Data analysis with Jupyter Notebook for Open Science

Prof Hans Fangohr
European X-ray Free Electron Laser
Photon and Neutron Open Science Cloud

Amsterdam, 7 May 2019

Hans.Fangohr@xfel.eu
https://fangohr.github.io
@ProfCompMod

Photon and Neutron Open Science Cloud – team effort

- Sandor Brockhauser (EuXFEL)
- Aidan Campbell (ESRF)
- Hans Fangohr (EuXFEL)
- Andy Götz (ESRF)
- Jamie Hall (ILL)
- Jerome Kieffer (ESRF)
- Thomas Kluyver (EuXFEL)
- Eric Pellegrini (ILL)
- Jean-François Perrin (ILL)
- Carlos Reis (CERIC-ERIC)
- Thomas Rod (ESS)
- Robert Rosca (EuXFEL)
- Jesper Selknaes (ESS)
- Krzysztof Wrona (EuXFEL)

- ESRF: European Synchrotron Radiation Facility
- ILL: Institut Laue-Langevin
- EuXFEL: European X-ray Free Electron Laser Facility
- ESS: The European Spallation Source
- CERIC-ERIC: The Central European Research Infrastructure Consortium
- ELI: Extreme Light Infrastructure

Outline

- Open Science, Data Analysis, Jupyter
- PaNOSC requirements and vision
- Challenges
- Summary

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

Data Analysis for Open Science

- (FAIR) data is central for Open Science
- *Data* provides potential for new understanding
- *Data analysis* extracts the meaning from the data
- Publications based on data
  - Data sources should be known
  - Central findings (figures, tables, numbers) should be reproducible

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

Jupyter Notebook

- Cells contain
  - (formatted) text
  - Code
  - Output from code execution
  - (multimedia)
- Can execute cells interactively
- Can save, close, load and re-execute
- Can export to other file formats
- Remote execution via https possible

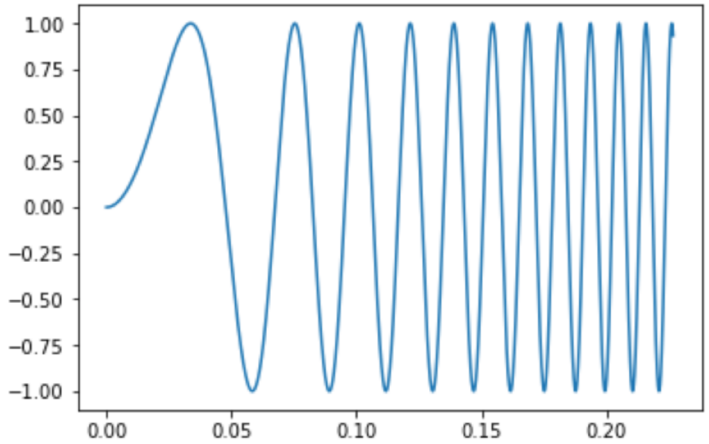
Cells can contain text and latex equations such as $f(x) = \sin(2\pi\omega t^2)$ and $\omega = 220$ Hz.
We can use code to define the corresponding functions:

In [2]:
```
import numpy as np
def f(t):
    omega = 220
    return np.sin(2 * np.pi * omega * t**2)
```

Let's compute the data and plot the beginning of it:

In [3]:
```
t = np.linspace(0, 2, 44100)
y = f(t)
## Show plots inside the notebook
%matplotlib inline
import pylab
pylab.plot(t[0:5000], y[0:5000])
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x1047ada58>]
```



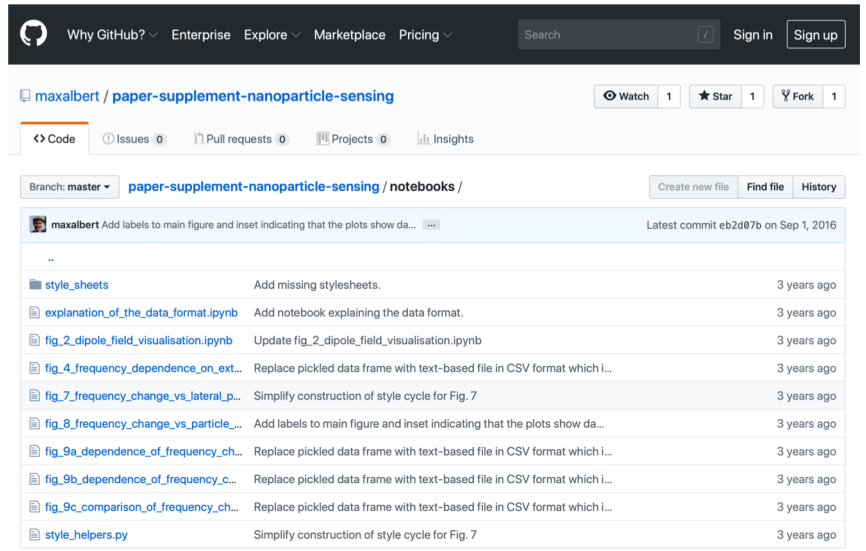
We can integrate media: images, videos, interactive elements and sound:

In [4]:
```
from IPython.display import Audio
Audio(y, rate=44100)    # plays the data in y as audible signal
```

Out[4]:

Jupyter Notebook for Open Science

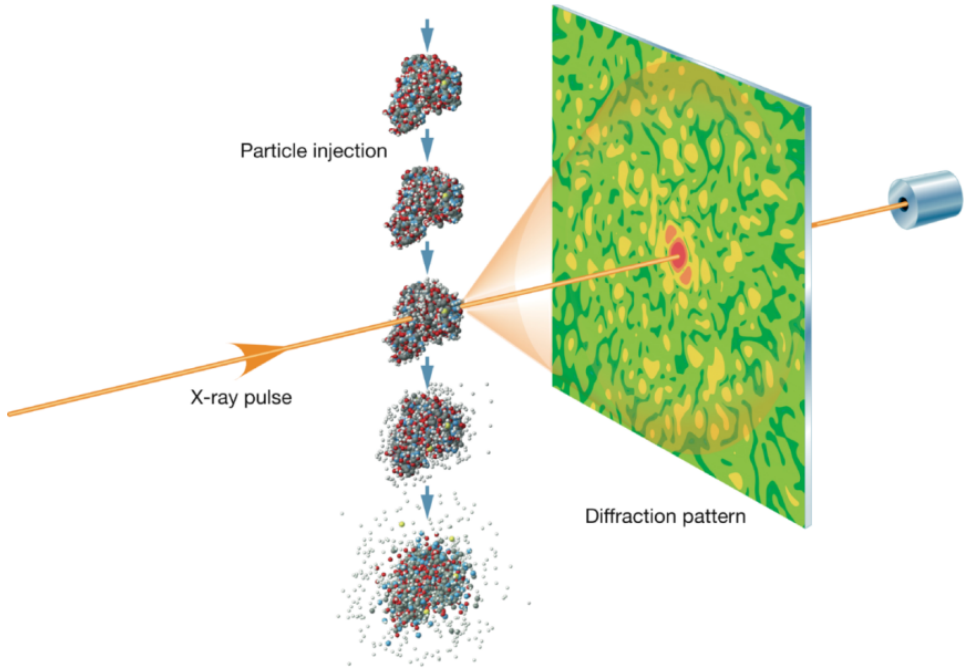
- Combination of code, output and annotation *in one document*
- If used appropriately, makes publications *reproducible*
  - For example: one notebook per figure in publication (examples: [1], [2])
- Notebooks from reproducible publications make the work *re-usable*
  - Currently, lots of time is used by researchers to repeat the work of others, before they can advance science.



[1] Appl. Phys. Lett. 109, 122401 (2016), https://doi.org/10.1063/1.4962726, https://github.com/fangohr/paper-supplement-2016-dmi-nanocylinder-hysteresis
[2] Nanotechnology 27, 455502 (2016), https://doi.org/10.1088/0957-4484/27/45/455502 https://github.com/maxalbert/paper-supplement-nanoparticle-sensing

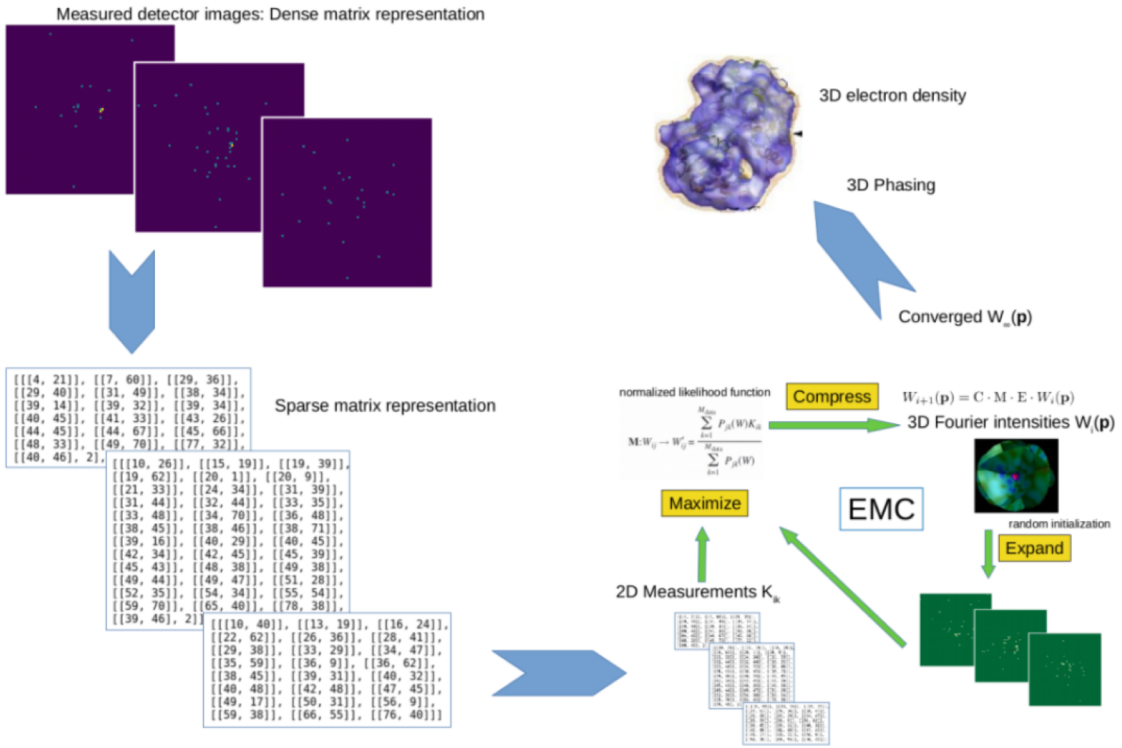
Photon and Neutron Open Science Cloud (PaNOSC)

- Research facilities using Photons and Neutrons for imaging
  - Underpins a wide range of fundamental and applied research, from basic physics to drug design
- European XFEL: use short-wavelength photons to image small things
  - Infrastructure
    - 3.4km tunnel
    - Accelerate electrons, then create photons from electrons
  - Data volume
    - Typical detector: 1 million pixels, each using 2 bytes
    - up to 27,000 X-ray pulses per second
    - → 2 byte * 1,000,000 * 27,000 / s = 54 GB/s
    - → 194 TB/h (theoretical peak)



Current state of data analysis: example European XFEL

- Data source:
  - 2d image detectors (up to 27,000 images /s)
  - Scalar values (typically 10 values per second)
  - others
- Data analysis
  - Calibration, data filtering, reduction, azimuthal integration, real space reconstruction, …
- Many steps involving different programs
  - connect `via ssh -X` to HPC cluster
  - some programs use GUIs (X display at EuXFEL)
  - some programs use command line
  - Increasingly Jupyter Notebooks



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

First, let's load a run containing LPD data:

https://karabo-data.readthedocs.io/en/latest/index.html

[1]:
```
from karabo_data import RunDirectory, by_index

run = RunDirectory('fxe_example_run/')
# Using only the first three trains to keep this example light:
run = run.select_trains(by_index[:3])

run.instrument_sources
```

[1]:
```
frozenset({'FXE_DET_LPD1M-1/DET/0CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/10CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/11CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/12CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/13CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/14CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/15CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/1CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/2CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/3CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/4CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/5CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/6CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/7CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/8CH0:xtdf',
           'FXE_DET_LPD1M-1/DET/9CH0:xtdf',
           'FXE_XAD_GEC/CAM/CAMERA:daqOutput',
           'FXE_XAD_GEC/CAM/CAMERA_NODATA:daqOutput',
           'SA1_XTD2_XGM/DOOCS/MAIN:output',
           'SPB_XTD9_XGM/DOOCS/MAIN:output'})
```

Normal access methods give us each module separately:

[2]:
```
data_module0 = run.get_array('FXE_DET_LPD1M-1/DET/0CH0:xtdf', 'image.data')
data_module0.shape
```

[2]:
```
(384, 1, 256, 256)
```

The class `karabo_data.components.LPD1M` can piece these together:

[3]:
```
from karabo_data.components import LPD1M
lpd = LPD1M(run)
lpd
```

[3]:
```
<LPD1M: Data interface for detector 'FXE_DET_LPD1M-1' with 16 modules>
```

[4]:
```
image_data = lpd.get_array('image.data')
```

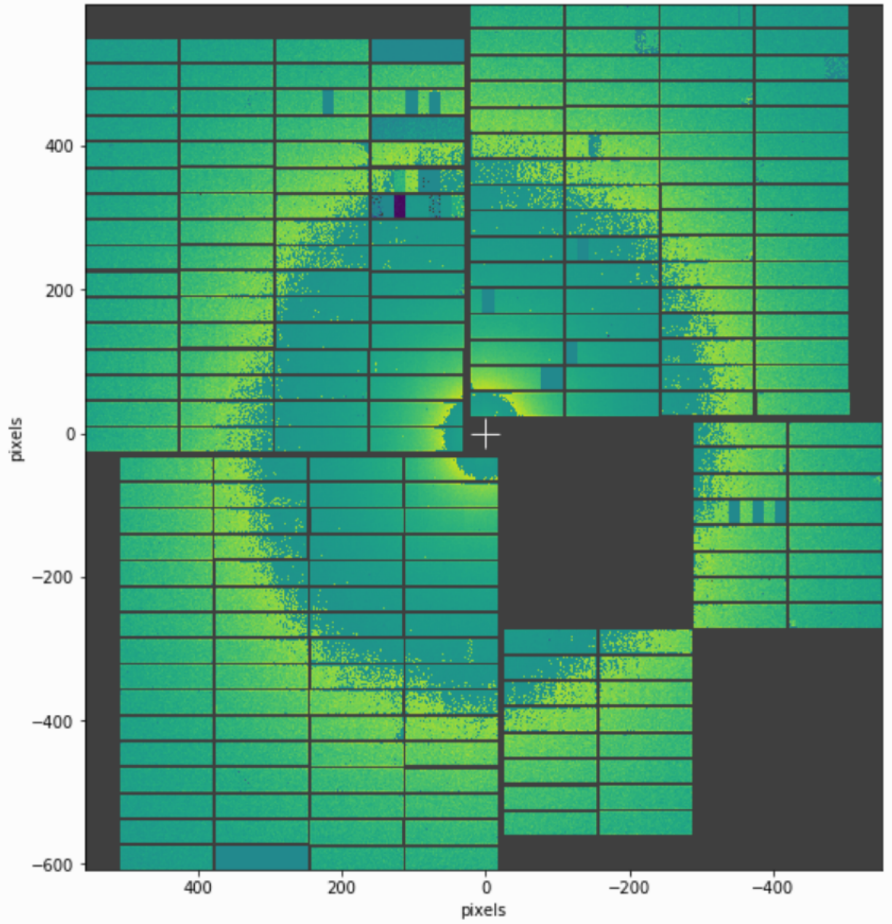
[10]:
```
# From March 18; converted to XFEL standard coordinate directions
quadpos = [(11.4, 299), (-11.5, 8), (254.5, -16), (278.5, 275)]  # mm

geom = LPD_1MGeometry.from_h5_file_and_quad_positions('lpd_mar_18_axesfixed.h5', quadpos)
```

Reassemble and show a detector image using the geometry:

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

[11]:



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

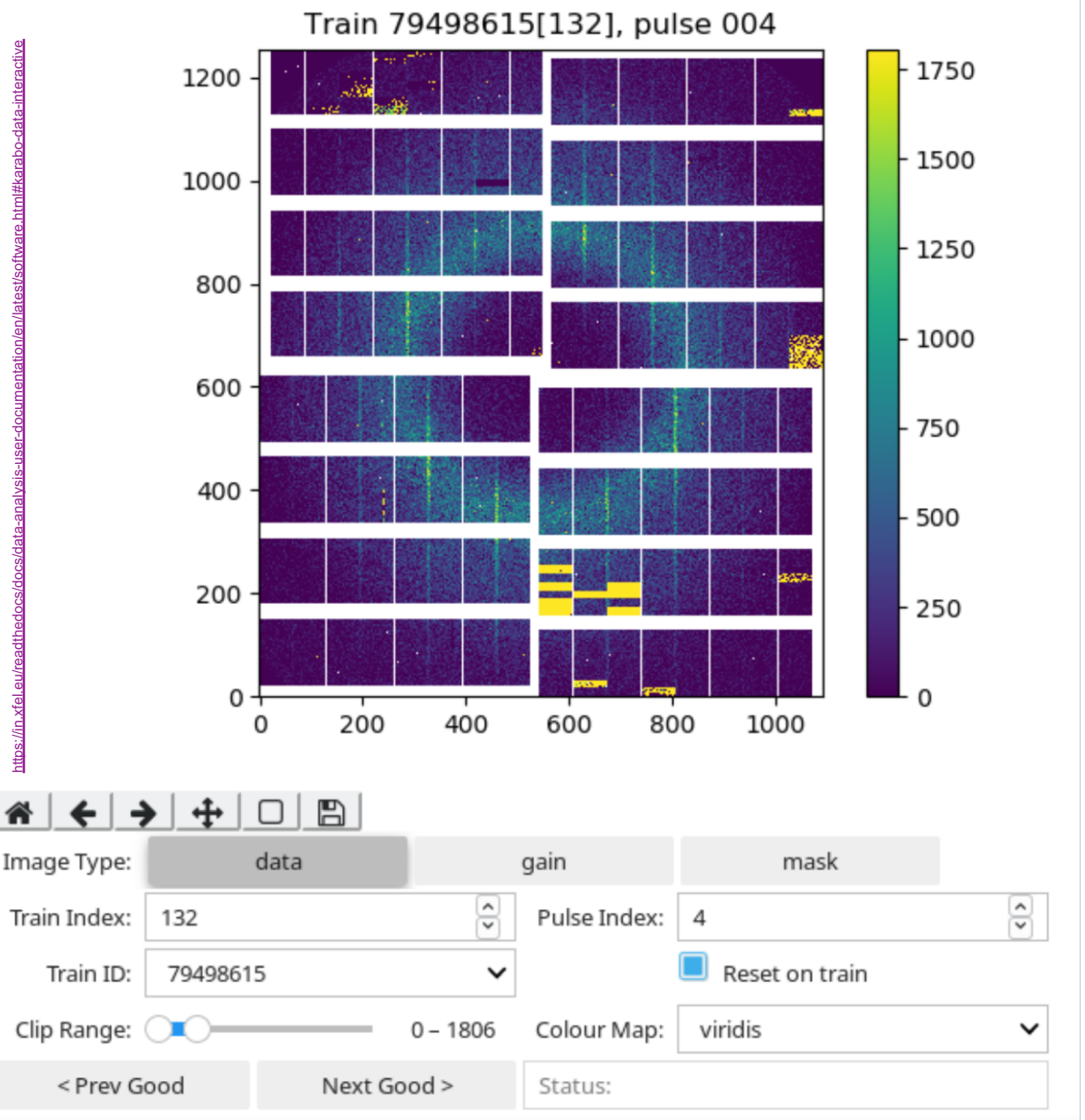
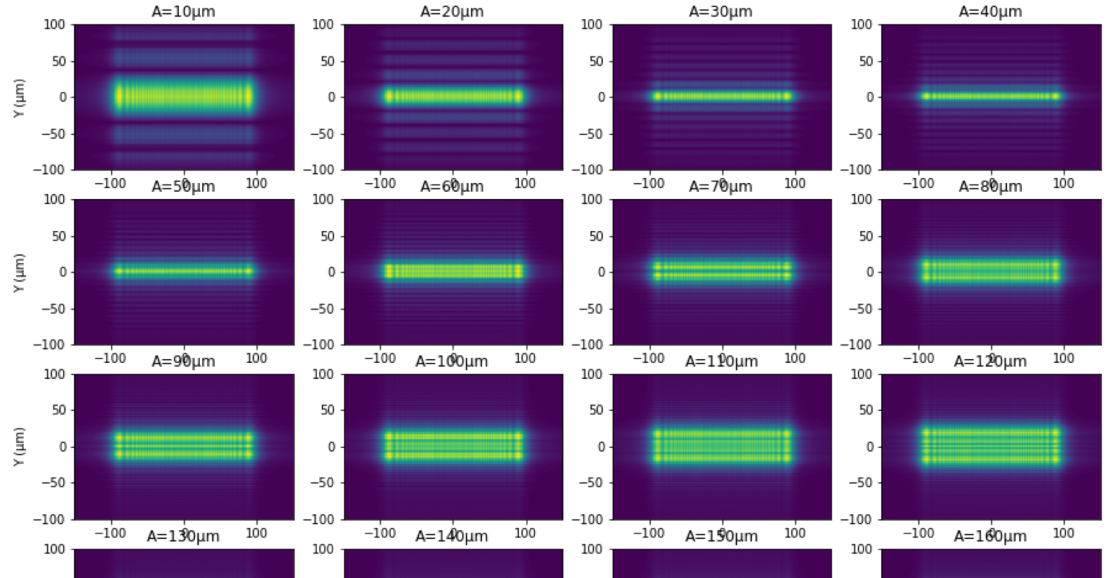


Fig. 6.2 Example image showing the karabo data interactive widget and plot layout

```
In [7]: # propagation of a stack of A x 200 microns apertures, varying A
w = Wavefront(d=np.zeros((16, 512, 512), dtype=np.complex64), pixel_size=1e-6, wavelength
=1.5e-10)
x, y = w.get_x_y()
d = w.get()
for i in range(16):
    a = 5e-6 / 2 * (i + 1)
    d[i] = ((abs(y) < a) * (abs(x) < 100e-6)).astype(np.complex64)
w.set(d)
w = PropagateFRT(1.2) * w
figure(figsize=(15, 10))
x, y = w.get_x_y()
x *= 1e6
y *= 1e6
for i in range(16):
    subplot(4, 4, i + 1)
    imshow(abs(fftshift(w.get()[i])), extent=(x.min(), x.max(), y.min(), y.max()), origin
='lower')
    title("A=%dμm" % (10 * (i + 1)))
    if i >= 12:
        xlabel('X (μm)')
    if i % 4 == 0:
        ylabel('Y (μm)')
    xlim(-150, 150)
    ylim(-100, 100)
suptitle("Fractional Fourier propagation (0.5m) of a A x 200 μm aperture")
```

```
Out[7]: <matplotlib.text.Text at 0x1141b3208>
```



Solution architecture for PaNOSC vision:

- Finding data:
  - Web interface with data base of experiment metadata
- Exploring and analysing data remotely (=in the cloud):
  - JupyterHub serving relevant notebooks
  - Move data analysis code into the notebook
  - remote desktop in browser, connected to Desktop of virtual machine

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

PaNOSC Use case 1: reproducibility and re-usability published results

- For a given publication based on facility data, users can
  - Find the data (through the EOSC web portal or URL/DOI in paper)
  - Access the data through web portal
  - Inspect the data analysis (notebook) that led to key figures / statements in the publication
  - Re-execute the data analysis through (→ reproducibility)
  - Modify and extend the notebook (→ reusability)

- Users may include scientists, interested public, journal editors and reviewers, representatitves from research councils, . . .

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

PaNOSC Use case 2: enable new data analysis on existing data sets

- Users can
  - Search and find data sets from experiments through web portal
  - Access the data through web portal
  - Choose from appropriate selection of data analysis tools (=Jupyter Notebook templates)
  - Execute the notebook
  - Modify and extend the notebook

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

Challenges 1

- Different facilities
  - Currently 6 facilities involved
  - Generally use different ways to store meta data data
  - Common way of classifying data sets (and experiment types) ?

- Data scale
  - For some data sets, the data cannot be moved to the compute resource

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

Challenges 2

- Analysis in Notebook
  - Computational environment – software (containers?)
    - Need to provide the right computational environment for each analysis type
    - How can we maintain computational environments in the future (Binder-like?)
    - Extending up to the life-time of publications and data sets
  - Which analysis is appropriate for data set → classification of data sets
  - Making analysis capabilities available in the Jupyter Notebook
    - Command line drived and Python based computation straightforward,
    - GUI-based tool more difficult / impossible
  - What to do with resulting analysis?
  - Some analysis notebooks require significant HPC resources (execute jobs from notebook?)
  - Computational environment – hardware, GPUs?

panosc

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

Challenges 3

- Concurrent development with EOSC hub
- Complicated access rights for data sets
  - Data policies with embargo period
- Publication use case:
  - Requires collaboration with scientists
    - Preparation of data analysis notebooks
  - Social / cultural challenge
  - Helped by changing metrics and expectations from funding bodies and journals
  - Research facilities can lead by example

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

Summary

- Introduction PaNOSC project (Photon and Neutron Open Science Cloud), http://panosc-eu.github.io
- Focus on data analysis
- Use cases
  - Make publications reproducible
    - EOSC MyBinder instance as first step?
  - Allow convenient exploration of existing data sets
- Contributions / brain storming / collaboration welcome

Contact presenter:
Hans.Fangohr@xfel.eu,
@ProfCompMod,

Contact PaNOSC project:
Andy Götz,
Andy.Gotz@esrf.fr

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823852.