JPKay Documentation

Release 0.1

Tony Fischer, Jeremy Perez

April 29, 2016

Contents

This Project aims to provide a comprehensive toolbox to load, analyze and plot JPK CellHeasion200 force-files. It is currently under heavy development.

CHAPTER 1

Documentation

1.1 User Guide

1.1.1 The CellHesion Class

This is the main data-class that provides all functionality to load, analyze and display a single JPK CellHesion200 force file archive.

Attributes

The following attributes are available:

- archive: an instance of *ForceArchive*
- properties: an instance of *Properties*
- data: pandas.DataFrame

ForceArchive Attribute

This is the internal jpk-force file archive handling object and should only be used to re-load data. This can be achieved via *load_data()*, for example:

```
>>> jpk_file = r'path/to/jpk-force/file'
>>> sample = CellHesion(force_file=jpk_file)
>>> sample.data.retract.force = pd.Series(np.random.rand(10))
>>> sample.load_data()
```

For more info, see *ForceArchive*.

Properties Attribute

```
>>> jpk_file = r'path/to/jpk-force/file'
>>> sample = CellHesion(force_file=jpk_file)
>>> print(sample.properties.units["vDeflection"])
V
>>> print(sample.properties.general["timestamp"])
2014-12-11 18:19:11 UTC+0000
>>> print(sample.properties.segments['retract']['force-segment-header.num-points'])
78635
```

```
>>> print(sample.properties.segments['contact']['name_jpk'])
pause-at-end
```

For more info, see *Properties*.

Data Attribute

The data segments are called:

- approach: cantilever approaches sample
- contact: cantilever is in contact with the sample
- retract: cantilever retracts from the sample
- pause: cantilever pauses between consecutive probings

Each segment holds both the force and height signal respectively. The force signal is in units of Newton (N), the height signal is in units of Meter (m).

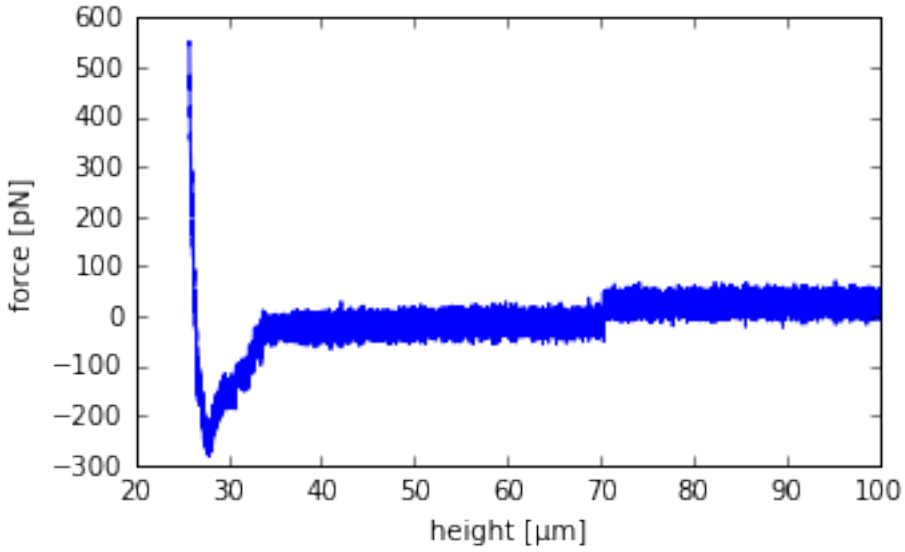
segment	approach		contact		retract		pause	
channel	force	height	force	height	force	height	force	height
0	4e-11	0.0001	5e-11	0.0001	5e-11	0.0001	4e-11	0.0001
...	...	...	...	...	...	...	...	...

The DataFrame has a hierarchical MultiIndex as column names and can be accessed using both standard DataFrame column indexing methods `sample.data.retract.force` or `sample.data['retract']['force']`. Manipulating data in-place has to happen using the `loc` method due to the usage of MultiIndexes (see official documentation for further explanation).

```
>>> jpk_file = r'path/to/jpk-force/file'
>>> sample = CellHesion(force_file=jpk_file)
>>> sample.data.retract.force.head()  # access using method
>>> sample.data['retract']['force'].head()  # access using dict-keys
>>> sample.data.loc[0, ('retract', 'force')] *= 10**12  # convert to pN
```

Example Usage

```
>>> jpk_file = r'path/to/jpk-force/file'
>>> sample = CellHesion(force_file=jpk_file)
>>> import matplotlib.pyplot as plt
>>> x = sample.data.retract.height * 10**6
>>> y = sample.data.retract.force * 10**12
>>> plt.plot(x, y)
>>> plt.xlabel("height [µm]"); plt.ylabel("force [pN]")
```



1.2 API Documentation

class JPKay.core.data_structures.**CellHesion**(*force_file*)

This is the main data-class that provides all functionality to load, analyze and display a single JPK CellHesion200 force file archive.

Attributes

The following attributes are available:

- archive: an instance of *ForceArchive*
- properties: an instance of *Properties*
- data: pandas.DataFrame

Example Usage

```
>>> jpk_file = r'path/to/jpk-force/file'
>>> sample = CellHesion(force_file=jpk_file)
>>> import matplotlib.pyplot as plt
>>> x = sample.data.retract.height * 10**6
>>> y = sample.data.retract.force * 10**12
>>> plt.plot(x, y)
>>> plt.xlabel("height [µm]"); plt.ylabel("force [pN]")
```

static construct_df()

Construct a pandas DataFrame to store force and height data for each segment.

Returns DataFrame blueprint

Return type pandas.DataFrame

convert_data(*channel*, *data*)

Convert specific data from specific channel from encoded integer format to physical quantity.

Each channel has it's own conversion factors and formulas, so the correct channel has to be provided.

Parameters

- **channel** (*str*) – data channel
- **data** (*numpy.ndarray*) – encoded data

Returns converted data

Return type numpy.array

load_data()
Load converted data to DataFrame. See *construct_df()* for DataFrame structure.

Returns force/height data

Return type pandas.DataFrame

load_encoded_data_segment(*segment*)
Loads the raw, encoded vertical deflection and height data of the specified segment.

This has to be converted using *convert_data()* to make use of it.

Parameters **segment** (*str*) – data segment to load

Returns vDeflection and height

class JPKay.core.data_structures.**ForceArchive**(*file_path*)
Object to handle reading contents of a jpk-force zipped file.

•Methods

•ls: list archive contents

•read_properties: read utf-8 string decoded content of a property file, one property per list entry

•read_data: read encoded raw data, must be converted to appropriate physical quantity!

ls()
List all files contained in this force-archive

read_data(*content_path*)
Reads the raw integer-encoded data of the specified data file inside a force-archive.

Parameters **content_path** (*str*) – internal path to the force-archive file

Returns raw data

Return type numpy.ndarray

read_properties(*content_path*)
Reads a property file form the force-archive.

The contents of the property file are elements of a list. Each entry is already decoded to utf-8.

Parameters **content_path** (*str*) – internal path to the force-archive file

Returns property list

Return type dict

class JPKay.core.data_structures.**Properties**(*file_path*)
Object to automatically extract and conveniently use relevant JPK force file header information.

This comprises things like conversion factors for raw data, units, and so on

•attributes

–vDeflection_channel_number: internal number of vDeflection channel raw data

–conversion_factors: dictionary containing important information

–units: dictionary containing channel units

•**example usage**:

```
>>> force_file = r"path/to/jpk-force-file"
>>> props = Properties(file_path=force_file)
>>> print(props.units["vDeflection"])
V
>>> print(props.conversion_factors["vDeflection"]["force multiplier"])
0.01529211140472191
```

static **convert_segment_name**(*jpk_name*)
Convert JPKs segment names to useful ones

extract_conversion_factors()
Extracts all conversion factors for the raw data channels. Currently, only vDeflection channel is extracted, because it is the only one calibrated during AFM measurements

Returns dict with conversion factors

Return type dict

extract_segment_props()
Extract properties for each data segment. Additionally, JPKs segment names are converted to a more useful naming scheme: approach, contact, retract, pause. Also the much needed segment number is stored to use during data loading. Properties for each segment are stored in a dictionary under the respective segment names as key.

Returns per-segment properties

Return type dict

extract_specs()
Extracts any kind of infos from the header, like units and the like

get_channel_numbers()
Extracts the channel numbers for each channel.

Returns dictionary with channel numbers

Return type dict

load_general_props()
This actually loads the props file on disk from jpk-force zip-file. Parses all java-properties info and the timestamp from the header of the header.

Returns props dictionary

Return type dict

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

j

C

E

F

G

J

L

P

R