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

ICALEPCS 2019, New York, US, 8 October 2019

Data Exploration and Analysis with Jupyter Notebooks

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. Fortmann[4], D. Fulla Marsa, G. Giovanetti, D. Goeries, S. Hauf, D. G. Hickin, T. Jarosiewicz[5], E. Kamil, M. Karnevskiy, Y. Kirienko, A. Klimovskaia, T. A. Kluyver, M. Kuster, L. Le Guyader, A. Madsen, L. G. Maia, D. Mamchyk, 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. Spirzewski[5], J. Sztuk, J. Szuba, S. Trojanowski[5], 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, Andrzeja 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:

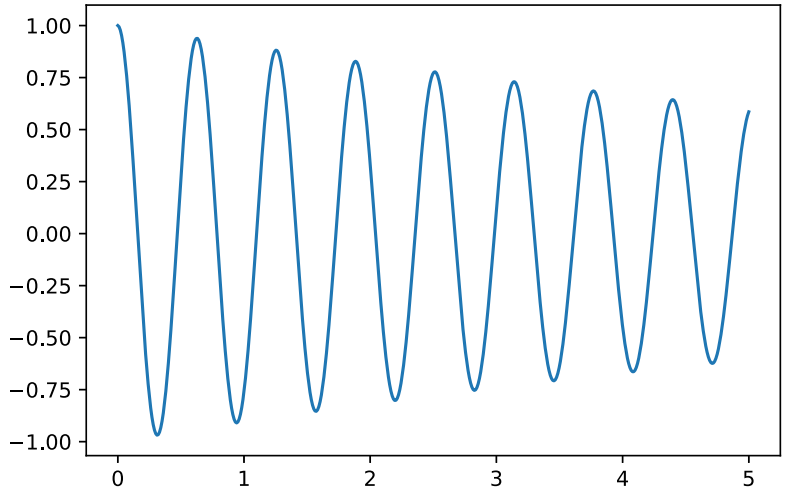
In [3]:
```
f(t=0.1, alpha=1, omega=10)
```

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:

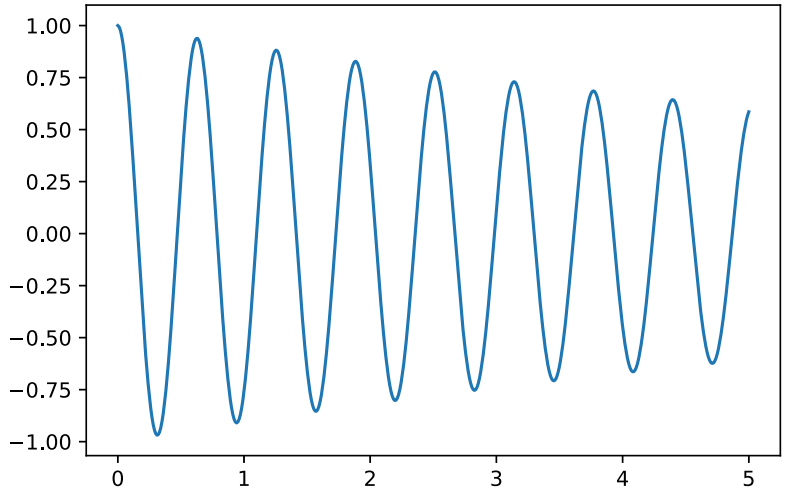
In [3]:
```
f(t=0.1, alpha=1, omega=10)
```

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

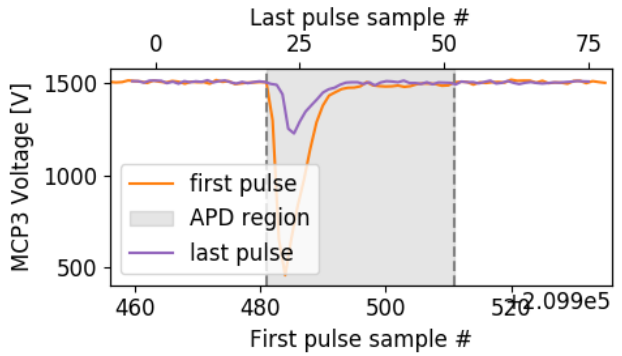
In [4]:
```
proposalNB = 900074
semesterNB = 201930
runNB = 487
topic = 'SCS'
fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
              validate=True, display=False)
nrun = tb.matchXgmTimPulseId(run)
```

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

Step 2: check the pulse integration window

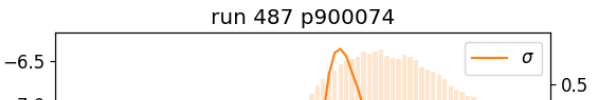
In [5]:
```
tb.checkTimApdWindow(nrun, mcp=3)
```

```
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

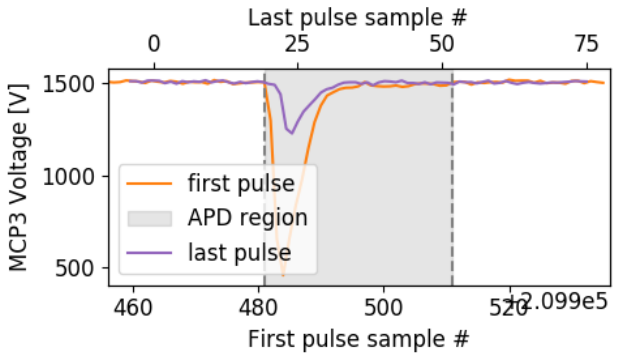
In [4]:
```
proposalNB = 900074
semesterNB = 201930
runNB = 487
topic = 'SCS'
fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
              validate=True, display=False)
nrun = tb.matchXgmTimPulseId(run)
```

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

Step 2: check the pulse integration window

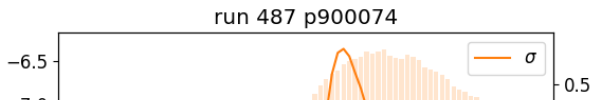
In [5]:
```
tb.checkTimApdWindow(nrun, mcp=3)
```

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



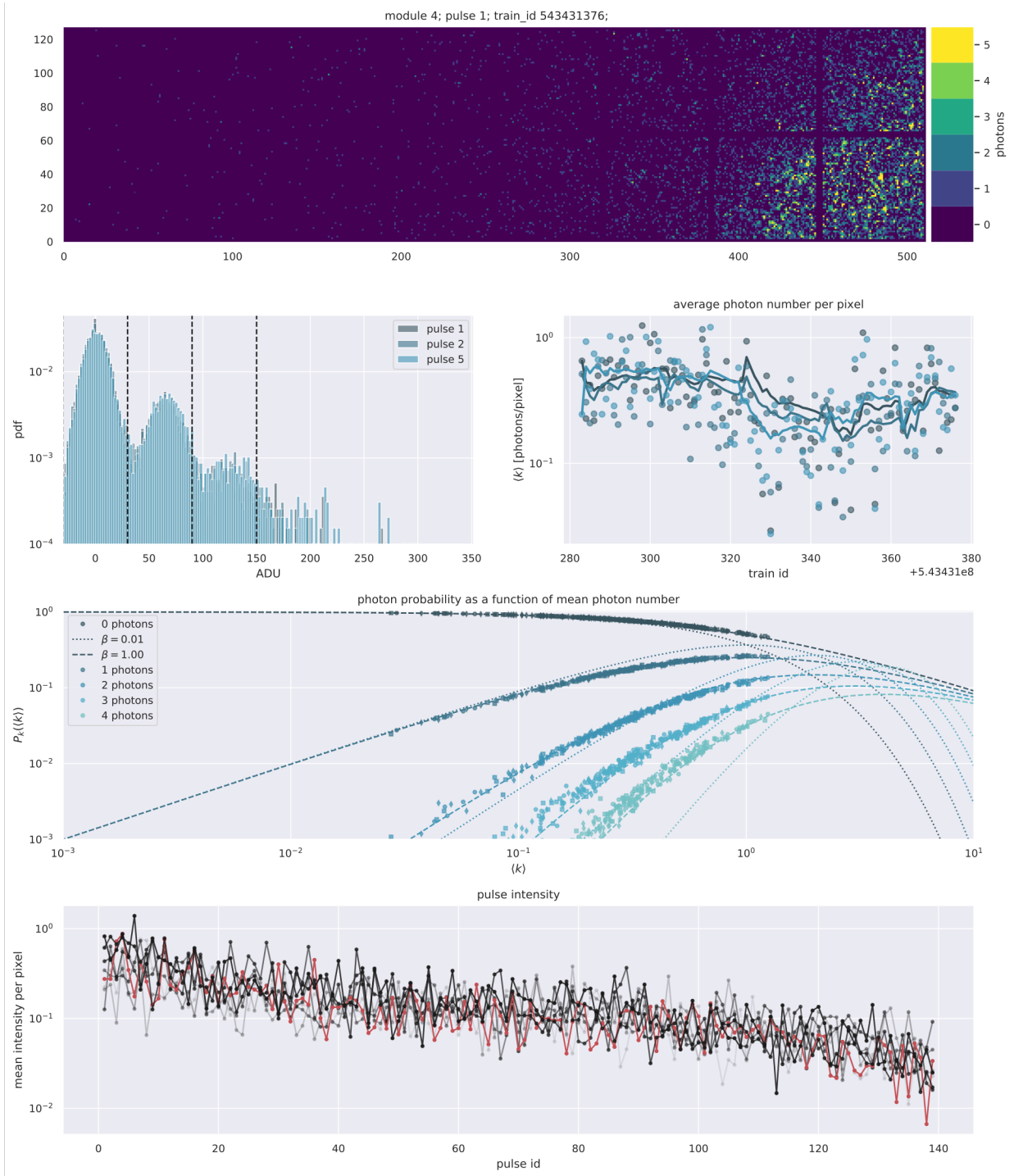
Step 3: bin the data and plot the XAS spectrum

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



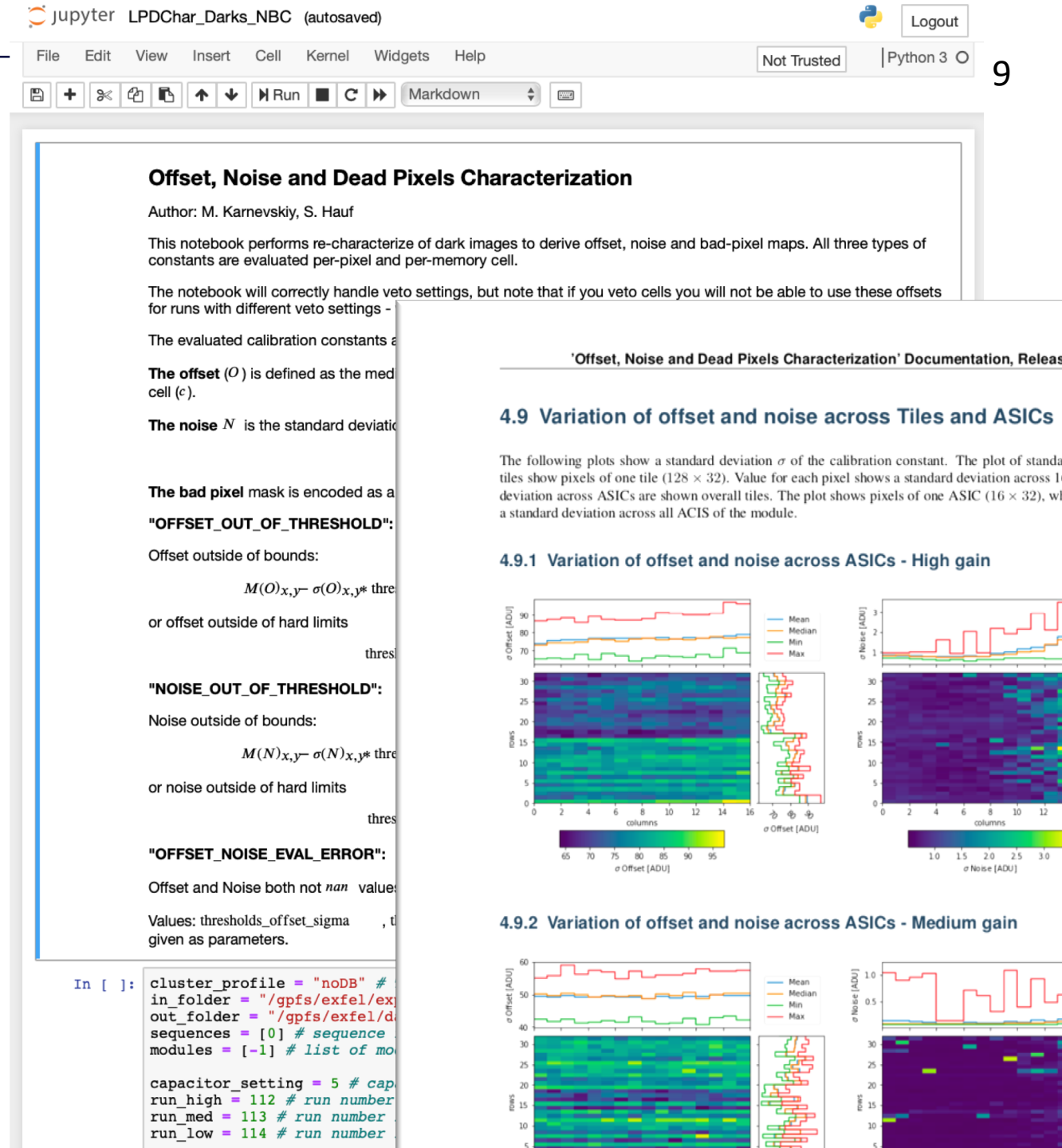
Use case 3: online visualisation

- 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)



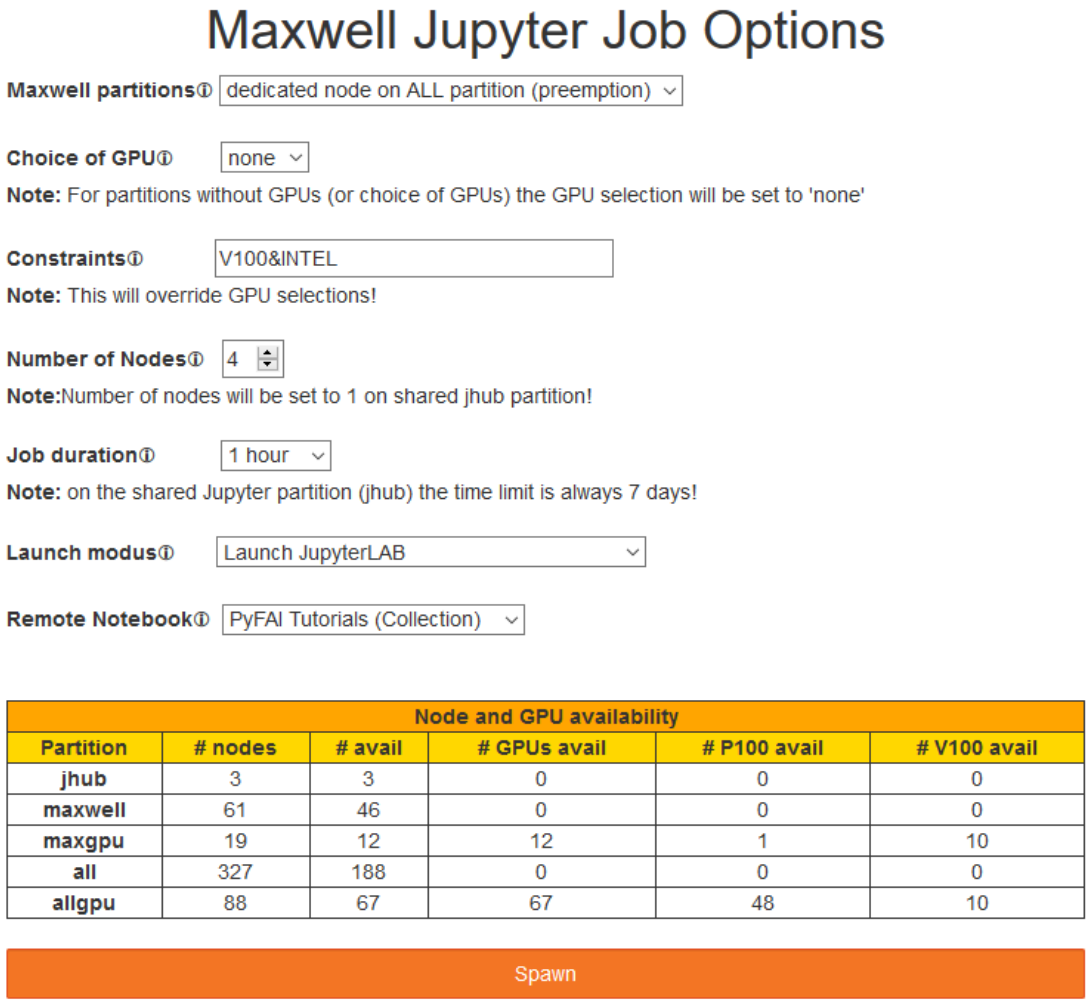
Use case 4: notebooks as a script

- 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 `nbparametrize` to insert run parameters into notebook before execution
  - executes on HPC facility
  - after execution, turn notebook into pdf report
  - error messages embedded in output



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
  - sses Maxwell HPC cluster
- Popular with users:
  - no software installation & browser of choice
  - works locally and remotely the same
- Model for European Open Science Cloud?

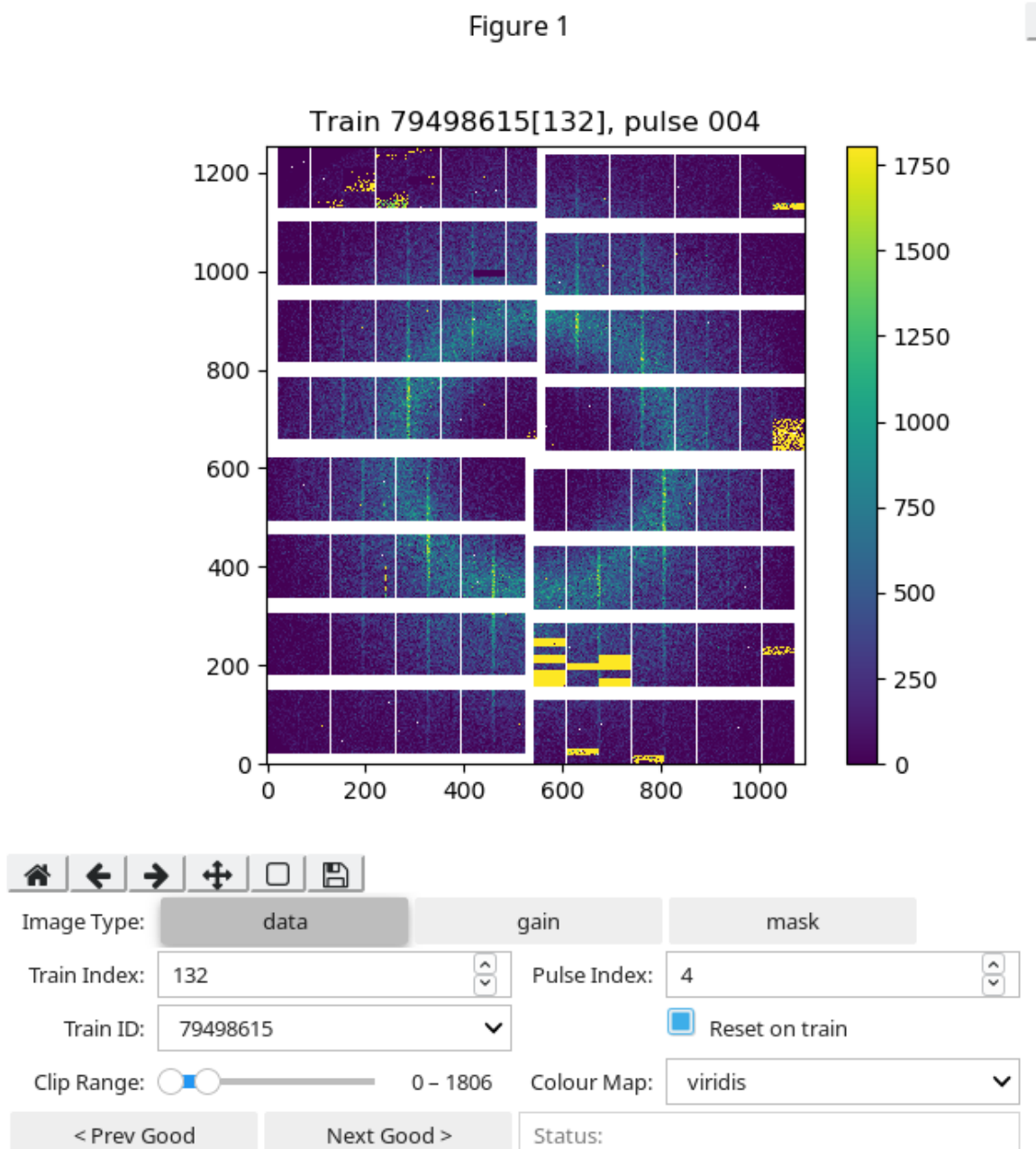


Use case 6: blending GUI and script

- 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)

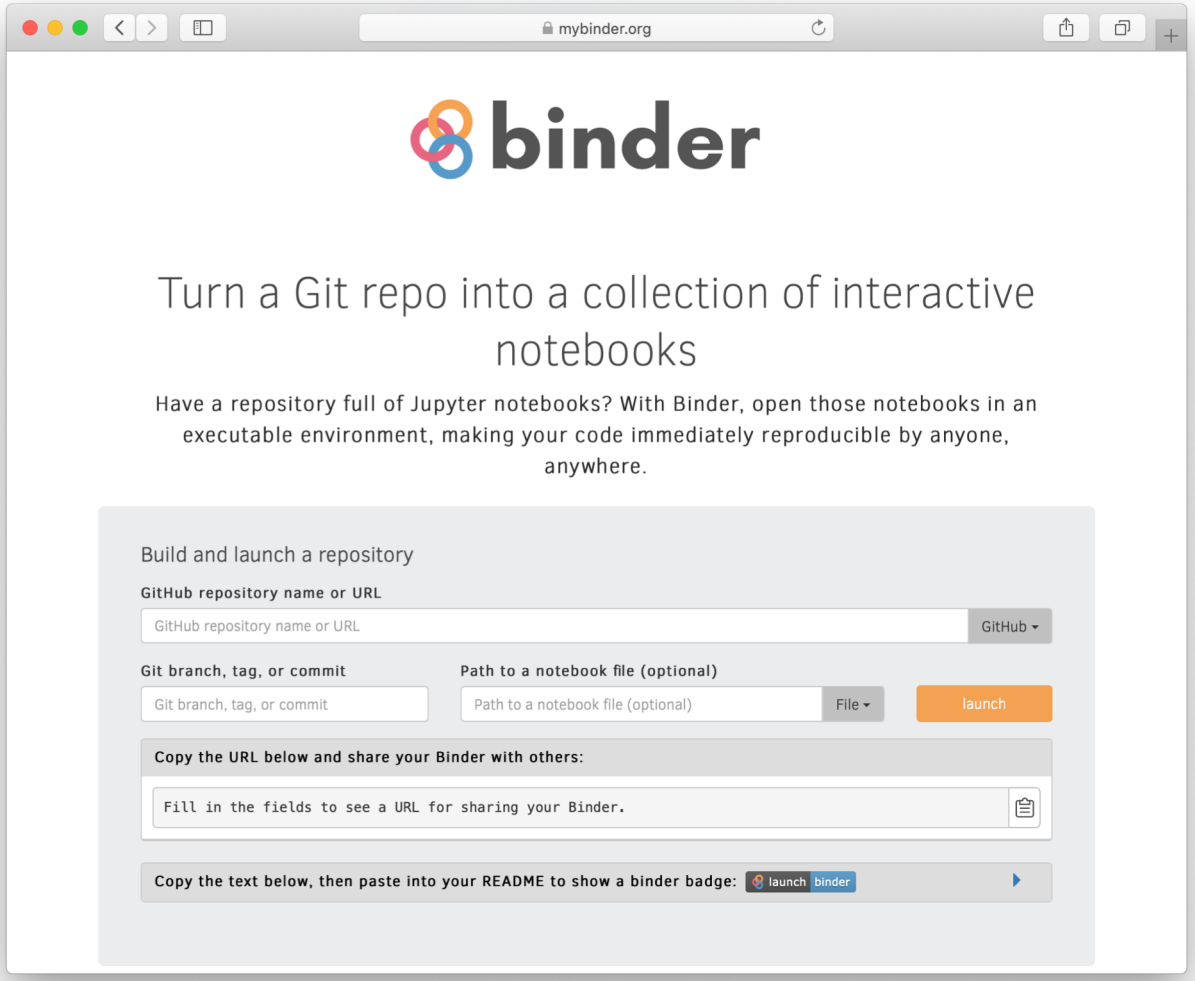
```
[6]: demo = InteractiveGeom(geom, run)

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



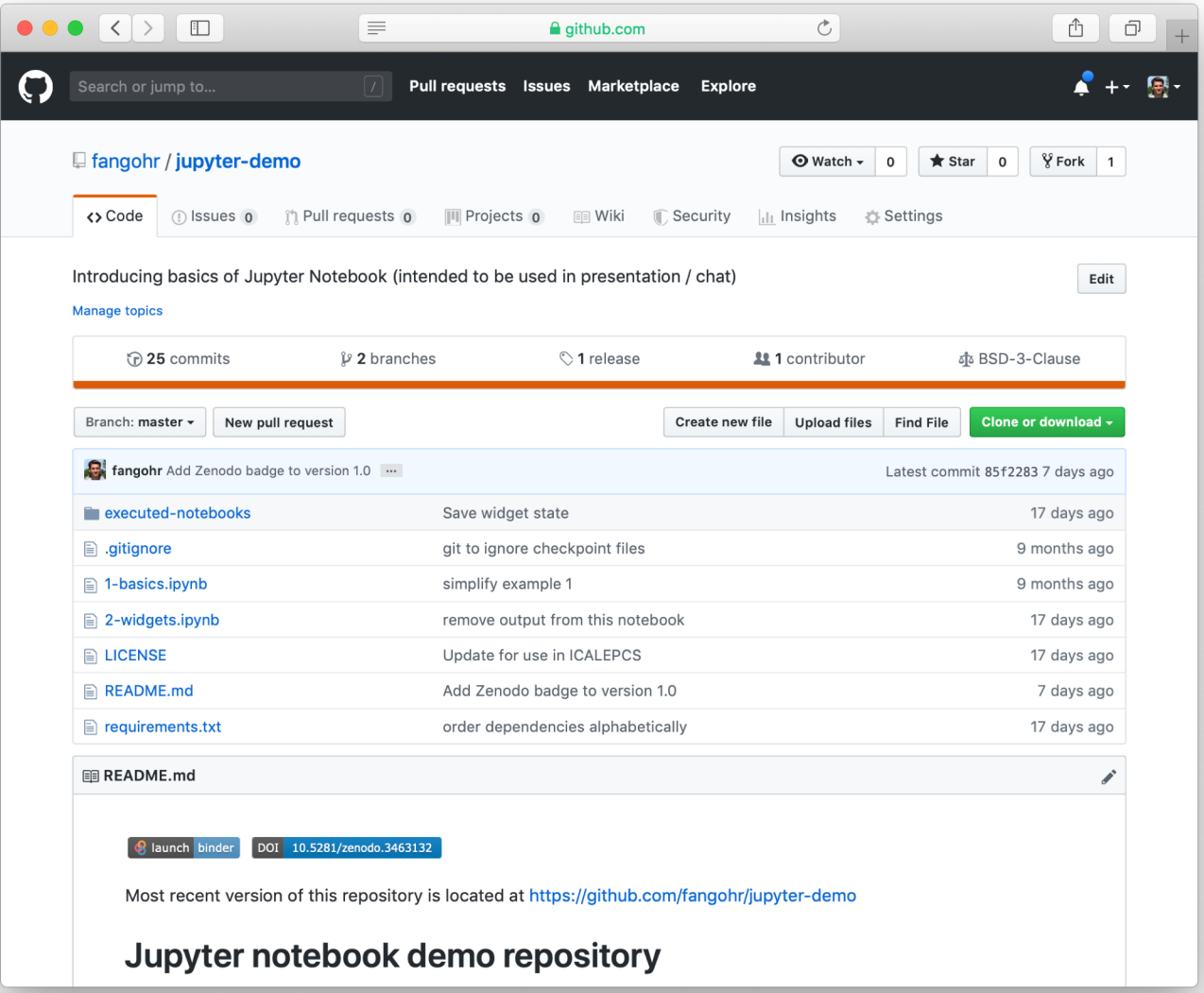
Binder project

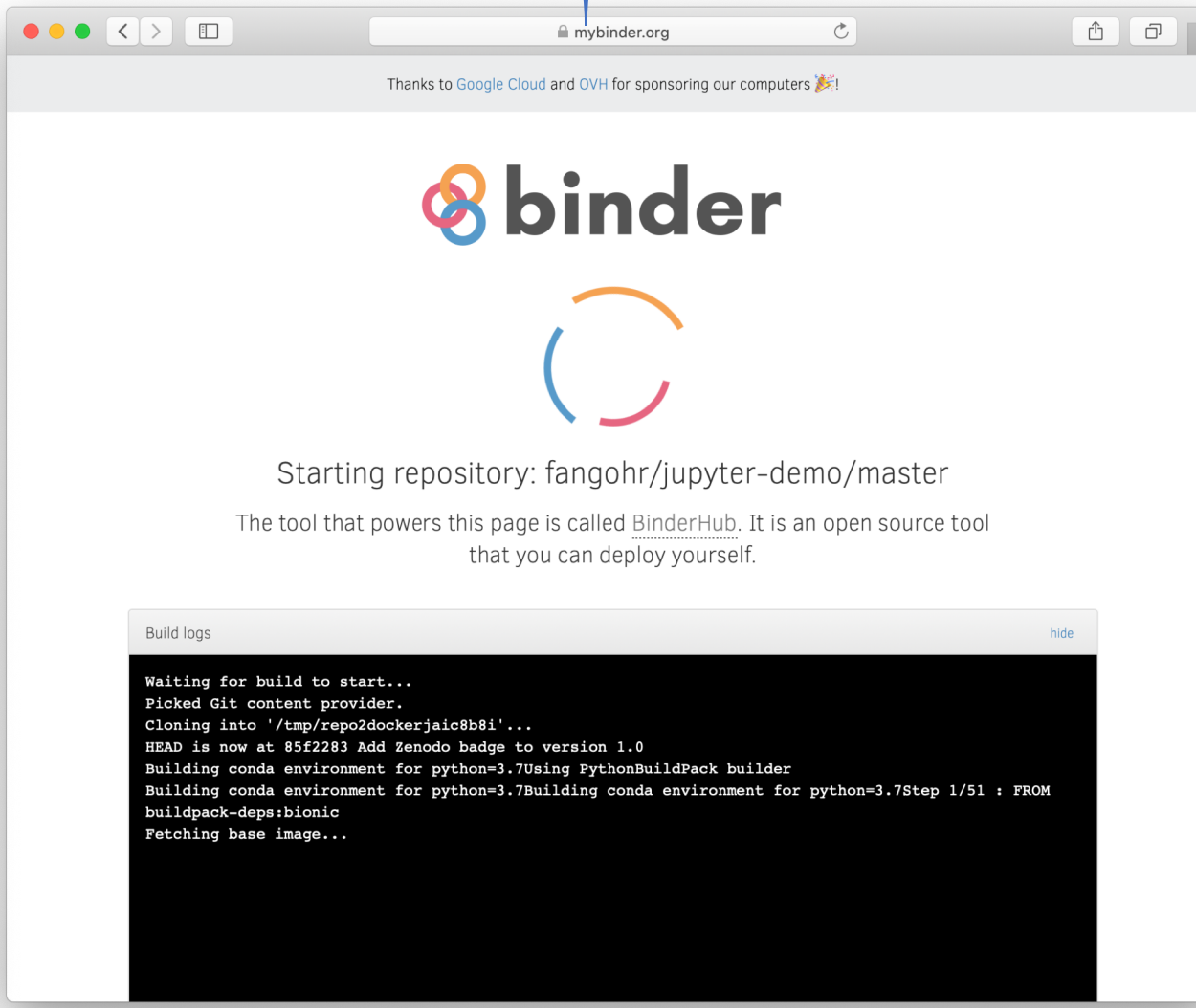
- Given a (github) repository with
  - Jupyter notebooks
  - software requirements (`Dockerfile`, `requirements.txt`, `environment.yml`)
- Binder service
  - builds a container with the required software
  - starts Jupyter notebook server in that container offering the notebooks
- Binder project provides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



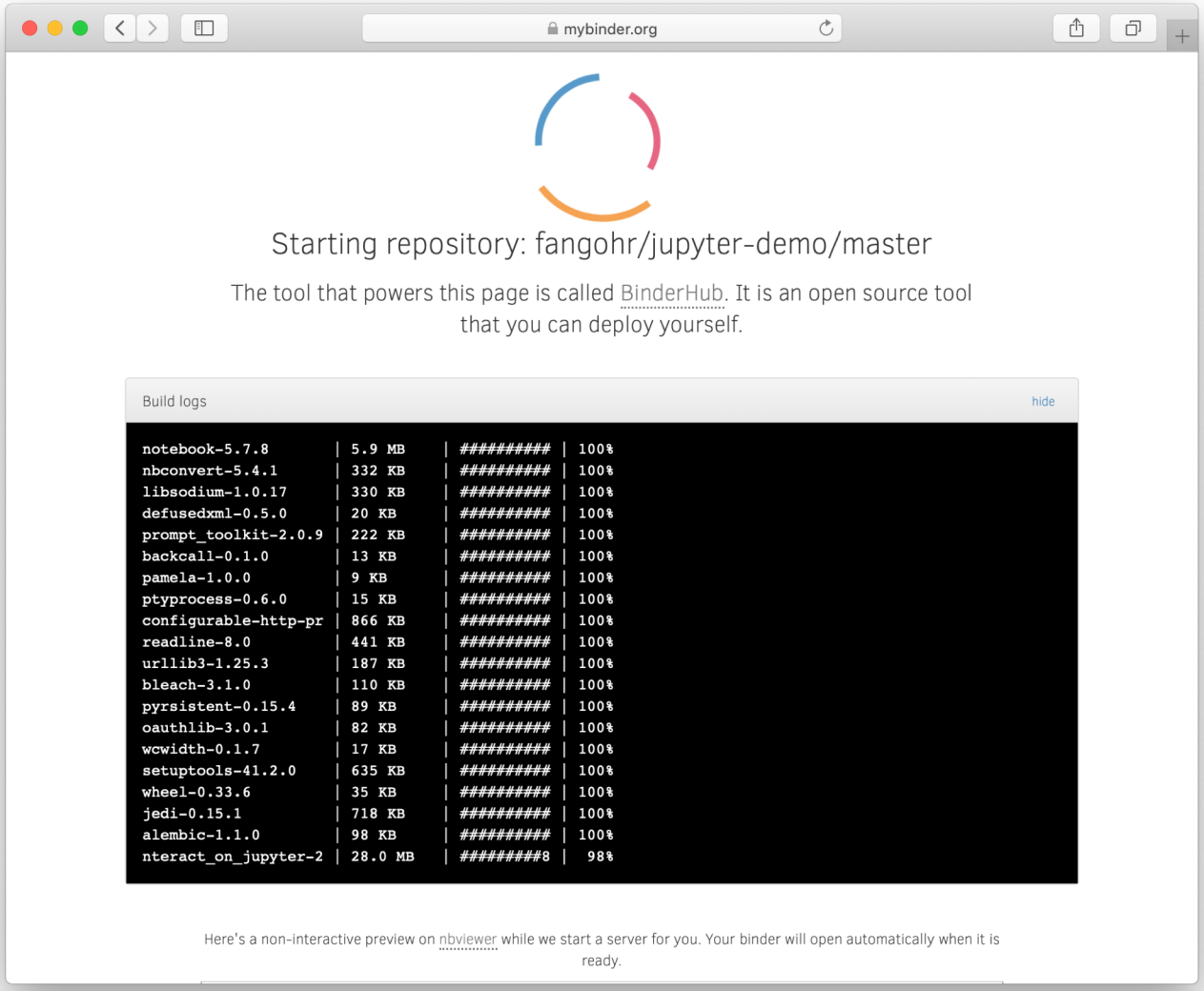
Example: https://github.com/fangohr/jupyter-demo

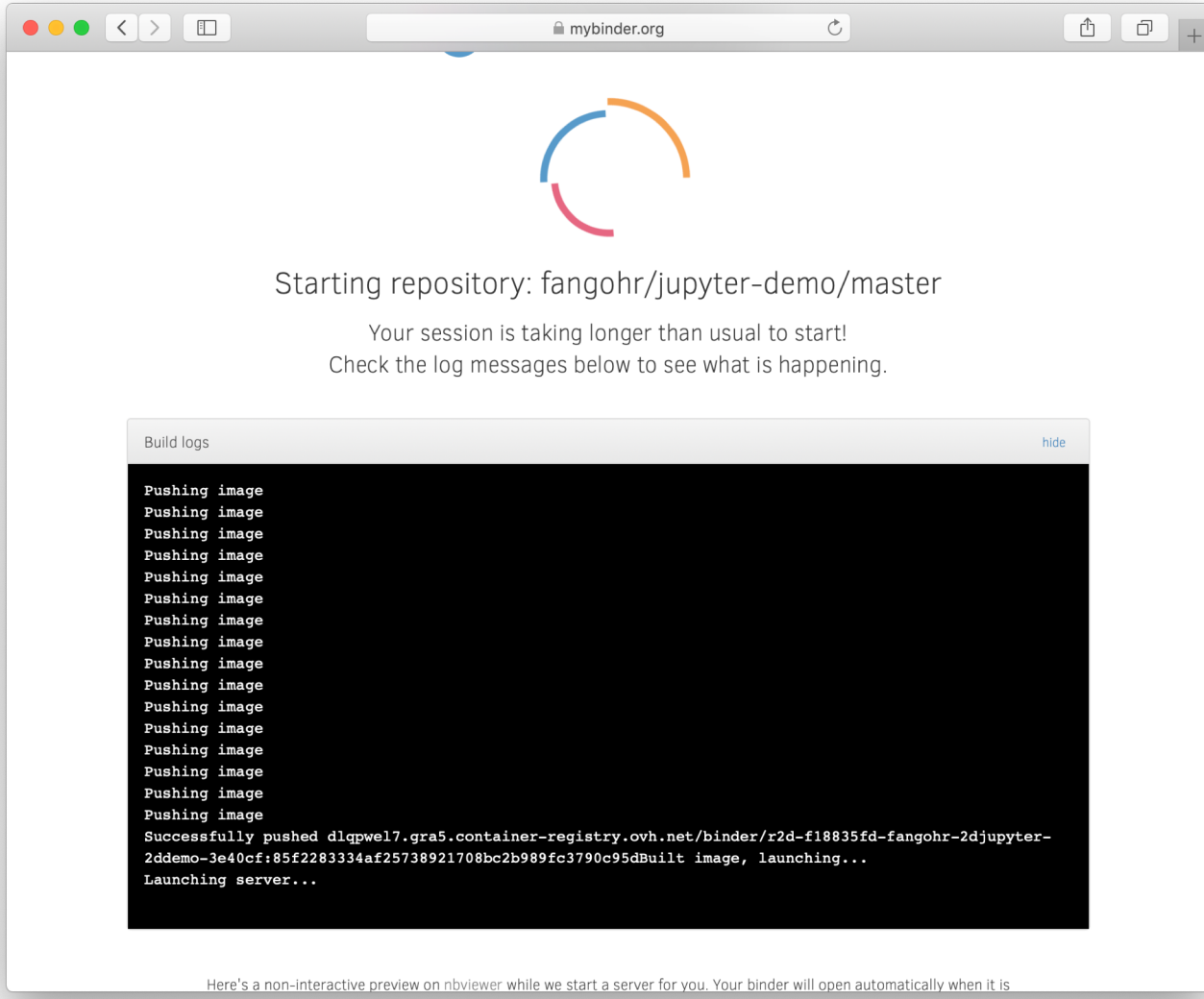
https://mybinder.org/v2/gh/fangohr/jupyter-demo/master



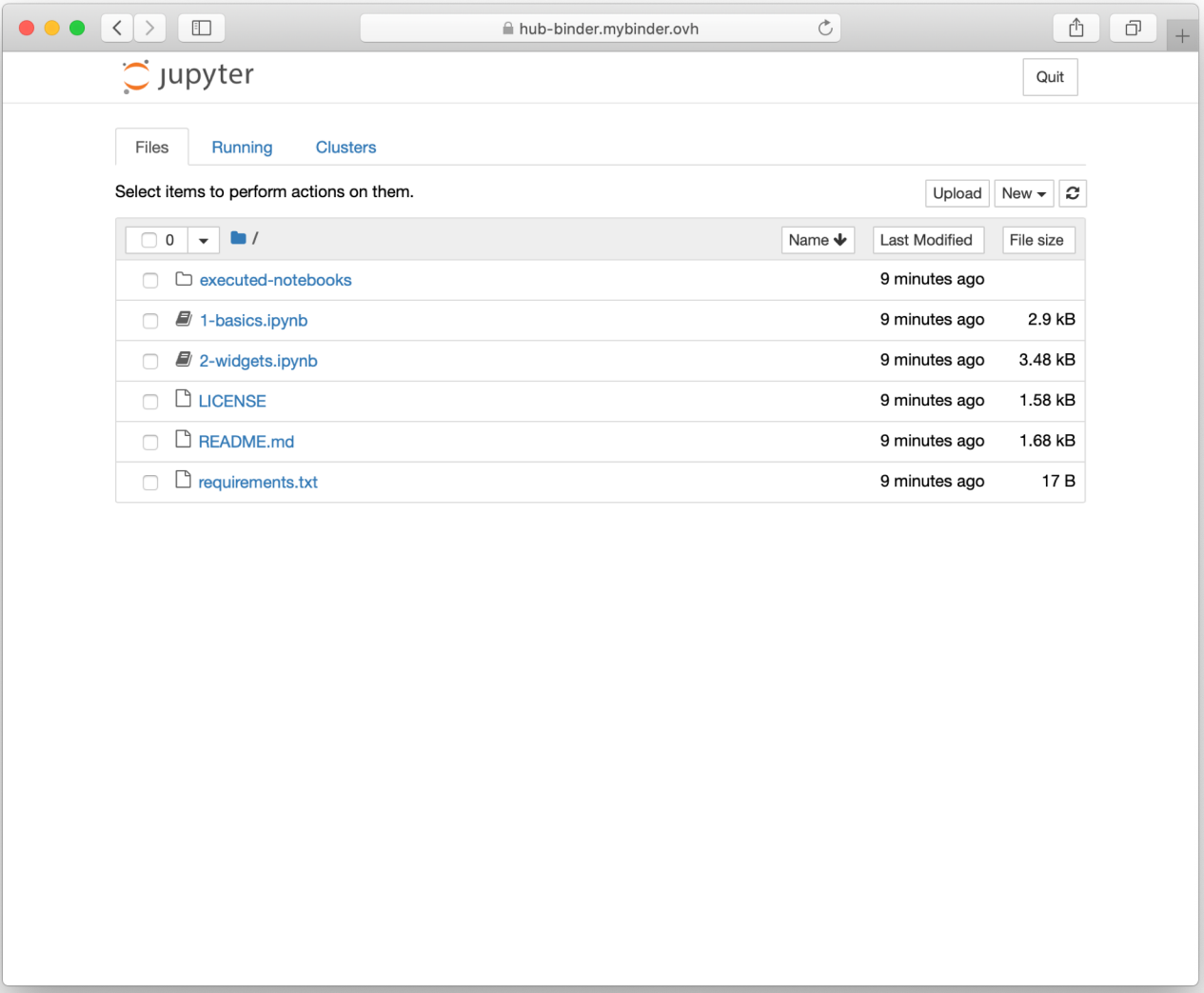


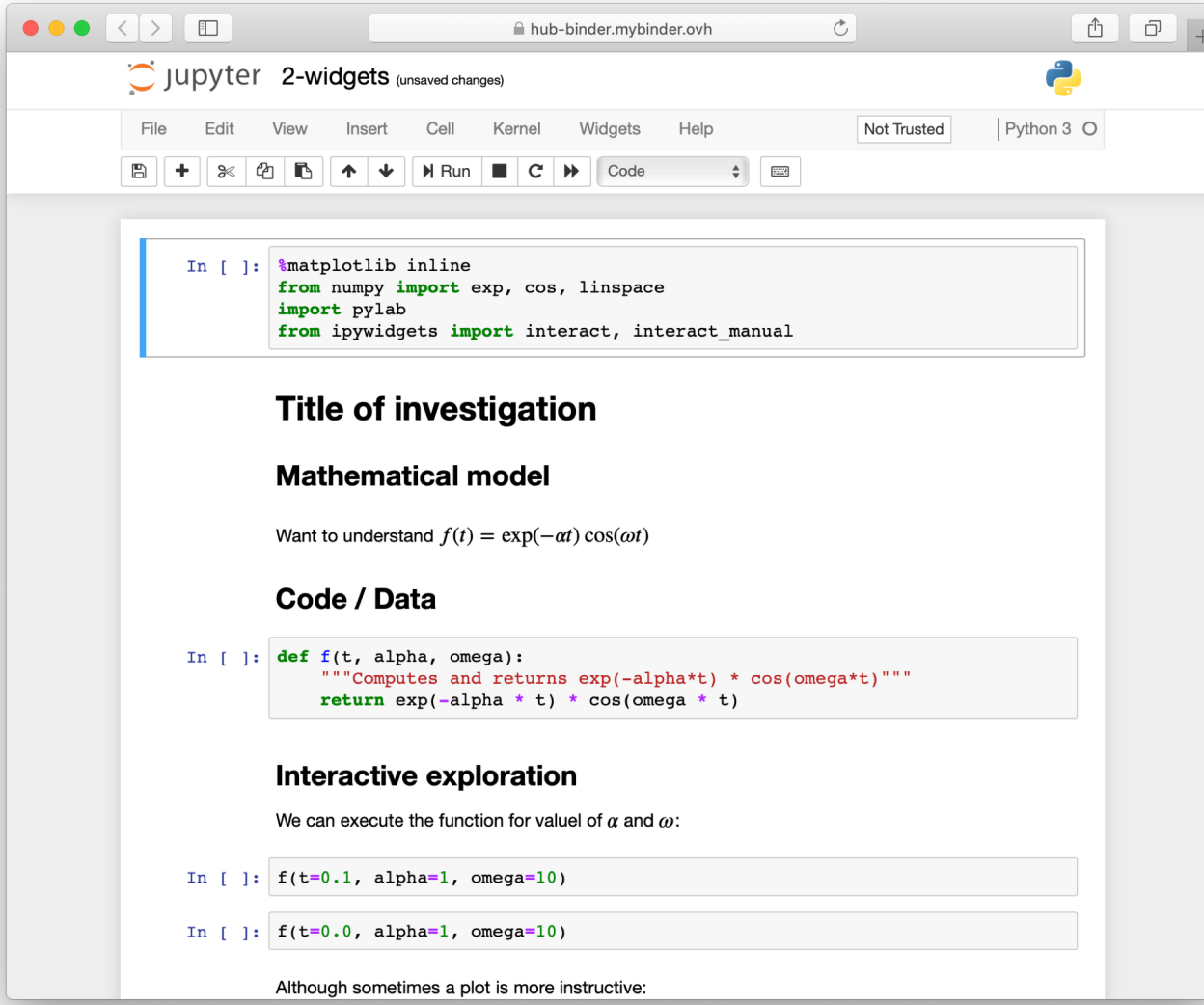
Example: https://github.com/fangohr/jupyter-demo





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

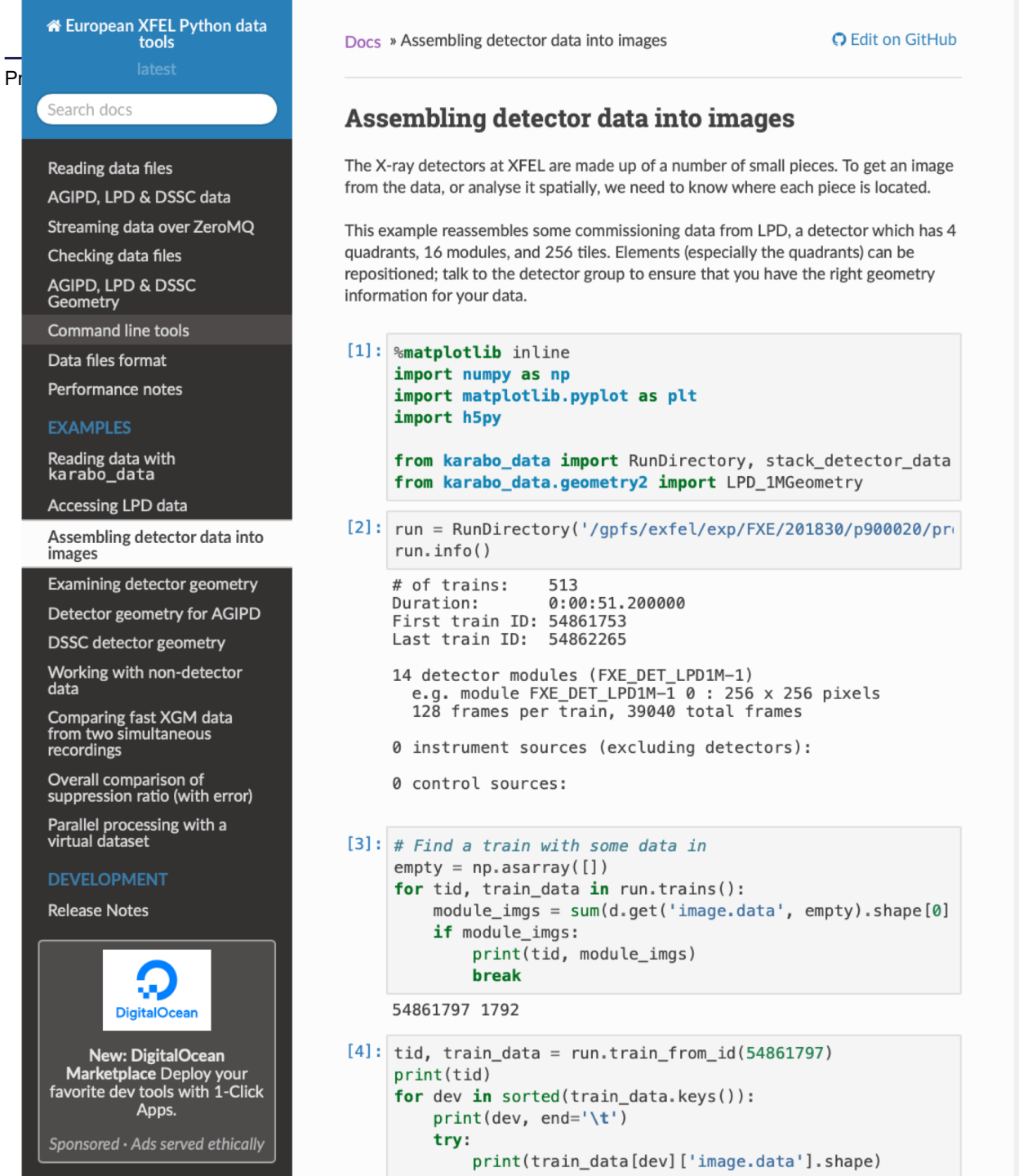




Example: https://github.com/fangohr/jupyter-demo

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



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

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

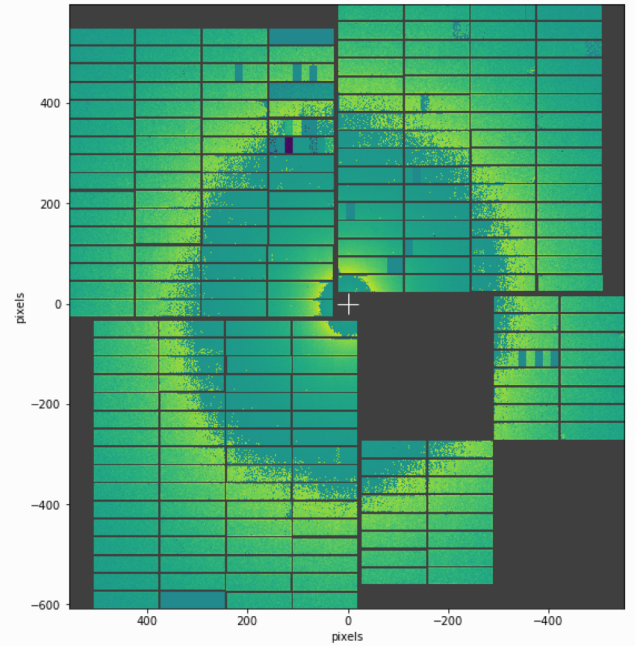
In the future, geometry information will be stored in the calibration catalogue.

```
[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>
```

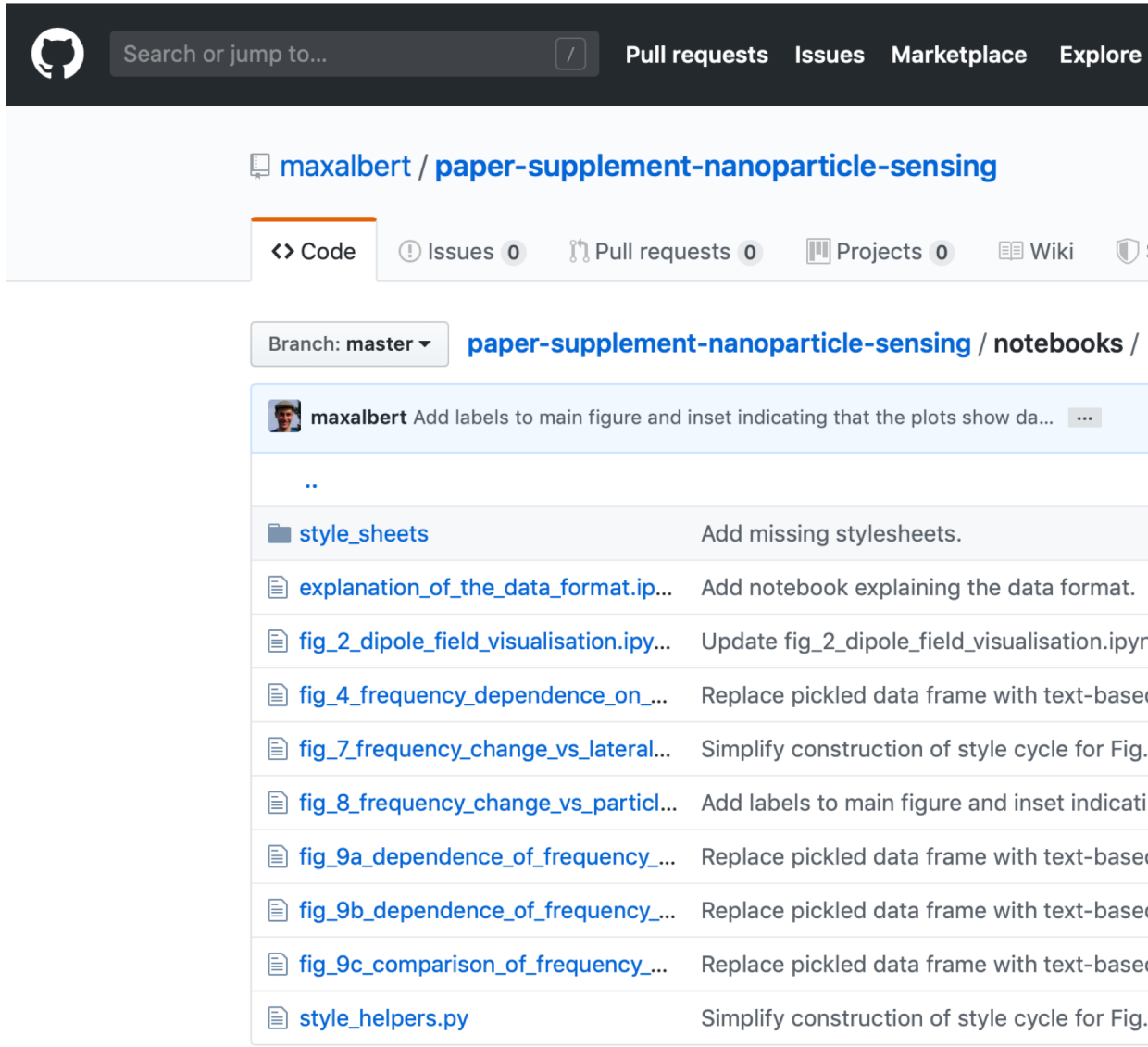


Reassemble detector data into a numpy array for further analysis. The areas without data have the special value ``*nan*`` to mark them as missing.

```
[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



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

Nanotechnology **27** (2016) 455502

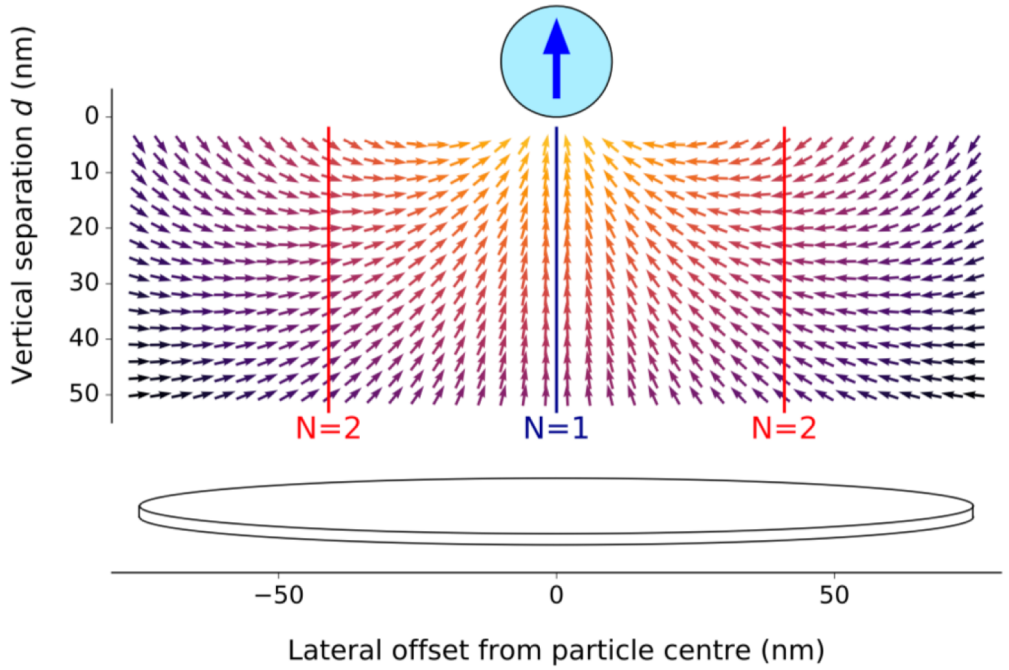
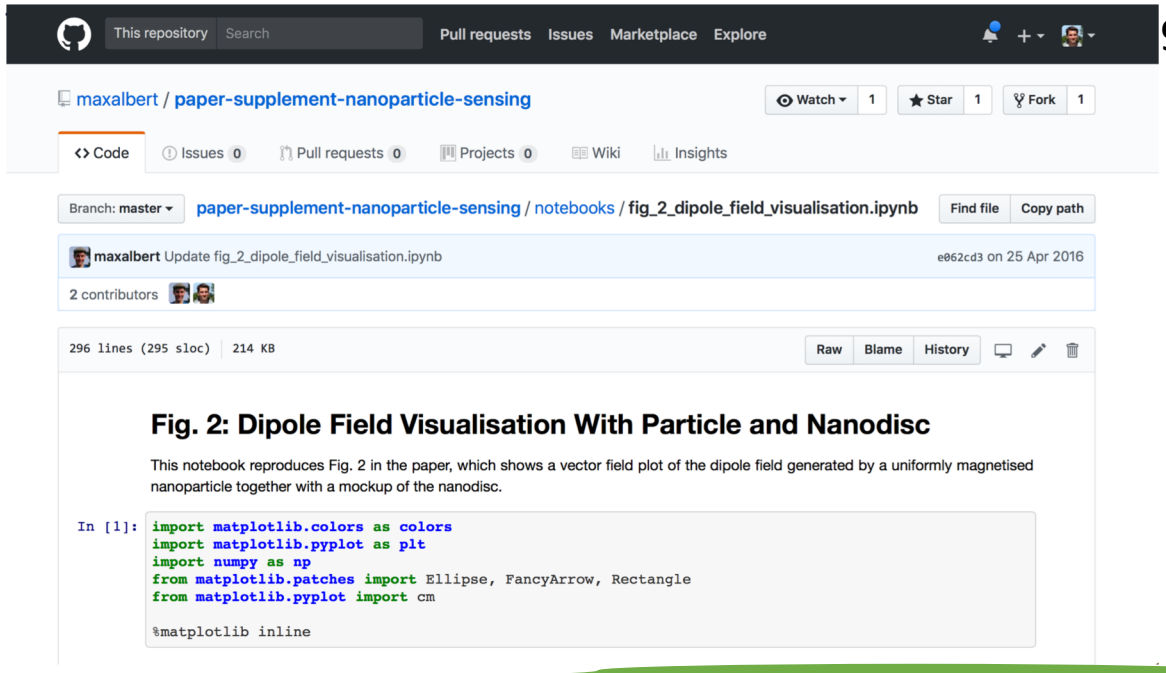
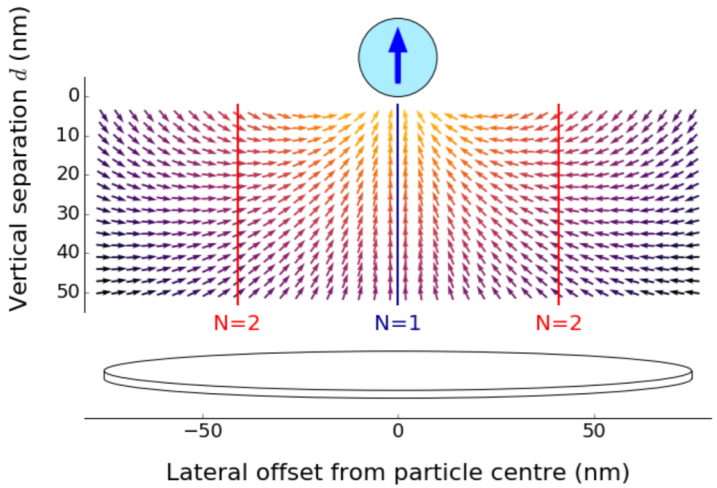


Figure 2. Vector field plot of the dipole field generated by a uniformly $+z$-magnetised MNP. The vectors are scaled to uniform length with their colour indicating the field strength (orange is high and violet/black is low). The vertical lines correspond to the x-values where modes 1 and 2 have maxima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




```
                                xmin_fld, xmax_fld, nx_fld,
                                ymin_fld, ymax_fld, ny_fld)
draw_vertical_line(ax, x=  0, ymin=ymin_fld, ymax=ymax_fld, annotation='N=1', color='darkblue')
draw_vertical_line(ax, x=-41, ymin=ymin_fld, ymax=ymax_fld, annotation='N=2', color='red')
draw_vertical_line(ax, x=+41, ymin=ymin_fld, ymax=ymax_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 preprint arxiv: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