but may be strongly dependent on wavelength (e.g. serum albumin $\alpha_{\text{SA}@366\text{nm}} = 0.198\text{mL/g}$ and $\alpha_{\text{SA}@656\text{nm}} = 0.179\text{mL/g}$) [BJ54].
- **The refractive index of the intracellular fluid** in DryMass is assumed to be $n_{\text{intra}} = 1.335$, an educated guess based on the refractive index of phosphate buffered saline (PBS), whose osmolarity and ion concentrations match those of the human body.
- **Dry mass and actual mass** of a cell differ by the weight of the intracellular fluid. This weight difference is defined by the volume of the cell minus the volume of the protein and DNA content. While it seems to be difficult to define a partial specific volume (PSV) for DNA, there appears to be a consensus regarding the PSV of proteins, yielding approximately 0.73mL/g (see e.g. reference [Bar57] as well as [HGC94] and [question 843 of the O-manual](#) referring to it). For example, the protein and DNA of a cell with a radius of 10 μm and a dry mass of 350pg (cell volume 4.19pL, average refractive index 1.35) occupy approximately $0.73\text{mL/g} \cdot 350\text{pg} = 0.256\text{pL}$ (assuming the PSV of protein and DNA are similar). Therefore, the actual volume of the intracellular fluid is 3.93pL (94% of the cell volume) which is equivalent to a mass of 3.93ng resulting in a total (actual) cell mass of 4.28ng. Thus, the dry mass of this cell makes up approximately 10% of its actual mass which leads to a total mass that is about 2% heavier than the equivalent volume of pure water (4.19ng).

3.1 Installing DryMass

DryMass is written in pure Python and supports Python version 3.6 and later. DryMass depends on several other scientific Python packages, including:

- `numpy`,
- `scikit-image` (segmentation).
- `qpimage` (phase data manipulation),
- `qpformat` (file formats),
- `qpsphere` (refractive index analysis, segmentation),

To install DryMass, use one of the following methods (package dependencies will be installed automatically):

- **from PyPI:** `pip install drymass`
- **from sources:** `pip install .orpython setup.py install`

3.1.1 Upgrade

If you have installed an older version of DryMass and wish to upgrade to the latest version, use

```
pip uninstall drymass followed by  
pip install drymass.
```

If you wish to install a specific version of DryMass (e.g. 0.3.0), use

```
pip install 'drymass==0.3.0'.
```

3.1.2 Known issues

- If you try to install from PyPI and get an error message similar to

*“Could not find a version that satisfies the requirement drymass (from versions:)
No matching distribution found for drymass”,*

please make sure that you are using Python version 3.6 or later with `python --version`. If that is already the case, please run `pip -vvv install drymass` and create an [issue](#) with the error messages (e.g. as a screenshot) that you get.

- If you are using Windows and the installation fails because *scikit-image* cannot be installed, e.g.

*” Rolling back uninstall of scikit-image
Command python.exe [...] --compile failed with error code 1 in [...] /scikit-image”,*

and you are using the [Anaconda Python distribution](#), please install *scikit-image* via `conda install scikit-image`. If you are not using Anaconda, you can install one of the [wheels provided by Christoph Gohlke](#) (download e.g. “`scikit_image0.14.0cp36cp35mwin_amd64.whl`” if you have installed the 64bit version of Python 3.6, navigate to the download directory and run `pip install scikit_image0.14.0cp35cp36mwin_amd64.whl`).

3.2 Command line interface

DryMass comes with a command line interface (CLI). To use the CLI, a command shell is required, such as the command prompt [Cmd.exe](#) on Windows, or [Terminal.app](#) on MacOS. Please note that if DryMass is installed in a [virtual environment](#), then the DryMass CLI is only available if this environment is activated.

3.2.1 dm_convert

This command converts experimental data on disk to the TIF file format for use with [Fiji/ImageJ](#) and to the hdf5-based [qimage](#) data file format. The experimental data files are loaded with the [qpformat](#) library, which supports several quantitative phase imaging file formats. If a specific format is not supported, please create an [issue](#) at the [qpformat issue page](#). A typical use case of `dm_convert` on Windows is

```
dm_convert "d:\\data\\path\\to\\experiment"
```

which is equivalent to

```
d:  
cd "data\\path\\to"  
dm_convert experiment
```

If this command is run initially for an experimental data set, the user is asked to enter or confirm imaging wavelength and detector pixel size. Then, a new directory `d:\\data\\path\\to\\experiment_dm` is created with the following files:

drymass.cfg the user-editable ***drymass configuration file*** which is used in subsequent analysis steps

sensor_data.h5 the experimental data (including meta data) in the hdf5-based **qpimage** data file format

sensor_data.tif the experimental phase and amplitude series data as a tif file, importable in **Fiji/ImageJ**

Note that it is possible to edit the *drymass.cfg* file and to re-run the `dm_convert` command (or any other of the commands below) with these updated parameters.

3.2.2 dm_extract_roi

This command automatically finds and extracts regions of interest (ROIs) and performs an automated background correction for single-cell analysis. The usage is the same as that of `dm_convert`:

```
dm_extract_roi "d:\\data\\path\\to\\experiment"
```

The command `dm_extract_roi` automatically runs `dm_convert` if it has not been run before. If ROI detection fails, the search parameters have to manually be updated in the *drymass configuration file*. The most important parameter is the diameter of the specimen in microns (“*size um*” in the *specimen* section); all other parameters are defined in the *roi* section. Note that the default parameters for the *roi* section are not written to the configuration file until `dm_extract_roi` is run. The following files are created by `dm_extract_roi`:

roi_data.h5 the extracted, background-corrected ROI data (including meta data) in the hdf5-based **qpimage** data file format

roi_data.tif the extracted, background-corrected ROI data as a tif file, importable in **Fiji/ImageJ**

roi_slices.txt the locations of the ROIs found as a txt file

sensor_roi_images.tif rendered sensor phase images with labeled ROIs; only created if “*roi images*” is set to “*True*” in the *output* section of the *drymass configuration file*

3.2.3 dm_analyze_sphere

This command is used for the analysis of spherical phase objects such as liquid droplets, beads, or suspended cells. The basic principle is thoroughly described in reference [SSM+16]. In short, this approach assumes that the objects found with `dm_extract_roi` are homogeneous and spherical which allows to extract parameters such as radius and refractive index from a single phase image (as opposed to tomographic approaches that require an acquisition of multiple phase images from different directions). The parameters for the sphere analysis, such as analysis method and scattering model, are defined in the *sphere* section of the *drymass configuration file*. For an overview of the available models, please refer to the **qpsphere docs**. The following files are created by `dm_analyze_sphere` (*METHOD* is the analysis method and *MODEL* is the scattering model defined in *drymass.cfg*):

sphere_METHOD_MODEL_data.h5 the quantitative sphere simulation data using *MODEL* with the parameters obtained with the combination of *METHOD* and *MODEL* for each of the ROIs obtained with `dm_extract_roi` in the hdf5-based **qpimage** data file format

sphere_METHOD_MODEL_images.tif rendered phase and intensity images of the input ROIs and the corresponding simulation, a difference, and a line plot through the phase image for visual inspection as a tif file, importable in **Fiji/ImageJ**

sphere_METHOD_MODEL_statistics.txt the analysis results, including refractive index, radius, and *relative and absolute dry mass* as a text file.

3.3 Configuration file

The DryMass configuration file *drymass.cfg* is located in the root of the output folder (“*_dm*” appended to the data path). The configuration file is divided into sections.

3.3.1 [bg] Background correction

DryMass uses the Python library `qpimage` for background correction. For detailed information on the algorithms (and the corresponding keyword arguments) used, please see `qpimage.bg_estimate`.

- **amplitude data** = none (*int_or_path*) – Amplitude bg correction file or index
Image indexing starts with 1. If a file, either specify the full path or the relative path to the directory containing the input data.
- **amplitude offset** = mean (*lcstr*) – Amplitude bg correction offset method
Valid values are defined in `qpimage.bg_estimate.VALID_FIT_OFFSETS`.
- **amplitude profile** = tilt (*lcstr*) – Amplitude bg correction profile method
Valid values are defined in `qpimage.bg_estimate.VALID_FIT_PROFILES`.
- **amplitude binary threshold** = nan (*float_or_str*) – Binary image threshold value or method
If not *nan*, defines either a threshold for background segmentation or a method in `skimage.filters`.
- **amplitude border perc** = 10 (*float*) – Amplitude bg border region to analyze [%]
- **amplitude border px** = 5 (*int*) – Amplitude bg border region to analyze [px]
- **enabled** = True (*fbbool*) – Enable bg correction globally
Set to *False* when manually editing *roi_slices.txt*.
- **phase data** = none (*int_or_path*) – Phase bg correction file or index
Image indexing starts with 1. If a file, either specify the full path or the relative path to the directory containing the input data.
- **phase offset** = mean (*lcstr*) – Phase bg correction offset method
Valid values are defined in `qpimage.bg_estimate.VALID_FIT_OFFSETS`.
- **phase profile** = tilt (*lcstr*) – Phase bg correction profile method
Valid values are defined in `qpimage.bg_estimate.VALID_FIT_PROFILES`.
- **phase binary threshold** = nan (*float_or_str*) – Binary image threshold value or method
If not *nan*, defines either a threshold for background segmentation or a method in `skimage.filters`.
- **phase border perc** = 10 (*float*) – Phase bg border region to analyze [%]
- **phase border px** = 5 (*int*) – Phase bg border region to analyze [px]

3.3.2 [holo] Hologram analysis

These parameters tune the hologram analysis step (if applicable). The parameters shown are passed to `qpimage.holo.get_field()`.

- **filter name** = disk (*str*) – Filter name for sideband isolation
- **filter size** = 1/3 (*float*) – Filter size (fraction of the sideband frequency)
- **sideband** = 1 (*floattuple_or_one*) – Sideband ± 1 or frequency coordinates

3.3.3 [meta] Image meta data

This section contains meta data of the experiment.

- **medium index** = nan (*float*) – Refractive index of the surrounding medium
- **pixel size um** = nan (*float*) – Detector pixel size [μm]

- **wavelength nm** = nan (*float*) – Imaging wavelength [nm]

3.3.4 [output] Supplementary data output

This section defines what additional data are written to disk.

- **roi images** = True (*bool*) – Rendered phase images with ROI location
- **sphere images** = True (*bool*) – Phase/Intensity images for sphere analysis
- **sensor tif data** = True (*bool*) – Phase/Amplitude sensor tif data

3.3.5 [roi] Extraction of regions of interest

The extraction of ROIs is done in `drymass.extractroi.extract_roi()`.

- **dist border px** = 10 (*int*) – Minimum distance of objects to image border [px]
- **eccentricity max** = 0.7 (*float*) – Allowed maximal eccentricity of the specimen
- **enabled** = True (*bool*) – Perform automated search for ROIs
If set to *False*, the file “roi_slices.txt” must contain ROIs.
- **exclude overlap px** = 30.0 (*float*) – Allowed distance between two objects [px]
- **force** = () (*tuple**tuple**int*) – Force ROI coordinates (x1,x2,y1,y2) [px]
- **pad border px** = 40 (*int*) – Padding of object regions [px]
- **size variation** = 0.5 (*float**01*) – Allowed variation relative to specimen size

3.3.6 [specimen] Specimen parameters

Prior information about the analyzed object(s).

- **size um** = 10 (*float*) – Approximate diameter of the specimen [μ m]
This is used as the initial value for the sphere analysis.

3.3.7 [sphere] Sphere-based image analysis

Retrieval of refractive index and radius is done with the Python module `qpsphere`. The parameters either apply to `qpsphere.edgefit.contour_canny()` or to `qpsphere.imagefit.alg.match_phase()`, depending on which analysis approach is used.

- **edge coarse** = 0.4 (*float*) – Coarse edge detection filter size
- **edge fine** = 0.1 (*float*) – Fine edge detection filter size
- **edge clip radius min** = 0.9 (*float*) – Interior edge point filtering radius
- **edge clip radius max** = 1.1 (*float*) – Exterior edge point filtering radius
- **edge iter** = 20 (*int*) – Maximum number iterations for coarse edge detection
- **image fit range position** = 0.05 (*float*) – Fit interpolation range for radius
- **image fit range radius** = 0.05 (*float*) – Fit interpolation range for radius
- **image fit range refractive index** = 0.10 (*float*) – Fit interpolation range for refractive index

- **image fix phase offset** = False (*bool*) – Fix the simulation background phase to zero
- **image iter** = 100 (*int*) – Maximum number of iterations for image fitting
- **image stop delta position** = 1 (*float*) – Stopping criterion for position
- **image stop delta radius** = 0.0010 (*float*) – Stopping criterion for radius
- **image stop delta refractive index** = 0.0005 (*float*) – Stopping criterion for refractive index
- **image verbosity** = 1 (*int*) – Verbosity level of image fitting algorithm
- **method** = edge (*lctr*) – Method for determining sphere parameters
Valid values are *edge* (edge-detection approach) or *image* (2D phase image fitting).
- **model** = projection (*lctr*) – Physical sphere model
Valid values are defined in `qpsphere.models.available`. If *method=edge*, then *model* must be set to *projection*. If *method=image*, setting *model* to *rytov-sc* has the best trade-off between accuracy and speed.
- **refraction increment** = 0.18 (*float*) – Refraction increment [mL/g]
- **radial inclusion factor** = 1.2 (*float*) – Radial inclusion factor for dry mass computation

3.4 Troubleshooting

3.4.1 Freezing / no output

If the *CLI* freezes and you see a blinking cursor for a very long time without any files being created or changed in the output directory (*_dm* appended to the input data set):

1. Remove all files in the output directory, **except** for *drymass.cfg*.
2. Run the analysis again.

3.4.2 None of the above

Please [create an issue](#) on GitHub with a screenshot (or copy-paste) of the error message.

4.1 T1: Bead analysis (CLI)

4.1.1 Introduction

Microgel beads are transparent, homogeneous, and spherical objects that are ideal test objects for quantitative phase imaging. The DryMass command *dm_analyze_sphere* can estimate the average refractive index of such homogeneous objects. This is a short tutorial that will reproduce the data presented in [supplementary figure 2a](#) of reference [\[SCG+17\]](#).

4.1.2 Prerequisites

For this tutorial, you need:

- Python 3.5 or above and DryMass version 0.1.1 or above (see [Installing DryMass](#))
- [Fiji](#) or Windows Photo Viewer (for data visualization)
- Experimental data set: [QLSR_PAA_beads.zip](#)

4.1.3 Execute *dm_analyze_sphere*

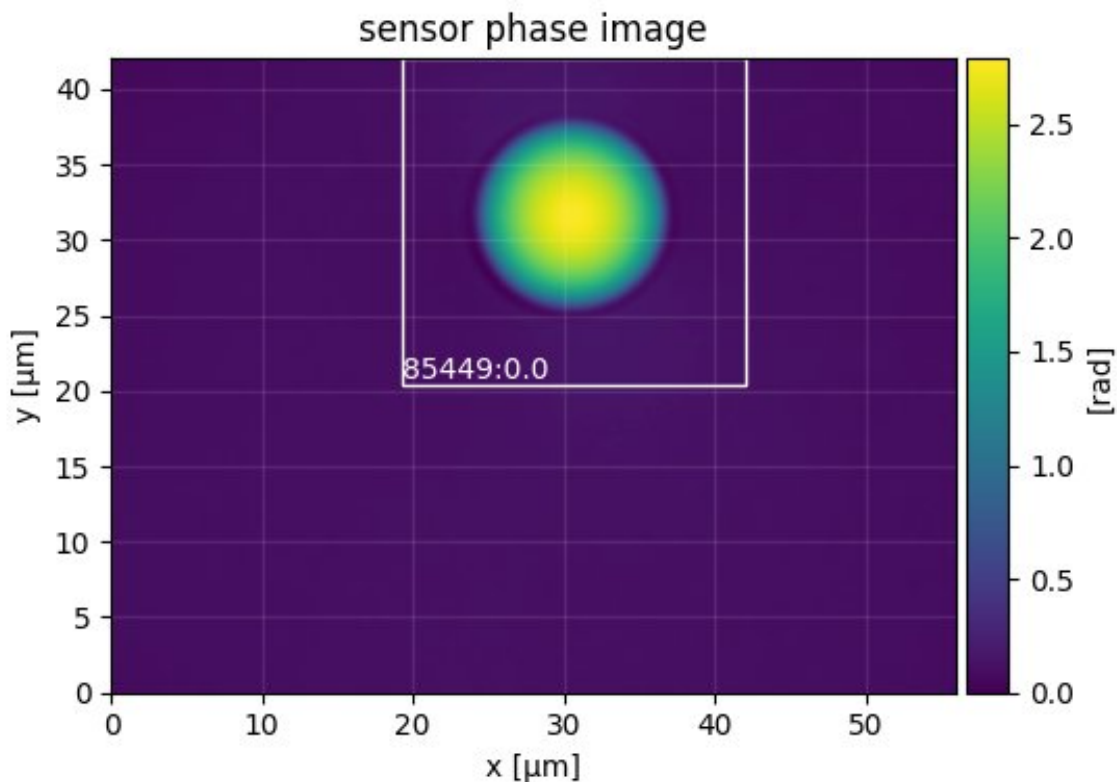
DryMass comes with a *Command line interface* (CLI) which is made available after the installation. We will use the DryMass command *dm_analyze_sphere* to extract the refractive index values of a population of microgel beads. Using the command shell of your operating system, navigate to the location of [QLSR_PAA_beads.zip](#) and execute the command *dm_analyze_sphere* with *QLSR_PAA_beads.zip* as an argument. You will be prompted for the refractive index of the surrounding medium (1.335), the detector pixel size in microns (0.14), and the wavelength in nanometers (647). Simply type in these values (press the *Enter* key to let DryMass acknowledge each input). On Windows, this will look similar to this (2.5x time lapse):

DryMass has created a directory called `QLSR_PAA_beads.zip_dm` (the input argument with `_dm` appended) which contains the following files

- **drymass.cfg**: DryMass *Configuration file*
- **roi_data.h5**: regions of interest (ROIs)
- **roi_data.tif**: phase and amplitude data of the ROIs as a tif file
- **roi_slices.txt**: positions of the ROIs
- **sensor_data.h5**: full sensor QPI data
- **sensor_data.tif**: full sensor phase and amplitude data as a tif file
- **sensor_roi_images.tif**: plotted full sensor phase images with ROIs
- **sphere_edge_projection_data.h5**: simulated (projection) phase and amplitude data
- **sphere_edge_projection_images.tif**: visualization of the sphere analysis of the ROIs as a tif file
- **sphere_edge_projection_statistics.txt**: sphere analysis results as a text file

4.1.4 Examine the results

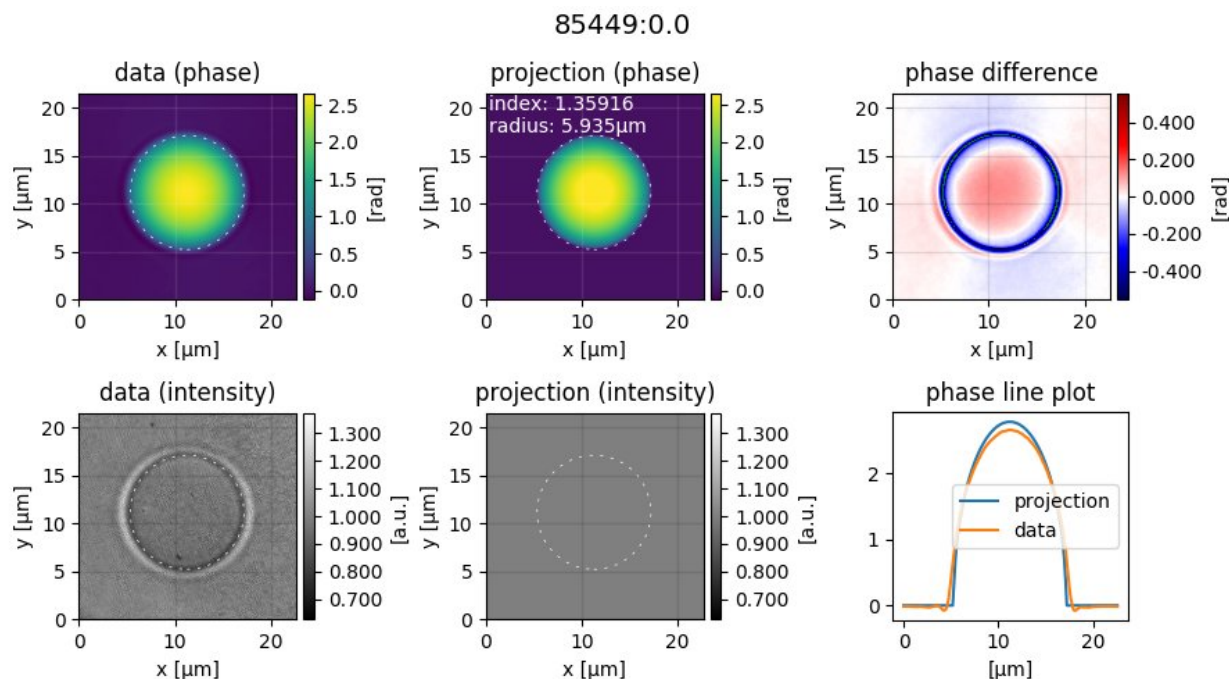
Let's have a look at `sensor_roi_images.tif` (using Fiji or Windows Photo Viewer). This is the first image stored in the tif file:



It shows the full sensor phase image of the first bead. The white rectangle indicates the ROI that was found by DryMass, labeled with the identifier `85449:0.0`. The file `sensor_roi_images.tif` allows you to check that DryMass has

correctly found the objects that you are interested in. If the beads were not detected correctly, we would probably have to adjust the size parameter in the *Configuration file* (see also *dm_extract_roi*).

It appears that DryMass has correctly found all beads with the default settings. Next, open the file *sphere_edge_projection_images.tif*. The identifier of the first ROI is shown at the top, the first column contains phase and intensity of the experimental data, the second column contains the modeled data (with refractive index and radius used for the simulation), and the third column shows the phase-difference as well as line plots through the phase images.



Note that the modeled intensity image is all-one, because the projection model only models the optical thickness and thus only affects the phase data. Also, note that the phase-difference image between data and model only has small deviations in the background phase. If there were large ramp-like structures or offsets, we would have to modify the *background correction*.

4.1.5 Post-processing

A closer examination of the phase-difference images shows that there seem to be either deformed beads or imaging artifacts in the images with the identifiers (prepend 85449:): 3.0, 6.0, 23.0, 25.0, 26.0, 34.0, 35.0, 38.0, 39.0, 50.0, 51.0, 54.0, 57.0, 59.0, 63.0, 66.0, and 70.0. Due to their asymmetry we ignore these images in our analysis by removing the respective rows from *sphere_edge_projection_statistics.txt* (Note that this file will be overridden when *dm_analyze_sphere* is executed again). We can then load the statistics file into a statistical analysis application and compute the average and the standard deviation of the refractive index. In Python, this can be done with

```
import numpy as np
ri = np.loadtxt("sphere_edge_projection_statistics.txt", usecols=(1,))

print("average: ", np.average(ri))
print("standard deviation: ", np.std(ri))
```

which will yield a refractive index of 1.357 ± 0.004 which agrees well with the value given in reference [SCG+17] (1.356 ± 0.004); The small difference can be explained by a slightly modified analysis pipeline and originally more strict selection criteria.

4.2 T2: HL60 cell analysis (CLI)

4.2.1 Introduction

HL60 cells in suspension are inhomogeneous, almost-spherical objects. To estimate an average refractive index (RI) of an HL60 cell population, the DryMass command `dm_analyze_sphere` can be used. This tutorial reproduces data presented in figure 5d of reference [MSG+18].

4.2.2 Prerequisites

For this tutorial, you need:

- Python 3.5 or above and DryMass version 0.1.4 or above (see [Installing DryMass](#))
- Fiji or Windows Photo Viewer (for data visualization)
- Experimental data set: [DHM_HL60_cells.zip](#)

4.2.3 Find regions of interest

Note: You can skip this part by copying `roi_slices.txt` from `DHM_HL60_cells.zip` into the `DHM_HL60_cells.zip_dm` folder, running `dm_convert` and manually adding the `[roi]` section to `drymass.cfg` with `enabled = False`.

We proceed slightly different than in [tutorial 1](#). Before we use the command `dm_analyze_sphere` to extract the RI values of the HL60 cells, we have to modify our configuration. We start by executing `dm_extract_roi DHM_HL60_cells.zip` which prompts us for the *pixel size* (0.107 μ m), and the *wavelength* (633nm), which can be found in the `readme.txt` file inside the zip archive. This command imports the raw data and searches for cells in the phase data. Opening the file `sensor_roi_images.tif`, we realize that the search parameters are not set optimally. This is image 27:

We want to exclude small ROIs and ROIs with a large overlap. Furthermore, we want to include all large cells (see e.g. image 39). Thus, we change the following configuration keys in `drymass.cfg`:

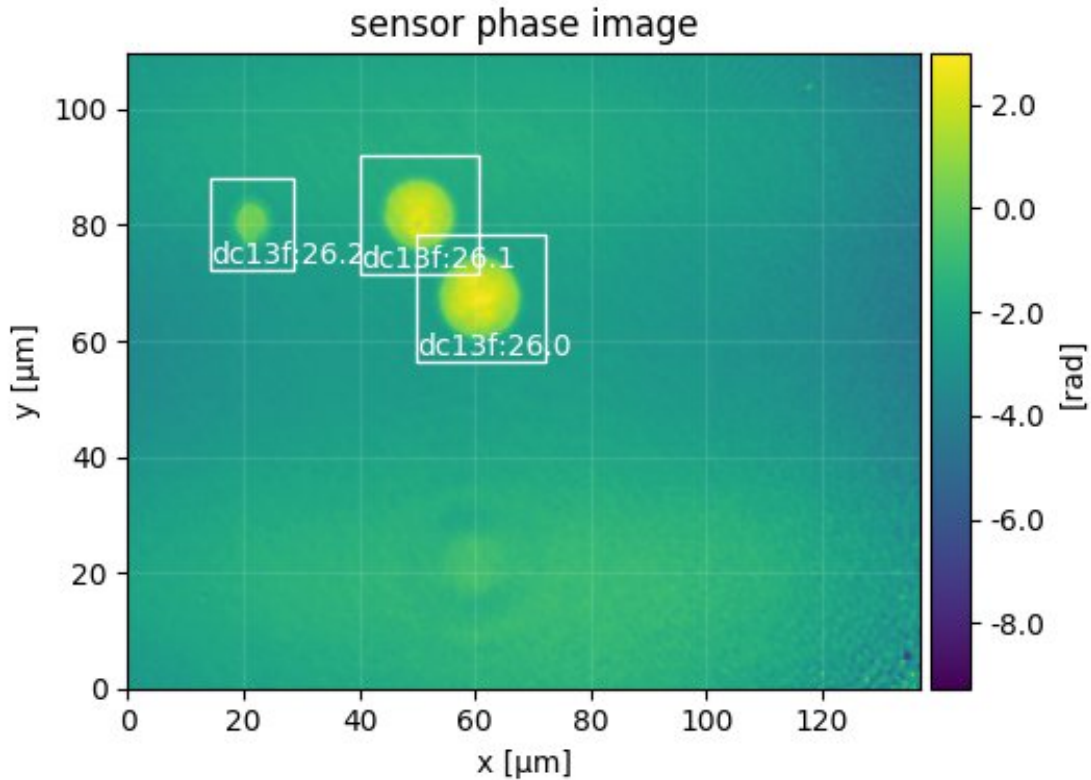
```
[specimen]
size um = 13          # approximate cell diameter we are looking for [ $\mu$ m]

[roi]
pad border = 80       # increase border size around cells
size variation = 0.2   # do not allow large variations of specimen size
exclude overlap = 100  # exclude ROIs with an overlap > 100px
```

With the new configuration, we run `dm_extract_roi DHM_HL60_cells.zip` again. Now all cells are detected. However, we want to exclude a few due to artifacts or shape issues. To achieve that, we disable the automatic search for ROIs in `drymass.cfg`

```
[roi]
enabled = False       # use existing roi_slices.txt
```

and remove the undesired ROIs from `roi_slices.txt`, which are 7.3, 14.1, 17.1, 17.2 and 34.1. There should now be a total of 87 ROIs.



4.2.4 Set 2nd order polynomial background correction

The default setting for background correction in DryMass is *tilt* which means that all phase data are corrected by fitting a 2D tilt image to the image borders. For the present dataset, a second order polynomial fit is a better approach, because the background phase does not follow a linear trend. Thus, we choose the *poly2o* profile and additionally set the fitting border width to 30 pixels. These are the updated lines in the `[bg]` section of `drymass.cfg`:

```
[bg]
phase border px = 30
phase profile = poly2o
```

4.2.5 Perform sphere analysis

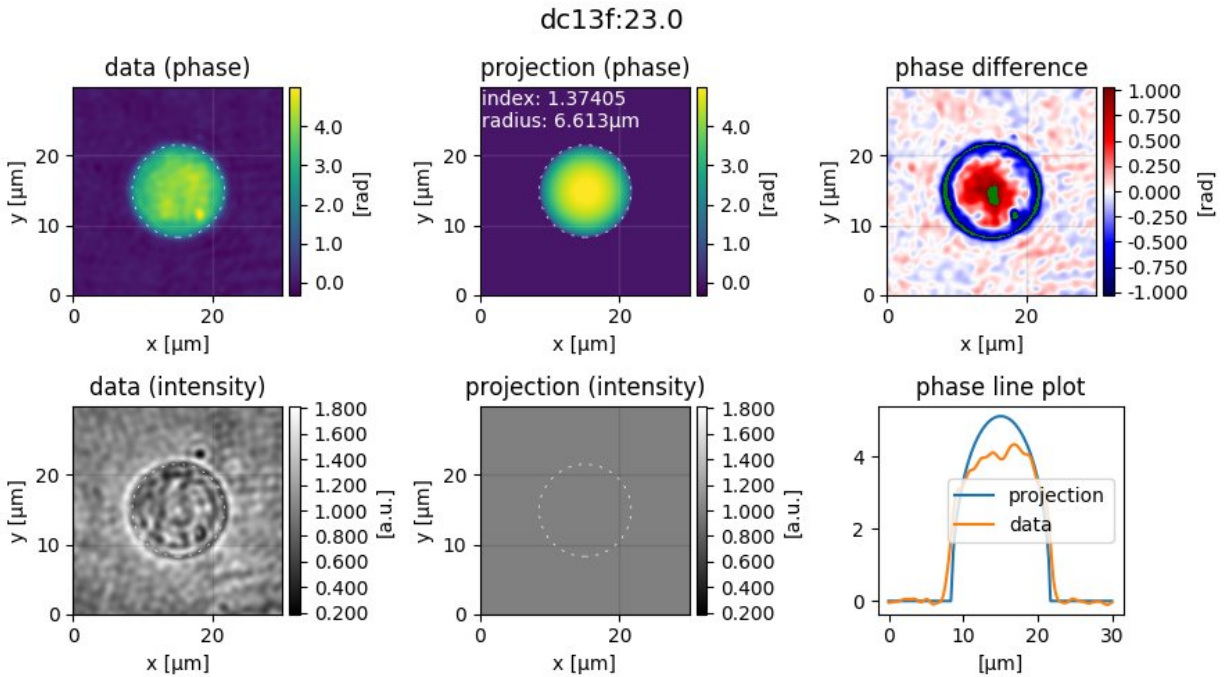
We now run `dm_analyze_sphere DHM_HL60_cells.zip` and are asked to enter the RI of the medium (1.335). By default, the RI of the cells is computed according to [\[SSM+15\]](#). The following files are created during this step:

- `sphere_edge_projection_data.h5`: QPI data
- `sphere_edge_projection_images.tif`: data visualization
- `sphere_edge_projection_statistics.txt`: results

Note: Warnings about *slice* and *QPIImage* identifiers can safely be ignored. Setting the RI of the medium changes the internal ROI identifiers. Since we have fixed the ROIs, the identifiers do not match anymore, but the enumeration

is still correct.

Let's have a look at the visualization of ROI 23.0 in *sphere_edge_projection_images.tif*.



The first column shows the experimental data, the second column shows the modeled data (with the cell perimeter indicated by a dashed circle), and the third column contains a residual image (pay attention to the colorbar, green means that the values are outside of the displayed range) and a line plot through the center of the cell. What is most striking about these data is that the RI is overestimated while the radius is underestimated by the edge-projection model. The explanation is that the radius of the cell is determined with an edge-detection algorithm applied to the phase image. Since the edge-detection algorithm determines the edge on the slope of the phase profile and not where the phase profile starts to deviate from the background, it underestimates the radius. The solution to this problem is to take into account the full phase image when determining RI and radius [KKL+07] [MSG+18].

This can be achieved by modifying the [sphere] section of *drymass.cfg*. In figure 5d of reference [MSG+18], multiple RI-retrieval methods are applied and compared for the same cell population. To reproduce these data, we run `dm_analyze_sphere DHM_HL60_cells.zip` three more times with a modified [sphere] section (note that this may take a while).

- Run 1: phase image fit with a projection model

```
[sphere]
method = image
model = projection
```

which produces the files

- *sphere_image_projection_data.h5*
- *sphere_image_projection_images.tif*
- *sphere_image_projection_statistics.txt*

- Run 2: phase image fit with the Rytov approximation


```
[sphere]
method = image
model = rytov
```

which produces the files

- *sphere_image_rytov_data.h5*
- *sphere_image_rytov_images.tif*
- *sphere_image_rytov_statistics.txt*

- Run 3: phase image fit with the systematically corrected Rytov approximation

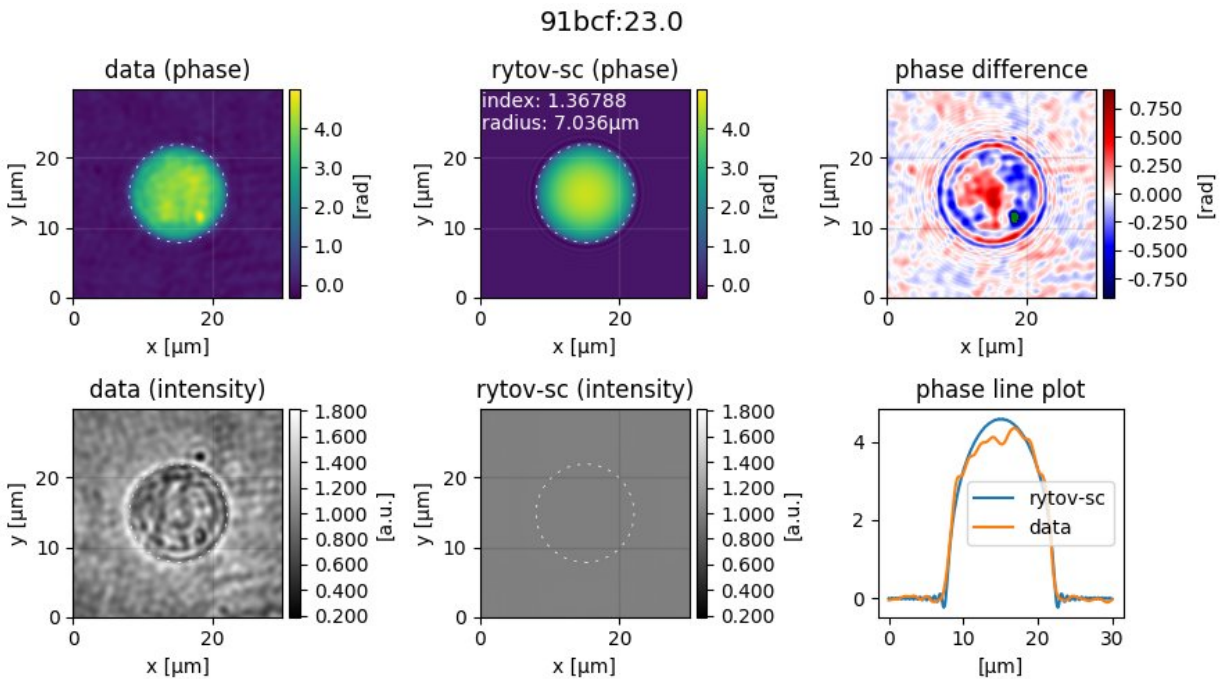
```
[sphere]
method = image
model = rytov-sc
```

which produces the files

- *sphere_image_rytov-sc_data.h5*
- *sphere_image_rytov-sc_images.tif*
- *sphere_image_rytov-sc_statistics.txt*

Note: We omitted the case `model = mie-avg` which is part of figure 5d in reference [MSG+18], because of the long fitting time.

To verify that the full-phase-image-based approaches indeed yield lower residuals than the edge-detection approach, let's have a look at ROI 23.0 of *sphere_image_rytov-sc_images.tif*.



The phase difference and the phase line plots look much better now. Observed deviations mostly originate from the inhomogeneity of the cell.

4.2.6 Plot the results

To plot the results, we use the following Python script.

```

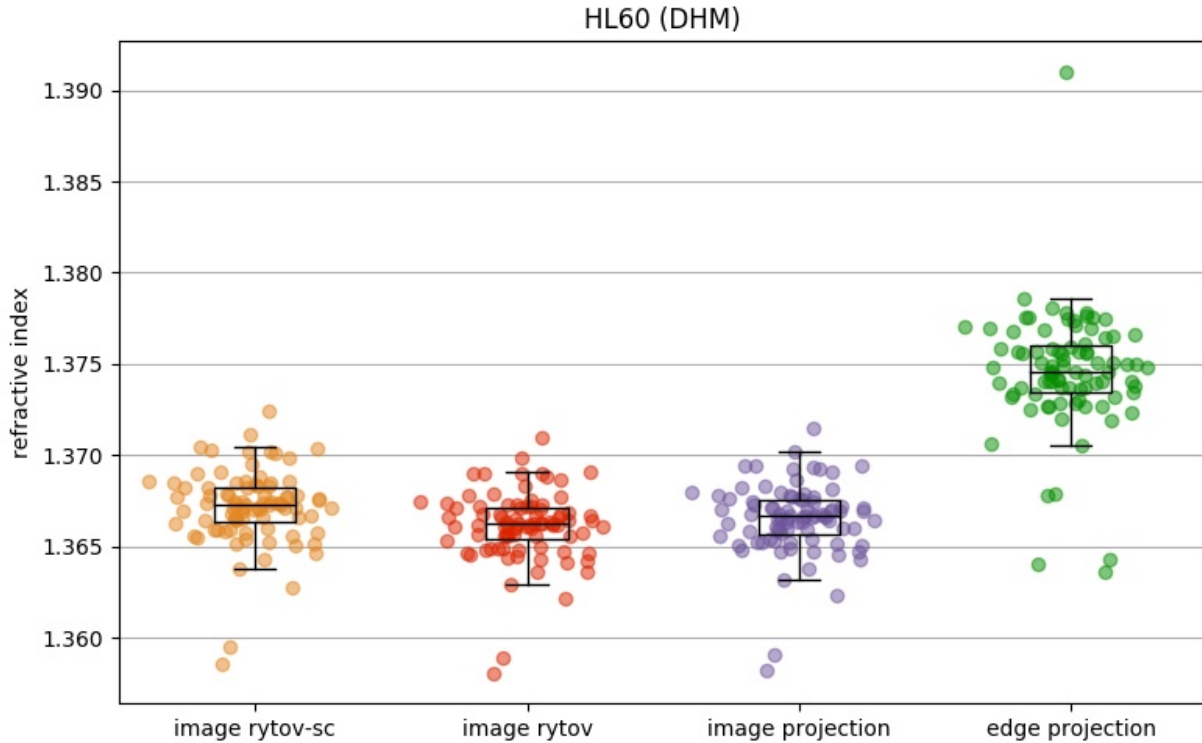
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4
5 def dot_boxplot(ax, data, colors, labels, **kwargs):
6     """Combined box and scatter plot"""
7     box_list = []
8
9     for ii in range(len(data)):
10         # set same random state for every scatter plot
11         rs = np.random.RandomState(42).get_state()
12         np.random.set_state(rs)
13         y = data[ii]
14         x = np.random.normal(ii+1, 0.15, len(y))
15         plt.plot(x, y, 'o', alpha=0.5, color=colors[ii])
16         box_list.append(y)
17
18     ax.boxplot(box_list,
19               sym="",
20               medianprops={"color": "black", "linestyle": "solid"},
21               widths=0.3,
22               labels=labels,
23               **kwargs)
24     plt.grid(axis="y")
25
26
27 if __name__ == "__main__":
28     ri_data = [
29         np.loadtxt("sphere_image_rytov-sc_statistics.txt", usecols=(1,)),
30         np.loadtxt("sphere_image_rytov_statistics.txt", usecols=(1,)),
31         np.loadtxt("sphere_image_projection_statistics.txt", usecols=(1,)),
32         np.loadtxt("sphere_edge_projection_statistics.txt", usecols=(1,)),
33     ]
34     colors = ["#E48620", "#DE2400", "#6e559d", "#048E00"]
35     labels = ["image rytov-sc", "image rytov",
36              "image projection", "edge projection"]
37
38     plt.figure(figsize=(8, 5))
39     ax = plt.subplot(111, title="HL60 (DHM)")
40     ax.set_ylabel("refractive index")
41     dot_boxplot(ax=ax, data=ri_data, colors=colors, labels=labels)
42     plt.tight_layout()
43     plt.show()

```

4.2.7 Discussion

The above figure correctly reproduces the message conveyed with figure 5d of reference [\[MSG+18\]](#). There are only minor differences that can be explained by a slightly different analysis pipeline:

- In [\[MSG+18\]](#), 84 cells were analyzed as opposed to the 87 cells shown here. This can be attributed to the improved object detection pipeline introduced in DryMass 0.1.4.
- In [\[MSG+18\]](#), the phase data were background-corrected with background data (not included in



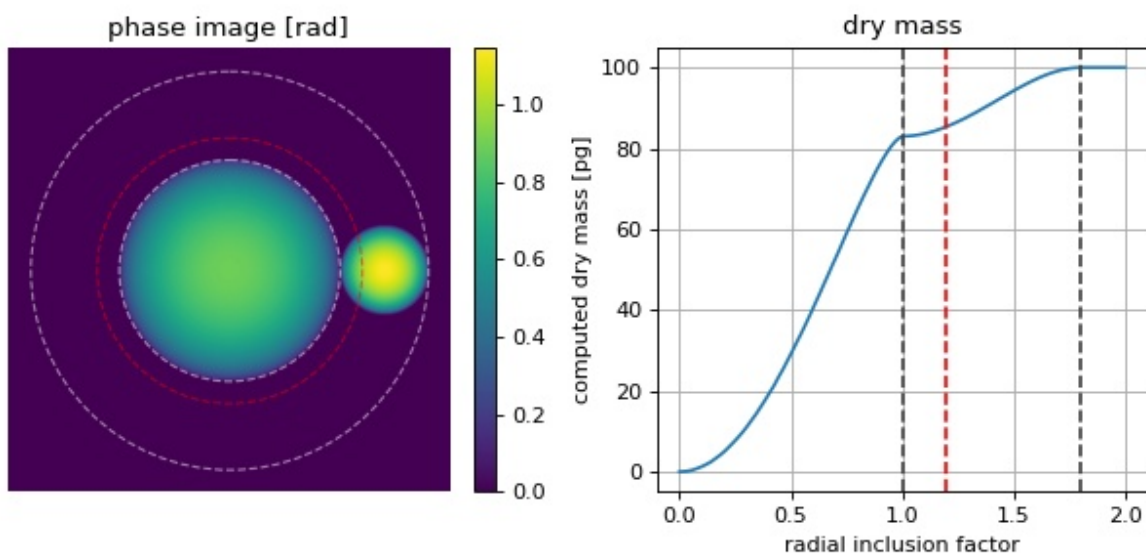
DHM_HL60_cells.zip) and a linear model (`phase profile = tilt`) as opposed to a second order polynomial model (which was introduced in DryMass 0.1.3). However, this does not seem to have any significant effect on the results, which indicates that the analysis methods are robust.

- There is a prominent outlier in the *edge projection* results set. The reason for this outlier is a falsely detected contour (see ROI 1.0). This ROI was not included in the analysis of [\[MSG+18\]](#).
- Other minor differences might originate from the fact that the hologram data is processed differently (`[holo]` section of *drymass.cfg*). In [\[MSG+18\]](#), a gaussian filter is used whereas DryMass defaults to a disk filter. For more information on this topic, see e.g. [Hologram filter choice](#).

5.1 Dry mass computation with radial inclusion factor

This examples illustrates the usage of the “radial inclusion factor” which is defined in the configuration section “sphere” and used in `drymass.ansphere.relative_dry_mass()` with the keyword argument *rad_fact*.

The phase image is computed from two spheres whose dry masses add up to 100pg with the larger sphere having a dry mass of 83pg. The larger sphere is located at the center of the image which is also used as the origin for dry mass computation. The radius of the larger sphere is known (10 μm). Thus, the corresponding radius (inner circle) corresponds to a radial inclusion factor of 1. In DryMass, the default radial inclusion factor is set to 1.2 (red). In some cases, this inclusion factor must be increased or decreased depending on whether additional information (the smaller sphere) should be included in the dry mass computation or not.



mass_radial_inclusion_factor.py

```

1  from drymass.anasphere import relative_dry_mass
2  import matplotlib
3  import matplotlib.pyplot as plt
4  import numpy as np
5  import qpimage
6  import qpsphere
7
8  # refraction increment
9  alpha = .18 # [mL/g]
10
11 # general simulation parameters
12 medium_index = 1.333
13 model = "projection"
14 wavelength = 500e-9 # [m]
15 pixel_size = 1e-7 # [m]
16 grid_size = (400, 400) # [px]
17
18 # sphere parameters
19 dry_masses = [83, 17] # [pg]
20 radii = [10, 4] # [μm]
21 centers = [(200, 200), (200, 340)] # [px]
22
23 phase_data = np.zeros(grid_size, dtype=float)
24 for m, r, c in zip(dry_masses, radii, centers):
25     # compute refractive index from dry mass
26     r_m = r * 1e-6
27     alpha_m3g = alpha * 1e-6
28     m_g = m * 1e-12
29     n = 1.333 + 3 * alpha_m3g * m_g / (4 * np.pi * (r_m**3))
30     # generate example dataset
31     qpi = qpsphere.simulate(radius=r_m,
32                             sphere_index=n,
33                             medium_index=medium_index,
34                             wavelength=wavelength,
35                             pixel_size=pixel_size,
36                             model=model,
37                             grid_size=grid_size,
38                             center=c)
39     phase_data += qpi.pha
40
41 qpi_sum = qpimage.QPImage(data=phase_data,
42                             which_data="phase",
43                             meta_data={"wavelength": wavelength,
44                                         "pixel size": pixel_size,
45                                         "medium index": medium_index})
46
47 # compute dry mass in dependence of radius
48 mass_evolution = []
49 mass_radii = []
50 for rad_fact in np.linspace(0, 2.0, 100):
51     dm = relative_dry_mass(qpi=qpi_sum,
52                             radius=radii[0] * 1e-6,
53                             center=centers[0],
54                             alpha=alpha,
55                             rad_fact=rad_fact)
56     mass_evolution.append(dm * 1e12)

```

(continues on next page)

(continued from previous page)

```

57     mass_radai.append(rad_fact)
58
59 # plot results
60 fig = plt.figure(figsize=(8, 3.8))
61 matplotlib.rcParams["image.interpolation"] = "bicubic"
62 # phase image
63 ax1 = plt.subplot(121, title="phase image [rad]")
64 ax1.axis("off")
65 map1 = ax1.imshow(qpi_sum.pha)
66 plt.colorbar(map1, ax=ax1, fraction=.048, pad=0.05)
67 # dry mass vs. inclusion factor
68 ax2 = plt.subplot(122, title="dry mass")
69 ax2.plot(mass_radai, mass_evolution)
70 ax2.set_ylabel("computed dry mass [pg]")
71 ax2.set_xlabel("radial inclusion factor")
72 ax2.grid()
73 # radius indicators
74 for r in [100, 180]:
75     cx = centers[0][0] + .5
76     cy = centers[0][1] + .5
77     circle = plt.Circle((cx, cy), r,
78                         color='w', fill=False, ls="dashed", lw=1, alpha=.5)
79     ax1.add_artist(circle)
80     ax2.axvline(r / 100, color="#404040", ls="dashed")
81 # add default
82 circle = plt.Circle((cx, cy), 120,
83                     color='r', fill=False, ls="dashed", lw=1, alpha=.5)
84 ax1.add_artist(circle)
85 ax2.axvline(1.2, color="r", ls="dashed")
86
87 plt.tight_layout()
88 plt.show()

```

5.2 Comparison of relative and absolute dry mass

Relative dry mass is the dry mass computed relative to the surrounding medium. If the refractive index of the surrounding medium does not match that of the intracellular fluid (approximately 1.335), then the relative dry mass underestimates the actual dry mass. For a spherical cell, the absolute (corrected) dry mass can be computed as described in the theory section on *dry mass computation*.

This examples compares the relative dry mass (`drymass.ansphere.relative_dry_mass()`) to the absolute dry mass corrected for a spherical phase object (`drymass.ansphere.absolute_dry_mass_sphere()`). From simulated phase images (projection approach, wavelength 550nm) of two cell-like spheres with a radius of 10 μ m and dry masses of 50pg (n1.337) and 250pg (n1.346), the absolute and relative dry masses are computed with varying refractive index of the medium.

At the refractive index of phosphate buffered saline (PBS), absolute and relative dry mass are equivalent. As the refractive index of the medium increases, the relative drymass decreases linearly (independent of dry mass), underestimating the actual dry mass.

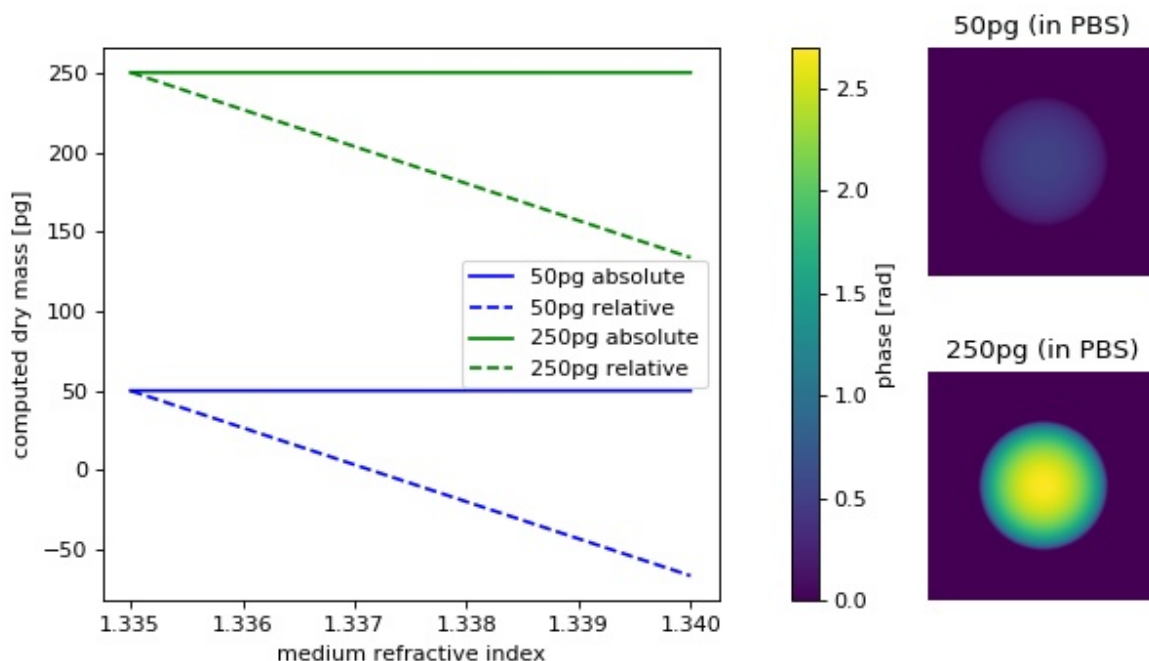
mass_relative_vs_absolute.py

```

1 from drymass.ansphere import absolute_dry_mass_sphere, relative_dry_mass
2 import matplotlib

```

(continues on next page)



(continued from previous page)

```

3 import matplotlib.pyplot as plt
4 import numpy as np
5 import qpsphere
6
7 # refraction increment
8 alpha = .18 # [mL/g]
9
10 # general simulation parameters
11 model = "projection"
12 wavelength = 500e-9 # [m]
13 pixel_size = 1.8e-7 # [m]
14 grid_size = (200, 200) # [px]
15
16 # sphere parameters
17 radius = 10 # [μm]
18 center = (100, 100) # [px]
19
20 dry_masses = [50, 250] # [pg]
21 medium_indices = np.linspace(1.335, 1.34, 5)
22
23 qpi_pbs = {}
24 m_abs = {}
25 m_rel = {}
26 phase_data = np.zeros(grid_size, dtype=float)
27 for m in dry_masses:
28     # initiate results list
29     m_abs[m] = []
30     m_rel[m] = []
31     # compute refractive index from dry mass
32     r_m = radius * 1e-6
33     alpha_m3g = alpha * 1e-6

```

(continues on next page)

(continued from previous page)

```

34     m_g = m * 1e-12
35     n = 1.335 + 3 * alpha_m3g * m_g / (4 * np.pi * (r_m**3))
36     for medium_index in medium_indices:
37         # generate example dataset
38         qpi = qpsphere.simulate(radius=r_m,
39                                sphere_index=n,
40                                medium_index=medium_index,
41                                wavelength=wavelength,
42                                pixel_size=pixel_size,
43                                model=model,
44                                grid_size=grid_size,
45                                center=center)
46
47         # absolute dry mass
48         ma = absolute_dry_mass_sphere(qpi=qpi,
49                                     radius=r_m,
50                                     center=center,
51                                     alpha=alpha,
52                                     rad_fact=1.2)
53
54         m_abs[m].append(ma * 1e12)
55         # relative dry mass
56         mr = relative_dry_mass(qpi=qpi,
57                               radius=r_m,
58                               center=center,
59                               alpha=alpha,
60                               rad_fact=1.2)
61
62         m_rel[m].append(mr * 1e12)
63         if medium_index == 1.335:
64             qpi_pbs[m] = qpi
65
66     # plot results
67     fig = plt.figure(figsize=(8, 4.5))
68     matplotlib.rcParams["image.interpolation"] = "bicubic"
69
70     # phase images
71     kw = {"vmax": qpi_pbs[dry_masses[1]].pha.max(),
72          "vmin": qpi_pbs[dry_masses[1]].pha.min()}
73
74     ax1 = plt.subplot2grid((2, 3), (0, 2))
75     ax1.set_title("{}pg (in PBS)".format(dry_masses[0]))
76     ax1.axis("off")
77     map1 = ax1.imshow(qpi_pbs[dry_masses[0]].pha, **kw)
78
79     ax2 = plt.subplot2grid((2, 3), (1, 2))
80     ax2.set_title("{}pg (in PBS)".format(dry_masses[1]))
81     ax2.axis("off")
82     ax2.imshow(qpi_pbs[dry_masses[1]].pha, **kw)
83
84     # overview plot
85     ax3 = plt.subplot2grid((2, 3), (0, 0), colspan=2, rowspan=2)
86     ax3.set_xlabel("medium refractive index")
87     ax3.set_ylabel("computed dry mass [pg]")
88     for m, c in zip(dry_masses, ["blue", "green"]):
89         ax3.plot(medium_indices, m_abs[m], ls="solid", color=c,
90                 label="{}pg absolute".format(m))
91         ax3.plot(medium_indices, m_rel[m], ls="dashed", color=c,
92                 label="{}pg relative".format(m))
93     ax3.legend()
94     plt.colorbar(map1, ax=ax3, fraction=.048, pad=0.1,

```

(continues on next page)

(continued from previous page)

```
91         label="phase [rad]")
92
93 plt.tight_layout()
94 plt.subplots_adjust(wspace=.14)
95 plt.show()
```


6.1 cli.parse_funcs

These methods are used to parse the values set in the *Configuration file* and convert them to the correct type.

`drymass.cli.parse_funcs.fbool (value)`

Boolean value from string or number

`drymass.cli.parse_funcs.fintlist (alist)`

List of integers from string or list of strings/integers

`drymass.cli.parse_funcs.float01 (flt)`

Float value between 0 and 1

`drymass.cli.parse_funcs.float_or_str (flt_or_str)`

Float value from string or number

`drymass.cli.parse_funcs.floattuple_or_one (fii)`

Tuple of two floats or ± 1 from a string or a number

`drymass.cli.parse_funcs.int_or_path (intpath)`

Integer or string from a string or a number

`drymass.cli.parse_funcs.lcstr (astr)`

Convert a string to lower-case

`drymass.cli.parse_funcs.tupletupleint (items)`

A tuple containing x- and y- slice tuples from a string or tuple

List of changes in-between DryMass releases.

7.1 version 0.3.0

- drop support for Python 3.5
- ref: background correction in `convert.py` uses background indices starting at 0 while the CLI keeps using indices starting at 1
- fix: hologram keyword arguments were not used for background correction
- fix: case where index-based background correction does not work
- feat: sphere analysis is picked up where left off in previous run
- command line interface:
 - fix: allow white spaces for input data path argument
 - fix: improve verbosity of status messages
- docs: add troubleshooting section

7.2 version 0.2.0

- feat: reuse computed ROI data from previous runs
- ref: update `[roi]` configuration key names to reflect pixel units: “dist border px”, “exclude overlap px”, “pad border px”
- fix: improve verbosity of error messages
- fix: config value “[roi] size variation” must be in interval [0,1]
- fix: handle ROIs with failed edge detection in “dm_analyze_sphere”

7.3 version 0.1.4

- feat: add local thresholding step before segmentation
- fix: check number of qpimages in sensor_data.h5 during dm_convert
- fix: “exclude overlap” only used for background image
- fix: not possible to analyze phasics data folder when raw data is present (qpformat 0.1.5)

7.4 version 0.1.3

- automatically fix inverted ranges when loading “roi_slices.txt”
- remove “holo” section from drymass.cfg if not relevant
- allow user-defined ROI selection via “[roi] enabled = False”
- added hologram analysis parameter tuning (qpimage 0.1.6)
- added 2D model fitting to sphere analysis (qpsphere 0.1.4)
- fixed *cfg_funcs.int_or_str*

7.5 version 0.1.2

- allow to disable sensor image tif export
- support input QPI data with different shapes
- fixed wrong assumption about definition of dry mass in cells (water vs. intracellular salt solution)
- allow to use separate file or series index for background correction with experimental data (#6)
- small visualization improvements

7.6 version 0.1.1

- add file name to image identifier and update plotting (#4)
- renamed “object” to “identifier” in statistics output
- binary-based background correction (manual or automated threshold)
- allow to disable background correction with “[bg] enabled = False”
- ask for missing meta data keys in CLI only when they are required

7.7 version 0.1.0

- initial release

CHAPTER 8

Bibliography

CHAPTER 9

Indices and tables

- `genindex`
- `modindex`
- `search`

Bibliography

- [Bar52] R. Barer. Interference Microscopy and Mass Determination. *Nature*, 169(4296):366–367, 1952. doi:10.1038/169366b0.
- [Bar53] R. Barer. Determination of dry mass, thickness, solid and water concentration in living cells. *Nature*, 172:1097–1098, 1953. doi:10.1038/1721097a0.
- [Bar57] R. Barer. Refractometry and interferometry of living cells. *Journal of the Optical Society of America*, 47(6):545, jun 1957. doi:10.1364/josa.47.000545.
- [BJ54] R. Barer and S. Joseph. Refractometry of Living Cells Part I. Basic Principles. *Quarterly Journal of Microscopical Science*, 171(2):38P–39P, 1954. doi:10.1038/171720a0.
- [HGC94] Y. Harpaz, M. Gerstein, and C. Chothia. Volume changes on protein folding. *Structure*, 2(7):641–649, jul 1994. doi:10.1016/s0969-2126(00)00065-4.
- [KKL+07] B. Kemper, S. Kosmeier, P. Langehanenberg, G. von Bally, I. Bredebusch, W. Domschke, and J. Schnekenburger. Integral refractive index determination of living suspension cells by multifocus digital holographic phase contrast microscopy. *Journal of Biomedical Optics*, 12(5):054009, 2007. doi:10.1117/1.2798639.
- [KvB07] B. Kemper and G. von Bally. Digital holographic microscopy for live cell applications and technical inspection. *Applied Optics*, 47(4):A52, oct 2007. doi:10.1364/ao.47.000a52.
- [LL02] M. Lehmann and H. Lichte. Tutorial on off-axis electron holography. *Microscopy and Microanalysis*, 8(06):447–466, dec 2002. doi:10.1017/s1431927602020147.
- [MSG+18] P. Müller, M. Schürmann, S. Girardo, G. Cojoc, and Guck J. Accurate evaluation of size and refractive index for spherical objects in quantitative phase imaging. *Optics Express*, 26(8):10729–10743, 2018. doi:10.1364/OE.26.010729.
- [SCG+17] M. Schürmann, G. Cojoc, S. Girardo, E. Ulbricht, J. Guck, and P. Müller. Three-dimensional correlative single-cell imaging utilizing fluorescence and refractive index tomography. *Journal of Biophotonics*, 11(3):e201700145, aug 2017. doi:10.1002/jbio.201700145.
- [SSM+15] M. Schürmann, J. Scholze, P. Müller, C. J. Chan, A. E. Ekpenyong, K. J. Chalut, and J. Guck. Chapter 9 - Refractive index measurements of single, spherical cells using digital holographic microscopy. In Ewa K Paluch, editor, *Biophysical Methods in Cell Biology*, volume 125 of *Methods in Cell Biology*, pages 143–159. Academic Press, 2015. doi:10.1016/bs.mcb.2014.10.016.
- [SSM+16] M. Schürmann, J. Scholze, P. Müller, J. Guck, and C. J. Chan. Cell nuclei have lower refractive index and mass density than cytoplasm. *Journal of Biophotonics*, 9(10):1068–1076, oct 2016. doi:10.1002/jbio.201500273.

d

`drymass.cli.parse_funcs`, [29](#)

D

`drymass.cli.parse_funcs` (module), [29](#)

F

`fbool()` (in module `drymass.cli.parse_funcs`), [29](#)

`fintlist()` (in module `drymass.cli.parse_funcs`), [29](#)

`float01()` (in module `drymass.cli.parse_funcs`), [29](#)

`float_or_str()` (in module `drymass.cli.parse_funcs`), [29](#)

`foattuple_or_one()` (in module `drymass.cli.parse_funcs`),
[29](#)

I

`int_or_path()` (in module `drymass.cli.parse_funcs`), [29](#)

L

`lcstr()` (in module `drymass.cli.parse_funcs`), [29](#)

T

`tupletupleint()` (in module `drymass.cli.parse_funcs`), [29](#)