

Data Exploration and Analysis with Jupyter Notebooks (TUCPR02)



Hans Fangohr

Senior Data Analysis Scientist

European X-ray Free Electron Laser (EuXFEL), Germany

Professor of Computational Modelling

University of Southampton, United Kingdom

fangohr.github.io, hans.fangohr@xfel.eu, [@ProfCompMod](https://twitter.com/ProfCompMod)

ICALEPCS 2019, New York, US, 8 October 2019

H. Fangohr^{*1}, M. Beg, M. Bergemann, V. Bondar, S. Brockhauser^{2,3}, C. Carinan, R. Costa, F. Dall'Antonia, C. Danilevski, J. C. E, W. Ehsan, S. G. Esenov, R. Fabbri, S. Fangohr, G. Flucke, C. Fortmann⁴, D. Fulla Marsa, G. Giovanetti, D. Goeries, S. Hauf, D. G. Hickin, T. Jarosiewicz⁵, E. Kamil, M. Karnevskiy, Y. Kirienko, A. Klimovskaia, T. A. Kluyver, M. Kuster, L. Le Guyader, A. Madsen, L. G. Maia, D. Mamchuk, L. Mercadier, T. Michelat, J. Möller, I. Mohacsi, A. Parenti, M. Reiser, R. Rosca, D. B. Rueck, T. Rüter, H. Santos, R. Schaffer, A. Scherz, M. Scholz, A. Silenzi, M. Spirzewski⁵, J. Sztuk, J. Szuba, S. Trojanowski⁵, K. Wrona, A. A. Yaroslavtsev, and J. Zhu

European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, Germany

¹ also at University of Southampton, SO17 1BJ, Southampton, United Kingdom

² also at University of Szeged, Arpad ter 2, 6701 Szeged, Hungary

³ also at Biological Research Center of the Hungarian Academy of Sciences, Szeged, Hungary

⁴ also at Max-Planck-Inst. for Evolutionary Biology, August-Thienem.-Strasse 2, 24306 Plön, Germany

⁵ also at NCBJ, Andrzejka Sołtana 7, 05-400 Otwock, Poland

J. Reppin, F. Schlünzen, and M. Schuh
DESY, Notkestr. 85, 22607 Hamburg, Germany

E. Fernandez-del-Castillo and G. Sipos
EGI Foundation, 140 Science Park, 1098XG Amsterdam, Netherlands

T. H. Rod, J. R. Selknaes, and J. W. Taylor
ESS, Ole Maaløes Vej 3, Copenhagen, Denmark

A. Campbell, A. Götz, and J. Kieffer
ESRF, 71 Avenue des Martyrs, 38000 Grenoble, France

J. Hall, E. Pellegrini, and J. F. Perrin
ILL, 71 Avenue des Martyrs, 38000 Grenoble, France

Outline

■ Introduction Jupyter Notebook

■ Use cases

■ Summary

Jupyter Notebook

- Document hosted in web browser
- Combines
 - text (markdown with LaTeX support)
 - computer code (Python)
 - output from code
- Saved in one *.ipynb file (IPYthon NoteBook)
 - combines input and output for each cell
- Re-use
 - load, re-execute, modify
 - export to static formats (html, pdf, py, ...)
- Demo at <https://github.com/fangohr/jupyter-demo>

```
In [1]: # Import Python and libraries we need later
%matplotlib inline
from numpy import exp, cos, linspace
import pylab
from ipywidgets import interact
```

Mathematical model: We would like to understand $f(t, \alpha, \omega) = \exp(-\alpha t) \cos(\omega t)$

Code: Here is an implementation:

```
In [2]: def f(t, alpha, omega):
        """Computes and returns exp(-alpha*t) * cos(omega*t)"""
        return exp(-alpha * t) * cos(omega * t)
```

Interactive exploration: We can execute the function for values of t , α and ω :

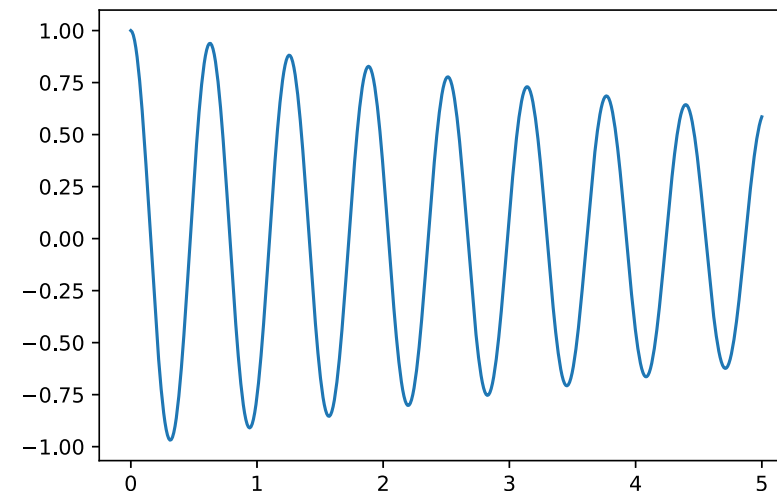
```
In [3]: f(t=0.1, alpha=1, omega=10)
```

```
Out[3]: 0.48888574340060287
```

Or produce a plot (in a function `plot_f` so it can be re-used for different parameters):

```
In [4]: def plot_f(alpha, omega):
        ts = linspace(0, 5, 500) # 500 points in interval [0, 5]
        ys = f(ts, alpha, omega)
        pylab.plot(ts, ys, '-')
```

```
In [5]: plot_f(alpha=0.1, omega=10) # call function and create plot
```



Jupyter Notebook

Supports a wide range of languages

Julia

Python (→ JuPyTer)

Shell, R, Matlab

C++

► <https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

A number of useful tools and extensions available

Some mentioned in talk:

nbconvert, JupyterHub, Binder, nbval,

Demo at <https://github.com/fangohr/jupyter-demo>

```
In [1]: # Import Python and libraries we need later
%matplotlib inline
from numpy import exp, cos, linspace
import pylab
from ipywidgets import interact
```

Mathematical model: We would like to understand $f(t, \alpha, \omega) = \exp(-\alpha t) \cos(\omega t)$

Code: Here is an implementation:

```
In [2]: def f(t, alpha, omega):
        """Computes and returns exp(-alpha*t) * cos(omega*t)"""
        return exp(-alpha * t) * cos(omega * t)
```

Interactive exploration: We can execute the function for values of t , α and ω :

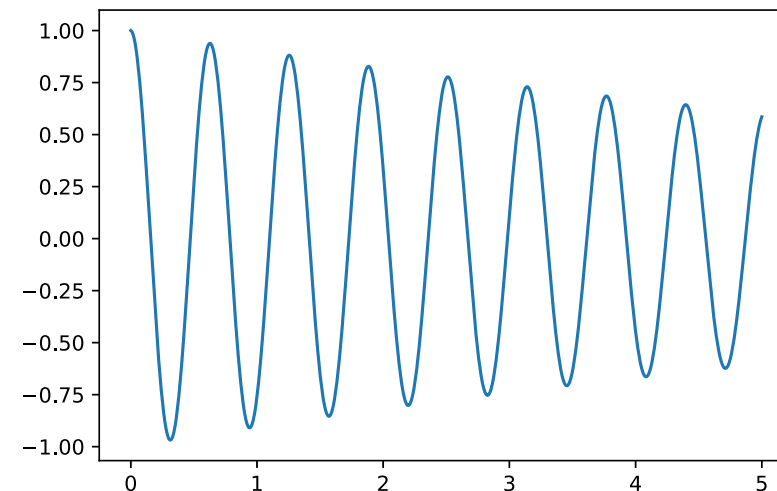
```
In [3]: f(t=0.1, alpha=1, omega=10)
```

```
Out[3]: 0.48888574340060287
```

Or produce a plot (in a function `plot_f` so it can be re-used for different parameters):

```
In [4]: def plot_f(alpha, omega):
        ts = linspace(0, 5, 500) # 500 points in interval [0, 5]
        ys = f(ts, alpha, omega)
        pylab.plot(ts, ys, '-')
```

```
In [5]: plot_f(alpha=0.1, omega=10) # call function and create plot
```



Use case 1: data analysis in notebook

- Explorative data analysis
- Convenient combination of processing, results and interpretation
- *Complete* capture of all computational steps
 - good record for *reproducibility* and *re-use*
 - ▶ FAIR data
- Through export to HTML, easy to share with collaborators & supervisors
- Scientists are confident drivers of this
 - example on the right from SCS instrument

```
In [3]: import ToolBox as tb
```

X-ray Absorption Spectroscopy

Step 1: Load data and align them by train id and pulse id

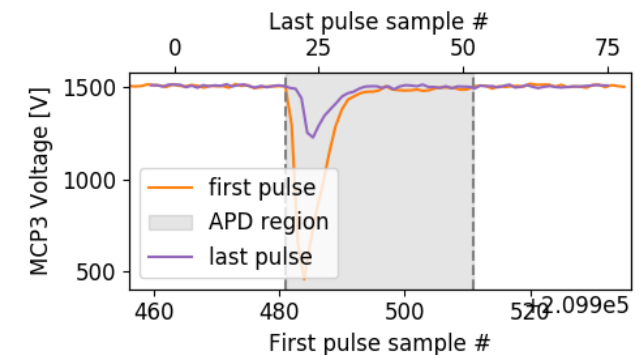
```
In [4]: proposalNB = 900074
semesterNB = 201930
runNB = 487
topic = 'SCS'
fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
             validate=True, display=False)
nrun = tb.matchXgmTimPulseId(run)
```

Checking run directory: /gpfs/xfel/exp/SCS/201930/p900074/raw/r0487/
No problems found

Step 2: check the pulse integration window

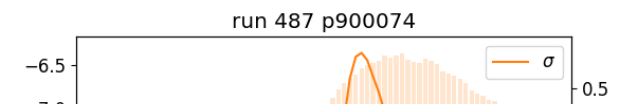
```
In [5]: tb.checkTimApdWindow(nrun, mcp=3)
```

no raw data for MCP3. Loading trace from MCP3



Step 3: bin the data and plot the XAS spectrum

```
In [6]: nrj = np.linspace(nrun.nrj.min(), nrun.nrj.max(), 80)
xas = tb.xas(nrun, nrj, plot=True)
```



Use case 2: notebooks as recipes

- Pre-populate notebook with cells to carry out a particular type of data analysis
 - provide a directory full of such recipes to users
 - users execute cells during beamtime and later

- Convenient compromise between
 - static recipe (=script)
 - interactive exploration

- Experience
 - keep code in notebook cells short and
 - move functionality into library (here “ToolBox“)
 - archive directory of modified recipes with data

```
In [3]: import ToolBox as tb
```

X-ray Absorption Spectroscopy

Step 1: Load data and align them by train id and pulse id

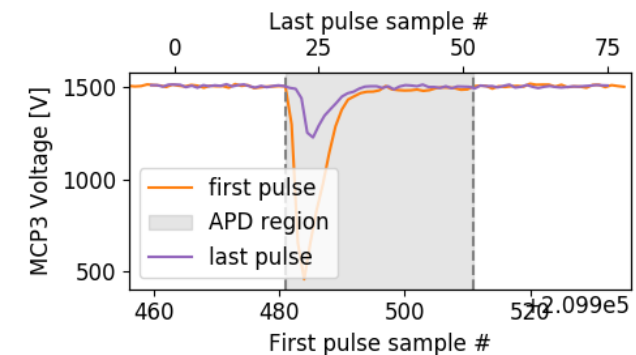
```
In [4]: proposalNB = 900074
semesterNB = 201930
runNB = 487
topic = 'SCS'
fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
             validate=True, display=False)
nrun = tb.matchXgmTimPulseId(run)
```

```
Checking run directory: /gpfs/xfel/exp/SCS/201930/p900074/raw/r0487/
No problems found
```

Step 2: check the pulse integration window

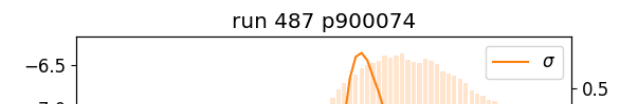
```
In [5]: tb.checkTimApdWindow(nrun, mcp=3)
```

```
no raw data for MCP3. Loading trace from MCP3
```



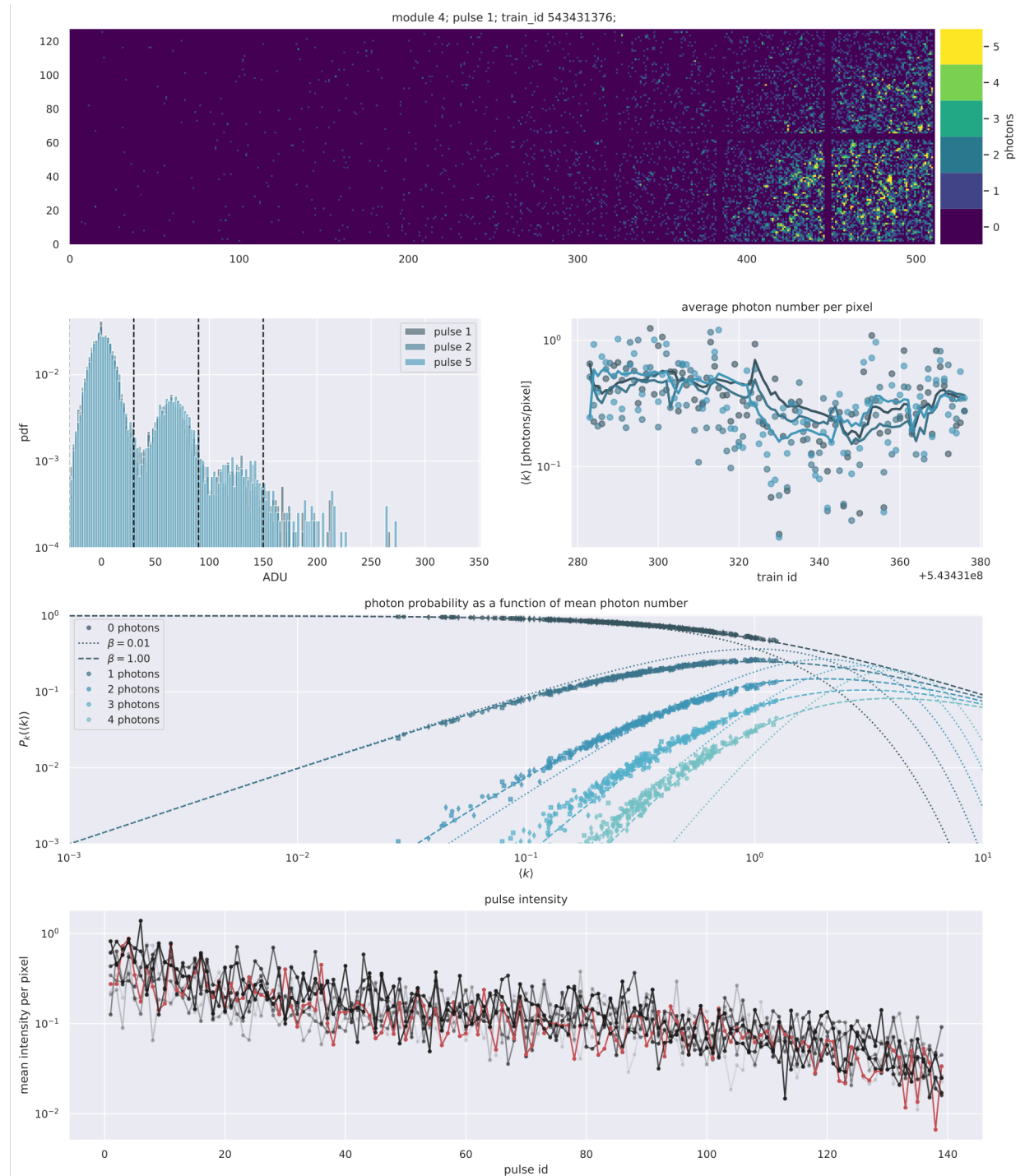
Step 3: bin the data and plot the XAS spectrum

```
In [6]: nrj = np.linspace(nrun.nrj.min(), nrun.nrj.max(), 80)
xas = tb.xas(nrun, nrj, plot=True)
```



Use case 3: online visualisation

- Use Jupyter Notebook to
 - fetch live experiment data from network
 - carry out some processing and update plots
- “plots” are chosen from library of processing and visualisation units
- Selection of plots, and distribution in rows and columns can be done by instrument scientist
 - Attractive because it provides flexibility
- Piloted at MID instrument at EuXFEL (Mario Reiser, Johannes Möller)



Use case 4: notebooks as a script

- Use Jupyter Notebook as a script
 - can execute using `nbconvert` to take commands in notebook, execute them, save resulting notebook
 - can create data files and plots in process
- Use case: detector calibration pipeline
 - use of `nbparameterize` to insert run parameters into notebook before execution
 - executes on HPC facility
 - after execution, turn notebook into pdf report
 - error messages embedded in output

Jupyter LPDChar_Darks_NBC (autosaved)
Logout

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3

Run
Markdown

Offset, Noise and Dead Pixels Characterization

Author: M. Karnevskiy, S. Hauf

This notebook performs re-characterize of dark images to derive offset, noise and bad-pixel maps. All three types of constants are evaluated per-pixel and per-memory cell.

The notebook will correctly handle veto settings, but note that if you veto cells you will not be able to use these offsets for runs with different veto settings -

The evaluated calibration constants a

The offset (O) is defined as the med cell (c).

The noise N is the standard deviat

The bad pixel mask is encoded as a

"OFFSET_OUT_OF_THRESHOLD":
Offset outside of bounds:

$$M(O)_{x,y} - \sigma(O)_{x,y} * \text{thre}$$

or offset outside of hard limits

"NOISE_OUT_OF_THRESHOLD":
Noise outside of bounds:

$$M(N)_{x,y} - \sigma(N)_{x,y} * \text{thre}$$

or noise outside of hard limits

"OFFSET_NOISE_EVAL_ERROR":
Offset and Noise both not *nan* values

Values: `thresholds_offset_sigma`, t

given as parameters.

```

In [ ] : cluster_profile = "noDB" #
in_folder = "/gpfs/exfel/ex
out_folder = "/gpfs/exfel/d
sequences = [0] # sequence
modules = [-1] # list of mo

capacitor_setting = 5 # cap
run_high = 112 # run number
run_med = 113 # run number
run_low = 114 # run number
                    
```

'Offset, Noise and Dead Pixels Characterization' Documentation, Release

4.9 Variation of offset and noise across Tiles and ASICs

The following plots show a standard deviation σ of the calibration constant. The plot of standard deviation across ASICs are shown overall tiles. The plot shows pixels of one ASIC (16×32), with a standard deviation across all ASICs of the module.

4.9.1 Variation of offset and noise across ASICs - High gain

4.9.2 Variation of offset and noise across ASICs - Medium gain

Use case 5: (remote) data analysis environment (JupyterHub)

JupyterHub

- users connect through browser and `https`
- serve notebooks on facility hardware
- use existing authentication systems
- connect to users' file storage

Example: JupyterHub at EuXFEL & DESY

- uses Maxwell HPC cluster

Popular with users:

- no software installation & browser of choice
- works locally and remotely the same

Model for European Open Science Cloud?

Maxwell Jupyter Job Options

Maxwell partitions:

Choice of GPU:

Note: For partitions without GPUs (or choice of GPUs) the GPU selection will be set to 'none'

Constraints:

Note: This will override GPU selections!

Number of Nodes:

Note: Number of nodes will be set to 1 on shared jhub partition!

Job duration:

Note: on the shared Jupyter partition (jhub) the time limit is always 7 days!

Launch modus:

Remote Notebook:

Node and GPU availability					
Partition	# nodes	# avail	# GPUs avail	# P100 avail	# V100 avail
jhub	3	3	0	0	0
maxwell	61	46	0	0	0
maxgpu	19	12	12	1	10
all	327	188	0	0	0
allgpu	88	67	67	48	10

Spawn

```
[6]: demo = InteractiveGeom(geom, run)
```

```
[7]: demo.interactive()
```

Use case 6: blending GUI and script

- JupyterWidgets provide graphical control elements in notebook
 - buttons, sliders etc trigger code execution and update of plot
- Useful for
 - data analysis of fixed type
 - data exploration of data sets
- Discussion
 - less powerful than, for example, QT GUI
 - popular with users due to
 - being embedded in notebook
 - no software installation (via JupyterHub)

Figure 1

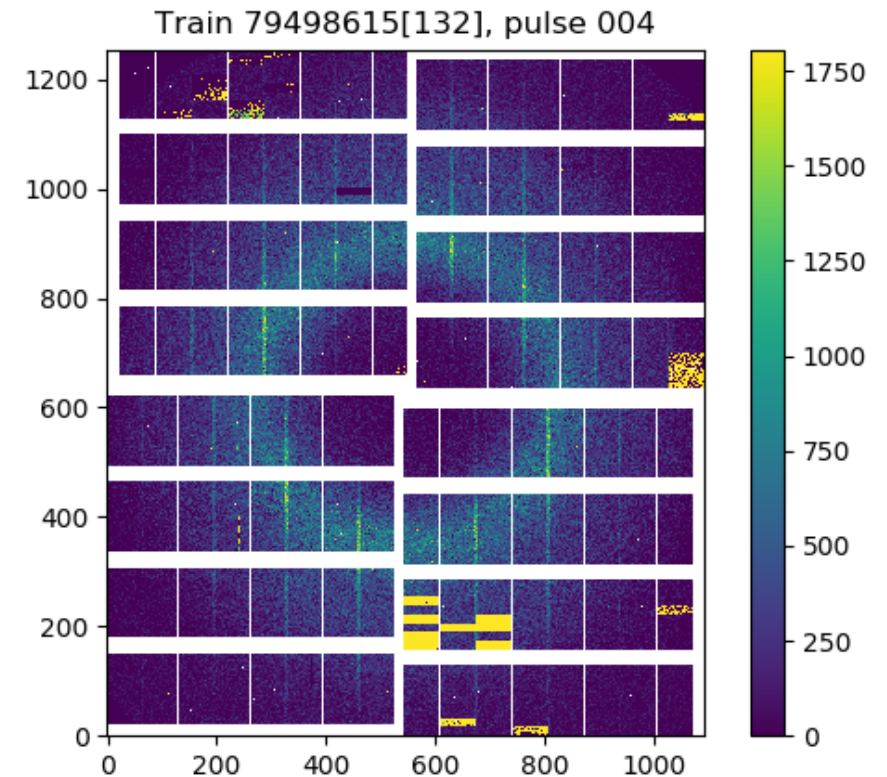
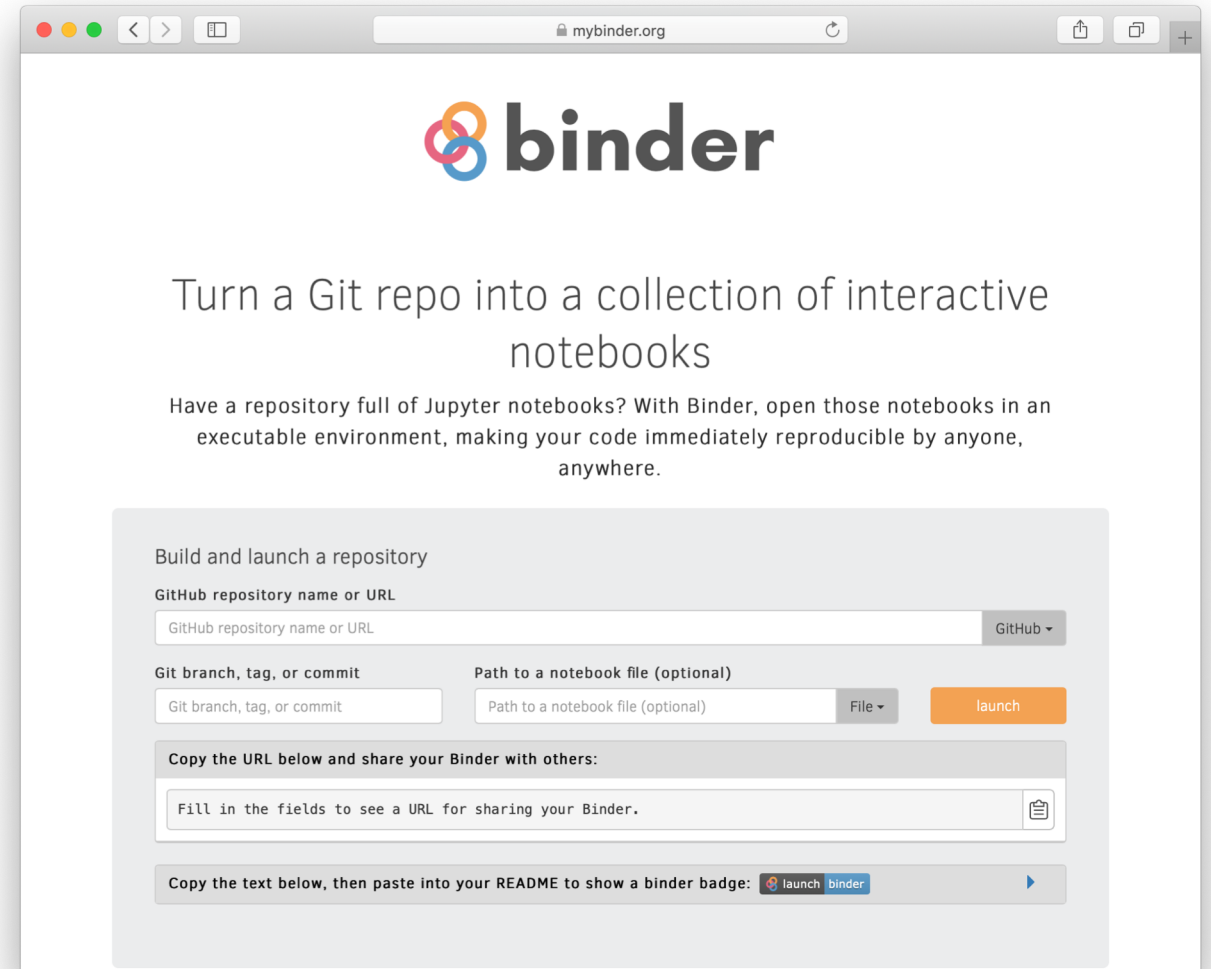


Figure 1 interactive control panel:

- Image Type: data gain mask
- Train Index: 132
- Pulse Index: 4
- Train ID: 79498615
- Reset on train
- Clip Range: 0 - 1806
- Colour Map: viridis
- < Prev Good
- Next Good >
- Status:

Binder project

- Given a (github) repository with
 - Jupyter notebooks
 - software requirements (Dockerfile, requirements.txt, environment.yml)
- Binder service
 - builds a container with the required software
 - starts Jupyter notebook server in that container offering the notebooks
- Binder project provides free pilot at
 - <https://mybinder.org>
- Institutional Binder instances are being deployed



Example: <https://github.com/fangohr/jupyter-demo>

The screenshot shows the GitHub repository page for 'fangohr / jupyter-demo'. The repository has 25 commits, 2 branches, 1 release, 1 contributor, and is licensed under BSD-3-Clause. The latest commit is by fangohr, titled 'Add Zenodo badge to version 1.0', made 7 days ago. A table of files is shown below the commit history:

File	Description	Last Modified
executed-notebooks	Save widget state	17 days ago
.gitignore	git to ignore checkpoint files	9 months ago
1-basics.ipynb	simplify example 1	9 months ago
2-widgets.ipynb	remove output from this notebook	17 days ago
LICENSE	Update for use in ICALEPCS	17 days ago
README.md	Add Zenodo badge to version 1.0	7 days ago
requirements.txt	order dependencies alphabetically	17 days ago

At the bottom of the repository page, there is a 'launch binder' button with a DOI of 10.5281/zenodo.3463132. Below this, it states: 'Most recent version of this repository is located at <https://github.com/fangohr/jupyter-demo>'.

Jupyter notebook demo repository

<https://mybinder.org/v2/gh/fangohr/jupyter-demo/master>

The screenshot shows the BinderHub landing page for the repository. The page features the Binder logo and the text: 'Starting repository: fangohr/jupyter-demo/master'. Below this, it states: 'The tool that powers this page is called BinderHub. It is an open source tool that you can deploy yourself.' At the bottom, there is a 'Build logs' section with a 'hide' button. The build logs show the following output:

```

Waiting for build to start...
Picked Git content provider.
Cloning into '/tmp/repo2dockerjaic8b8i'...
HEAD is now at 85f2283 Add Zenodo badge to version 1.0
Building conda environment for python=3.7Using PythonBuildPack builder
Building conda environment for python=3.7Building conda environment for python=3.7Step 1/51 : FROM
buildpack-deps:bionic
Fetching base image...

```

Example: <https://github.com/fangohr/jupyter-demo>

hub-binder.mybinder.ovh

jupyter Quit

Files Running Clusters

Select items to perform actions on them. Upload New ↕ ↻

	Name ↓	Last Modified	File size
<input type="checkbox"/>	0 /		
<input type="checkbox"/>	executed-notebooks	9 minutes ago	
<input type="checkbox"/>	1-basics.ipynb	9 minutes ago	2.9 kB
<input type="checkbox"/>	2-widgets.ipynb	9 minutes ago	3.48 kB
<input type="checkbox"/>	LICENSE	9 minutes ago	1.58 kB
<input type="checkbox"/>	README.md	9 minutes ago	1.68 kB
<input type="checkbox"/>	requirements.txt	9 minutes ago	17 B

hub-binder.mybinder.ovh

jupyter 2-widgets (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Code

```
In [ ]: %matplotlib inline
from numpy import exp, cos, linspace
import pylab
from ipywidgets import interact, interact_manual
```

Title of investigation

Mathematical model

Want to understand $f(t) = \exp(-\alpha t) \cos(\omega t)$

Code / Data

```
In [ ]: def f(t, alpha, omega):
        """Computes and returns exp(-alpha*t) * cos(omega*t)"""
        return exp(-alpha * t) * cos(omega * t)
```

Interactive exploration

We can execute the function for value of α and ω :

```
In [ ]: f(t=0.1, alpha=1, omega=10)
```

```
In [ ]: f(t=0.0, alpha=1, omega=10)
```

Although sometimes a plot is more instructive:

Use case 7: documenting software library

- Use notebook as chapter in documentation
 - supported by sphinx → html, pdf as usual
- Documentation easy to create:
 - enter commands in notebook
 - output is produced automatically
- updating docs means re-running notebook
- Can run regression test on documentation notebooks using Notebook VALidate (nbval)
- Can make documentation interactive using Binder

European XFEL Python data tools
latest

Search docs

Reading data files
AGIPD, LPD & DSSC data
Streaming data over ZeroMQ
Checking data files
AGIPD, LPD & DSSC Geometry
Command line tools
Data files format
Performance notes


EXAMPLES

Reading data with karabo_data
Accessing LPD data
Assembling detector data into images

Examining detector geometry
Detector geometry for AGIPD
DSSC detector geometry
Working with non-detector data
Comparing fast XGM data from two simultaneous recordings
Overall comparison of suppression ratio (with error)
Parallel processing with a virtual dataset

DEVELOPMENT

Release Notes


New: DigitalOcean Marketplace Deploy your favorite dev tools with 1-Click Apps.
Sponsored · Ads served ethically

Assembling detector data into images

The X-ray detectors at XFEL are made up of a number of small pieces. To get an image from the data, or analyse it spatially, we need to know where each piece is located.

This example reassembles some commissioning data from LPD, a detector which has 4 quadrants, 16 modules, and 256 tiles. Elements (especially the quadrants) can be repositioned; talk to the detector group to ensure that you have the right geometry information for your data.

```
[1]: %matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import h5py

from karabo_data import RunDirectory, stack_detector_data
from karabo_data.geometry2 import LPD_1MGeometry

[2]: run = RunDirectory('/gpfs/exfel/exp/FXE/201830/p900020/pr
run.info()

# of trains:      513
Duration:        0:00:51.200000
First train ID:  54861753
Last train ID:   54862265

14 detector modules (FXE_DET_LPD1M-1)
  e.g. module FXE_DET_LPD1M-1 0 : 256 x 256 pixels
    128 frames per train, 39040 total frames

0 instrument sources (excluding detectors):

0 control sources:

[3]: # Find a train with some data in
empty = np.asarray([])
for tid, train_data in run.trains():
    module_imgs = sum(d.get('image.data', empty).shape[0]
                      if module_imgs:
                        print(tid, module_imgs)
                        break

54861797 1792

[4]: tid, train_data = run.train_from_id(54861797)
print(tid)
for dev in sorted(train_data.keys()):
    print(dev, end='\t')
    try:
        print(train_data[dev]['image.data'].shape)
```


Use case 7: documenting software library

- Use notebook as chapter in documentation
 - supported by `sphinx` → html, pdf as usual
- Documentation easy to create:
 - enter commands in notebook
 - output is produced automatically
- updating docs means re-running notebook
- Can run regression test on documentation notebooks using `NoteBook VALidate (nbval)`
- Can make documentation interactive using Binder

European XFEL Python data tools

latest

Search docs

Reading data files

AGIPD, LPD & DSSC data

Streaming data over ZeroMQ

Checking data files

AGIPD, LPD & DSSC Geometry

Command line tools

Data files format

Performance notes

EXAMPLES

Reading data with `karabo_data`

Accessing LPD data

Assembling detector data into images

Examining detector geometry

Detector geometry for AGIPD

DSSC detector geometry

Working with non-detector data


Comparing fast XGM data from two simultaneous recordings

Overall comparison of suppression ratio (with error)

Parallel processing with a virtual dataset

DEVELOPMENT

Release Notes



New: DigitalOcean Marketplace Deploy your favorite dev tools with 1-Click Apps.

Sponsored · Ads served ethically

Load the geometry from a file, along with the quadrant positions used here.

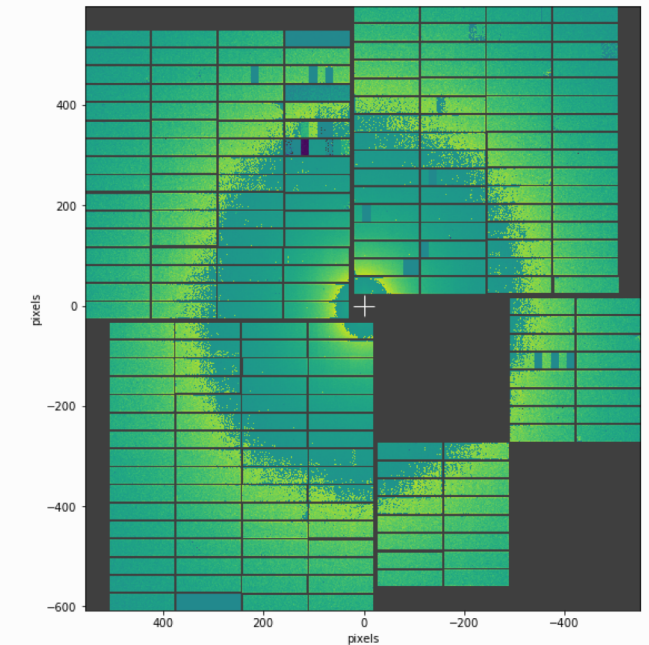
In the future, geometry information will be stored in the calibration catalogue.

```
[10] # From March 18; converted to XFEL standard coordinate di
quadpos = [(11.4, 299), (-11.5, 8), (254.5, -16), (278.5,
geom = LPD_1MGeometry.from_h5_file_and_quad_positions('lp
```

Reassemble and show a detector image using the geometry:

```
[11] geom.plot_data_fast(clip(modules_data[12], max=5000))
```

```
[11] <matplotlib.axes._subplots.AxesSubplot at 0x2b611ff3de48>
```



Reassemble detector data into a numpy array for further analysis. The areas without data have the special value `nan` to mark them as missing.

```
[12] res, centre = geom.position_modules_fast(modules_data)
print(res.shape)
plt.figure(figsize=(8, 8))
plt.imshow(clip(res[12, 250:750, 450:850], min=-400, max=!
```

(128, 1203, 1105)

```
[12] <matplotlib.image.AxesImage at 0x2b60ec9f4160>
```

Use case 8: reproducible publication

- Create github repository to complement publication
 - create one notebook per figure / main result
 - define software environment using Binder syntax

- Close to reproducible publication:
 - fully specified software environment
 - fully specified data analysis
 - data access problematic → PaNOSC, TUBPL02

- Zenodo for long term preservation
 - create Zenodo deposit for repository
 - cite Zenodo DOI in publication

The screenshot shows the GitHub interface for the repository 'maxalbert / paper-supplement-nanoparticle-sensing'. The repository is on the 'master' branch. The file list includes:

- ..
- style_sheets
- explanation_of_the_data_format.ip...
- fig_2_dipole_field_visualisation.ipy...
- fig_4_frequency_dependence_on...
- fig_7_frequency_change_vs_lateral...
- fig_8_frequency_change_vs_particl...
- fig_9a_dependence_of_frequency...
- fig_9b_dependence_of_frequency...
- fig_9c_comparison_of_frequency...
- style_helpers.py

Each file has a brief description of its content, such as 'Add missing stylesheets.', 'Add notebook explaining the data format.', and 'Update fig_2_dipole_field_visualisation.ipyn...'.

■ Example:

<https://github.com/maxalbert/paper-supplement-nanoparticle-sensing>

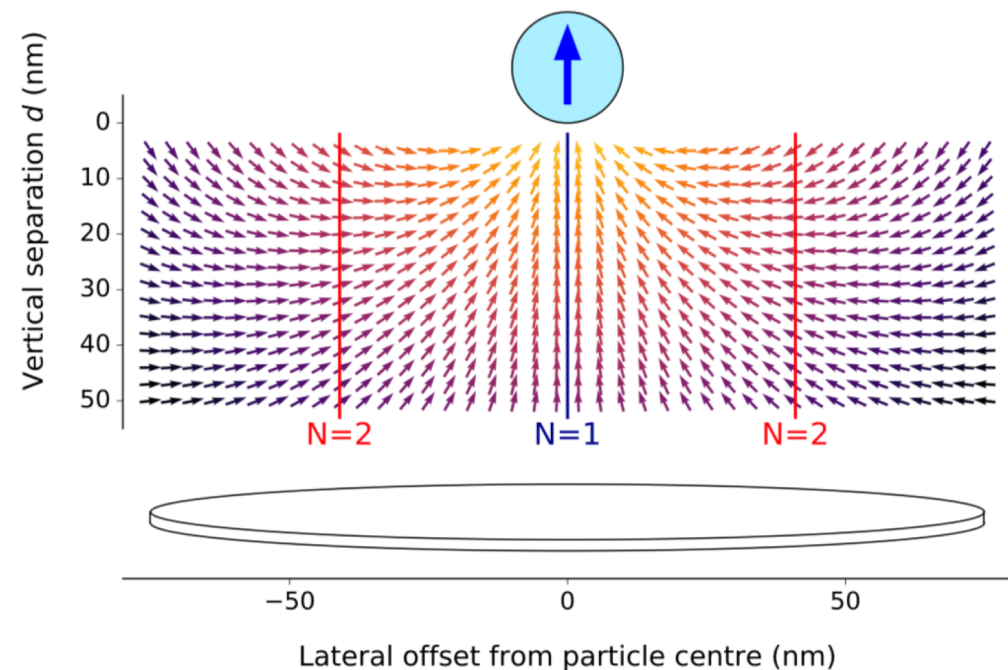


Figure 2. Vector field plot of the dipole field generated by a uniformly $+z$ -magnetised MNP. The vectors are scaled to uniform length with their colour indicating the field strength (orange is high and violet/black is low). The vertical lines correspond to the x -values where modes 1 and 2 have maxima in their spin precession amplitude (see figures 3(a)–(b)). A schematic of the nanodisc is shown at the bottom.

This repository Search Pull requests Issues Marketplace Explore

maxalbert / paper-supplement-nanoparticle-sensing Watch 1 Star 1 Fork 1

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Branch: master paper-supplement-nanoparticle-sensing / notebooks / fig_2_dipole_field_visualisation.ipynb Find file Copy path

maxalbert Update fig_2_dipole_field_visualisation.ipynb e062cd3 on 25 Apr 2016

2 contributors

296 lines (295 sloc) 214 KB Raw Blame History

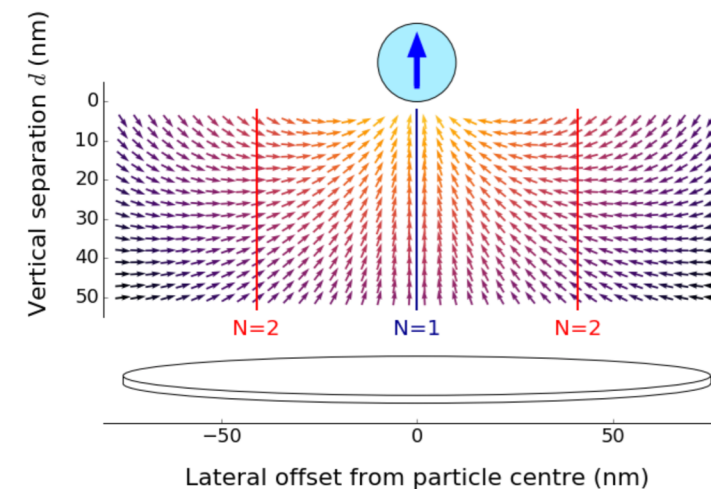
Fig. 2: Dipole Field Visualisation With Particle and Nanodisc

This notebook reproduces Fig. 2 in the paper, which shows a vector field plot of the dipole field generated by a uniformly magnetised nanoparticle together with a mockup of the nanodisc.

```
In [1]: import matplotlib.colors as colors
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import Ellipse, FancyArrow, Rectangle
from matplotlib.pyplot import cm

%matplotlib inline
```

```
draw_normalised_dipole_field(ax, dipole_x, dipole_y,
                             xmin fld, xmax fld, nx fld,
                             ymin fld, ymax fld, ny fld)
draw_vertical_line(ax, x= 0, ymin=ymin fld, ymax=yymax fld, annotation='N=1', color='darkblue')
draw_vertical_line(ax, x=-41, ymin=ymin fld, ymax=yymax fld, annotation='N=2', color='red')
draw_vertical_line(ax, x=+41, ymin=ymin fld, ymax=yymax fld, annotation='N=2', color='red')
```



Frequency-based nanoparticle sensing over large field ranges using the ferromagnetic resonances of a magnetic nanodisc: supplementary material

DOI [10.5281/zenodo.60605](https://doi.org/10.5281/zenodo.60605) preprint [arxiv:1604.07277](https://arxiv.org/abs/1604.07277) launch [binder](#) license [MIT](#)

This repository accompanies the paper "*Frequency-based nanoparticle sensing over large field ranges using the ferromagnetic resonances of a magnetic nanodisc*", published in *Nanotechnology*, Volume 27, Number 45. It provides the data underlying the figures in the paper, as well as [Jupyter](#) notebooks to reproduce those figures.

The latest version of this repository can be found at <https://github.com/maxalbert/paper-supplement-nanoparticle-sensing>

Authors: Maximilian Albert, Marijan Beg, Dmitri Chernyshenko, Marc-Antonio Bisotti, Rebecca L. Carey, Hans Fangohr and Peter Metaxas.

Contents

The directory `notebooks/` contains Jupyter notebooks for the relevant figures in the paper. On Github you can view them directly in the browser:

- [Fig. 2: Dipole field visualisation](#)
- [Fig. 4: Frequency dependence on external field strength](#)
- [Fig. 7: Frequency change vs. lateral particle position](#)

Summary

- Jupyter Notebook and ecosystem increasingly popular in academia and industry
 - Welcome by users, and driven by users
 - Combines code, results and interpretation
 - Strong technology candidate for European Open Science Cloud and remote analysis portals?

- A quickly moving field
 - significant potential to enable better science
 - many open questions

- This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 654220 (PaNOSC).

- Contact: Hans Fangohr, hans.fangohr@xfel.eu, <http://fangohr.github.io>, [@ProfCompMod](https://twitter.com/ProfCompMod)