

# Reproducible Science with Jupyter

Carsten Fortmann-Grote

Scientific Computing Services

Max-Planck-Institut for Evolutionary Biology, Plön, Germany

[carsten.fortmann-grote@evolbio.mpg.de](mailto:carsten.fortmann-grote@evolbio.mpg.de)



**MAX PLANCK INSTITUTE  
FOR EVOLUTIONARY BIOLOGY**

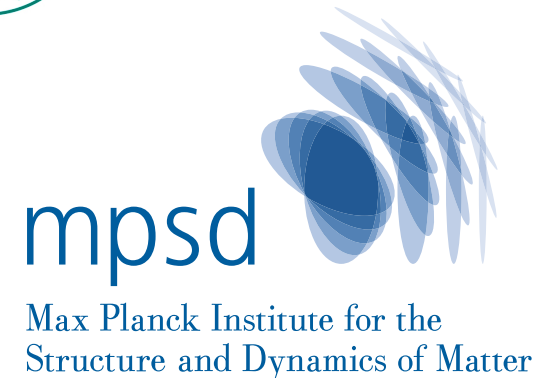
Hans Fangohr

Scientific Support Unit for Computational Science

Max-Planck Institute for Structure and Dynamics of Matter, Hamburg, Germany

University of Southampton, Southampton, United Kingdom

@ProfCompMod, [hans.fangohr@mpsd.mpg.de](mailto:hans.fangohr@mpsd.mpg.de)



Max Planck Institute for the  
Structure and Dynamics of Matter

UNIVERSITY OF  
**Southampton**

Research Data Management workshop 5 and 6 May 2021

# Outline

- Brief introduction Jupyter Notebook
- Some use cases
- Jupyter Notebooks for (more) reproducible science
- Binder, and example
- Format:
  - Presentation
  - Tutorial
  - Discussion
- Informal: Ask questions immediately



# Jupyter Notebook

- Purpose: data analysis, exploration, visualisation, ...
- Document hosted in web browser (demo)
- Combines
  - text (markdown with LaTeX support)
  - Computer code (Python)
  - Output from code
- Saved in one document
  - `*.ipynb` [IPYthon NoteBook] (JSON format on disk)
  - Can be re-loaded and re-executed

<http://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  $t$ ,  $\alpha$  and  $\omega$ :

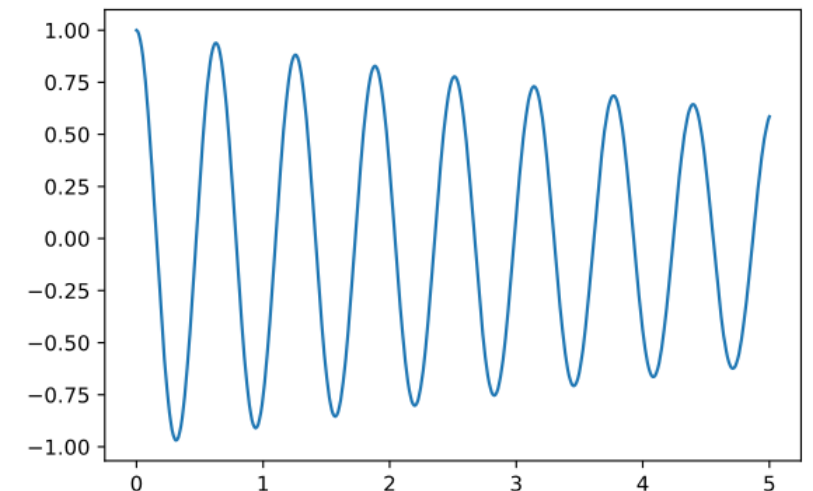
```
In [3]: f(t=0.1, alpha=1, omega=10)
```

```
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
```



# Terminology project Jupyter

- Jupyter Notebook: document in browser / file format
- Jupyter Lab - newer interface in browser  
(includes Jupyter Notebook)
- JupyterHub:  
allows to use Jupyter notebooks remotely



# JupyterLab

File Edit View Run Kernel Tabs Settings Help

Files

Running

Commands

Cell Tools

Tabs

Python 3

In this Notebook we explore the Lorenz system of differential equations:

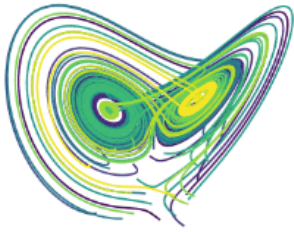
$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

In [4]: `from lorenz import solve_lorenz  
t, x_t = solve_lorenz(N=10)`

Output View

sigma 10.00  
beta 2.67  
rho 28.00



lorenz.py

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x_y_z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15 to 15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```



# History of Jupyter

- IPython -> IPYthon NoteBook -> \*.ipynb
- community began adding other languages (“kernels”) to the notebooks, starting with Julia and R
- JULia, PYThon, and R -> Jupyter
- “Jupyter” is also a homage to Galileo’s notebooks which recorded the discovery of Jupiter’s moons

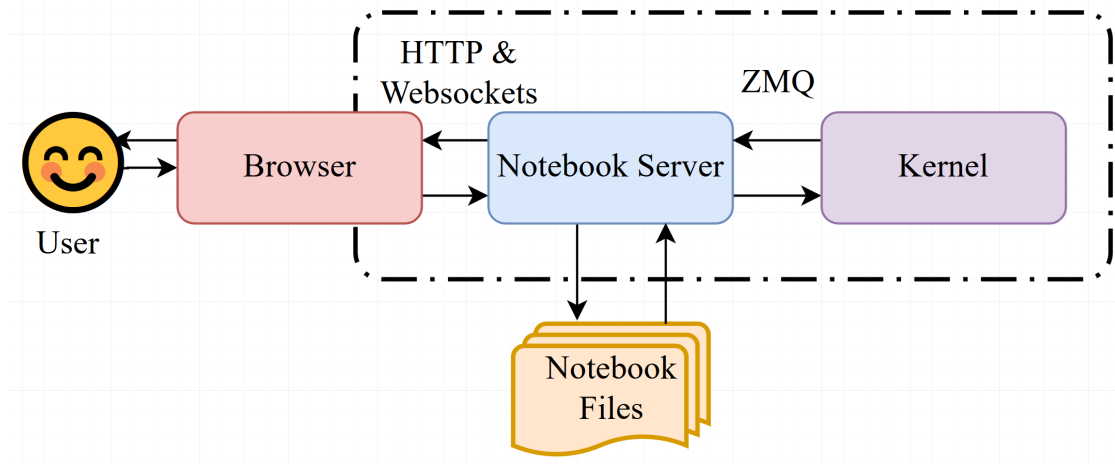
IP[y]:  
IPython

IP[y]: Notebook

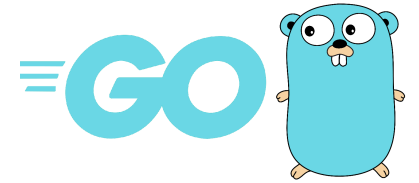


# \* How does it work?

- Starting jupyter-notebook starts the notebook server
- Opening a notebook starts up the “kernel”
- The kernel communicates with the notebook server via ZMQ
- The notebook server shows content via the browser
- JavaScript used in Browser



# \* Supported Kernels



- 50+ languages supported.
- More complete list at
- <https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>



# Use case: data analysis

- Explorative data analysis
- Convenient combination of processing, results and interpretation
- Complete capture of all computational steps
- Good record for reproducibility
- Share with collaborators & supervisors (html, latex, pdf)

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

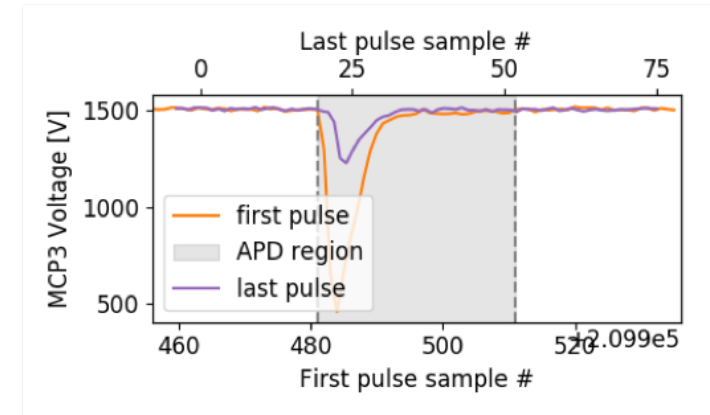
```
In [4]: proposalNB = 900074
semesterNB = 201930
runNB = 487
topic = 'SCS'
fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
              validate=True, display=False)
nrun = tb.matchXgmTimPulseId(run)

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

Step 2: check the pulse integration window

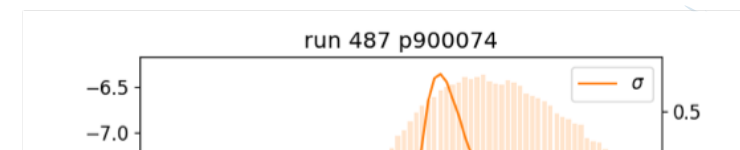
```
In [5]: tb.checkTimApdWindow(nrun, mcp=3)

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: recipes

- For recurring tasks, pre-populate notebook with cells to carry out a particular type of data analysis
- Create collection of such notebook recipes
- Compromise between static recipe (=script) and interactive exploration

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

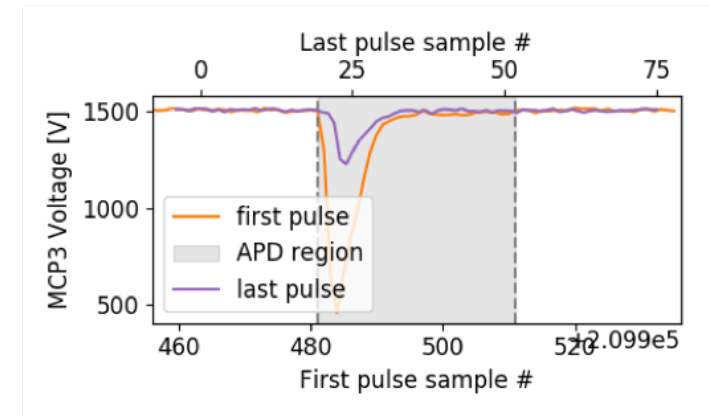
```
In [4]: proposalNB = 900074
semesterNB = 201930
runNB = 487
topic = 'SCS'
fields = ["SCS_photonFlux", "SCS_XGM", "MCP3apd", "nrj"]
run = tb.load(fields, runNB, proposalNB, semesterNB, topic,
              validate=True, display=False)
nrun = tb.matchXgmTimPulseId(run)

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

Step 2: check the pulse integration window

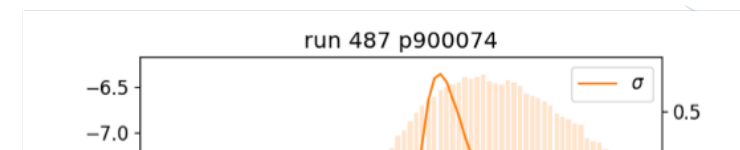
```
In [5]: tb.checkTimApdWindow(nrun, mcp=3)

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: report generator

- Use Jupyter Notebook as a report generator
  - \*execute using `nbconvert` & save resulting notebook or create html/pdf
  - \*Can modify parameters in beginning of notebook (`nbparametrize` or `papermill`)
- Error messages embedded in output

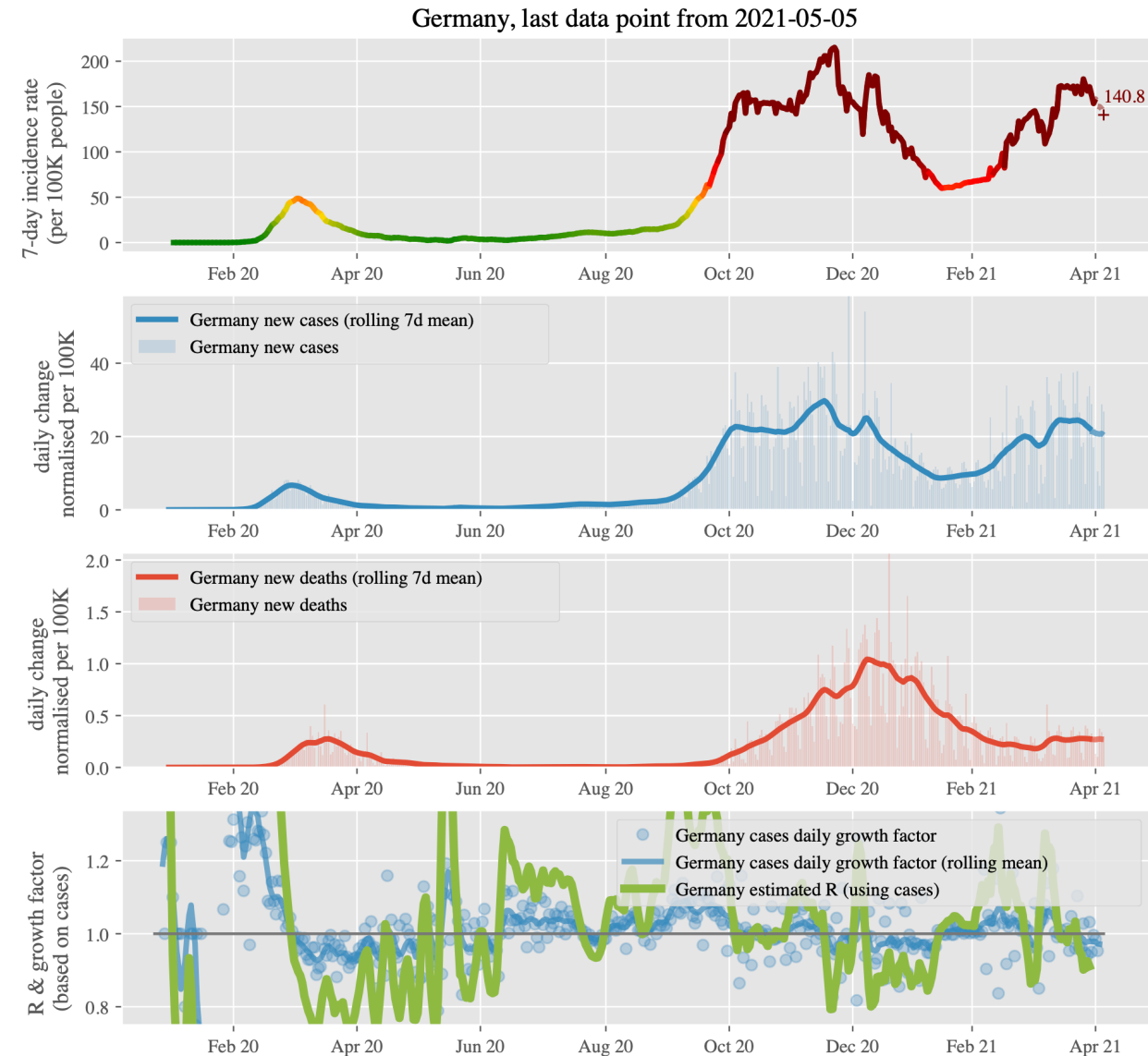
- Use cases:

- Open Science COVID Analysis

<https://oscovida.github.io>

<https://oscovida.github.io/html/Germany.html>

```
[4]: overview("Germany");
```



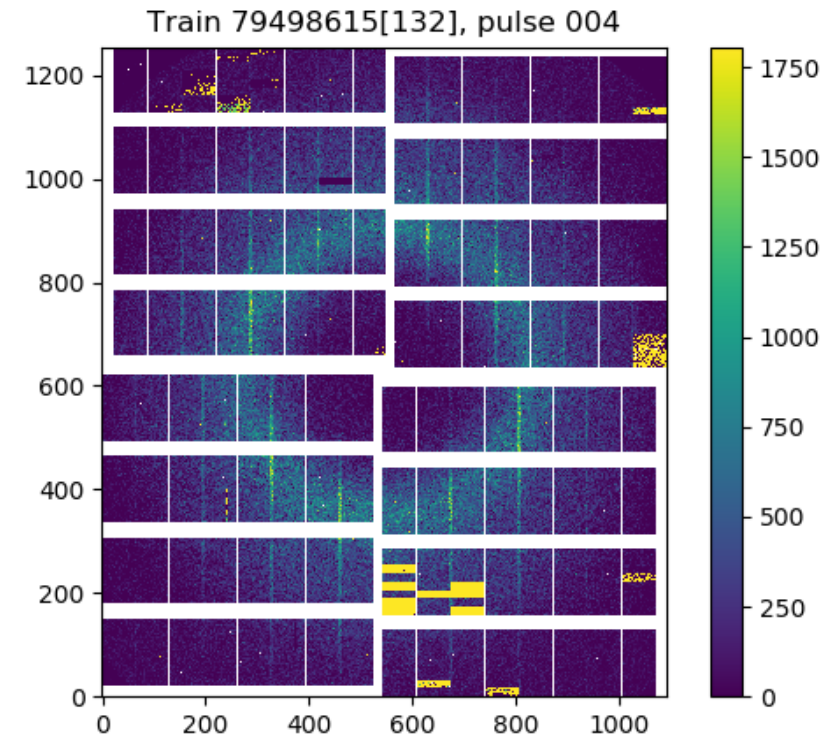
# Use case: blending GUI and script

- “Widgets” provide graphical control elements in notebooks
- Buttons, sliders etc trigger code execution / plotting
- Discussion
  - Reproducibility is difficult at the moment

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

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

Figure 1



Home Left Right Pan Zoom Save

Image Type: data gain mask

Train Index: 132 Pulse Index: 4

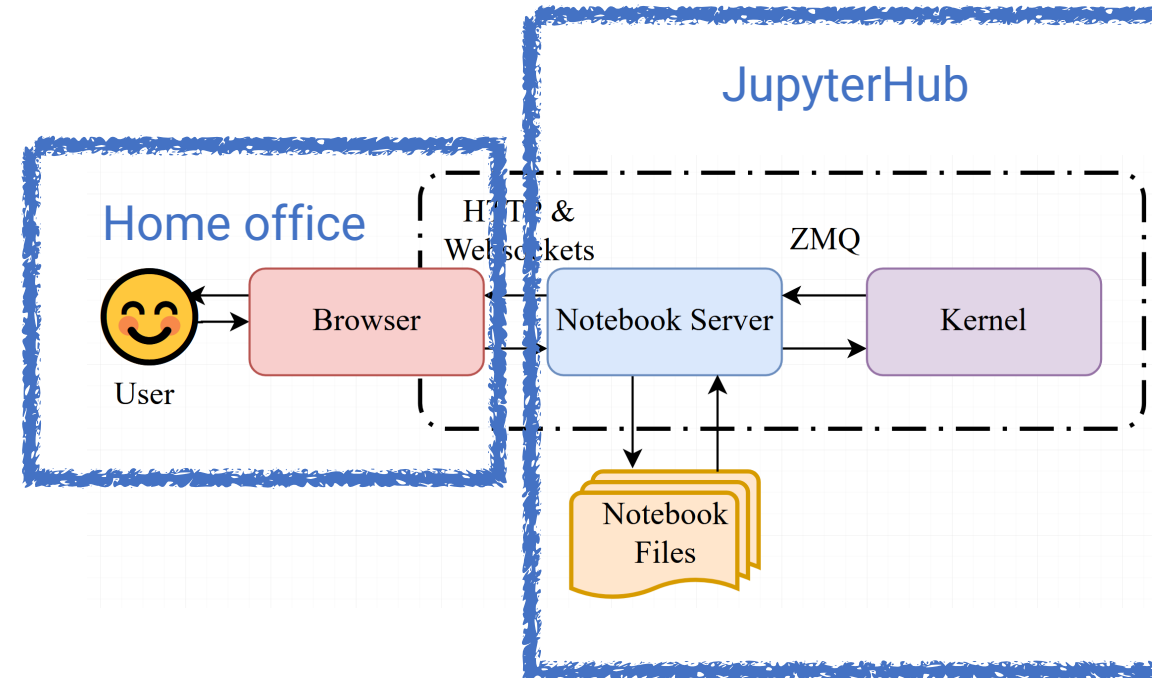
Train ID: 79498615 ☒ Reset on train

Clip Range:  0 - 1806 Colour Map: viridis

< Prev Good Next Good > Status:

# \*JupyterHub: use remote compute environment

- JupyterHub allows
  - users to connect through browser to High Performance Computing (HPC) resources
  - Use storage of HPC systems for notebooks and data
- Example: JupyterHub @ GWDG:  
<https://jupyter-cloud.gwdg.de>



# JupyterHub example EuXFEL / DESY

- Access to ~30PB of data
- Many thousand compute cores
- Select appropriate hardware before launching

Data exploration and analysis with Jupyter notebooks, Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS2019, TUCPR02 <http://accelconf.web.cern.ch/icalepcs2019/doi/JACoW-ICALEPCS2019-TUCPR02.html>, pdf (2020)

## Maxwell Jupyter Job Options

Maxwell partitions①

Choice of GPU①

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

Constraints①

Note: This will override GPU selections!

Number of Nodes①

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

Job duration①

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

Launch modus①

Remote Notebook①

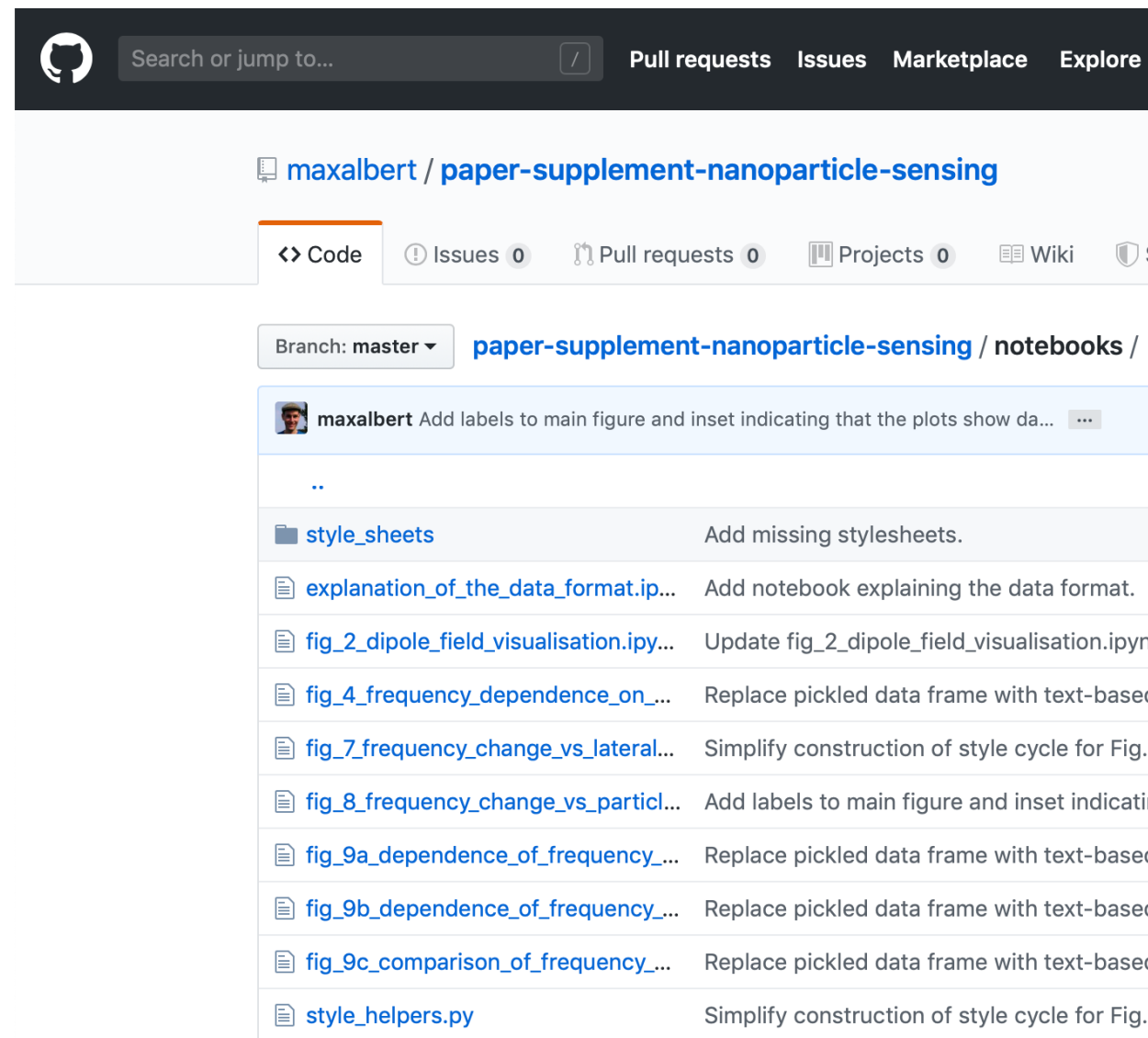
| Node and GPU availability |         |         |              |              |              |
|---------------------------|---------|---------|--------------|--------------|--------------|
| Partition                 | # nodes | # avail | # GPUs avail | # P100 avail | # V100 avail |
| jhub                      | 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



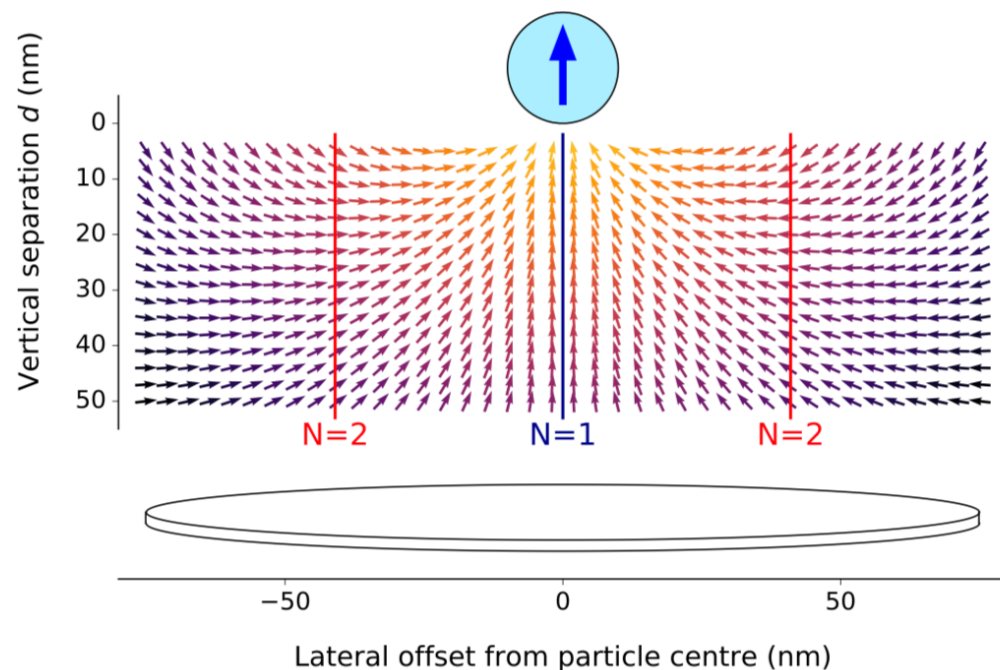
# Use case: reproducible publication

- Create github repository to complement publication, containing
- one notebook per figure / main result
- Zenodo for long term preservation

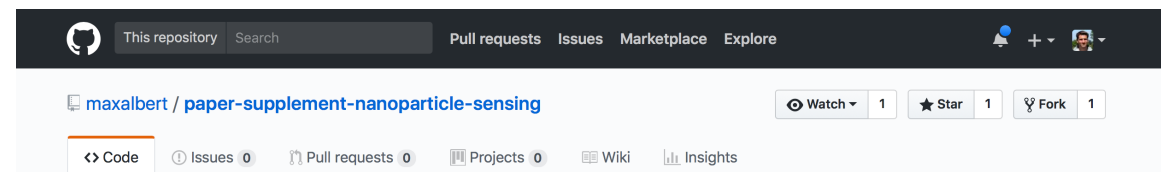


Example:  
<https://github.com/maxalbert/paper-supplement-nanoparticle-sensing>





**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.



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

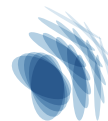
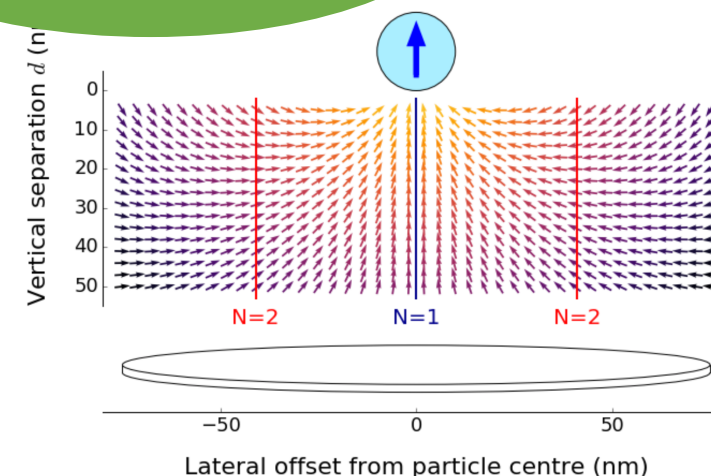
Raw Blame History

## 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.

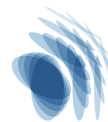
```
In [1]: import matplotlib.colors as colors
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.patches import Ellipse, FancyArrow,
from matplotlib.pyplot import cm
%matplotlib inline
```

```
... fld,
... y_fld)
... ymax_fld, annotation='N=1', color='darkblue')
... ymax_fld, annotation='N=2', color='red')
... ymax_fld, annotation='N=2', color='red')
```



# One-study one-document

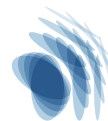
- *One-study one-document:*  
Notebook combines (and logs) equations, code, results, interpretation
- Much easier than manually tracking post-processing, analysis, plot-creation scripts, figure files, and LaTeX / Word files.
- *Data exploration and analysis with Jupyter notebooks, [10.18429/JACoW-ICALEPCS2019-TUCPR02](#) (2020)*



# Jupyter for Reproducible Science

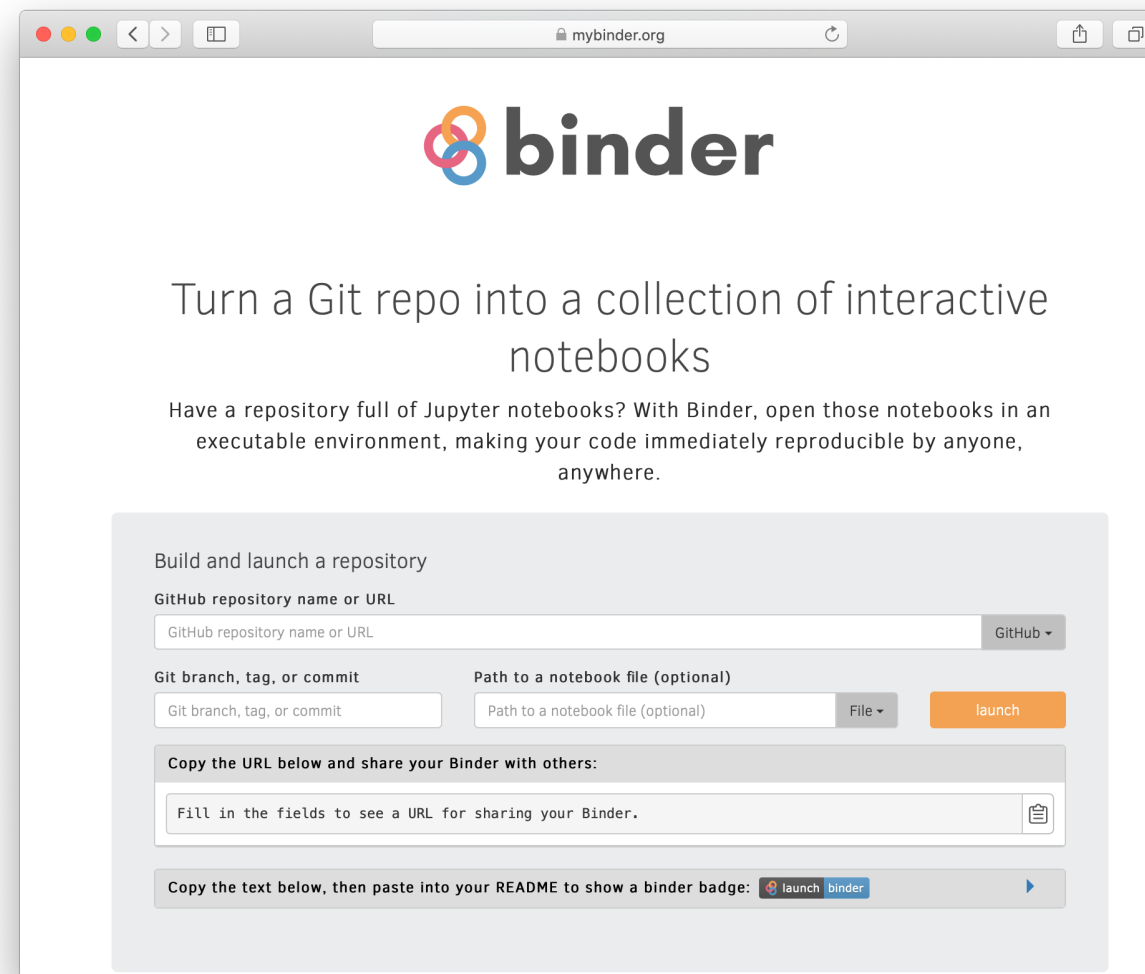
- One-study one-document: research driven from notebook
- Archive notebooks (and software environment) with data
- Publish notebooks with manuscript
  - For example one notebook per key statement / figure / table
- Strategy:

*Using Jupyter for reproducible scientific workflows.* Computing in Science & Engineering, [10.1109/MCSE.2021.3052101](https://doi.org/10.1109/MCSE.2021.3052101) (2021)

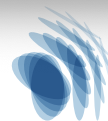


# \*Binder project

- Given a (Github) repository with
  - Jupyter notebooks
  - Software requirements (Dockerfile, requirements.txt, environment.yml)
- Binder service
  - builds a container with the required software (repo2docker)
  - starts Jupyter notebook server in that container offering the notebooks
- Alternative to Google Colab
- Public Binder service: <https://mybinder.org>



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



github.com

Search or jump to... Pull requests Issues Marketplace Explore

**fangohr / jupyter-demo** Watch 0 Star 0 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Introducing basics of Jupyter Notebook (intended to be used in presentation / chat) [Edit](#)

[Manage topics](#)

25 commits 2 branches 1 release 1 contributor BSD-3-Clause

Branch: master New pull request Create new file Upload files Find File Clone or download

**fangohr** Add Zenodo badge to version 1.0 Latest commit 85f2283 7 days ago

|                    |                                   |              |
|--------------------|-----------------------------------|--------------|
| 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  |

README.md

launch binder DOI 10.5281/zenodo.3463132

Most recent version of this repository is located at <https://github.com/fangohr/jupyter-demo>

## Jupyter notebook demo repository

mybinder.org

Thanks to Google Cloud and OVH for sponsoring our computers 🙌!

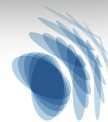
# binder

Starting repository: fangohr/jupyter-demo/master

The tool that powers this page is called BinderHub. It is an open source tool that you can deploy yourself.

Build logs [hide](#)

```
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...
```



The screenshot shows a web browser window with the address bar displaying "mybinder.org". The main content area features a large, colorful circular loading spinner. Below the spinner, the text reads: "Starting repository: fangohr/jupyter-demo/master". Underneath this, a message states: "Your session is taking longer than usual to start! Check the log messages below to see what is happening." Below the message is a terminal window titled "Build logs" with a "hide" link in the top right corner. The terminal output shows a series of "Pushing image" messages, followed by a "Successfully pushed" message with a long repository URL, and finally "Launching server...". At the bottom of the page, a small line of text says: "Here's a non-interactive preview on nbviewer while we start a server for you. Your binder will open automatically when it is ready."



hub-binder.mybinder.ovh

jupyter Quit

Files Running Clusters

Select items to perform actions on them. Upload New ↻

| <input type="checkbox"/> | 0 ▾ /              | Name ▾ | Last Modified | File size |
|--------------------------|--------------------|--------|---------------|-----------|
| <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)
```

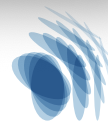
## Interactive exploration

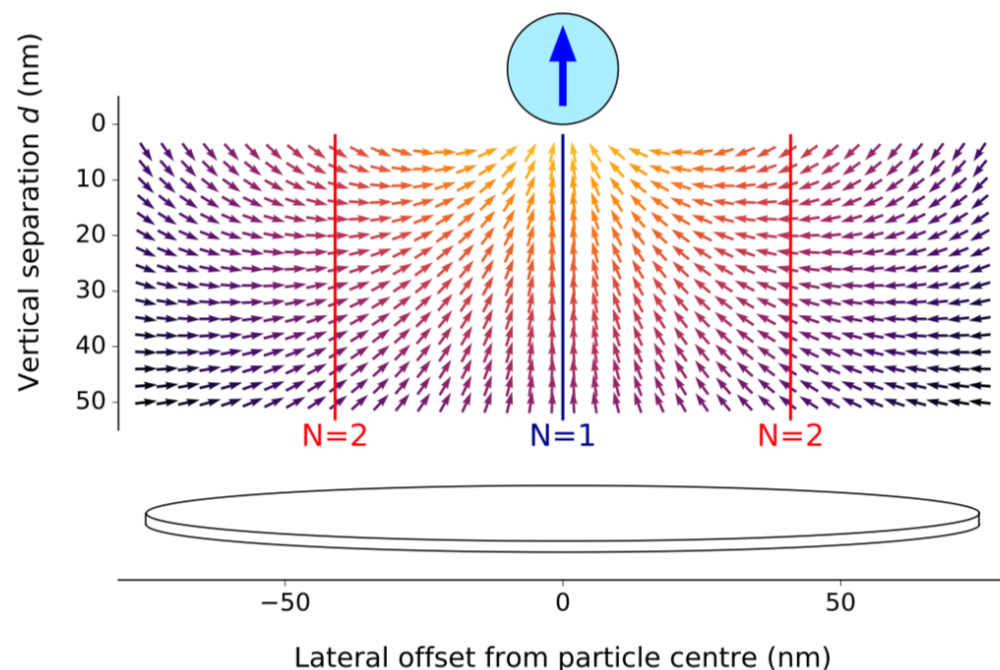
We can execute the function for value of  $\alpha$  and  $\omega$ :

```
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:

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





**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

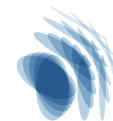
296 lines (295 sloc) 214 KB Raw Blame History

### 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,
from matplotlib.pyplot import cm
%matplotlib inline
```

```
... fld,
... y_fld)
... ymax_fld, annotation='N=1', color='darkblue')
... x=ymax_fld, annotation='N=2', color='red')
... x=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](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.

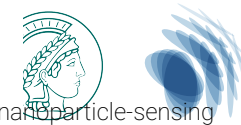
The latest version of this repository can be found at <https://github.com/maxalbert/paper-supplement-nanoparticle-sensing>

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

## Contents

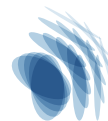
The directory `notebooks/` contains Jupyter notebooks for the relevant figures in the paper. On Github you can view them directly in the browser:

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



# Summary – Jupyter notebook use cases

- Data analysis, One-study one-document
- Provision of recipes, report generator
- Remote data analysis (JupyterHub)
- Documentation of Software (also publishing of books)
- Reproducibility



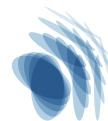
# Additional remarks

- A wide ecosystem of tools is emerging for Project Jupyter
  - JupyterLab - a new interface to Jupyter Notebooks (Window manager in browser)
  - Many discipline specific libraries; rapid development; can be difficult to track
- Observed issues
  - Multiplication of code and notebooks, for example during intense data analysis or experiment period
  - State of notebook may be tricky: Good practice to restart and re-run to check



# Summary Jupyter for Reproducible Science

- “One-study one-notebook” concept is powerful
- Some open issues:
  - Jupyter Notebook does not work for (i) all applications and (ii) all people
  - Archiving software environment
  - Studies with large data and large compute
- Binder shows interesting perspectives (NFDI, EOSC, GWDG, MPCDF, ...?)



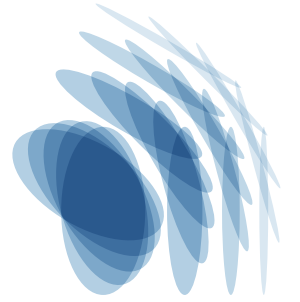
# Acknowledgements

- Marijan Beg, Thomas Kluyver, Robert Rosca
- OpenDreamKit Horizon 2020, European Research Infrastructures project (No 676541), <http://opendreamkit.org>
- PaNOSC: Photon and Neutron Open Science Cloud, European Union's Horizon 2020 research and innovation programme under grant agreement (No 823852), <http://panosc.eu>

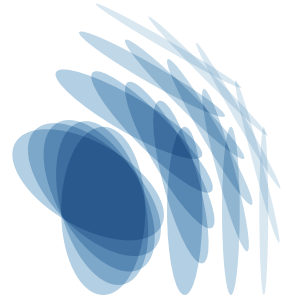
# References

- Hans Fangohr et al, *Data exploration and analysis with Jupyter notebooks*, Proceedings of the 17th International Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS2019, TUCPR02, [10.18429/JACoW-ICALEPCS2019-TUCPR02](https://doi.org/10.18429/JACoW-ICALEPCS2019-TUCPR02) (2020)
- Marijan Beg et al, *Using Jupyter for reproducible scientific workflows*, Computing in Science & Engineering, vol. 23, no. 2, pp. 36-46, 10.1109/MCSE.2021.3052101 (2021)





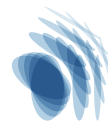
# Tutorial



# Appendix

# Example text books using Jupyter

- François Chollet: *Deep Learning with Python*, <https://github.com/fchollet/deep-learning-with-python-notebooks>
- Wes McKinney: *Python for Data Analysis*, <https://github.com/wesm/pydata-book>
- Aurélien Géron: *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow*, <https://github.com/ageron/handson-ml2>
- Hans Fangohr: *Introduction to Python for Computational Science and Engineering*, <https://github.com/fangohr/introduction-to-python-for-computational-science-and-engineering/blob/master/Readme.md>



# Use case: documenting software library

- Use notebook as chapter in documentation
  - Supported by sphinx → html, pdf
- 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**)

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

 DigitalOcean

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.

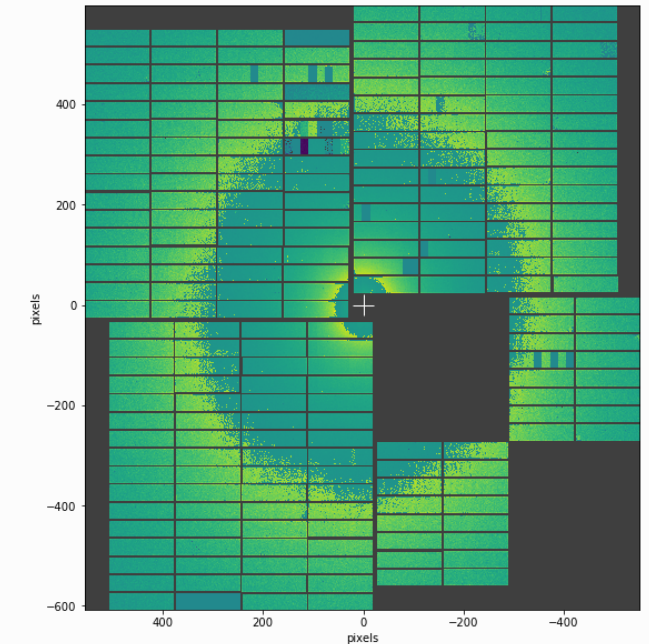
In the future, geometry information will be stored in the calibration catalogue.

```
[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=!
```

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

0\_Jupyter\_and\_Colab Last Checkpoint: 2 minutes ago (unsaved changes)

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

Toolbar

Notebook name

Kernel name

## Jupyter Notebooks and Colab

Jupyter Notebooks are a widely used tool for data science. They contain both code and rich-text elements like paragraphs, equations and charts. Notebooks are both (a) human-readable documents used to present analysis and results; and (b) executable documents that your computer can be run.

Take 5 minutes to explore!

Markdown cells

Currently selected cell

### Important shortcuts

| Action                                       | Colab Shortcut |
|----------------------------------------------|----------------|
| Executes current cell                        | <CTRL-ENTER>   |
| Executes current cell and moves to next cell | <SHIFT-ENTER>  |
| Insert cell above                            | <CTRL-M> <A>   |
| Append cell below                            | <CTRL-M> <B>   |
| Convert cell to code                         | <CTRL-M> <Y>   |
| Convert cell to Markdown                     | <CTRL-M> <M>   |
| Autocomplete                                 | <TAB>          |
| Goes from edit to "command" mode             | <ESC>          |
| Goes from "command" to edit mode             | <ENTER>        |

Note: On OS X use <COMMAND> instead of <CTRL>

### Types of cell

Notebooks are made up of a series of cells. There are two types:

- **Code cells.** These contain executable commands in Python.
- **Text ('markdown') cells.** These include plain text, or you can use markdown conventions to add formatting.

Code cell

Code output

```
In [6]: 1 print('Hello, I am a code cell')
        2 print('Running me executes the code and renders the output below.')
```

Hello, I am a code cell  
Running me executes the code and renders the output below.

# Sharing and exporting notebooks

- Share \*.ipynb files
  - Can be displayed using `jupyter-notebook`
  - Code can be re-executed on collaborator's machine
    - Only if software is available, and data is available, and collaborators know how to start notebook
- “Static sharing” through html and email (for example)
  - Often sufficient – does not require installation or additional skills
    - Effective to communicate with supervisors, line managers etc
  - Convert using menu “File -> Download as -> html”
  - Or using `nbconvert`:

```
$ jupyter-nbconvert --to html 2-widgets.ipynb
[NbConvertApp] Converting notebook 2-widgets.ipynb to html
[NbConvertApp] Writing 382609 bytes to 2-widgets.html
```



# Summary – some tools in ecosystem

- Jupyter Lab – next generation Jupyter user interface
- Jupyter Hub – serving notebook from compute facility for multiple users
- NBDIME – DIffing and MErging tools
- NBVAL – VALidation tool; use each cell as a test
- NBCONVERT – conversion of notebooks to other formats & execution
- NBParametrize and Papermill – inject parameters into notebook files
- IPyWidgets – GUI like elements in notebook
- Binder – Cloud hosted execution of notebooks from github repositories
- There is much more.



## \* Executing a notebook from the command line (nbconvert)

- Convert from ipynb to html:

```
$ jupyter-nbconvert --to html 2-widgets.ipynb
```

- Can optionally set output name

```
$ jupyter-nbconvert --to html --output myout.html 2-widgets.ipynb
```

- Can execute notebook before conversion:

```
$ jupyter-nbconvert --execute --to html --output myout.html 2-widgets.ipynb
```

- Execute notebook and save results as notebook:

```
$ jupyter-nbconvert --execute --to ipynb --output myout.ipynb 2-widgets.ipynb
```

```
[NbConvertApp] Converting notebook 2-widgets.ipynb to ipynb
```

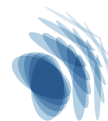
```
[NbConvertApp] Executing notebook with kernel: python3
```

```
[NbConvertApp] Writing 103398 bytes to myout.ipynb
```



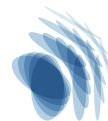
# nbconvert

- **nbconvert** is a tool which can *execute and convert* notebooks to various other formats such as:
  - HTML, LaTeX, PDF, Markdown, ReStructuredText, Python, ...
- Static HTML pages can be created as documentation or tutorial
  - **Nbsphinx** -> Jupyter notebook provides section in sphinx documentation
- Homepage: <https://github.com/jupyter/nbconvert>
- Notebook as a script approach
  - Can change parameters inside notebook before execution (**nbparametrize** or **papermill**)



# nbval

- py.test is a popular Python testing framework
- nbval is a py.test plugin which lets py.test recognise and collect Jupyter notebooks
- In each notebook, each cell is a test:
  - The test passes if execution of the input creates the stored output
  - The test fails otherwise
- There is a variety of configuration parameters
- Home page: <https://github.com/computationalmodelling/nbval>

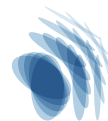


# nbval example

```
$ py.test --verbose --nbval 2-widgets.ipynb
===== test session starts =====
platform darwin -- Python 3.6.8, pytest-3.10.0, py-1.7.0, pluggy-0.8.0 -- /Users/fangohr/anaconda3/bin/python
rootdir: /Users/fangohr/Desktop/jupyter-demo, inifile:
plugins: remotedata-0.3.1, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2, nbval-0.9.1
collected 9 items

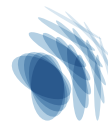
2-widgets::ipynb::Cell 0 PASSED [ 11%]
2-widgets::ipynb::Cell 1 PASSED [ 22%]
2-widgets::ipynb::Cell 2 PASSED [ 33%]
2-widgets::ipynb::Cell 3 PASSED [ 44%]
2-widgets::ipynb::Cell 4 PASSED [ 55%]
2-widgets::ipynb::Cell 5 PASSED [ 66%]
2-widgets::ipynb::Cell 6 PASSED [ 77%]
2-widgets::ipynb::Cell 7 PASSED [ 88%]
2-widgets::ipynb::Cell 8 PASSED [100%]

===== 9 passed, 1 warning in 2.11 seconds =====
(base) [20:49:09] fangohr:jupyter-demo git:(master*) $
```



# Jupyter Lab

- Next generation Jupyter notebook interface
- Window manager embedded in browser
- “classic notebook” in one window
- Additional features in other windows
  - File browser
  - Extensions
  - CSV viewer.



FileEditViewRunKernelTabsSettingsHelp

0\_JUPYTER\_AND\_COLAB.IPYNB

<>M↓⋮

1. Jupyter Notebooks and Colab

Jupyter Notebooks are a widely used tool for data science. They contain both code and rich-text elements like paragraphs, equations and charts. Notebooks are both (a) human-readable documents used to present analysis and results; and (b) executable documents that your computer can be run.

1.0.1. Important shortcuts

1.0.1.1. Types of cell

Notebooks are made up of a series of cells. There are two types:

Table of Contents extension

0\_Jupyter\_and\_Colab.ipynb1\_Python\_code\_EXERCISES.i

Python 3

1. Jupyter Notebooks and Colab

Jupyter Notebooks are a widely used tool for data science. They contain both code and rich-text elements like paragraphs, equations and charts. Notebooks are both (a) human-readable documents used to present analysis and results; and (b) executable documents that your computer can be run.

Take 5 minutes to explore!

1.0.1. Important shortcuts

| Action                                        | Colab Shortcut |
|-----------------------------------------------|----------------|
| Executes current cell                         | <CTRL-ENTER>   |
| Executes current cell and moves to next cell  | <SHIFT-ENTER>  |
| Insert cell above                             | <CTRL-M> <A>   |
| Append cell below                             | <CTRL-M> <B>   |
| Convert cell to code                          | <CTRL-M> <Y>   |
| Convert cell to Markdown                      | <CTRL-M> <M>   |
| Autocomplete                                  | <TAB>          |
| Goes from edit to "command" mode              | <ESC>          |
| Goes from "command" to edit mode              | <ENTER>        |
| Note: On OS X use <COMMAND> instead of <CTRL> |                |

1.0.1.1. Types of cell

Notebooks are made up of a series of cells. There are two types:

- Code cells. These contain executable commands in Python.
- Text ('markdown') cells. These include plain text, or you can use markdown conventions to add formatting.

[6]:

```
print('Hello, I am a code cell')
print('Running me executes the code and renders the output below.')
```

Hello, I am a code cell

Running me executes the code and renders the output below.

rosca@DESKTOP-P9E1B7I: ~

Multiple panels

Explore files (e.g. CSV)

BLI2015.csv

Delimiter: ,

|    | "LOCATION" | Country        | INDICATOR | Indicator                          |
|----|------------|----------------|-----------|------------------------------------|
| 1  | AUS        | Australia      | HO_BASE   | Dwellings without basic facilities |
| 2  | AUT        | Austria        | HO_BASE   | Dwellings without basic facilities |
| 3  | BEL        | Belgium        | HO_BASE   | Dwellings without basic facilities |
| 4  | CAN        | Canada         | HO_BASE   | Dwellings without basic facilities |
| 5  | CZE        | Czech Republic | HO_BASE   | Dwellings without basic facilities |
| 6  | DNK        | Denmark        | HO_BASE   | Dwellings without basic facilities |
| 7  | FIN        | Finland        | HO_BASE   | Dwellings without basic facilities |
| 8  | FRA        | France         | HO_BASE   | Dwellings without basic facilities |
| 9  | DEU        | Germany        | HO_BASE   | Dwellings without basic facilities |
| 10 | GRC        | Greece         | HO_BASE   | Dwellings without basic facilities |
| 11 | HUN        | Hungary        | HO_BASE   | Dwellings without basic facilities |
| 12 | ISL        | Iceland        | HO_BASE   | Dwellings without basic facilities |
| 13 | IRL        | Ireland        | HO_BASE   | Dwellings without basic facilities |
| 14 | ITA        | Italy          | HO_BASE   | Dwellings without basic facilities |
| 15 | JPN        | Japan          | HO_BASE   | Dwellings without basic facilities |
| 16 | KOR        | Korea          | HO_BASE   | Dwellings without basic facilities |
| 17 | LUX        | Luxembourg     | HO_BASE   | Dwellings without basic facilities |
| 18 | MEX        | Mexico         | HO_BASE   | Dwellings without basic facilities |
| 19 | NLD        | Netherlands    | HO_BASE   | Dwellings without basic facilities |
| 20 | NZL        | New Zealand    | HO_BASE   | Dwellings without basic facilities |
| 21 | NOR        | Norway         | HO_BASE   | Dwellings without basic facilities |
| 22 | POL        | Poland         | HO_BASE   | Dwellings without basic facilities |

12

BLI2015.csv

# Jupyter Notebook vs. JupyterLab

- JupyterLab is the newer interface to the notebooks
- Window manager in browser
- It provides a more flexible interface closer to a modern IDE
- Flexible layouts and panes
- Console
- Drag/Drop/Expand/Collapse cells
- Themes
- Extensions



# Nbdime – NoteBook DIff and MErge

- Jupyter notebooks are rich media documents, stored as plain text JSON files
- Basic diff and merge tools (such as that used by git) do not handle this format well
  - Small changes to text/plots result in unreadable diffs
- **nbdime** provides multiple tools to help with “content-aware” diffing and merging for Jupyter notebook files
- Homepage: -<https://github.com/jupyter/nbdime>
- **nbdiff** compare notebooks in a terminal-friendly way
- **nbmerge** three-way merge of notebooks with automatic conflict resolution
- **nbdiff-web** shows you a rich rendered diff of notebooks
- **nbmerge-web** gives you a web-based three-way merge tool for notebooks
- **nbshow** present a single notebook in a terminal-friendly way



notebook-v1.ipynb × notebook-v2.ipynb ×

Markdown Python 3

# 1. This is a placeholder title!

Here's some description text, maybe has a typo in it...

Here's some text that won't change.

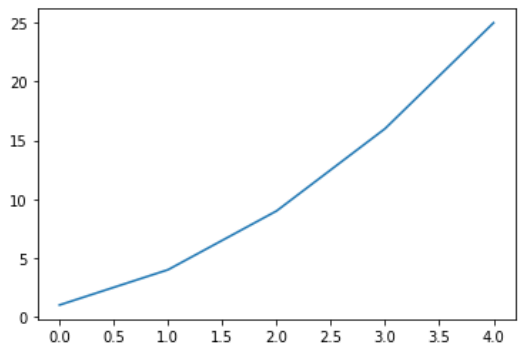
```
[1]: import matplotlib.pyplot as plt
```

```
[2]: amazing_math = [1, 2, 3, 4, 5]
```

```
      amazing_math = [x**2 for x in amazing_math]
```

```
[3]: plt.plot(amazing_math)
```

```
[3]: [<matplotlib.lines.Line2D at 0x7f30ab9b86d8>]
```



TODO:

- ☐ Better title
- ☐ Fix typo
- ☐ Section headings
- ☐ Go up to 6 squared!

Mode: Command Ln 1, Col 1 notebook-v1.ipynb

notebook-v1.ipynb × notebook-v2.ipynb ×

Markdown Python 3

# 1. Example of Notebook Diffing!

Here's some description text, now typo free!

Here's some text that won't change.

```
[1]: import matplotlib.pyplot as plt
```

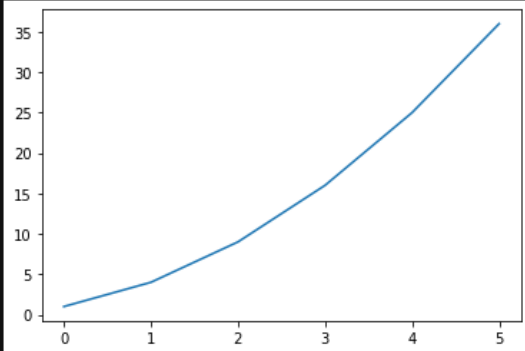
```
[2]: amazing_math = [1, 2, 3, 4, 5, 6]
```

```
      amazing_math = [x**2 for x in amazing_math]
```

## 1.1. A Plot

```
[3]: plt.plot(amazing_math)
```

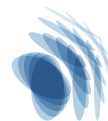
```
[3]: [<matplotlib.lines.Line2D at 0x7f77cbc8395f8>]
```



TODO:

- ☒ Better title
- ☒ Fix typo
- ☒ Section headings
- ☒ Go up to 6 squared!

Mode: Command Ln 1, Col 1 notebook-v2.ipynb



```
diff notebook-v1.ipynb notebook-v2.ipynb
7c7
<      "# This is a placeholder title!"
---
>      "# Example of Notebook Diffing!"
14c14
<      "Here's some description text, maybe has a typo in it ... \n",
---
>      "Here's some description text, now typo free!\n",
34c34
<      "amazing_math = [1, 2, 3, 4, 5]\n",
---
>      "amazing_math = [1, 2, 3, 4, 5, 6]\n",
39a40,46
>      "cell_type": "markdown",
>      "metadata": {},
>      "source": [
>          "## A Plot"
>      ]
>      },
>      {
47c54
<          "[<matplotlib.lines.Line2D at 0x7f30ab9b86d8>]"
---
>          "[<matplotlib.lines.Line2D at 0x7f7cbc8395f8>]"
56c63
<      "image/png": "iVBORw0KGGoAAAANSUHEUGAAAAAAD4CAYAAAD1jb0+AAAABHNCSVQICAgIFAhkIAAAAAAwSFLzAAAEgAACxIB0
t1+/AAAAADh0RVh0U29mdHdhcmUAUABWf0cGxvdGxpYiB2ZXJzaW9uMy4xLjEsIGh0dHA6Ly9tYXRwbG90bGliLm9yZy8QZhcZAAAgAELEQVR4nO3d
eXWu9eH/8dcHEgjhCfDcWhHCHe4rYERBFKWgVqpWBZT1BvRUElatb9ae2E961UFRFR07zsggopY0cORA0G+SuGc5IDcu5/fH9n6o5RASHZ3dpP
38/HI8g3MjPN2zLwzmZ35jLHWiIiIwae00wFERKRW4iEiQUoGLIAQpFbiISJA6a4EbY9oZY74xxmw2xmwxtzrmf6EeagMwa95+Ny38cVEZH/q
xcHgvWrPHnKkVEgp4xZu/ppusUiOhIkFKBi4gEKRW4iEiQUoGLIAQpFbiISJA6a4EbY9oZY74xxmw2xmwxtzrmf6EeagMwa95+Ny38cVEZH/q
MxLhGXA9bZaZGNMY2CtMwaxZ97z1tpnfBdPREQctYjcGtturU22fM6H9gCtPF1MBGRmqCgpIwnPt1EbmGp17/30Z0DN8bEAF2BLZ5JdxtjNhpj
ZhpjmlXwNZONMMWuMMWuysrKqFVZEJjGcLy7j5pmrefvHPazde9Tr37/SBW6MaQR8ANxnrc0d/gV0AvoB6cCzp/s6a+00a22CtTYhMvJ/7gQVEam
RcgtLmfDGStbu08aL4/pzSXwrr6+jUrfSG2NCKS/v0dbadWgStYdPmj8d+Nzr6UREglB0QqM/mrmKLeL5vDj+AKN6RftkPZW5CsUAbwBbrLXPnT
Q95gTFrgZSvR9PRCS4HD1RwvjpK0LLz+e1mw6rLhYhckfGfWATgBRjzHrPtMeAccaYfoAF9gB3+CSHiEiQyMov5sYZK9h7pIAZEXMY1tW3p43PW
uDW2uWA0c2sL70fR0Qk0B30K2L89BUcyinizZsHMaRzS5+v06/DyYqI1ESHcgoZP30FWfnFvHxRYAZ3a06X9arARUSYqf/RASZNX0FuQSLv33Ye
A9uf9opqn1CBi4hU0Z7sE4yfoITJ57mTDqPPm2b+nX9KnARkSrYkXmcG2esoNRlmTvpPHq2jvB7BhW41Mg52pqrZ40zVgKwEzMS6Rbd2JECkNa
RkXOW+VAen72xkpA6hrmTzqdzVCPHsmg8CBGRSko5kMu46SuoH1KHbXc4W96gI3ARKUpJ3neMiTNXEdEglHmTEmnXPnzpSDoCFxE5m9V7jjJhxx
qaN6zHgjjv0D4jyBh2Bi4ic0b93ZnPBrdXENA1j7u2JREeEOR3pJzoCFxGpwLJtWdzy5mraNmva/MmBVd6gI3ARKdNamnaY099jpLNUi2bfNpgWj
eo7HeL/qMBFRE6xaFMgd89NjJ66Ce/cNpm4fWcJnRaKnARkZn8sTGde+evo3fbcGbdMpiIBqFOR6qQzoGLiHh8v04g98xLpn9sU96+NbDLG3QE
LiTCwltr9vPBxtJ7NCCGRMTaFg/80sx8BOKiPJy3JX7e0yJfIZ2acm0Cqk0qFFX6UiVogIXkVrtrX/v4fFPN3FJfBSv3jiaASNDGKG9gQytILTZ
92S7++uUWRvZoxcvjB1AvJLjeFLSbi0it9Mo303h60Vau6B3DC2P7EVo3uMobVOAiUstYa3nh6+38c8L2ftGvNc9c15eIQcXvUIGLSCL1ireXpRV
t59duD/HJgW566tg916xinY1WZCLx EagVrLX/9Ygszlu9m/Hmx/GVML+oECXmDCLxEagG32/Knz2b1o97uXLHI//vAFGBHD5gwpRG04t9vy+
49TmldqP50GduCxy7vXiPIGFbi1GAUt+WRDzby/toD3HVxJx4a2a3GLdeowEWhkipZuXnwvQ18sv4Q91/alD+M6FyyjhtU4CJSA5W63Nw3fz1f
pKtZ8KhuTBne2elIPqECF5EapbjMxdz17F482H+cEV3bh/a0eLIpMCF5Eao6jUxa9nr+WbVn86aqeTBwS53Qkn1KBi0iNUFjiyV7a1i+I5u
/Xd2b8eF0h3J51TgThL0THSxcdtbqm5+yj/uLpY1W0yczqS5X5+1AABjTdtjzDfGmM3GmE3GmHs905sbYxYbY7Z7/m3m+7giIv8tv6iUiTNXsX
rPMW64ov+tKW+o3CPVyoAHrbU9PGETgLMNMD+FBRYIm1tguxwP05Ij5f8BaWmUGNVazfn80LY/szpL8bpyP51Vkl3Fqbbq1N9zr0B7YAbYAxwFuex
d4CfuGrkCIipz2ooQbZ6xg06FcXr1xAff0iXE6kt+d0xiKxpg4oD+wEmhLRu33zMoAWLXwNZONMMWuMMWuysrKqEVVEpNyR48WMm67CbYePM21C
AiN7RjdydRGLNBjTCPGaA+Aa23eyf0sTrawp/s6a+00a22CtTYhMjKyWmFFRDZixg7bQV7jpzgjYkXJBwF5Qkx15qI0xoZS9xxr7YeeyYe
NMTG6e+TFAPm8iioiUy8gtYuzrKziYU8ibnW9maJfafBYmatQDPAGSMVa+9xz4FJnpeTwQd+8X48EZFYB3MKUWHaj2TmF/P2rYM5vLMLpyM5rj
LXgV8ATABSjDhrPdMeA67xpbjgP2Atf7JqKI1Hb7jhQwbv0K8opKee2wfSP1VXLUIkCt9YUByoawmuEd+0iPy33dknGB99BYWLubenkjvt
hFORwoYuhNTRALWjsx8xk9fSznBmvf2RHKq0buJ0pICiAheRGLQ1I58bZ6wADPMnJ9K1VW0nIwWcc7o0XETEHI1P5jJ22o/UrWNYCkIfuyIqcBEJ
KBv25zB++goahNZLweT26RTZy0LIAUunUeqYKZde4ybZ66iacNQ5t6e5U4U5HCmgcgBEJCCT3HEHWWauJahLGNvPo3XTBk5HCngcgBFx3A8
7srn9rTW0bhrGvEmJRDUJczpSUNA5cBFx1Hfbsrh1mpim4czf/L5Ku9zoCnWEXHM15sPM2VOMp2jgJH79NVo3rCe05G6Cio7ARcQRCLPTuXP2Wu
JjgJn3ksq7KNeQLIj+99mGQ9y3YD1920Yw69bBNALdTpSUFKBi4hffZhg8Ife20BC++bmVGuQjeqrhqpKW05E/Obdift55MONNn+xBTmJhBeT
xVUhdP6tUxI76zYyx8/TmV10mTRhWGHdpYPMFRW4iPjcz0W7eFLZZyIj+KVGweovL1EB54iPvX6d2v5e1tao3pG8+K4/tQLOcV3qIcXfG
eWnJdp5dvI0r+8Tw/A39CKZr8YmFbiEj21LxcUb+PFPtU4pn8b/VHLPoSovL10BS4iXmWt5amFW3ntu51cn9CW1/T7p1Knoqo1SHCLxEvmZ2
ay58/38LMH3zU2Is17VizoqB59RGUyIV7jdlsc/3cQ7K/ZyywVx/N+VPTBG5e1LKNaRqTa32/LYRynMX72f04215MH8SPvp1CBi0i1uNyW37
6/gQ+TD3LPJZ154LKuM8/UYGLSJWVudzc/+4GPttwiAcu68pvrRnR0LktogIXkSopKXNz7/x1JKVm80joe068qJPTKwodFbiInLPiMhd3zUnm6
y2Z/PHKhtx2YQenI9VKNaROSeFJ57unL2W77Z18ecpZLWfpzTktWotFbIVNqOz0PcNSeZbZn5TL2mN2MHxzodqVZTgYtIpXyy/iC/+zCfSNc6
zLpLMBdijXQ6Uqz2nAheRMyoqdfHnzczcz+U+Eto346Xx/YmJa0B0LEEFLLjnsPfICabMSWbToTzUuKgjD43spKtYKXEROa2Fq0r99byPGWPR
fJX8Zj1ZOR5JtNPVXqTFmpjEm0xiTetK0J4wx840x6z0fL/s2poj4S0mZmyc/28yds5PpGnmQL34zVOUdoCpZBD4LeBl4+5Tpz1trn/F6thFzxM
GcQu6ak8z6/TncPCSO310eT/0QPF4sUJ21wk21y4wxcb6PIjJOWpp2maFE3UCZY/LK+AFc05FG6UhyFtV5N+JuY8xGzymWZL5JCJ+VeZy89TCN
G6dtYaYaIAZ8ds+FKu8gUdUC/xQcGegHpApR21gMwYmWNaNMWZNLZVFCcnIr5wOK+I8TNW8g9vdzJucCwFrLChY5iNwnY4LLVSLq1CstYf/89oY
Mx34/AzLTgOmASQkJNiqrE9EvG/59mzunb+OghIXz9/Ql6v7t3U6kpjYKHW4MSbGWpvu+frqPVMvY4tI4HC5L58t3c4/L2ync2Qj5k8eQJdwZ2
0JVVw1gI3xswDhgMtjTEHMeB4cayFoAF9gB3+DCjiHhJ9vFi7pu/nuU7srmfxfv+cnUvwvudpdpBgVZmrUMadZvBpSgiIj60ctcr7pm3jtZCuQ
Ze05sbBrXtk30CnH71itRwrfl9WW7e0arrcQ2D2fWLYPp0bJ07HEC1TgIjXYSRMLPPjebPamZXF7xlmxtubxmgHtscSL1G8i9RQ6/Yd4+656
8jML+LJMT2ZkNhep0xqGBW4SA1jreXNH/pw96qtGoSxvt3DqFvuG20xxtfUIGL1CB5RaU8/N5GFm7K4NLU0t7X781wnKPKZSgYVUEKkH5ky
J5mDOYX8/vLu3D60g0621HAqCJEGZ61lzp9PPn52pqH12PB5EQS4po7HUV8QAuUeSR0Fjfx2EcpfLL+EMO6RtL891p0ai+07HE1TgIkFqa0Y
U+asZXf2CR4a2ZUpwztTp450mdQmKnCRIPT+2gP84eMUGtUPZfbt5zGku0unI4kDV0AiaQSwMXj6by7poDJH2szovj+hpVOMzpw0IQFbHkN
```

```
nbdiff notebook-v1.ipynb notebook-v2.ipynb
--- notebook-v1.ipynb 2019-09-24 22:43:55.669188
+++ notebook-v2.ipynb 2019-09-24 22:43:57.525607
## inserted before /cells/0:
+ markdown cell:
+ source:
+      "# Example of Notebook Diffing!"

## deleted /cells/0:
- markdown cell:
- source:
-      "# This is a placeholder title!"

## modified /cells/1/source:
@@ -1,3 +1,3 @@
-Here's some description text, maybe has a typo in it ...
+Here's some description text, now typo free!

Here's some text that won't change.

## modified /cells/3/source:
@@ -1,3 +1,3 @@
-amazing_math = [1, 2, 3, 4, 5]
+amazing_math = [1, 2, 3, 4, 5, 6]

amazing_math = [x**2 for x in amazing_math]

## inserted before /cells/4:
+ markdown cell:
+ source:
+      "# A Plot"

## modified /cells/4/outputs/0/data/text/plain:
- [<matplotlib.lines.Line2D at 0x7f30ab9b86d8>]
+ [<matplotlib.lines.Line2D at 0x7f7cbc8395f8>]

## inserted before /cells/4/outputs/1:
+ output:
+ output_type: display_data
+ data:
+      image/png: iVBORw0KGGo...<snip base64, md5=f6fafa16789fcf2488 ...>
+      text/plain: <Figure size 432x288 with 1 Axes>
+      metadata (unknown keys):
+      needs_background: light

## deleted /cells/4/outputs/1:
- output:
- output_type: display_data
- data:
-      image/png: iVBORw0KGGo...<snip base64, md5=01fb4236a7c64f48 ...>
-      text/plain: <Figure size 432x288 with 1 Axes>
-      metadata (unknown keys):
-      needs_background: light

## modified /cells/5/source:
@@ -1,5 +1,5 @@
TODO:
- [ ] Better title
- [ ] Fix typo
- [ ] Section headings
- [ ] Go up to 6 squared!

+ - [x] Better title
```