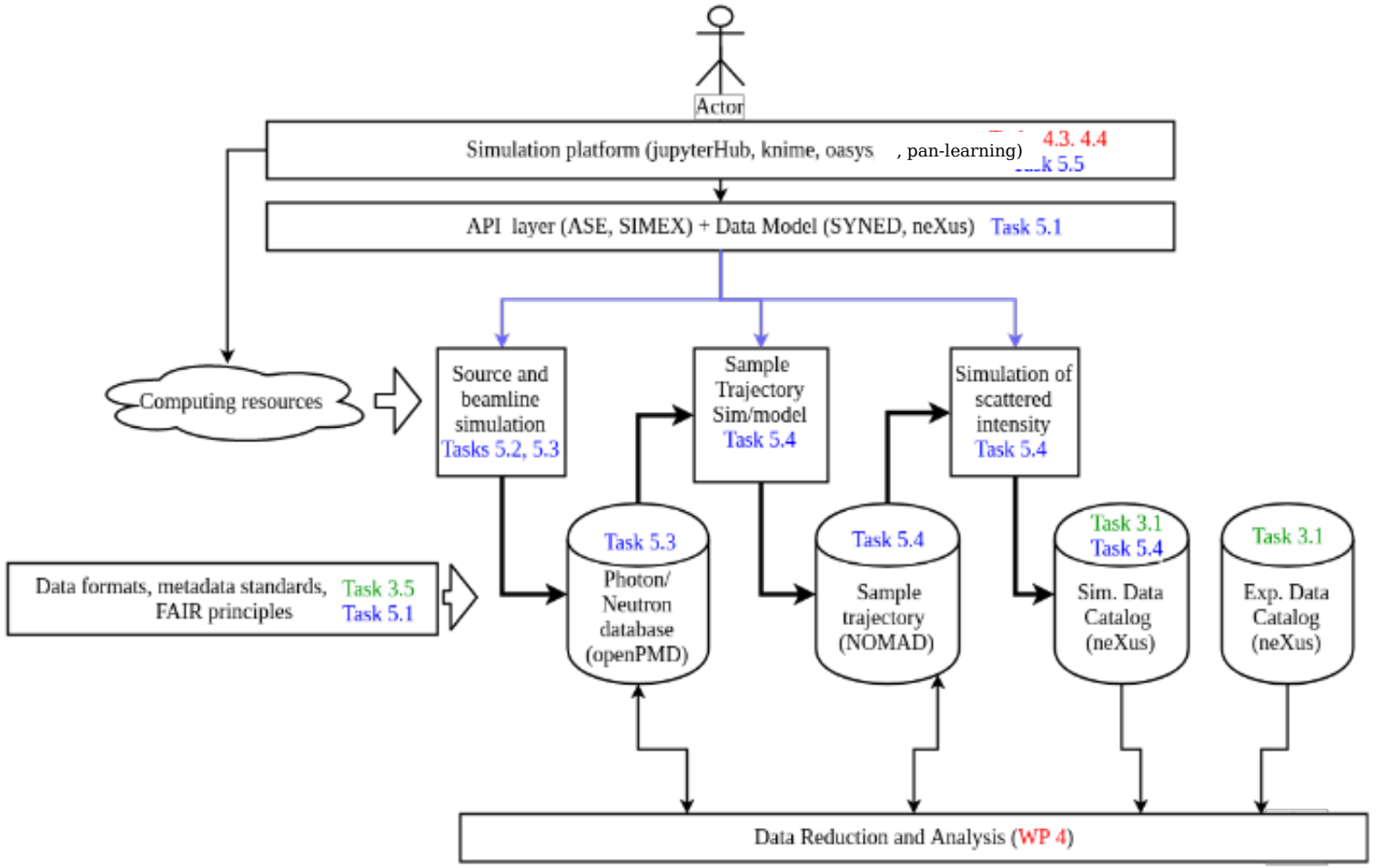
WP5
Virtual Neutron and Photon Laboratory

Carsten Fortmann-Grote, Juncheng E, Mads Bertelsen, Shervin Nourbakhsh, Mousumi Upadhyay Kahaly, Zsolt Lécz, Aljosa Hafner

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

A simulation platform for P&N science



Key Components

- SIMEX: Photon Experiment Simulation platform
- McStas: Neutron Raytracing engine
- OASYS: X-ray optics simulation environment
- OpenPMD: Metadata standard for particle-mesh data
- NEXUS: Metadata standard for experiments
- Jupyter: Data analysis platform

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

Experiment Simulation services add value to

- **RI Users:**
  - Support beamtime propopals
  - Assist in setting up experimental configurations
  - Estimate data quality, volume, scope, and range
- **RI operators:**
  - Optimize beamline optics by maximizing data yield and quality
  - Validate beamtime proposals
- **Students:**
  - Simulations in E-learning platform (→ WP8)
- **PaNOSC Project:**
  - Provide realistic test data for other work packages (→ WP4)

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

Experiment Simulations foster adoption of FAIR Data principles

- **Interoperable** simulation modules
  - Our simulation pipelines rely on standardized data interfaces to allow seamless connectivity between simulation modules
- **Reusable** workflows
  - Harmonized simulation API for self-documenting and self-contained workflows

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

Provision of Simulation Services

Jupyter notebook services

- Preconfigured simulation workflows, maintained in ViNyL github repository
- Accessible from cloud computing services (binder, jupyterHub)
- Seamless connection to data analysis services (WP4)
- Prototype: jupyterHub@DESY
  → collaboration with EXPANDS

Remote Desktop services

- Oasys x-ray optics simulation
- User connects to cloud computing service
- Open and run Oasys as if in local environment
- Prototype: RAFEC@CERIC-ERIC, integrated into Elettra's Virtual Unified Office

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

Deliverables

No.	Title	Due	Verification	Status
D5.1	Prototype simulation data formats as openPMD domain specific extensions including example datasets	M12	Written report	Done
D5.2	Release of documented simulation APIs	M24	Software	In Progress
D5.3	Repository of documented jupyter notebooks and Oasys canvases showcasing simulation tasks executable via JupyterHub or remote desktop.	M42	Software	Start after D5.2
D5.4	VINYL software tested, documented, and released, including integration into interactive data analysis workflow with feedback loop.	M48	Written report + Software	Start after integration

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

Example: Neutron simulation service

- Target simulation with Density-Functional Theory (ASE interface to QuantumEspresso)

```
In [8]: from ase import io, Atom, Atoms
        atomCIF = io.read(CIF_file)

        print(atomCIF)
        print(atomCIF.get_positions())

Atoms(symbols='N8', pbc=True, cell=[5.65, 5.65, 5.65])
[[0.415275 0.415275 0.415275]
 [2.409725 5.234725 3.240275]
 [3.240275 2.409725 5.234725]
 [5.234725 3.240275 2.409725]
 [5.43078  5.43078  5.43078 ]
 [3.04422  0.21922  2.60578 ]
 [2.60578  3.04422  0.21922 ]
 [0.21922  2.60578  3.04422 ]]

In [9]: from ase.build import bulk
        from ase.calculators.espresso import Espresso
        from ase.constraints import UnitCellFilter
        from ase.optimize import LBFGS

        pseudopotentials={'N': pseudopotfile}

In [10]: calc = Espresso(pseudopotentials=pseudopotentials,
                         tstress=True, tprnfor=True, kpts=(6,6,6),ecutrho=480,ecutwfc=60,ibrav=0,
                         nat=8,ntyp=1,
                         calculation='relax',occupations='smearing',smearing='cold',degauss=0.001,
                         outdir='tmp',pseudo_dir=pseudo_dir,
         conv_thr        = 0.0000001,
         mixing_mode     = 'plain',electron_maxstep = 80,
         mixing_beta     = 0.5,ion_dynamics='bfgs',
         )


         atom = atomCIF
         atom.calc = calc

         #atom.set_calculator(calc)
         #atom.get_potential_energy()
         #fermi_level = calc.get_fermi_level()
```

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

Example: Neutron diffraction

- Target simulation with Density-Functional Theory (ASE interface to QuantumEspresso)
- Neutron beam transport and diffraction simulation with McStas/McStas-Script

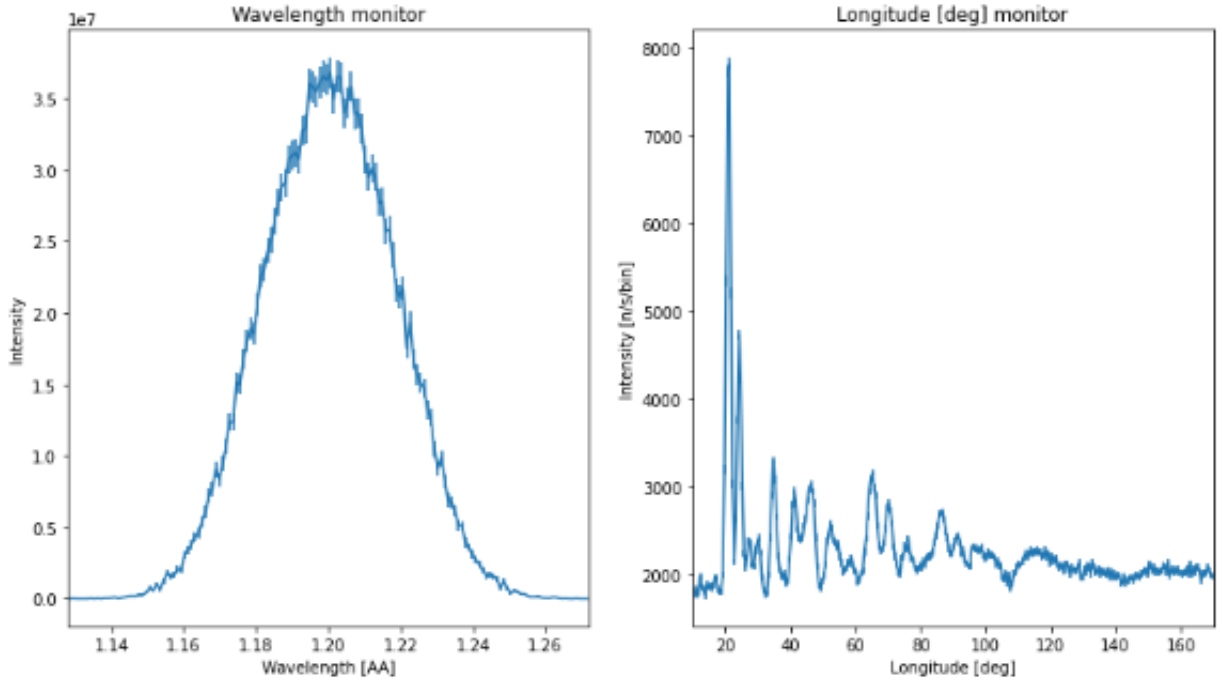
```
In [31]: data = Instr.run_full_instrument(ncount=5E6, mpi=4,
                                          foldername=mcstas_outdir, increment_folder_name=True)

         #plotter.make_sub_plot(data)


         # Instr.show_instrument() Uncomment to view instrum
```

```
In [32]: plotter.make_sub_plot(data)

number of elements in data list = 2
Plotting data with name L_mon
Plotting data with name monitor
```




```
Out[32]: <mcstasscript.interface.plotter.make_sub_plot at 0x7f96e7f5c5f8>
```

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

Summary

- Simulations provide core data service for photon and neutron scientists
- The two major backbone infrastructure components are
  - delivered: Standardized data interfaces → openPMD, D5.1
  - in progress: Simulation API → pyvinyl, D5.2
- Application examples demonstrate the usability and flexibility of our simulation environment.

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

Thank you!

carsten.grote@xfel.eu

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

Backup slides

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

Examples

Jupyter notebook services

- X-ray diffraction experiments at Free Electron Lasers
- X-ray diffraction experiments at Synchrotrons
- Density Functional Theory Target simulations
- Neutron diffraction at conventional neutron sources

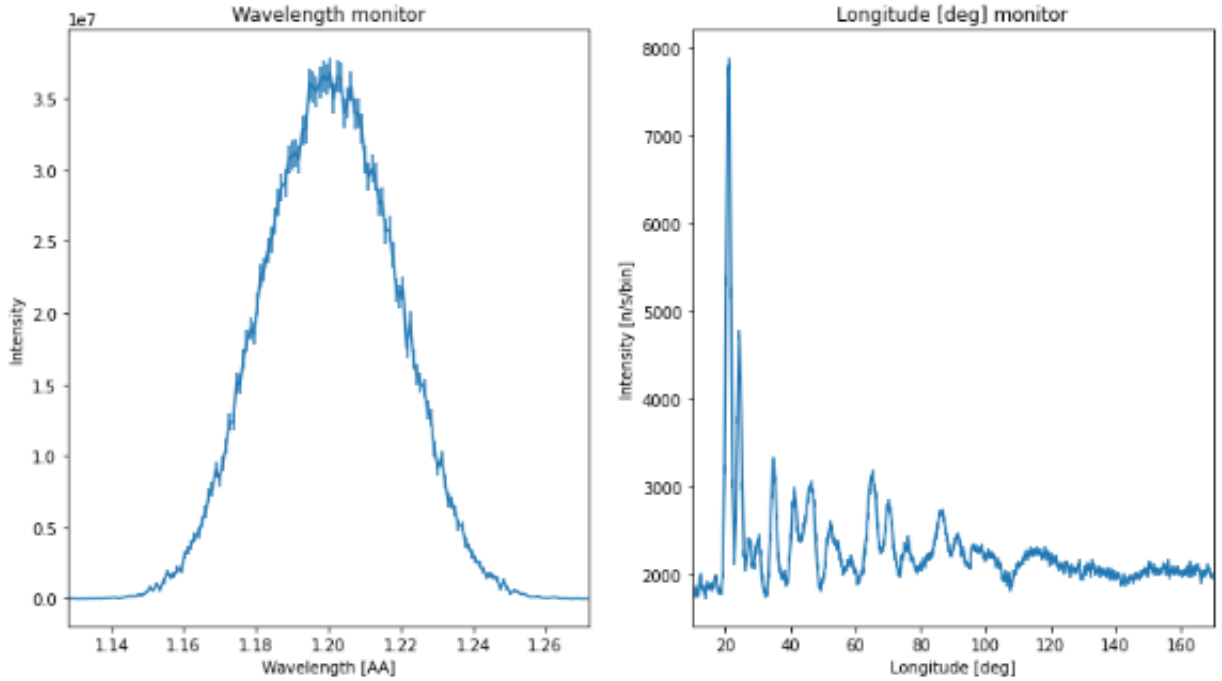
```
In [31]: data = Instr.run_full_instrument(ncount=5E6, mpi=4,
                                          foldername=mcstas_outdir, increment_folder_name=True)

         #plotter.make_sub_plot(data)


         # Instr.show_instrument() Uncomment to view instrum
```

```
In [32]: plotter.make_sub_plot(data)

         number of elements in data list = 2
         Plotting data with name L_mon
         Plotting data with name monitor
```




```
Out[32]: <mcstasscript.interface.plotter.make_sub_plot at 0x7f96e7f5c5f8>
```

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

Examples

Jupyter notebook services

- X-ray diffraction experiments at Free Electron Lasers

Initialize the Diffractor

```
In [21]: diffractor = SingFELPhotonDiffractor(parameters=diffraction_parameters,
                                              input_path='pmi',
                                              output_path="diffr")
```

```
WARNING: Geometry not set, calculation will most probably fail.
```

Run the scattering simulation

```
In [22]: diffractor.backengine()
Out[22]: 0
```

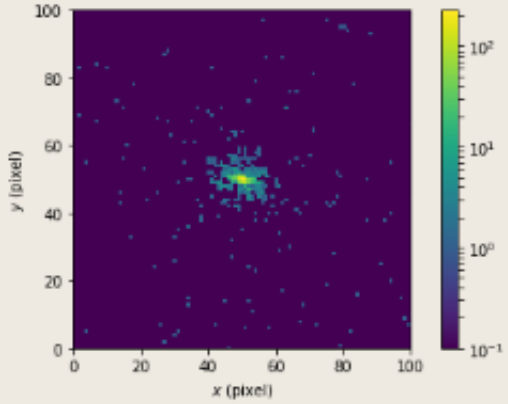
```
In [24]: diffractor.saveH5()
```

Analysis

Setup analysis object

```
In [25]: spi_analysis = DiffractionAnalysis(diffractor.output_path,
                                            pattern_indices=[1],
                                            poissonize=True)
```

```
In [26]: spi_analysis.plotPattern(logscale=True)
```



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

Examples

Jupyter notebook services

- X-ray diffraction experiments at Free Electron Lasers
- X-ray diffraction experiments at Synchrotrons

Detector setup

```
detector_panel = DetectorPanel(
            ranges={
                'fast_scan_min': 0,
                'fast_scan_max': 200,
                'slow_scan_min': 0,
                'slow_scan_max': 200
            },
            pixel_size=220e-6 * meter,
            photon_response=1.0,
            distance_from_interaction_plane=0.25 * meter,
            corners={
                'x': -100,
                'y': -100
            },
        )
detector_geometry = DetectorGeometry(panels=[detector_panel])
```

Polychromatic Beam setup

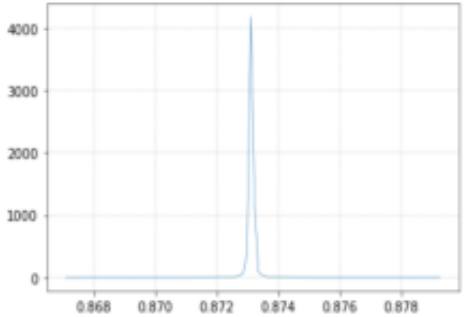
```
bean = './raytracing_out.h5'
```

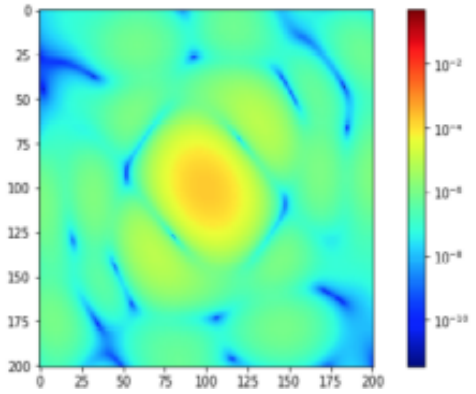
Diffractor setup

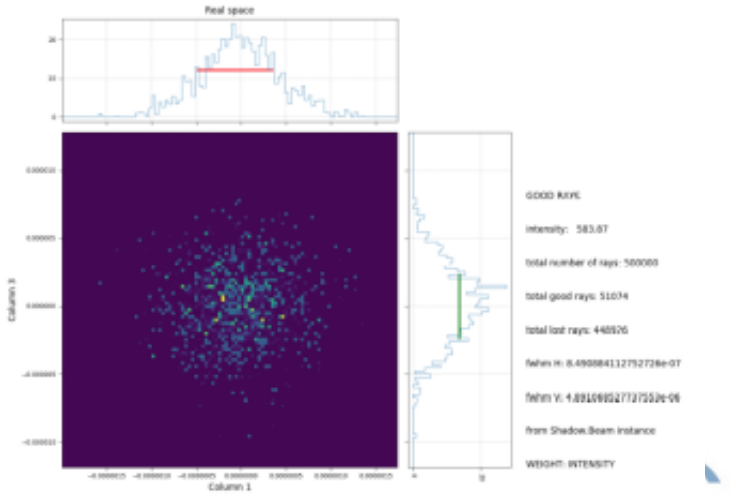
```
outfile = 'diffr_poly_2.txt'

parameters = GAPDPhotonDiffractorParameters(
            detector_geometry=detector_geometry,
            beam_parameters=beam,
            number_of_spectrum_bins = 100)

diffractor = GAPDPhotonDiffractor(parameters=parameters,
                                  input_path='1WUL.pdb',
                                  output_path=outfile)
```



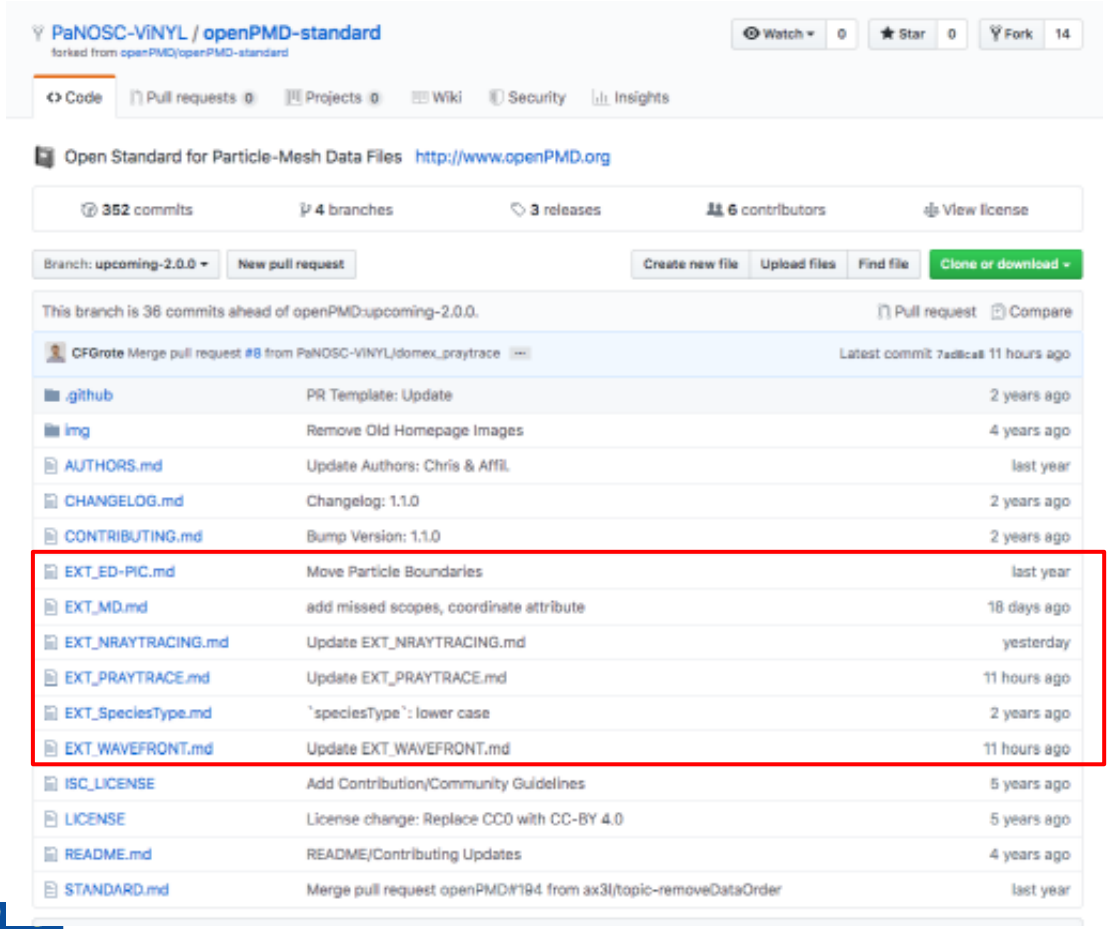




This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

D5.1: openPMD domain extensions

https://github.com/PaNOSC-ViNYL/openPMD-standard/tree/upcoming-2.0.0



- **OpenPMD domain extensions for:**
  - Molecular Dynamics Simulation
  - Neutron raytracing
  - Photon raytracing
  - Coherent Wavefront Propagation
- Integration into openPMI standard ongoing (→ KPIs)

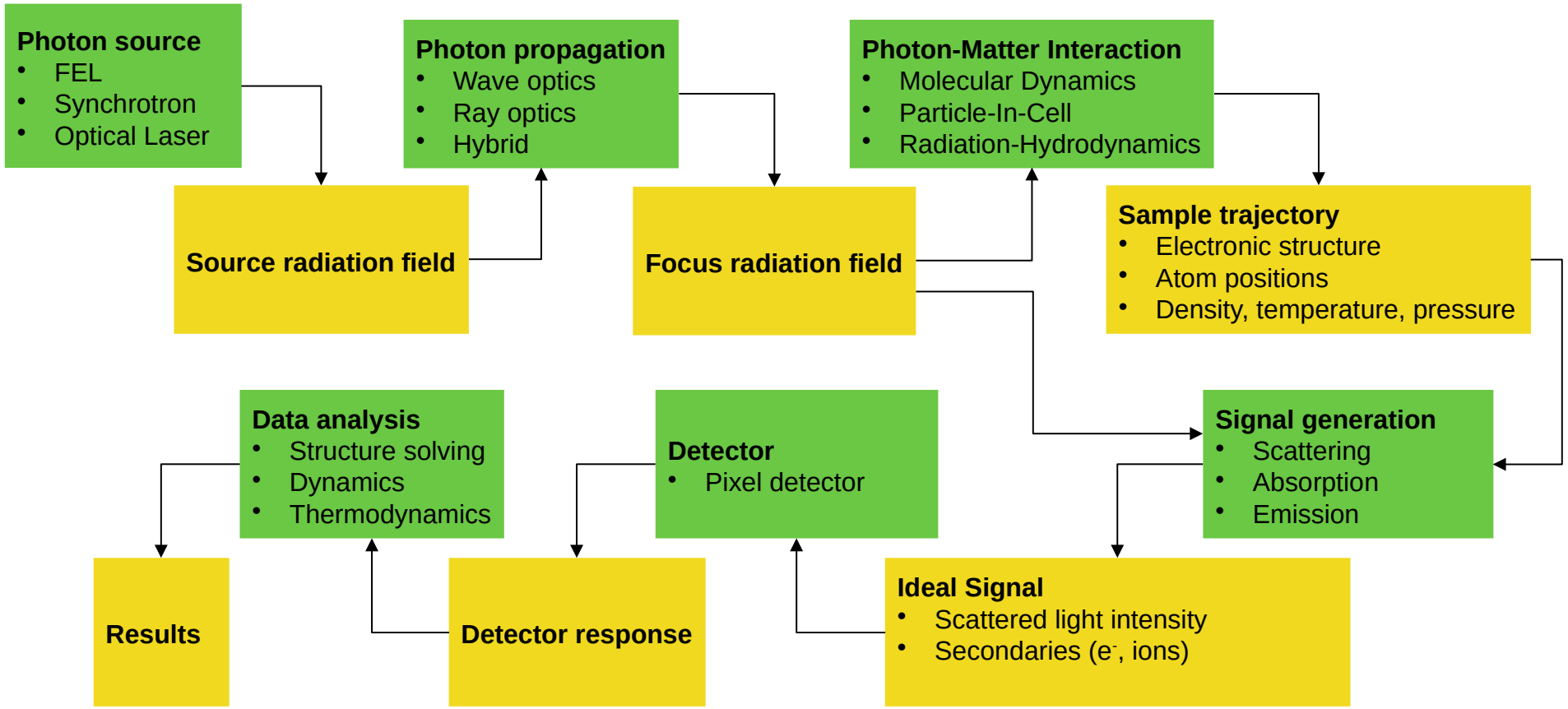
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

SimEx in ViNYL-project

- In collaboration with WP4, we are preparing a serial crystallography example using SimEx to demonstrate the possibility of enabling users to rapidly implement simulations (WP5) and test the data analysis workflow (WP4)
- To harmonize different simulation frameworks under one abstract pyvinyl API, we had a successful practice to bring the SimEx photon simulation framework and OASYS optical simulation environment together

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

SimEx: start to end simulations



Calculators: python APIs to advanced simulation codes

Data interfaces based on open metadata standards: openPMD

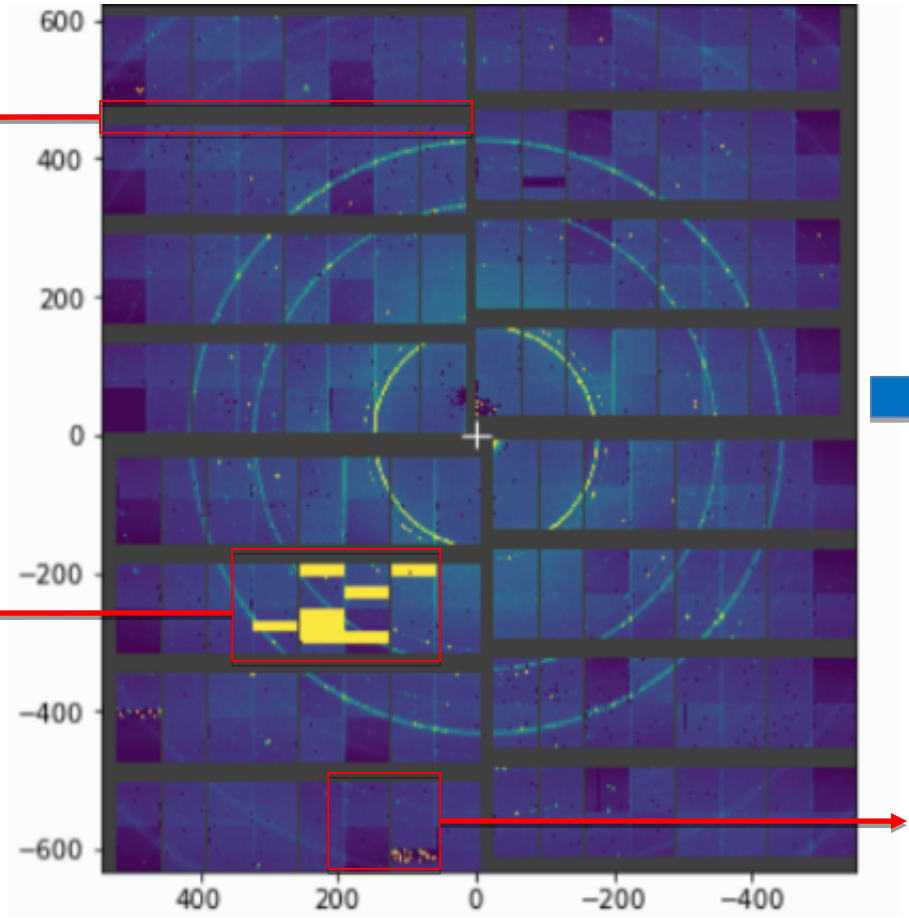
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Serial Crystallography Simulation and Analysis

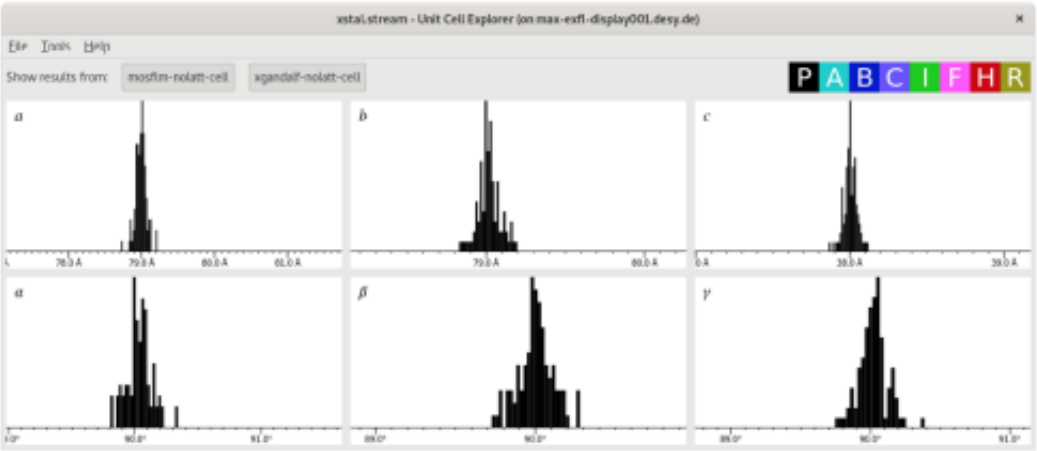
Demonstrate the effects of noise, panel gaps and masks on the results of crystallography analysis with SimEx simulation diffraction pattern results

Panel gaps

Masks



Noise



Scenarios:

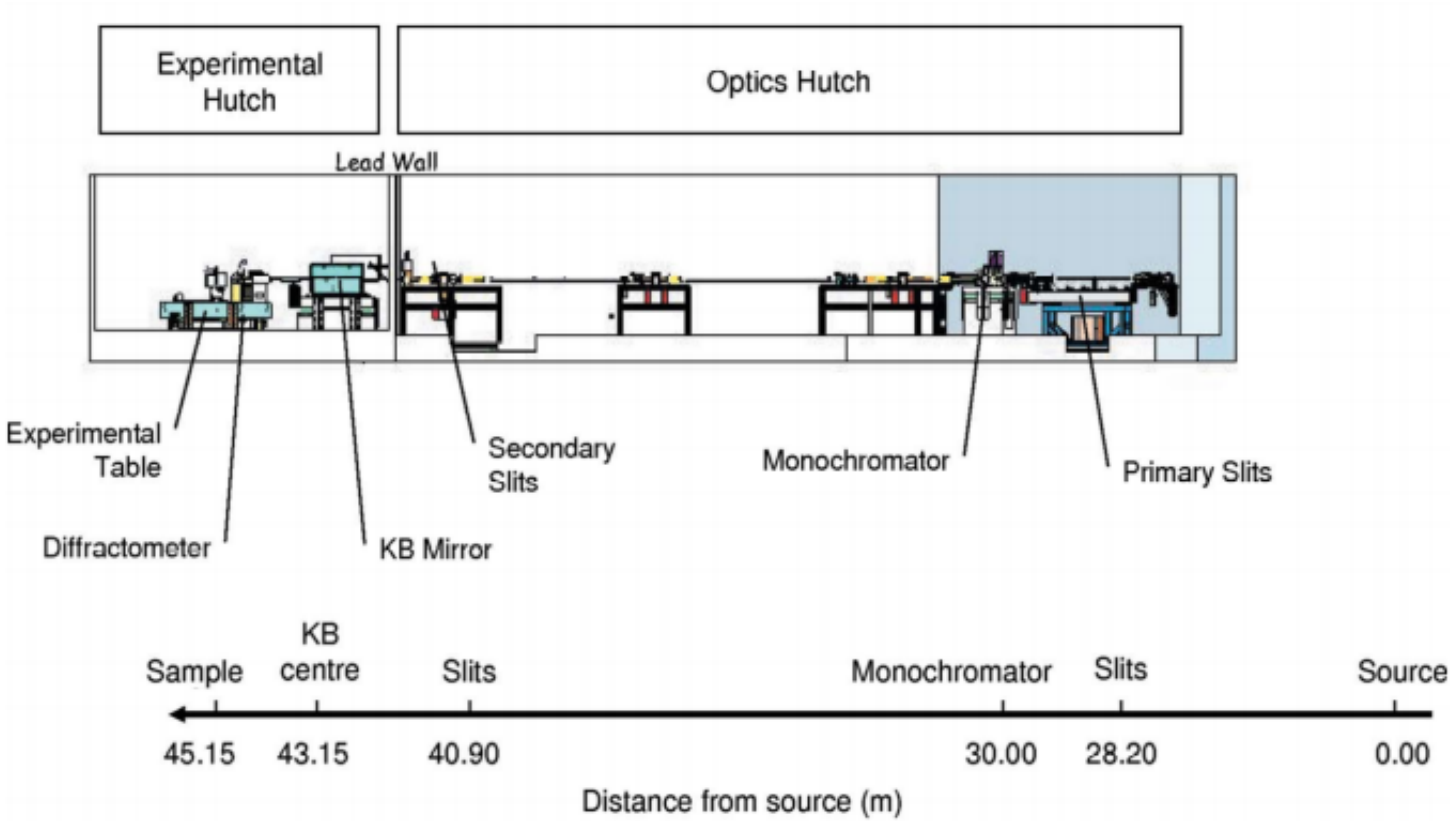
New researchers want to get hands-on experience of data processing for serial crystallography.

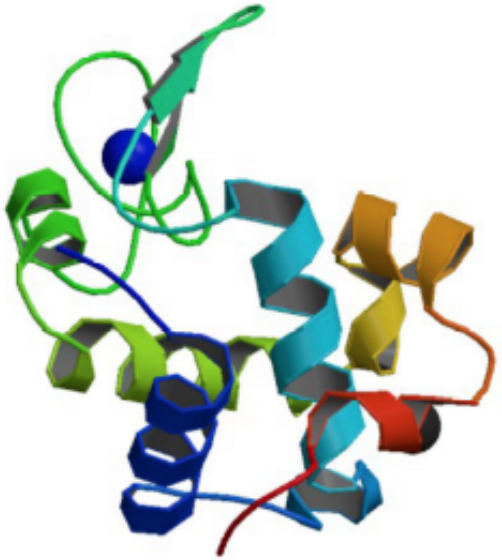
Starting from a working example of a simulation coupled to an analysis pipeline

Explore how different experimental conditions and different analysis options affect the results.

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

Polychromatic x-ray diffraction from lysozyme crystals at the ESRF

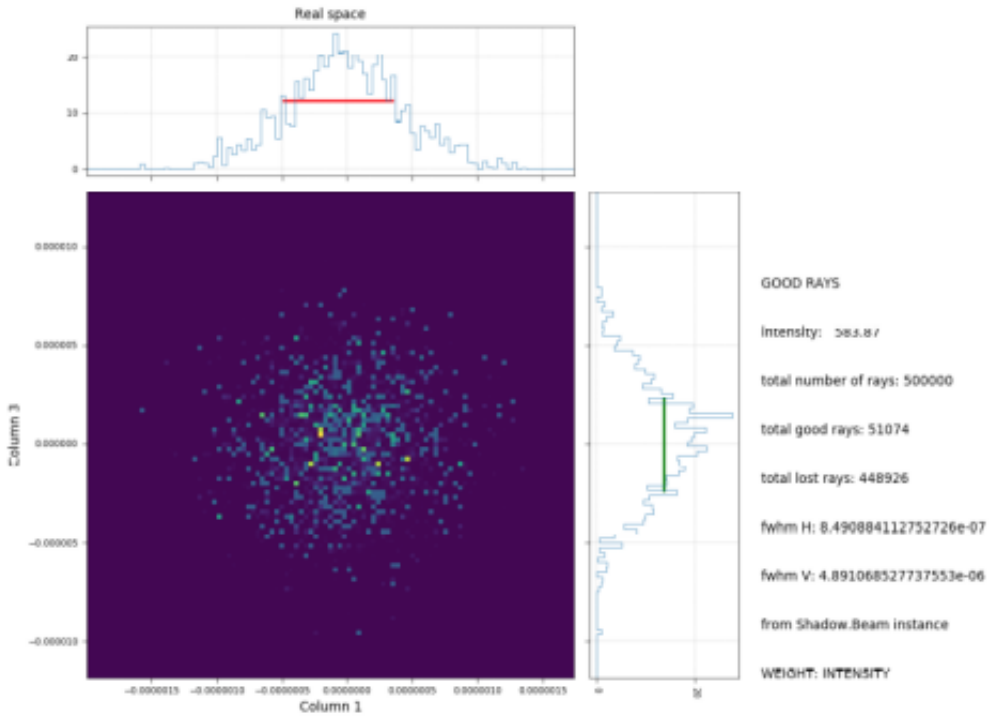




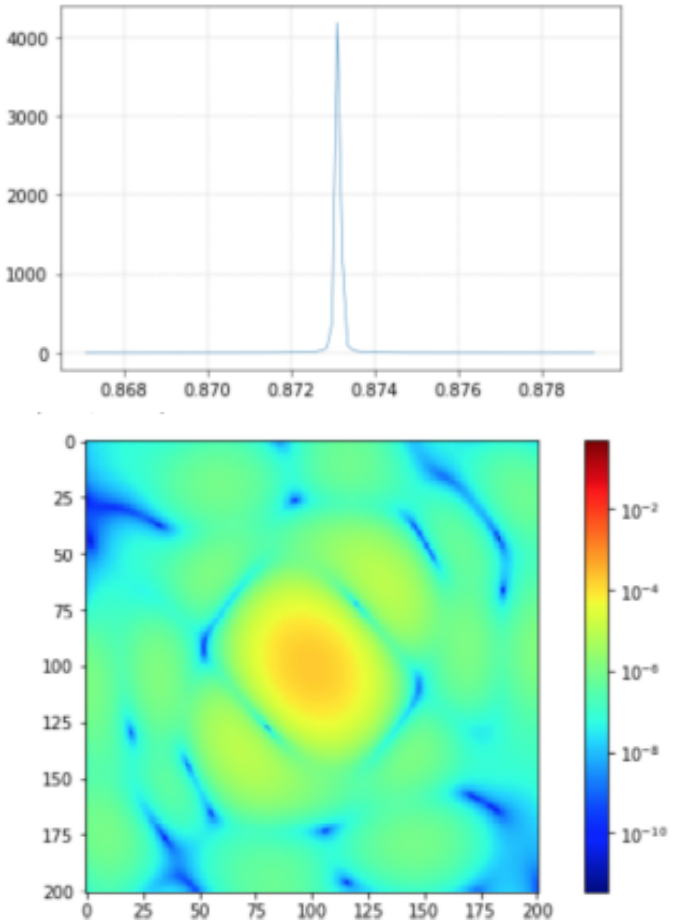
Lyzozyme

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

Polychromatic x-ray diffraction from lyzozyme crystals at the ESRF



OASYS



SimEx

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

Polychromatic x-ray diffraction from lyzozyme crystals at the ESRF

Interface: Jupyter notebook
Simulation:

- Setup detector parameters
- Read raytracing output generated by OASYS
- Setup diffractor parameters
- Run the diffraction backengine

A good practice to bring a harmonized API

Detector setup

```
detector_panel = DetectorPanel(
            ranges={
                'fast_scan_min': 0,
                'fast_scan_max': 200,
                'slow_scan_min': 0,
                'slow_scan_max': 200
            },
            pixel_size=220e-6 * meter,
            photon_response=1.0,
            distance_from_interaction_plane=0.25 * meter,
            corners={
                'x': -100,
                'y': -100
            },
        )
detector_geometry = DetectorGeometry(panels=[detector_panel])
```

Polychromatic Beam setup

```
beam = './raytracing_out.h5'
```

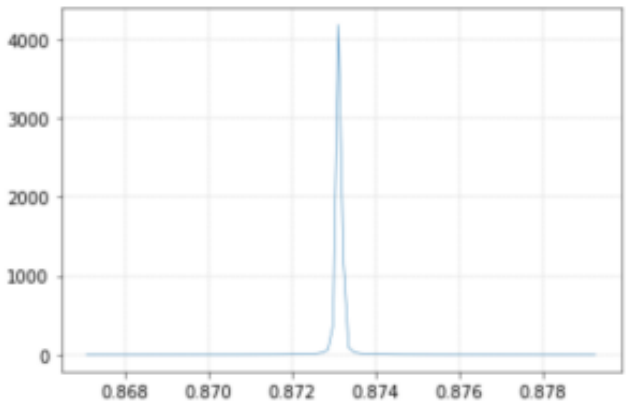
Diffractor setup

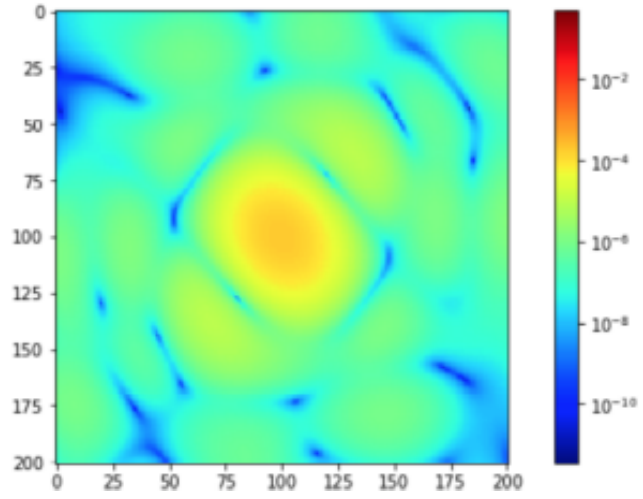
```
outfile = 'diffr_poly_2.txt'

parameters = GAPDPhotonDiffractorParameters(
            detector_geometry=detector_geometry,
            beam_parameters=beam,
            number_of_spectrum_bins = 100)

diffractor = GAPDPhotonDiffractor(parameters=parameters,
                                  input_path='3WUL.pdb',
                                  output_path=outfile)
```

```
diffractor.backengine()
```

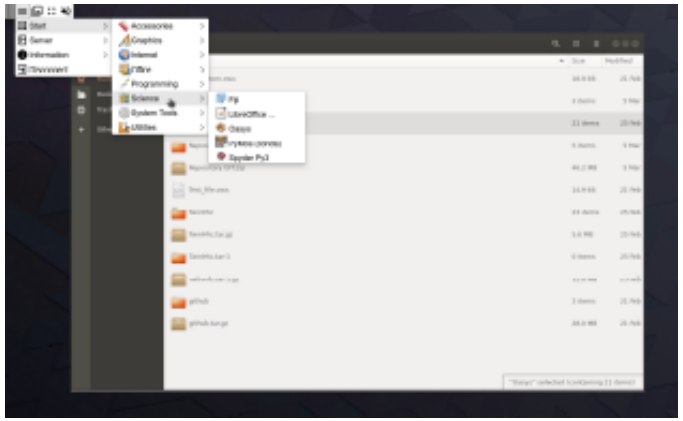




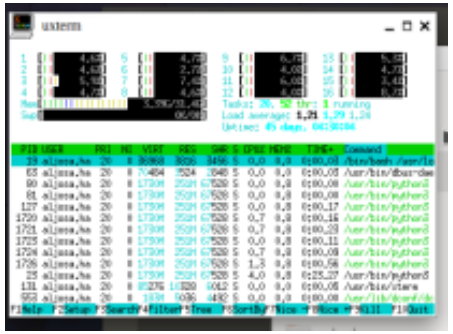
SimEx

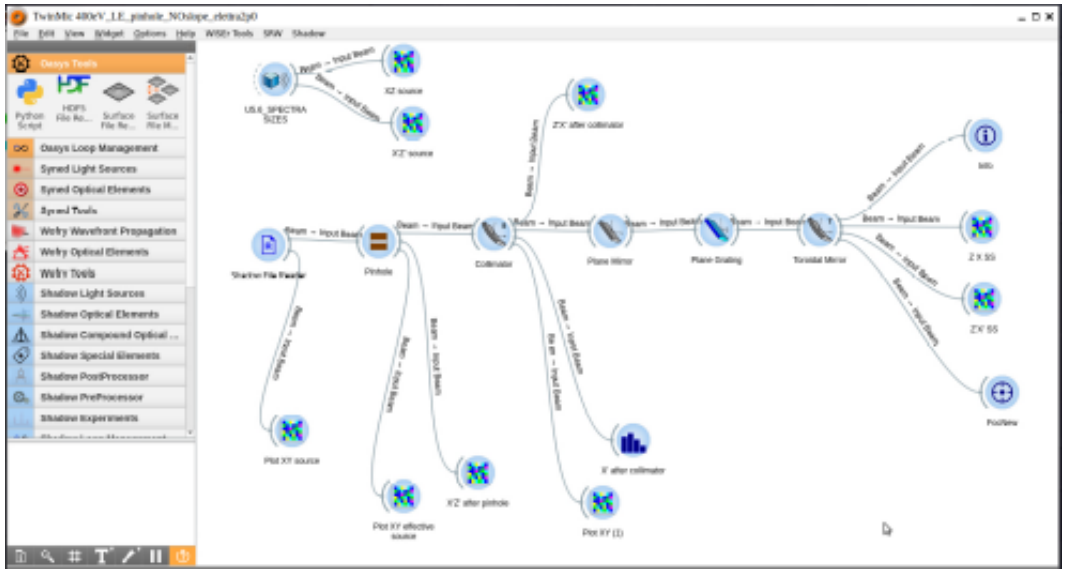
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

Remote access - RAFEC



- RAFEC – internally developed remote access technology built around Xpra and Kubernetes
- To be available through VUO
- Collaborative access



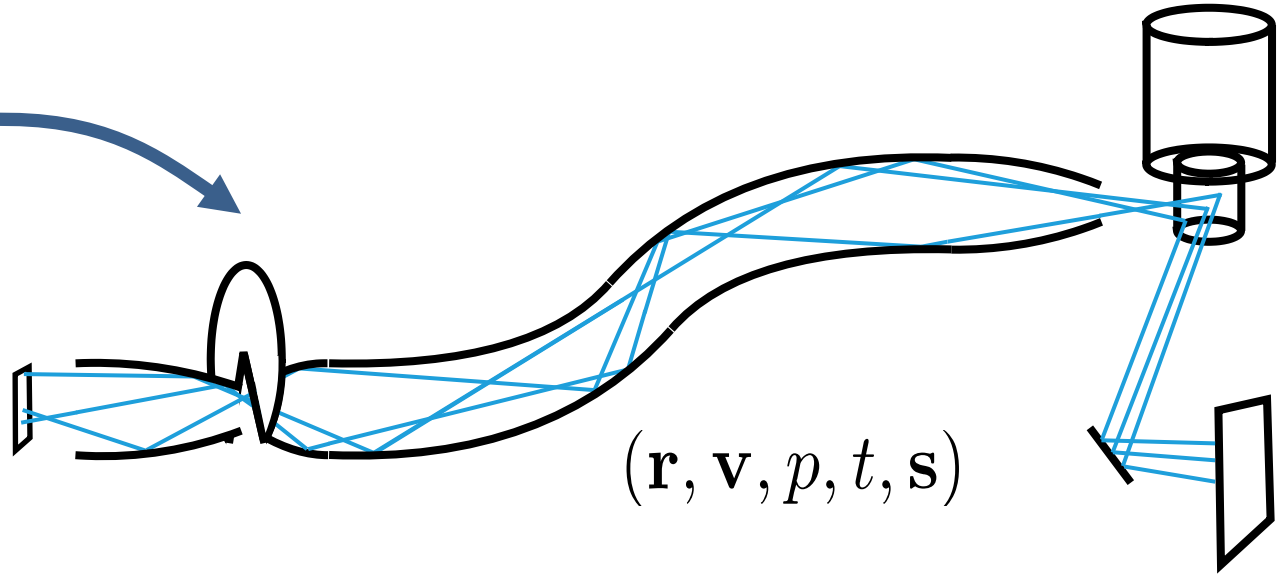


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- McStas is a Monte Carlo neutron raytracing simulation tool
- Used at almost all neutron sources in the world for instrument design
- Written in C, user writes simulation in meta c language
- Selected to be included in PaNOSC harmonized simulation package
- Necessary to make a python API before this can happen, McStasScript

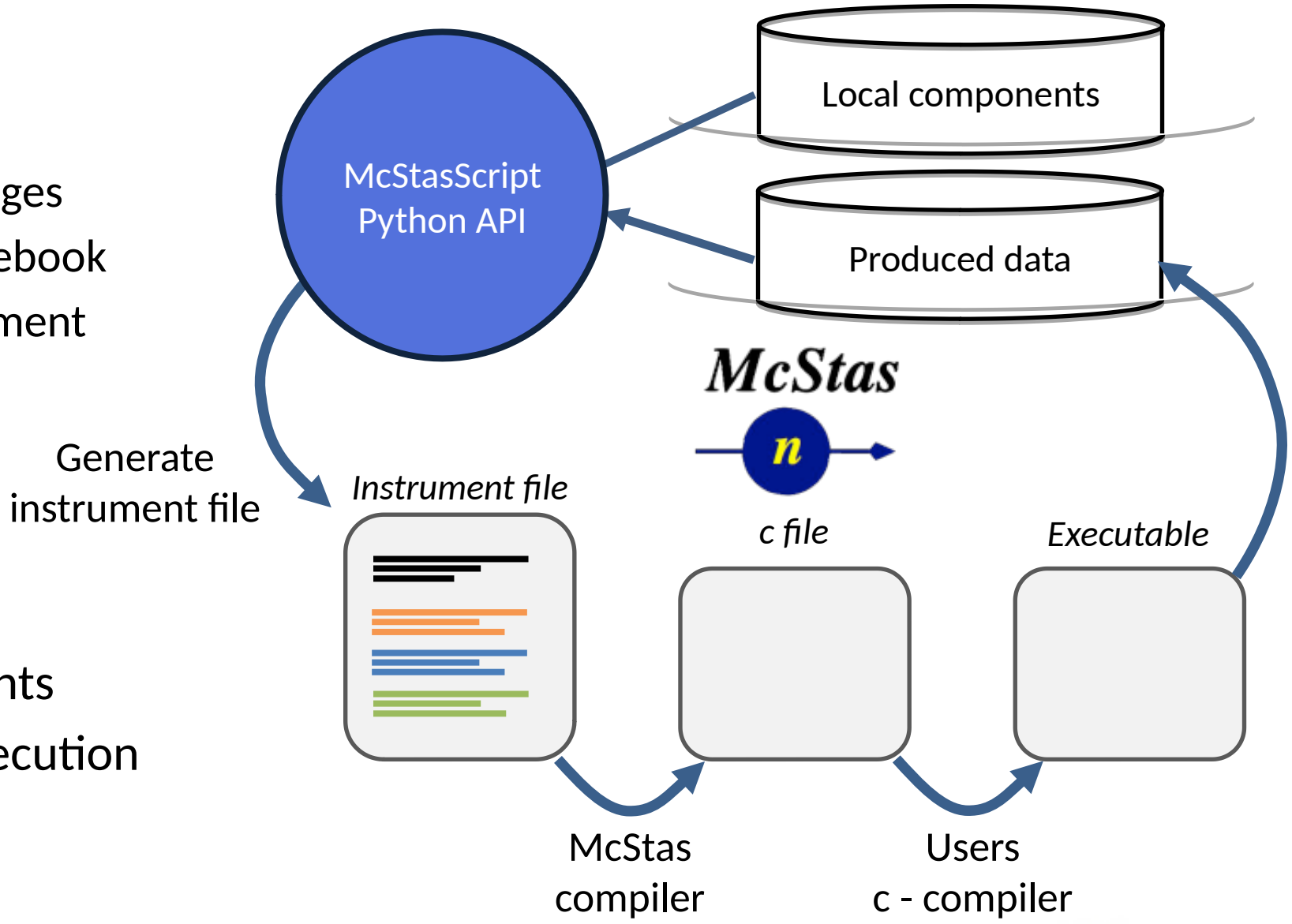
McStas simulation from source to detector including guide, chopper, sample environment and sample.



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

McStasScript

- Python McStas API
  - Traditional Python advantages
  - Run McStas in Jupyter Notebook
  - More choice in code placement



- Reads the local components
- Handles compilation / execution
- Handles import of data

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

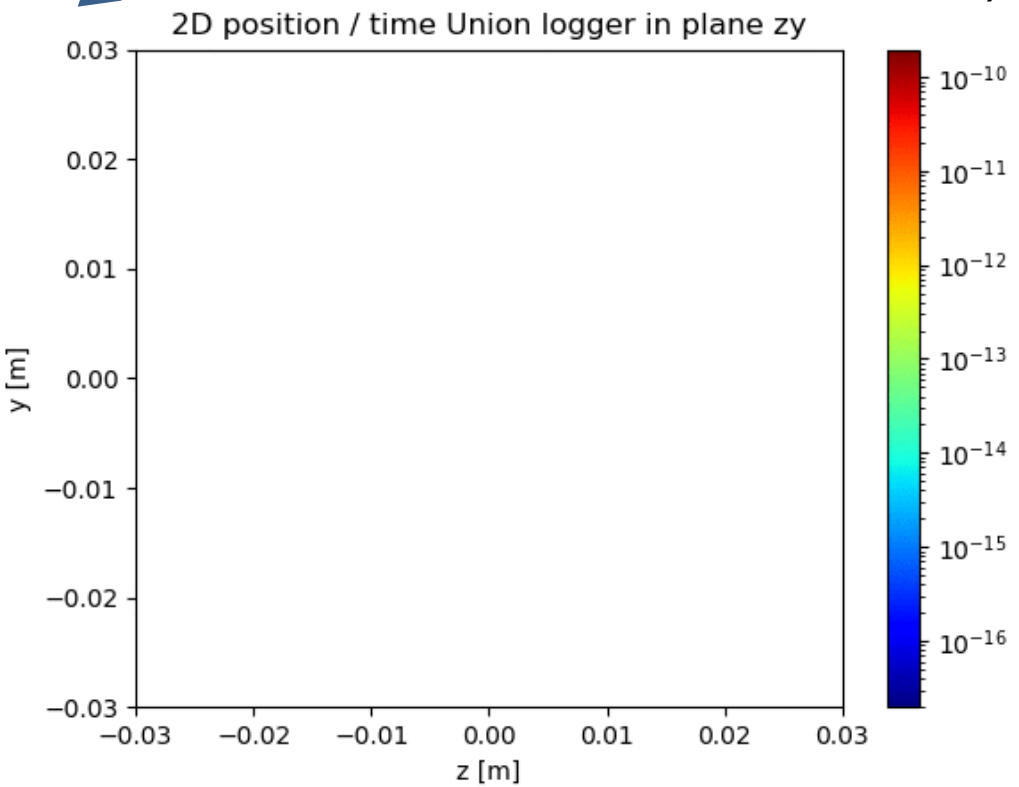
McStasScript

- Used at experiment @ SNS, Tennessee
  - Simulated diamond anvil and sample
  - Improved flexibility with python interface
  - Faster to implement changes in experimental setup
  - Data still being analyzed

Diamond anvil

McStas simulation performed with McStasScriot Diamond anvil

Scattered intensity



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

Materials Simulations through common platform for PaNOSC

Mousumi Upadhyay Kahaly, Computational Materials Science, Theory & Simulation, ELI-ALPS, Szeged, Hungary

WP5 ViNYL: Top level components

- SIMEX
- MCSTAS/ Mcstas script
- ASE
- OASYS

Current involvement from ELI-ALPS

Work plan for WP5 – part 1:

- ✓ Running materials simulations through a common platform, for users
- ✓ Convert the relaxed structure into format required by the subsequent simulation step (either neutron scattering or x-ray absorption)
- ✓ Deposit the final relaxed structure and on the NOMAD database (make an account for ViNYL)

Highlights of achievements so far: https://github.com/PaNOSC-ViNYL/workshop2020/blob/team2/demo/team2/ase.ipynb

- Running a structural relaxation, energy/force minimization with a DFT or TDDFT calculator within a common Atomistic simulation environment (ASE).
- Devise a workflow where data is split up into metadata (including the input parameters), raw data (sim. output), and reduced/analyzed data.
- Feeding DFT output structures to MCSTAS and simulating signals.
- Defining and implementing python classes which allows us to describe different stages of the interaction as well defined objects.
- A Jupyter notebook is prepared that represents an example of the possible work-flow : structural relaxation → photon-matter or neutron-matter interaction → simulation of signal.

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

https://github.com/PaNOSC-ViNYL/workshop2020/blob/team2/demo/team2/ase.ipynb

1900 lines (1900 sloc) | 149 KB

Authors:

- Mads Bertelsen (ESS)
- Mousumi Upadhyay Kahaly (ELI-ALPS)
- Shervin Nourbakhsh (ILL)

ASE input files

ASE can take in input several file formats. In this demo we will check that the conversio
Quantum-Espresso (QE) simulation can be carried on with them.

In the following only CIF files will be considered as inputs for the simulation workflow

Convert CIF to QE input file

ASE is able to convert from different formats. If you plan to run QE as a standalone pac
so you need to convert for example a CIF file into QE format. This can be done in th
`myfile.pwi`

If you run the simulation using ASE, this step is not needed since conversions are done i

Working directories setup and download of input files

In [1]:
```
import sys
import os
```

you need this if you have not installed Quantum-Espresso, but just compiled in local set here the PATH to the QE binaries, otherwise comment the following two lines

In [2]:
```
QE_bin_path = os.environ["HOME"]+"/PANOSC/bin"
os.environ['PATH']=os.environ['PATH']+":"+QE_bin_path

mcstas_outdir = "mcstas_output"
os.environ['PATH']=os.environ['PATH']+":/usr/lib64/mpich/bin:"
```

set here the path for your temporary files

In [4]:
```
tmpdir='/tmp/jupiter/'
print('Create temporary directory: '+tmpdir)
os.makedirs(tmpdir,exist_ok=True)
os.chdir(tmpdir)
os.makedirs(mcstas_outdir,exist_ok=True)
```

```
Create temporary directory: /tmp/jupiter/
```

Download here one CIF file

In [5]:
```
CIF_file = '1527603.cif'
print('Downloading CIF file '+CIF_file+' from crystallography.net')
os.system("wget -c https://www.crystallography.net/cod/"+CIF_file)
```

```
Downloading CIF file 1527603.cif from crystallography.net
```

Out[5]: 0

Download the pseudo potential for Nitrogen:

In [6]:
```
pseudopotfile = 'N.pbe-n-kjpaw_psl.1.0.0.UPF'
pseudo_dir = tmpdir+"/pseudo/"
os.makedirs(pseudo_dir,exist_ok=True)
os.chdir(pseudo_dir)
os.system("wget -c https://www.quantum-espresso.org/upf_files/"+pseudopotfile)
os.system("wget -c https://raw.githubusercontent.com/PaNOSC-ViNYL/workshop2020/team2/demo/team2/
e-n-radius_5.UPF")
os.chdir(tmpdir)
pseudopotfile = 'N.pbe-n-radius_5.UPF'
```

Setup the simulation

Atom from CIF file

In [8]:
```
from ase import io, Atom, Atoms
atomCIF = io.read(CIF_file)

print(atomCIF)
print(atomCIF.get_positions())
```

Importing relevant tools

In [9]:
```
from ase.build import bulk
from ase.calculators.espresso import Espresso
from ase.constraints import UnitCellFilter
from ase.optimize import LBFGS

pseudopotentials={'N': pseudopotfile}
```

Goes on...

DFT output structures fed into MCSTAS and simulating signals

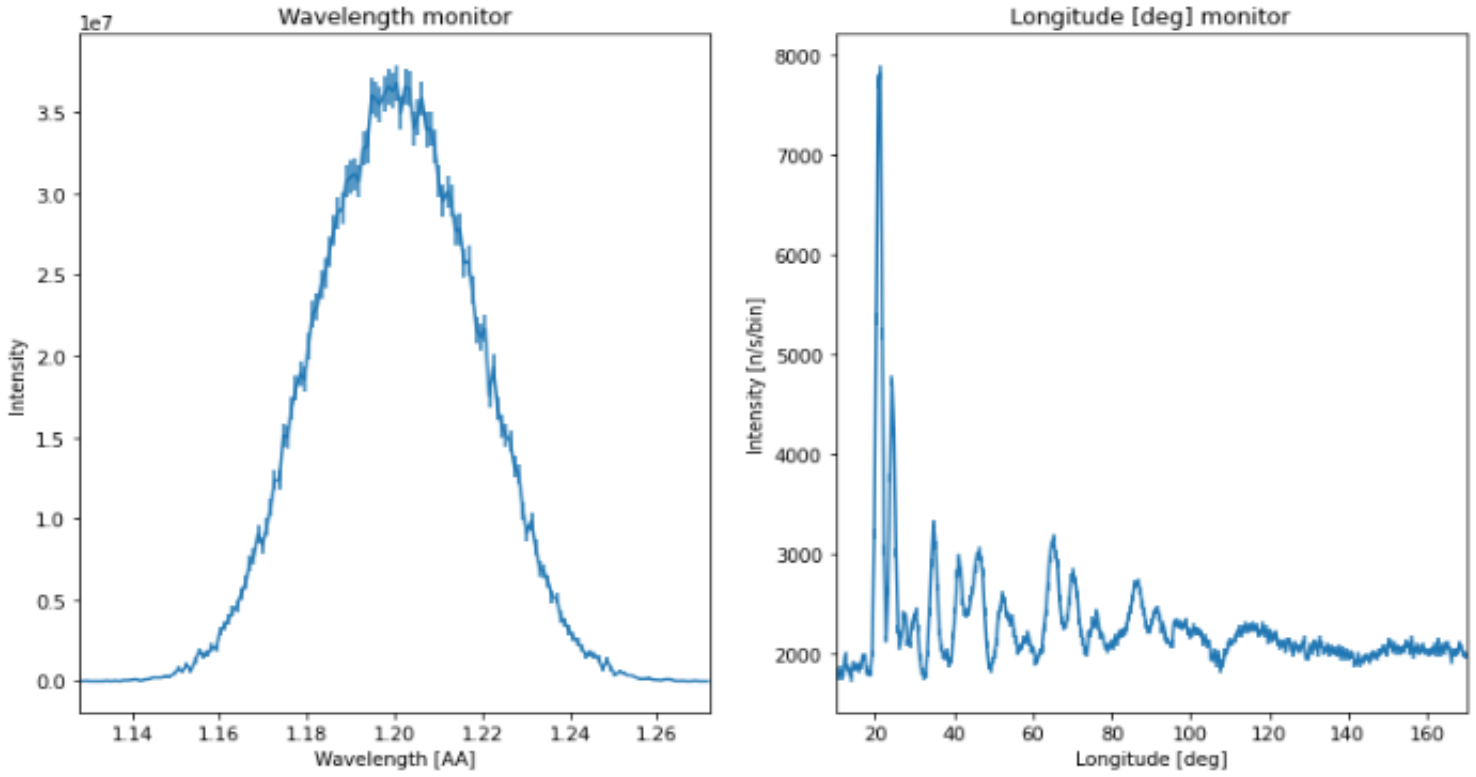
```
In [31]: data = Instr.run_full_instrument(ncount=5E6, mpi=4,
                                          foldername=mcstas_outdir, increment_folder_name=True)

         #plotter.make_sub_plot(data)


         # Instr.show_instrument() Uncomment to view instrum
```

```
In [32]: plotter.make_sub_plot(data)

         number of elements in data list = 2
         Plotting data with name L_mon
         Plotting data with name monitor
```




```
Out[32]: <mcstasscript.interface.plotter.make_sub_plot at 0x7f96e7f5c5f8>
```

See the demo version in:

https://github.com/PaNOSC-ViNYL/workshop2020/blob/team2/demo/team2/ase.ipynb

Laser-driven high repetition rate neutron sources at ELI-ALPS

Zsolt Lécz, Plasma Physics Group, ELI-ALPS, Szeged, Hungary

The plan for WP5 – part 2:

- production of neutrons with energy higher than ~MeV.
  The production rate should be >10^12/s
- these neutrons will be used for transmutation: nuclear reaction in order to reduce the lifetime of nuclear waste and to make them less radioactive.
- for producing high energy neutrons we need energetic ions first. One possible reaction candidate: D+D = He + n(2.45 MeV)
- high repetition rate lasers are needed to achieve the desired production rate

First steps:

- Study experimentally and theoretically the ion acceleration with the SYLOS laser shooting on micrometer thin aluminum foils
- Calculate the approximated neutron number using the typical proton energy distribution (assuming that they are deuteron)
- currently we estimate 10^10 neutrons/ second

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

panosc

Highlights of achievements:

- Implementation of a simple Monte-Carlo algorithm for generating the neutron distribution which is created from D+D reactions
- Implementation of Python classes which allows us to describe different stages of the interaction as well defined objects
- A Jupyter notebook represents an example of the possible work-flow : ion acceleration → ion-matter interaction → neutron data

https://github.com/PaNOSC-ViNYL/neutrontools/blob/master/Test2.ipynb

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

Jupyter notebook presenting the example workflow:

In [7]:
```
from SimEx.Parameters.IonMatterInteractorParameters import IonMatterInteractorParameters
from SimEx.Calculators.TNSAIonMatterInteractor import TNSAIonMatterInteractor
```

Define the "parameters" object: List of parameters: energy_bin, neutron_weight, ibeam_radius, target_length, target_density, xsec_file, ion_name

In [8]:
```
myparams = IonMatterInteractorParameters(ion_name='proton', neutron_weight=1.e4)
myparams.xsec_file = 'D_D_-_3He_n.txt'
```

Create the object doing the job:

In [9]:
```
mysource = TNSAIonMatterInteractor(parameters=myparams, input_path='Data/0010.sdf',
                                   output_path='Data/NeutronData.h5')
```

Another way to do it:

In [2]:
```
paramsdict={'target_density':2.e28, 'ion_name':'proton', 'xsec_file':'D_D_-_3He_n.txt'}
```

In [3]:
```
mysource = TNSAIonMatterInteractor(parameters=paramsdict, input_path='Data/0010.sdf',
                                   output_path='Data/NeutronData.h5')
```

Run the "engine":

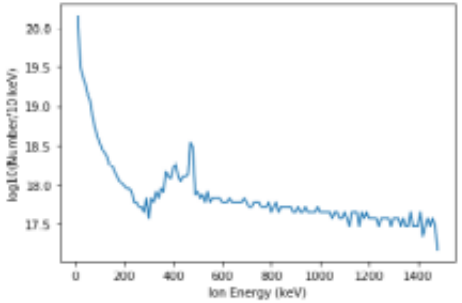
In [10]:
```
mysource.backengine()
```

```
Number of energy bins: 148
Number of neutron macroparticles: 1001
```

In [14]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

In [15]:
```
fig = plt.figure()
ax = plt.axes()

ax.plot(np.divide(mysource.binedges, 1000), np.log10(mysource.counts))
plt.xlabel("Ion Energy (keV)")
plt.ylabel("log10(Number/10 keV)")
plt.show()
```

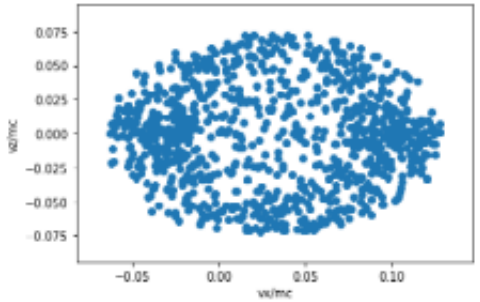


Ion energy spectrum

Check the result by plotting the Neutron distribution in velocity space:

In [16]:
```
fig1 = plt.figure()
ax1 = plt.axes()

ax1.scatter(mysource.data[3]/3.e8, mysource.data[5]/3.e8)
plt.xlabel("vx/mc")
plt.ylabel("vz/mc")
plt.show()
```



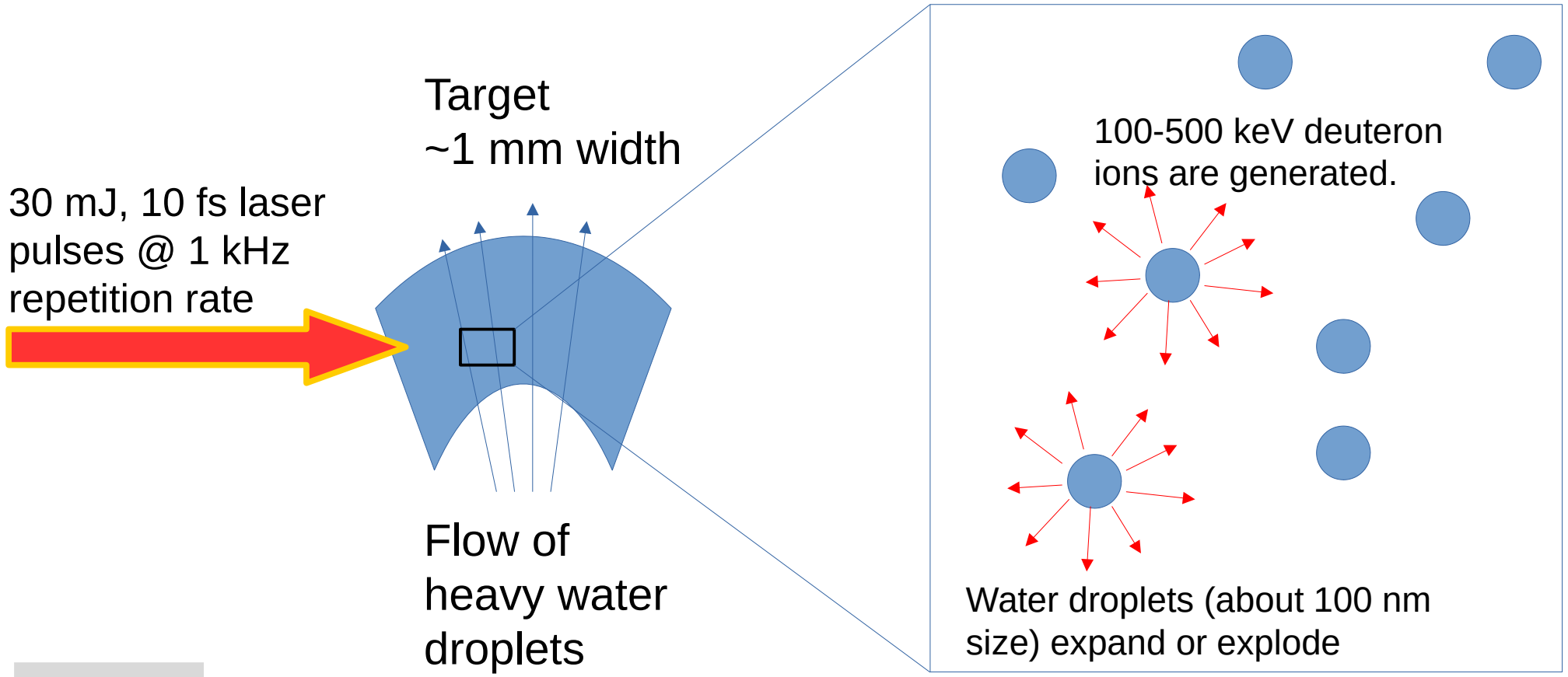
Generated neutrons in velocity space

In [8]:
```
mysource.saveH5()
```

The code uses the ion spectra obtained from PIC simulations and generates the neutron data, which is saved in hdf format.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under g

Next step: make it compatible with high rep-rate lasers using spray targets



Reaction: D(1,2)+D(1,2)=He(2,3)+n

Outlook:

- implementation of a kinetic approach in the base algorithm of the code, which takes into account the ion momentum distribution and their spatial distribution.
- incorporate it in PanOSC-Vynil

Thank you

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