PaNOSC

Photon and Neutron Open Science Cloud

H2020-INFRAEOSC-04-2018

Grant Agreement Number: 823852

D5.4: VINYL software tested, documented, and released, including interactive simulation and analysis workflow

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

Project Deliverable Information Sheet

Project Reference No.	823852
Project acronym:	PaNOSC
Project full name:	Photon and Neutron Open Science Cloud
H2020 Call:	INFRAEOSC-04-2018
Project Coordinator:	Andy Götz (andy.gotz@esrf.fr)
Coordinating Organization:	ESRF
Project Website:	www.panosc.eu
Deliverable No:	D5.4: VINYL software tested, documented, and released, including interactive simulation and analysis workflow
Deliverable Type:	Report
Dissemination Level:	Public
Contractual Delivery Date:	30/11/2022
Actual Delivery Date:	09/12/2022
EC project Officer:	Flavius Alexandru Pana

Document Control Sheet

Document	Title: D5.4: VINYL software tested, documented, and released, including interactive simulation and analysis workflow
	Version: 1
	Available at: https://github.com/panosc-eu/panosc
	Files: 1
Authorship	Written by: Carsten Fortmann-Grote
	Contributors: Mads Bertelsen, Stella d'Ambrumenil, Juncheng E, Aljoša Hafner, Gergely Norbert Nagy, Shervin Nourbakhsh, Mousumi Upadhyay Kahaly
	Reviewed by: Jordi Bodera Sempere, Andy Götz
	Approved: Andy Götz

List of participants

Participant No.	Participant organisation name	Country
1	European Synchrotron Radiation Facility (ESRF)	France
2	Institut Laue-Langevin (ILL)	France
3	European XFEL (XFEL.EU)	Germany
4	The European Spallation Source (ESS)	Sweden
5	ELI European Research Infrastructure Consortium (ELI-ERIC)	Belgium
6	Central European Research Infrastructure Consortium (CERIC-ERIC)	Italy
7	EGI Foundation (EGI.eu)	The Netherlands

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

Contents

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

1 Abstract

We report on the Deliverable D5.4 in work package 5 (Virtual Neutron and X–ray Laboratory – ViNYL) in the European Union Horizon 2020 project *Photon and Neutron Open Science Cloud – PaNOSC*. This report details the latest releases of software developed under the umbrella of ViNYL including test suites, documentation. In addition, we report on a use case where ViNYL software is employed in an iterative data analysis workflow in serial femtosecond crystallography. This is the final deliverable.

2 Introduction

The main purposes of the PaNOSC work package 5 are to harmonize the user interface for simulations of X-ray and neutron beamlines and to enhance the interoperability of simulation codes using community defined data formats and metadata standards. With the latest releases of simulation code projects and simulation infrastructure libraries, these ambitious goals have now been accomplished. The present report aims at presenting a comprehensive overview on the various software solutions and gives references and pointers to the respective resources for source code, documentation and testing.

The appendix contains a table (Tab. 6) listing all released versions of ViNYL software packages and their DOIs.

3 libpyvinyl

3.1 Scope and purpose

libpyvinyl, the python APIs for Virtual Neutron and x-raY Laboratory, is a python package providing a way to harmonize the user interfaces of neutron and X-ray simulation codes by defining essential base and auxiliary classes for developers of a start-to-end simulation platform.

3.2 Structure of the library

The structure of libpyvinyl is explained in detail in Deliverable D5.3 and in the documentation (see below in Sec 3.4). We reiterate the main concepts here:

3.2.1 Main classes

BaseCalculator the interface to set parameters and perform calculation.

BaseData the representation of the input/output data of a Calculator class.

BaseFormat the interface to exchange data between the memory and the file on the disk in a specific format.

3.2.2 Auxiliary classes

Parameter the Parameter class describes a single parameter with its limits (valid/invalid range) and unit (if applicable) defined.

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

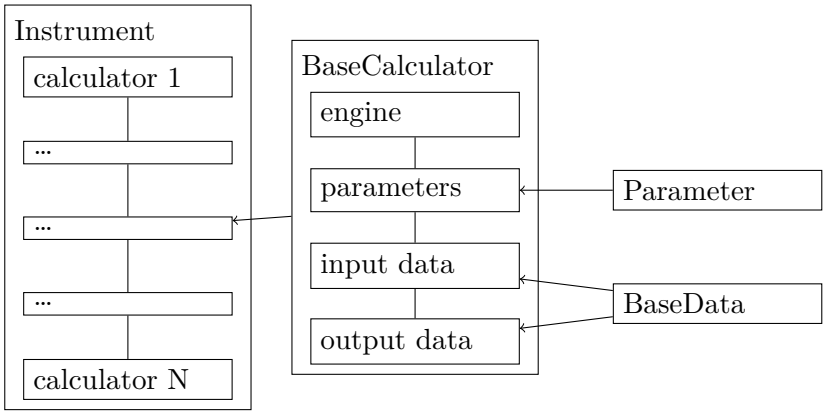


Figure 1: Simplified diagram showing the relations between the main libpyvinyl classes

Calculator Parameters a container for holding all the Parameter objects that pertain to a single Calculator.

DataCollection a thin layer interface between the Calculator and DataClass. It aggregates the input and output into a single variable, respectively.

Instrument a container for integrating different Calculator objects into a complete start-to-end simulation workflow.

Figure 1 illustrates the relations among the base classes in libpyvinyl.

The Parameter class describes a single parameter intended to be an input for a calculator with the following characteristics:

- no a priori for the type
- usually a physical quantity that can be expressed by a floating point number or a string
- units can be defined and unit conversions taken into account automagically to provide to the calculator the value in the units it expects

The following listing demonstrates the parameter handling in libpyvinyl:

```
energy_parameter = Parameter(name="energy", unit="meV", comment="Energy of emitted particles")
energy_parameter.value = 5.0

```

We defined the default unit to be meV, but since the object is aware of the unit, it is possible to provide in another energy unit using Pint.

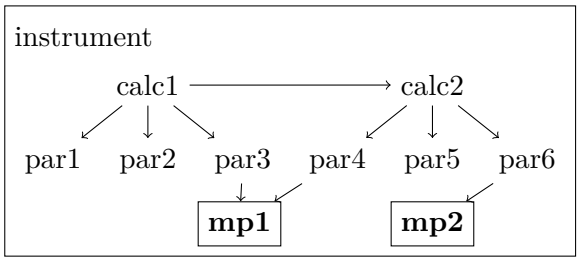
```
import pint
ureg = pint.UnitRegistry()
energy_parameter.value = 5.0*ureg.eV

```


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

3.3 Parameters common to multiple calculators

If multiple calculators share common parameters, they need to be changed together when running the simulation:



- Example: master is the neutron energy, calc1 is source and calc2 is a monochromator
- A single parameter promoted as "master" to distinguish parameters to be changed by the "end user" or for setting up the workflow.

```
myinstr.add_master_parameter("energy", {"calc1": "par3", "calc2":"par4"}, unit="GeV")
myinstr.master["energy"] = 4.5 * ureg.keV
```

3.4 Repository

The following table lists the relevant public resources for libpyvinyl's source code, documentation site, testing status and use case.

Table 1: Relevant libpyvinyl repositories.

Source code repository	https://github.com/PaNOSC-ViNYL/libpyvinyl
Release snapshots	https://github.com/PaNOSC-ViNYL/libpyvinyl/releases
Documentation	https://libpyvinyl.readthedocs.io
Test status	https://github.com/PaNOSC-ViNYL/libpyvinyl/actions/workflows/ci.yml
Usage illustration	https://github.com/PaNOSC-ViNYL/libpyvinyl/tree/master/tests/integration/plusminus

The current release of libpyvinyl is version 1.1.2. Installation proceeds either from the cloned repository, from the downloaded release tarball, or via the python package index as

```
pip install libpyvinyl
```

4 Instrument database

4.1 Scope and purpose

Facility users have little access to complete and up–to–date instrument descriptions to run sample simulations and they are difficult to manipulate for a user not expert in the specific simulation software. The libpyvinyl API offers the high level interface to harmonize the interaction with an instrument

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

described with different simulation software packages. The instrument database has been designed to collect in a central place instrument descriptions created with software adopting the libpyvinyl API. Users are then able to access the instrument with few lines of code and run the simulation with a set of implemented samples (expandable). The instrument comes with the set of high level parameters defining the settings of the instrument that the user is supposed to be able to manipulate at the facility for the data acquisition, while all internal complexities are hidden.

4.2 Current status of the project and contributions

The structure of the database have been defined and a dedicated API is available to list its content and access the required instrument description. Only few pilot instruments are described and are supposed to be used as examples for implementing new instruments. A complete validation of the instruments has not been performed.

The instrument database and its API designs have been led by ILL and co-designed with ESS and EuXFEL.

It is envisioned that several ESS instruments and sample environments will be submitted to the database soon, further testing and extending the system as needed. The instrument database addressed task 5.3 at a higher level of ambition than was foreseen at the start of the project.

On the ViNYL platform, in the API layer, SIMEX was used as the main simulation framework, which could efficiently mimic X-ray source based beamline (like XFEL), however it requires additional functionalities and incorporation of certain features to mimic IR laser source, which is widely useful to probe ultrafast dynamics in materials (as in ELI-ALPS). ELI colleagues partly addressed this problem, by testing and incorporating time-dependent density functional theory based tools within ASE (Atomistic Simulation Environment) to describe ultrafast temporal modulations in laser fields and its interaction with materials of different dimensionalities. Associated Jupyter notebooks are already uploaded in pan-learning platform, and were used for in PaNOSC summer school.

Additionally, ELI-ALPS colleagues implemented strategies for simulation of scattering intensities by calculating the form factor. Using the ASE platform for structure optimization of chosen structures, we obtained the modified geometry and resulting electronic charge density distribution which provides form factor. The output scattered field amplitude is thus modulated by charge, mass, and geometric factors, and reflects on the structure form factor due to the interaction of the molecules with other external factors (like incident electric field, or other neighbouring molecules/medium). A class with a streamlined front-end interface is implemented for reading and post-processing the charge density data, ultimately calculating the form factor and some other useful data, such as the integrated charge density. The class is also able to save the charge density in VTK format, so later it can be visualized with suitable software like ParaView.

4.3 Repository

The database is defined as a git repository hosted on the WP5 github organization as well as the API to access the instrument database and make instrument descriptions available.

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

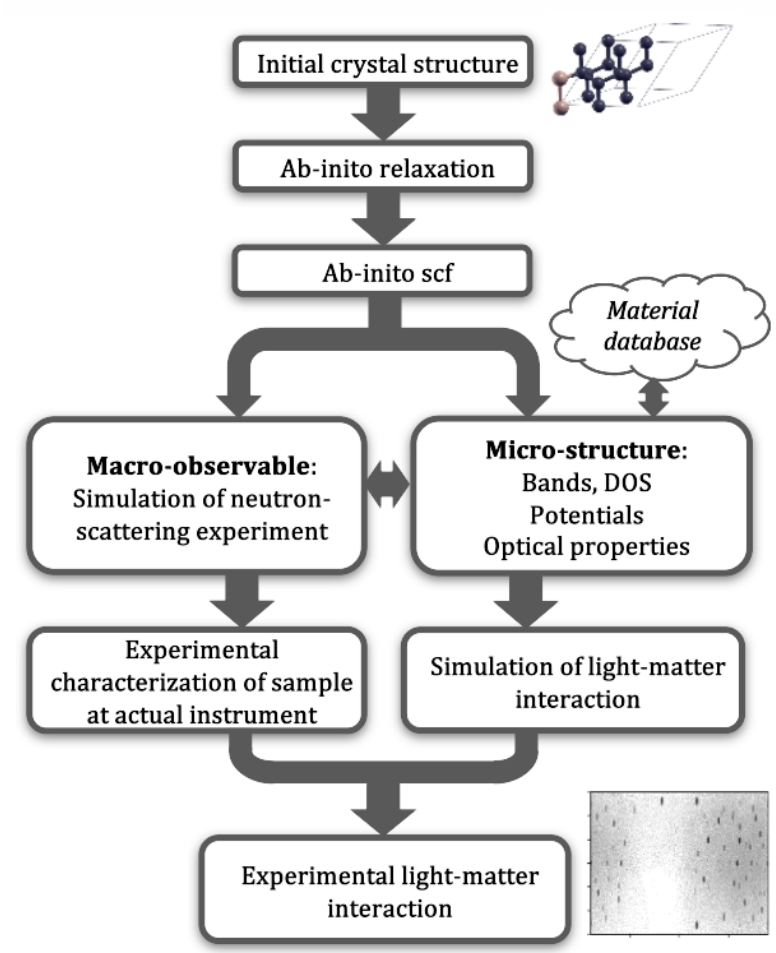


Figure 2: A visual representation of PaNOSC use case 18: In-silico Neutron Diffraction: precise tool for exact structural analysis and defect detection

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

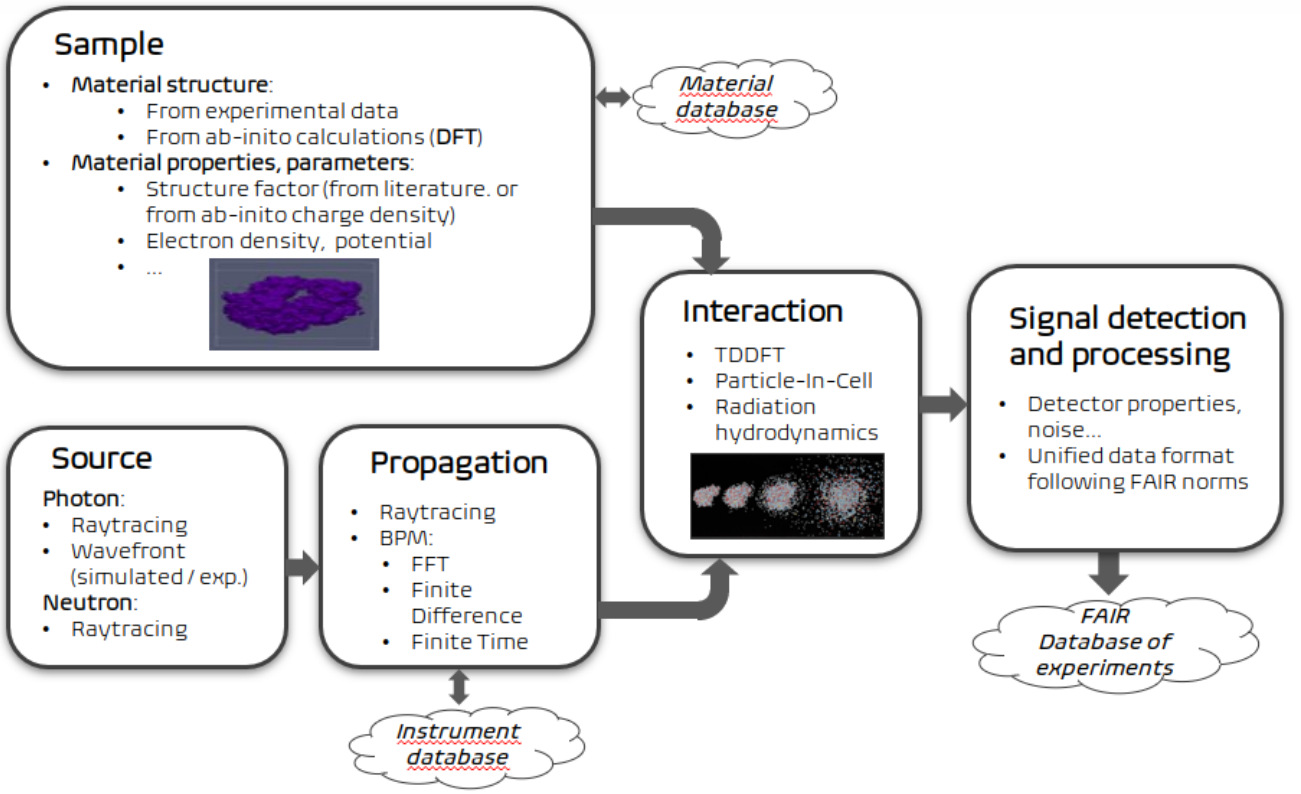


Figure 3: General workflow of the ViNYL simulation environment.

https://github.com/PaNOSC-ViNYL/instrument_database

https://github.com/PaNOSC-ViNYL/instrument_database_API

4.4 Structure

4.5 Installation and access

The repository can be cloned locally using git commands:

```
git clone  --recurse-submodules https://github.com/PaNOSC-ViNYL/instrument_database.git
```

4.6 Documentation

The documentation is self contained in the repository as Markdown file. The documentation targets both users of the instrument database and developers implementing instrument descriptions.

4.7 Dissemination

It is crucial to have a significant contribution from the research facilities by providing accurate and complete instrument descriptions and to maintain them. A workshop dedicated to WP5 tools and prospects would be ideal to reach both expert and non-expert simulation users as well as developers.

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

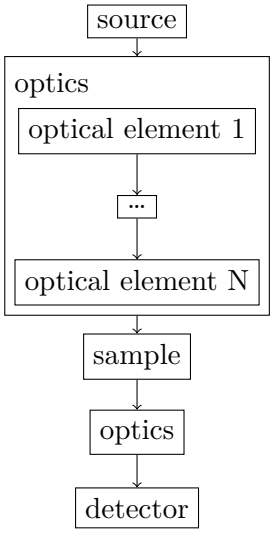


Figure 4: A simplified description of an instrument and its elements

4.8 User base

Currently limited but potentially every user submitting a proposal for beam time.

The current version of the repository is v0.1.0 and contains two fully implemented instruments:

- SPB-SFX for EuXFEL (using SimEx-lite)
- ThALES for ILL (using McStas)

For ILL's implementation a new instrument base class (McStasInstrumentBase) has been implemented inheriting from libpyvinyl Instrument class. McStasInstrumentBase offers convenient methods to desribe the instrument in a multi-calculator way, taking care of the input-output bridging between calculators. It is also a helper class describing samples and sample environments with their parameters with the related methods to select one among those implemented.

The most relevant methods are:

- `.set_sample_by_name(string)`: replacing the current sample with the one identified by the argument. Parameters are automatically added to the instrument.
- `.set_sample_environment_by_name(string)`: replacing the current sample environment by the one identified by the argument
- `.sample`: returning McStas component object representing the sample
- `.force_compile(boolean)`: switching the McStasScript flag on all the calculators
- `.sample_<box,cylinder,sphere>_shape(args)`: changing the shape and the value of the paramters of the sample
- `.sim_neutrons(integer)`: changing the McStasScript setting on all the calculators
- `set_seed(integer)`: changing the McStasscript setting on all the calculators

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

4.9 Instrument repository API

First, the API is initialized, which provides access to the instruments represented in the instrument repository.

```
from instrumentdatabaseapi import instrumentdatabaseapi as API
repo = API.Repository()
repo.init()
```

From this point onwards, users interact with libpyvinyl objects. In particular, they can load the instrument and units from a hierarchical tree where the different levels represent the Institute, the Instrument name, the Version (HEAD for the current or date of last day of validity YYYY-MM-DD) and the Flavour (e.g. detailed vs simplified) as demonstrated in the following listing:

```
myinstrument = repo.load("ILL","ThALES" , "HEAD", "mcstas", "full", dep=False)
import pint
ureg = pint.get_application_registry()
```

Simulation parameters are set to their values in the usual manner as defined by the libpyvinyl base class methods:

```
myinstrument.set_instrument_base_dir("/tmp/ThALES_scan/")
myinstrument.sim_neutrons(500000)
myinstrument.set_seed(654321)
```

The same goes for physical and instrument parameters:

```
myinstrument.master["a2"] = myinstrument.energy_to_angle(4.98 * ureg.meV)
myinstrument.master["a4"] = 60 * ureg.degree
```

as well as for sample parameters:

```
myinstrument.set_sample_by_name("vanadium")
myinstrument.sample_cylinder_shape(0.005, 0.01)
```

Finally, to run the simulation, again, the libpyvinyl method is used

```
np = (21 - 1) / 2
dEI = 0.05
import numpy
myinstrument.force_compile(false)
outputs = []
for energy in numpy.arange(4.98 - np * dEI, 4.98 + np * dEI, dEI):
  myinstrument.master["a6"] = myinstrument.energy_to_angle(energy * ureg.meV)
  myinstrument.run()
  outputs.append(myinstrument.output)
```

These listings demonstrate in a concise way, how the various parts in ViNYL are now integrated into a seamless simulation environment: Instrument parameters and settings that correspond to their actual physical counterparts at the RI are loaded from a database as a single source of truth, these parameters are then injected into the simulation setup and the actual simulation backengine is launched

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

all from within a common programming environment, e.g. a python script, interactive session or jupyter notebook.

We now continue with the summary of achievements by going through the individual simulation libraries for neutron and photon science.

5 McStasScript

For neutron scattering, the McStas package is the standard tool for instrument simulations, yet it has a specialized user interface built on the C programming language. McStas has a sister project called McXtrace that simulates X-ray instrumentation. The McStasScript Python API that allows nearly full access to McStas/*McXtrace* features through an elegant Python interface was mainly developed at the European Spallation Source (ESS).

5.1 Scope and purpose of the package

The McStasScript package aims to be an alternative user interface that covers the vast majority of McStas features. McStas itself is a Monte Carlo ray-tracing simulation code that covers neutron beamlines from source to detector, including sample scattering. Since the same software is used for the entire beamline, access to McStas from Python covers task 5.2 to 5.4, as it provides source simulation, beamline simulation and sample physics. Before McStasScript, McStas offered two ways of working with instrument simulations. Central to both are the instrument file that describes the instrument to be simulated in a C meta language. In this file the instrument is built from components and it is possible to have input parameters along with calculations. It is possible to run a simulation by using the command line tools to execute such a simulation by providing the instrument file, and then plot the resulting data using a different command line tool. McStas also provides a GUI with a text editor that provides some help features when inserting components, and then dialog boxes for running simulations and viewing the data. The data is always written to disk, usually as text files with metadata in a specialized format.

McStasScript thus provides a third way of writing McStas instruments, where everything can be done from the Python API. One defines an instrument object, adds components, parameters and calculations as necessary, and then runs the simulation from within the Python environment. The data is returned as objects with metadata and the numerical data is available as standard numpy arrays, making it easy to work with the generated data.

McStasScript was developed in parallel with the simulation base package libpyvinyl in order to ensure a harmonization of the different simulation package APIs. libpyvinyl provides base classes from which one can build simulation packages, and these simulations packages will thus follow a similar logic and share the most important method names. The libpyvinyl package is in this way the answer to the harmonization part of task 5.1, and McStasScript is built upon this base package.

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

5.1.1 Repository

The project is open source and available on the WP5 github repository. There are two repositories, one for the main package and a separate one for example notebooks, this makes it easier for users to get examples in a location more easily accessible than the install location. Links to the repositories are shown in table 2. The McStasScript-notebook repository provides simulation examples in the form of jupyter notebooks.

Name	Link
McStasScript	https://github.com/PaNOSC-ViNYL/McStasScript
McStasScript-notebooks	https://github.com/PaNOSC-ViNYL/McStasScript-notebooks

Table 2: Links to McStasScript repositories.

5.1.2 Tests

The package has good coverage with unit tests and these are used in continuous integration (CI). In addition, integration tests are available, as these do Monte Carlo simulations and require a local McStas installation, these are not included in CI.

5.1.3 Installation

It is possible to install the package through the Python package index pip as shown below. The package can also be installed from source by cloning the github repository. Regardless of the method, configuration is necessary to tell McStasScript where the local McStas/*McXtrace* installation is located.

```
pip install McStasScript
```

5.1.4 Documentation

The documentation is available at https://mads-bertelsen.github.io, where significant efforts have been made to provide comprehensive documentation. The documentation includes the purpose of the package, how to use it and tutorials on a large part of the McStas package as well as the McStas Union components. The release of the documentation along with online availability of the package was the ESS contribution to the completion of deliverable 5.2.

5.1.5 Dissemination

It is important to inform the community about the package in order to create a user community. Table 3 contains an overview of events where McStasScript was shown outside of PaNOSC events.

5.1.6 User base

It is difficult to judge the size of the current McStasScript user base as the download statistics include downloads of the package made through continuous integration, bots that mirror the package and so forth. There have been days with more than 300 downloads of the package, but it has not been possible

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

Event	Format
ECNS 2019	Poster
ICANS 2019	Poster and Presentation
SNS visit	Presentation for simulation group
HighNESS McStas course	Presentation
ISIS McStas course	Presentation
MLZ Garching group meeting	Presentation and online demo
PNPI group meeting	Presentation
ESS McStas days	Presentation and quiz exercises
ICNS 2022	Poster and award speech for Instrumentation and Innovation prize
ESS-ILL User meeting 2022	Poster

Table 3: Overview of McStasScript dissemination activities.

to find a total number of downloads across all versions. Users provide issues to the github project, pull requests are being submitted and support requests are regularly received by the author over e-mail. It can be concluded that the package is reaching some level of adoption within the McStas/*McXtrace* community. Known users are located at ILL, SNS, KU, DTU, MLZ, TUDelft where it has even been used to design X-ray optics for space based telescope.

5.2 McStasScript features

This section aims to provide an overview of the feature set of McStasScript.

5.2.1 Write instrument

An instrument is made by creating an instrument object, and this instrument object can then have components added. These are a sequence of smaller codes that describe the beamline, and are each represented by a file on the users disk. McStasScript parses these files to internalize information on their parameters and units. It is possible to add components to any point in the sequence of components, but it is most common to place new components at the end. The instrument object inherits from the libpyvinyl calculator class in order to adhere to the standardized user interface created by WP5.

5.2.2 Parameters

It is also possible to add parameters to an instrument, these can be changed easily between runs. When changing components, McStas requires the underlying C code to be recompiled, but this can be avoided when only changing parameters. McStasScript has yet to implement code that avoids compiling when unnecessary, this is still the users responsibility.

5.2.3 Error checking

When writing a McStas instrument there are many possible ways to make an error that would result in the generated C code to fail to compile. A number of these are recognized by McStasScript and reported immediately instead of only when the user attempts to run the simulation. The exceptions

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

raised by McStasScript are more specific that the compiler errors generated by McStas. Not all errors are caught, but the compiler output can be seen from McStasScript.

5.2.4 Help features

McStasScript contains a number of help features ranging from simple text overviews of the parameters, variables and components to diagrams of components included in an instrument. The help features can be used to find the right component in the library and to understand the parameters of components even before they are included in the instrument. When components are included in the instrument, colored text is used to show different states of the component parameters, such as default, required and set by user. The generated instrument diagrams such as those seen in figure 5 are great for getting a quick overview of an instrument and are only available in McStasScript.

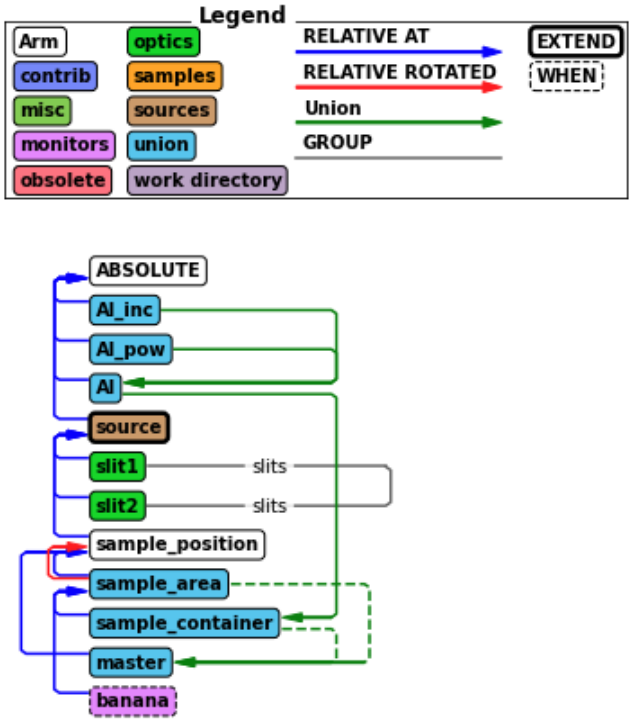


Figure 5: Example of diagram from McStasScript. In widget mode one can hover the mouse over the left side of the boxes to see further info on each component.

5.2.5 Run simulation

The simulation can be executed from the Python environment, this is achieved with a system call using the subprocess module. The user can use the same controls available on the command line tools, such as setting the number of rays, location of data output, enabling gravity and selecting how many CPU cores will be used. The actual run method is called backengine to conform the libpyvinyl standard.

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

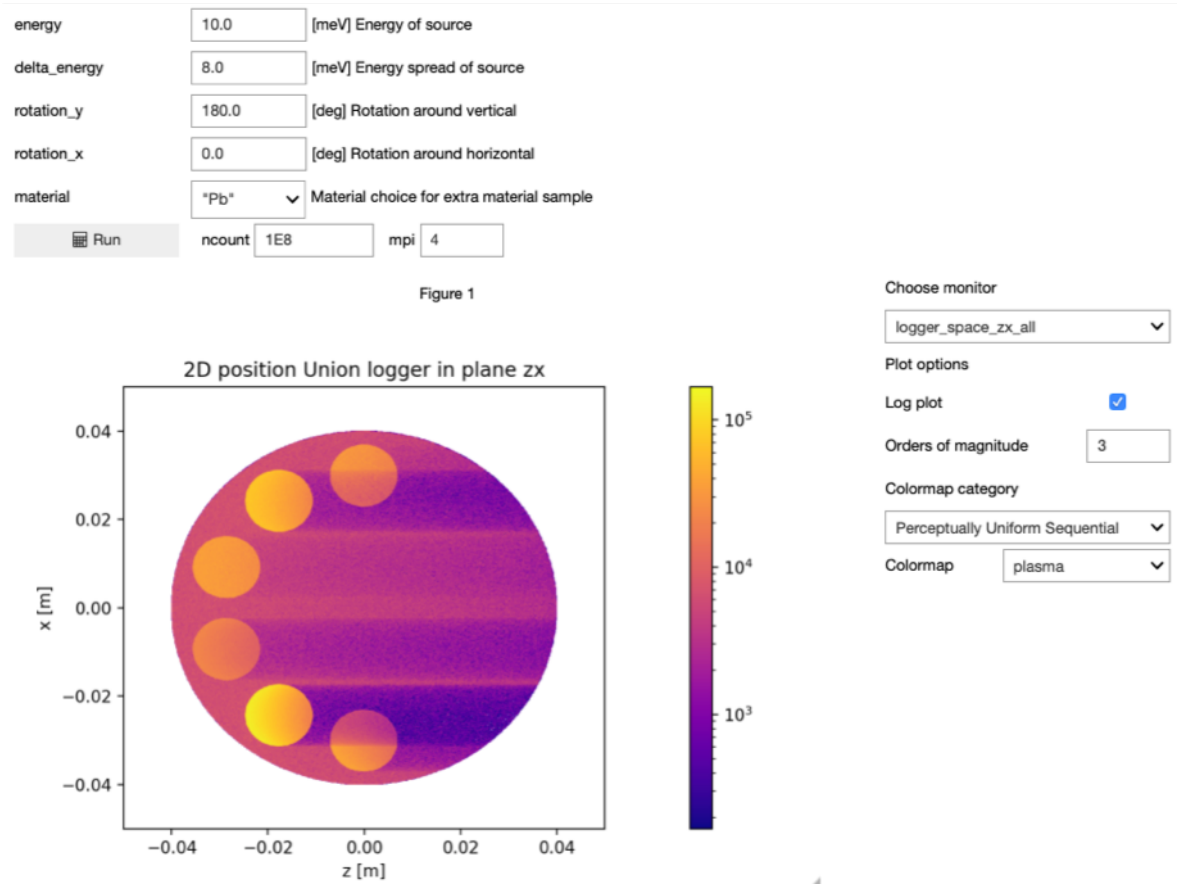


Figure 6: Example of run widget from McStasScript. Here showing scattering in an imaging calibration sample with several embedded materials. Notice the streaks of lower intensity behind materials that scatter / absorb more than the others.

5.2.6 Return of data

Running a simulation returns data as a list of Python objects corresponding to each monitor in the instrument. These contain metadata such as the used parameters and information about the monitor as well as the actual data in nunmpy arrays. The objects also contain preferences on how this object should be plotted so the plotting tools in McStasScript can use appropriate settings when plotting the included data.

5.2.7 Widgets

McStasScript also contains widgets specifically designed for use in Jupyter Notebooks. One widget can take a list of data objects and plots each of them with easy access to plotting options and colorscales in the widget. The other widget takes an instrument object and is able to run the simulation with parameters set by the user, this widget shows the data from the simulation just as the plotting widget would. An example of the run widget can be seen in figure 6.

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

6 openPMD

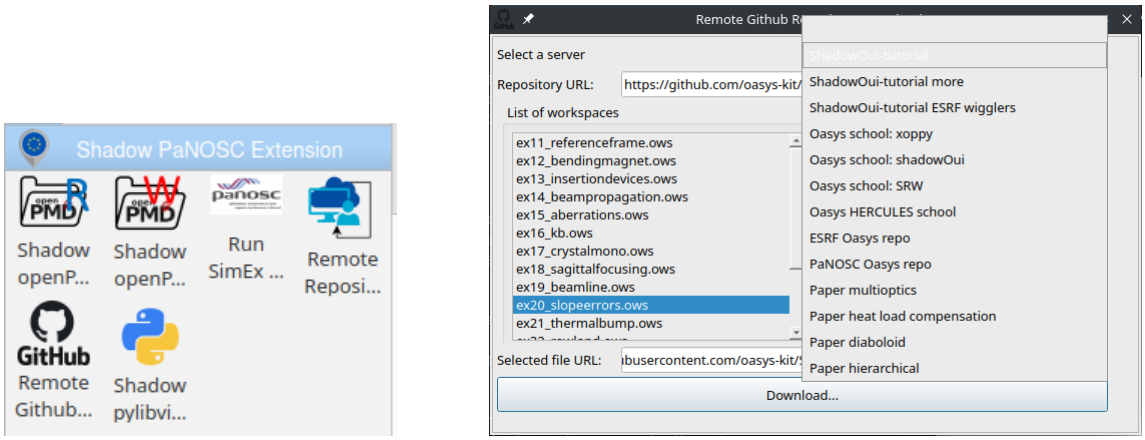
The openPMD https://www.openpmd.org format is to be used for transferring descriptions of beams between different simulations which was an objective of task 5.1 (see also the D5.1 report). Core development of OpenPMD is not performed under the umbrella of WP5 but we have contributed various domain specific extensions to the standard (molecular dynamics, coherent wavefront propagation, x–ray and neutron ray tracing). We mention openpmd here only for the sake of completeness. Further details are given in the D5.1 report.

7 OASYS

7.1 Introduction

OASYS (OrAnge SYnchrotron Suite) is an open source environment for modelling X-ray beamlines and experiments. It is based on the Orange (`https://orangedatamining.com/`) framework which provides an easy-to-use graphical user interface (GUI). Various OASYS widgets (optical elements) are connected to each other, thus creating a workflow (beamline). This is particularly useful for X-ray optics simulations, as the wavefront propagation direction is well-defined and known in advance (from the source to the experimental station).

Several successful calculation packages exist and are actively maintained by the community. Among the most popular are *SHADOW* for ray-tracing simulations and *SRW* for wavefront propagation. During the PaNOSC project, we have addressed several different topics. This chapter provides an overview of all the development. All PaNOSC-specific widgets can be installed through OASYS1-PaNOSC (shown in Fig. 7a package hosted on PyPI and available through the internal add-on manager. All contributions to the OASYS part described below come from CERIC-ERIC and ESRF.



(a) (b)

Figure 7: (a) PaNOSC widget toolbox. (b) Remote repository of OASYS workspaces (OWS) access through a widget in the GUI.

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

Table 4: Relevant OASYS repositories. The current release of OASYS1-PaNOSC is v0.3.2 and of OASYS1-oasyswiser v0.3.22.

OASYS Panosc toolbox code	https://github.com/PaNOSC-ViNYL/OASYS1-PaNOSC
Wiser (Library) repository	https://github.com/oasys-elettra-kit/WISEr
OasysWiser (GUI) repository	https://github.com/oasys-elettra-kit/OASYS1-Wiser
OASYS local Docker container	https://gitlab.elettra.eu/panosc/ceric/oasys-local-docker
OASYS Jupyter Docker container	https://gitlab.elettra.eu/panosc/jupyter-desktop-oasys
Dockerhub release	https://hub.docker.com/r/ceric/panosc-oasys-local
E-learning Docker container	https://github.com/pan-training/Docker/tree/main/wp8-summerschool
PaNOSC OWS database	https://github.com/PaNOSC-ViNYL/Oasys-PaNOSC-Workspaces

7.2 Wiser

Wiser is a new package for wavefront propagation calculations based on Python implementation of WISE calculation code which stemmed from and is still maintained by Elettra Synchrotron. It is targeted at simulating the optical performance of X-ray mirrors. Propagation is done using Huygens-Fresnel integral and allows for simultaneous 2-dimensional calculations of both figure errors (profiles) and roughness (statistical). Several optical elements have been implemented so far: plane mirror, elliptic mirror, spherical mirror, slit, detector, grating.

OASYS implementation consists of several abstraction layers of which the user interacts only with the top-most one - OASYS1-oasyswiser (available for installation on PyPI and through internal OASYS add-on manager). The back-end is a numerically optimized (able to use multi-core CPU processing) stand-alone Python library *Wiser* which then interacts with *WOFRY* package which strives to be a generic high-level abstraction of wavefront propagation workflows in OASYS.

The development was done in three stages, it first focused on numerical optimization of Wiser. Since solving the Huygens-Fresnel integral is very expensive, this was a crucial step in usability and accessibility of the code. A few different optimization methods have been used. Using the *numba* Python library for just-in-time compiling of calculation kernel, a speed-up on the order of 10-100x has been reached. Afterwards, several new optical elements had to be implemented and a testing phase was conducted. Once the numerical consistency had been confirmed, work started on the GUI part. At present, all the elements beside the grating are available and ready-to-use as widgets in OASYS.

The package has been presented at the SPIE 2020 conference [1] and since then used in several applications (see Section 7.7).

7.3 COMSYL

COMSYL (COherent Modes for SYnchrotron Light) is a software package to perform numerically the coherent mode decomposition of undulator radiation in a storage ring [2]. COMSYL requires the use

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

of high-performance computer resources. COMSYL is used for beamline modelling at the ESRF-EBS. Several tasks that directly concern COMSYL have been carried out in the context of PaNOSC. COMSYL has evolved from the old OAR task manager to the new SLURM at the ESRF clusters. The OASYS add-on has also been updated. But, with no doubts, the most important and useful development concerns the development of a fast and lightweight tool for partially coherent beamline simulations [3] based on the same concepts of COMSYL, but fully implemented in python. This method has been thoroughly benchmarked against COMSYL and other tools used to simulate partially coherent x-rays [3], and it is fully integrated in the OASYS package.

Not directly linked with COMSYL, but in the heart of OASYS, it is important to mention that the ray tracing engine SHADOW and the corresponding add-on ShadowOui have been upgraded to allow simulations for monochromators using crystals with high d-spacing [4], which is of particular interest for storage rings producing soft X-rays.

7.4 Remote usage

Being based on Orange which itself is using PyQT, OASYS is a standalone desktop application. By default it offers no remote access and/or collaborative development capabilites. A major portion of development has thus been focused on these features.

7.4.1 Remote workspace access

As part of PaNOSC deliverable D5.3, we have developed a widget for access to a remote repository of OWS workspaces (binary files in which OASYS projects are saved). The widget is able to read the file content of an arbitrary github repository where the files are be stored. The metadata description is read automatically from the workspace. It allows for access to several online repositories and opening of the workspace directly from within OASYS (Fig. 7b).

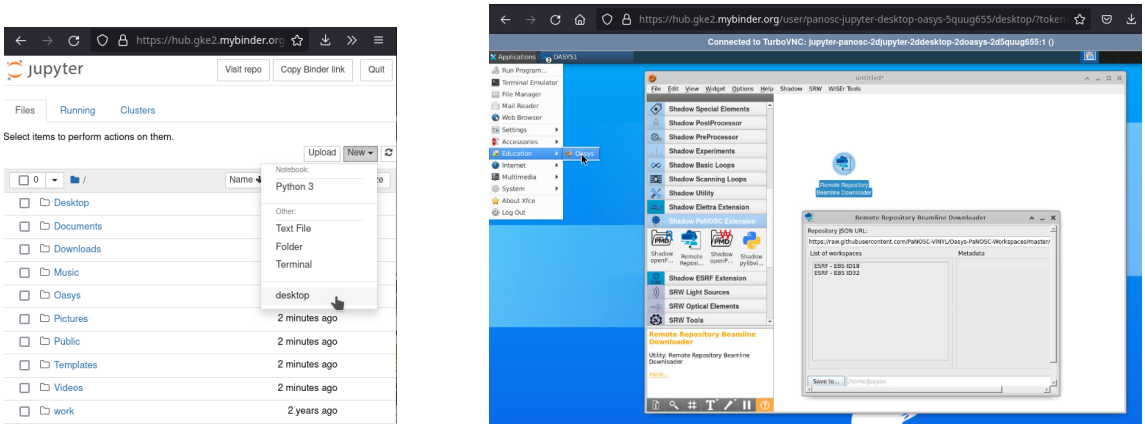
7.4.2 Access to computing resources

Running programs with rich GUIs remotely is non-trivial. One option is to use remote desktop solutions, but they are non-collaborative and require quite extensive user permissions. HPC servers normally also do not provide desktop environments. We have developed and experimented with several remote access options (see deliverable D5.2), but only the final one is presented and described here. A solution was developed through which arbitrary GUI software can be run as a Jupyter hub plugin as shown in Fig. 8a. This links OASYS package with the rest of the WP5 simulation codes and can be installed into any existing Jupyter hub instance and allows OASYS to be run in a web browser (Fig. 8b), accessing any dedicated computing resources. This also avoids complex user permissions and delegates them to the Jupyter hub instance.

7.4.3 E-learning

Building on the solution described in Sec. 7.4.2, a Docker container has been prepared for the purpose of E-learning. The said Docker container has been successfully used during the PaNOSC summer school

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



(a) (b)

Figure 8: (a) Running the XFCE desktop environment from within Jupyter hub. (b) A new browser tab opens with XFCE desktop, providing a way to run OASYS inside the web browser while sharing the computing resources with the Jupyter hub instance.

organized by the training work package, WP8. It was further enriched with all necessary software for the course: Crispy, McStas and OASYS. Students could then seamlessly access all the required software without any installation on local machines and sharing the remote computing resources.

7.5 openPMD

As a part of task 5.1 and deliverable D5.1, we (CERIC-ERIC) have developed OASYS support for openPMD format and contributed to the standard for ray-tracing simulations. Two specific widgets, one for reading and one for writing openPMD files, are a part of the PaNOSC toolbox (Fig. 7a).

7.6 Beyond optics - Full beamline simulations

Synergy between different simulation codes has been an important part of the developments in WP5. As OASYS simulates the X-ray spot parameters at various points along the beamline, the addition of beam-sample interaction through other codes was sought. Specifically, GAPD calculator found in Simex was used for demonstration purposes and is a part of deliverable D5.3. A prototype widget has been designed that can be seamlessly connected to the standard workflow. With this, one uses the result of X-ray propagation at the sample position and performs diffraction calculations on it. The architecture of the widget was further improved and it now allows for different calculators to be implemented in the future. A checking mechanism whether a given calculation code is properly installed on the system has also been implemented. The widget itself is shown in Fig. 9a and an example workflow in Fig. 9b.

7.7 OASYS PaNOSC use cases

A list of use cases using OASYS, submitted to PaNOSC website is provided in Table 5. Out of 31 submitted use cases, 5 of them are either direct contributions to OASYS code or connected to infrastructural uses and setup of OASYS, i. e. remote access, e-learning, etc. This amounts to 16% of all use cases, implying a significant impact in the community.

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

(a) (b)

Figure 9: (a) Widget for running SimEx calculation directly from Oasys. (b) Example workflow where a beamline simulation result is used as an input to SimEx GAPD calculator.

#	Name
8	Remote, collaborative access to beamline optics simulations
12	K-B System performance analysis
13	Effects on the spot quality of the mirror error profile and source pointing instability
16	Coherent mode decomposition of synchrotron emission by COMSYL
31	Seamless connection of Jupyter notebooks and GUI applications for e-learning purposes

Table 5: Links to PaNOSC use cases linked to OASYS. All use cases are published on `www.panosc.eu`.

8 SimEx-Lite

SIMEX [5] is a platform for simulation of experiments at advanced laser and X-ray light sources. SimEx-Lite is the core package of the SIMEX platform funded by the PaNOSC project to provide the calculator interfaces and data APIs for start-to-end simulation. By exposing all aspects of typical experiments at light source infrastructures from the source to the detector, it allows to build a workflow to simulate behaviours of an X-ray instrument.

8.1 Repository

SimEx-Lite is hosted at `https://github.com/PaNOSC-ViNYL/SimEx-Lite`, and its documentation can be found at `https://simex-lite.readthedocs.io/en/latest/`. The current release of SimEx-Lite is version-1.0.0.

8.2 Modules

There are several calculator and data APIs already developed in SimEx-Lite.

Calculators:

- SourceCalculators
  - GaussianSourceCalculator [6]
- PropagationCalculators

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

  - WPGPropagationCalculator [6]
- PMICalculators (PhotonMattterInteractionCalculators)
  - SimpleScatteringPMICalculator
- DiffractionCalculators
  - SingFELDiffractionCalculator [7]
  - CrystFELDiffractionCalculator [8]
- DetectorCalculators
  - GaussianNoiseCalculator [9]

Data APIs:

- WavefrontData
  - WPGFormat
- SampleData
  - ASEFormat (For atomic structure formats supported by ASE)
- PMIData (PhotonMatterInteractionData)
  - XMDYNFormat
- Diffractiondata
  - SingFELFormat
  - EMCFormat
- DetectorData
  - CXIFormat

8.3 Usage

A calculator can be easily constructed within SimEx-Lite. Here, we take the SingFELDiffractionCalculator as an example to explain its usage.

```
from SimExLite.PMIData import PMIData, XMDYNFormat
from SimExLite.DiffractionCalculators import SingFELDiffractionCalculator

input_data = PMIData.from_file("./testFiles/PMI.h5", XMDYNFormat, "PMI_data")
diffraction = SingFELDiffractionCalculator(
        name="SingFELCalculator",
        input=input_data,)

print(diffraction.parameters)
```


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

With `print(diffraction.parameters)` at the end, one can check the default values of the calculator's parameters. The output will look like this:

```
 - Parameters object -
random_rotation                 True                If it's False, the orientations...
calculate_Compton               False               If calculate the compton...
slice_interval                  100                 The slice interval of the pmi...
slice_index_upper               1                   The upper limit of the slice...
pmi_start_ID                    1                   The start ID of the pmi files
pmi_stop_ID                     1                   The stop ID of the pmi files
number_of_diffraction_patterns  10                  The number of diffraction...
pixel_size                      0.001       [meter] The pixel size of the detector
pixels_x                        10                  Number of pixels in x direction
pixels_y                        5                   Number of pixels in y direction
distance                        0.13        [meter] Sample to detector distance
mpi_command                     mpirun -n 2         The mpi command to run pysingfel
```

To run a simulation, as long as the corresponding backengine software is installed in the system, one can start the simulation with this function:

```
output = diffraction.backengine()
```

The output is a data class mapping to the native output of the simulation software, with the `output.get_data()` function, one can read the output data into memory as a python dictionary and conduct further analysis.

With different calculators representing different stages of an experiment, one can create an `Instrument` instance to perform start-to-end simulation. An example is provided at https://github.com/PaNOSC-ViNYL/instrument_database/blob/spb/institutes/EuXFEL/instruments/SPB-SFX/HEAD/simex-lite/SPB-SFX.py. One can load this defined instrument and run the defined simulation.

The packages integrated in the SIMEX platform have been applied in researches reported in several peer-reviewed publications[9–11].

9 Integrated simulation data workflows

EXtra-xwiz is a command-line tool for automated processing of the calibrated serial crystalography data[12]. As described in the Task 5.5 of the proposal of Work Package 5, we have created a workflow to couple SimEx-Lite and EXtra-xwiz together to optimize instrument parameters for serial femtosecond crystallography (SFX) experiments by analyzing the simulation data from SimEx-Lite with EXtra-xwiz.

As the schematic of the workflow shown in Fig. 10, the simulation parameters and sample structure (can be any format compatible with the PDB format) are set for the CrystFEL calculator in SimEx-Lite. After running the simulation, the diffraction patterns output in CXI format is analysed by EXtra-xwiz. By making use of the metrics obtained with EXtra-xwiz, we can optimize the parameters to maximize the metric of interest in a certain condition. The optimized parameters can be a useful reference to achieve better experiment quality.

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

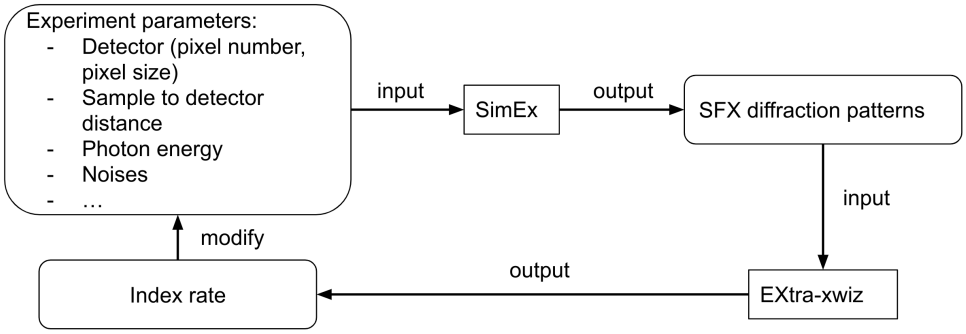
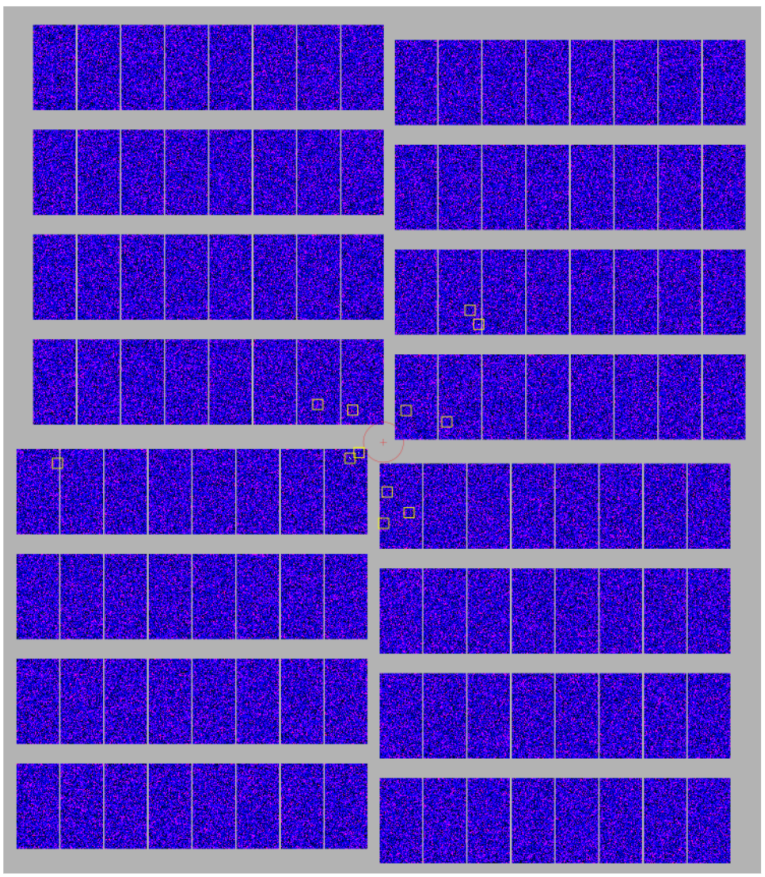


Figure 10: The schematic of the coupling of simulation software and data analysis tools.

One of the workflow's use cases is to optimize the sample-to-detector distance for SFX experiments to reach the highest index rate within an acceptable range of resolution. We scanned several data points of sample-to-detector distance for the SFX simulation and analyzed the output with EXtra-xwiz to get the index rate in the corresponding condition (Fig. 11) With different X-ray energy densities, the sample-to-detector distance needed to reach the highest index rate is different. One can then optimize these parameters based on the exact need.

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

(a)



(b)

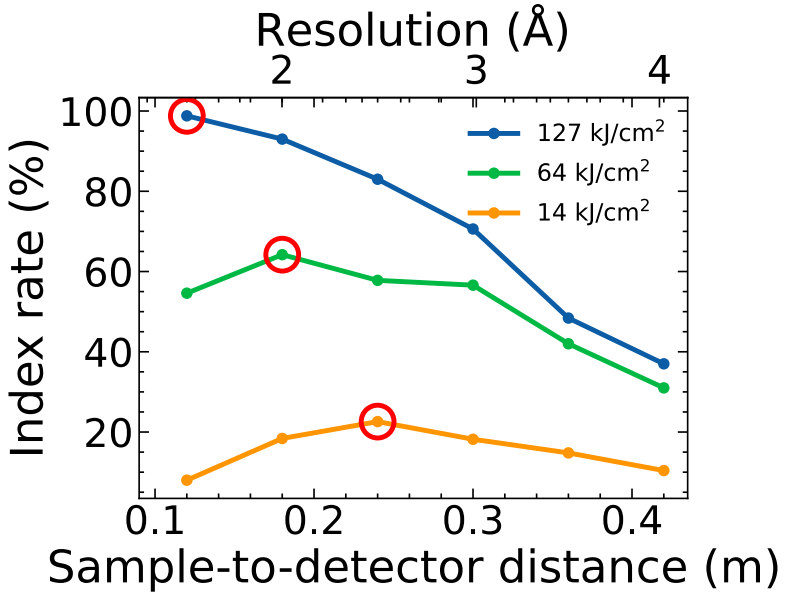


Figure 11: (a) An example SFX pattern for the simulation, the white rectangle boxes indicate found peaks. (b) The index rate curve as a function of the sample-to-detector distance with different X-ray energy densities.

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

10 Use case: Towards Digital Twin

10.1 Goal

The goal of this use case is to interface McStas simulations of an instrument from a research facility to the data acquisition system in order to offer to users the same experience as during their experiments. Users are then able to prepare proposals, preparing in advance their experiments optimizing the instrument settings to maximize their scientific outcome and reduce beam time needed.

10.2 Current status

The project is still in an advanced prototype state. A new tab dedicated to specific settings for the simulation mode has been added to the ILL data acquisition system Nomad (see Fig. 13.

The data acquisition is performed as for the real data and results are visualised and treated accordingly. In Fig. 14 a scan with the triple axis spectrometer ThALES is performed using a vanadium sample.

10.3 Repository

Currently in internal ILL gitlab, will be migrated to the PANOSC github repository in the next days.

Everything is released under the GPLv3 licence.

10.4 Structure

The framework consists in:

- one client library (C++)
- one simulation server
- one application manager server
- McStas instrument executables

The client is integrated in the data acquisition system at ILL, named Nomad, that can be started either in *REAL* mode to control the instrument and acquire real data, or in *SIMULATED* mode to communicate to the simulation server. The simulation server objective is to receive simulation requests from a client, schedule and launch the McStas instrument applications, read and cache the results, and send them back to the client.

The managing of the application is performed using the Cameo library developed at the ILL (open source and publicly available). Cameo allows not only to control other applications (start, stop, etc.) but also communication between them with requester–responder publisher–subscriber models implemented using the Zmq library.

The communication between the different components is sketched in Fig. 12

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

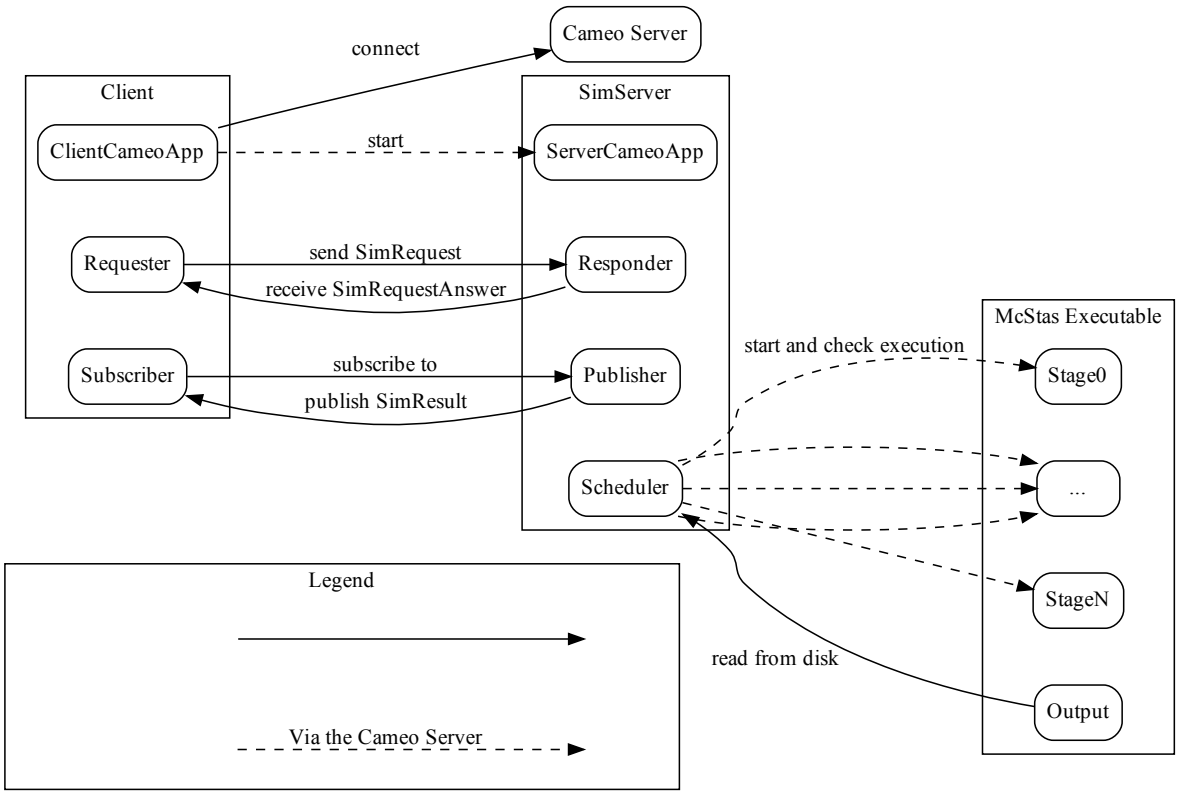


Figure 12: Digital Twin Communication Diagram.

11 Use case: Characterization of BN targets with neutron scattering

With the aforementioned achievements, the realization of scientific use cases, such as PaNOSC use case 18: "In-silico Neutron diffraction from Boro-carbon systems: precise and reliable tool for exact structural analysis and defect detection" are now within reach. While still work in progress, the following diagram illustrates the various components that needed to be provided and interfaced. This has now been achieved and the use case will soon be executed at ELI-ALPS.

12 Concluding remarks and future developments

libpyvinyl was developed mainly by the work package partners from Eu.XFEL, ILL, and ESS, important feature requests and tests were contributed by ESRF. To ensure that the library can fulfill its purpose as an harmonization interface for various simulation backengines, it is vital that the library continues to be developed in a decentralized manner at various places representing a diversity of physics applications, simulation codes and use cases. While the continuation of development of individual simulation codes and APIs (such as McStasScript, SIMEX, or Oasys) is secured at the level of individual RIs, this is, at the time of writing, much more uncertain in the case of libpyvinyl. It would be desirable that the development teams of individual simulation codes and frameworks can allocate sufficient resources

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

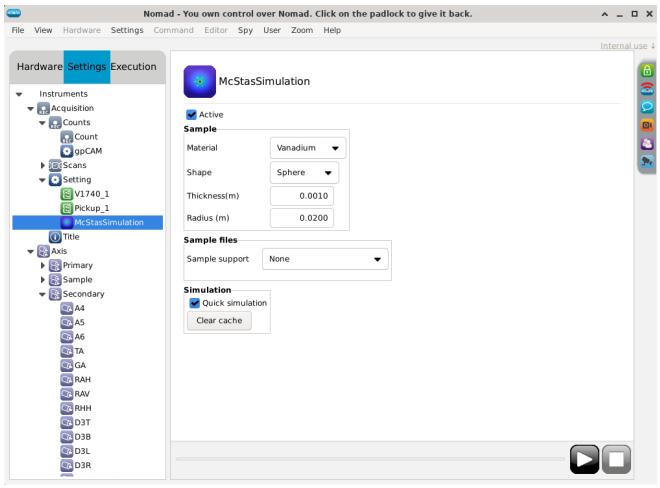


Figure 13: ILL's Nomad simulation settings tab

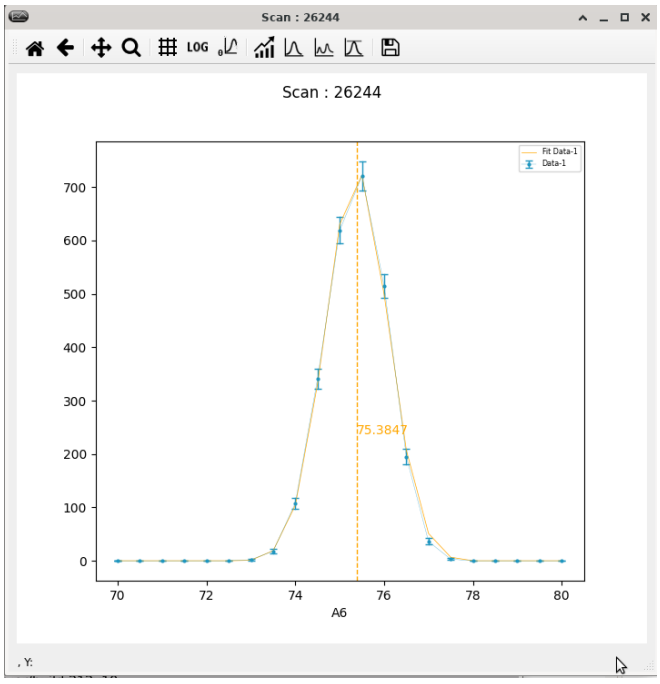


Figure 14: Plot of a scan with ThALES instrument with a Vanadium sample. The instrument description retrieved from the instrument database

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

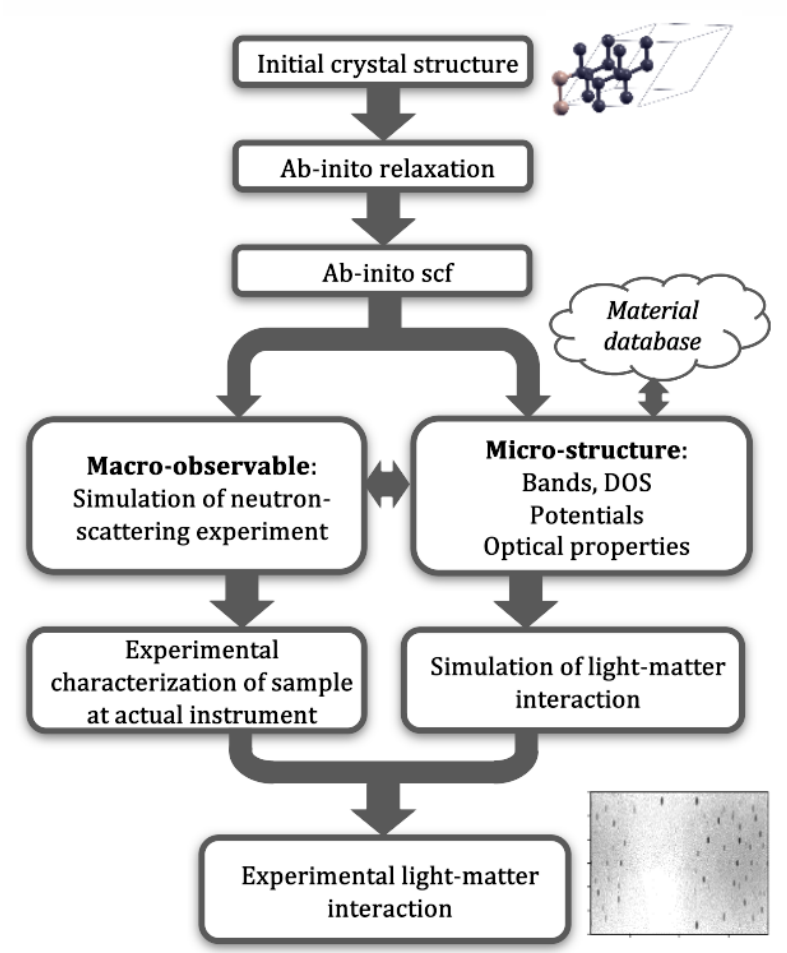


Figure 15: A visual representation of PaNOSC use case 18: In-silico Neutron Diffraction: precise tool for exact structural analysis and defect detection

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

to the maintenance and development of *libpyvinyl* within their budgets. Dedicated resources for the library do currently not exist. ELI colleagues prepared and performed a number of test cases for the GaussianSourceCalculator implementation in SimEx. These test cases include technical tests such as I/O validation, data flow validation, method calls, and default setups; and physics-based sanity checks on the results the calculator gives. The testing revealed some important hard-to-spot bugs related to the file locations. The test cases are currently merged into the main branch of SimExLite.

A ViNYL software releases

Package name	Latest Version at time of writing	DOI
libpyvinyl	1.1.2	10.5281/zenodo.6558163
instrumentDatabaseAPI	0.1.0	10.5281/zenodo.7373910
McStasScript	0.0.46	N.A[1]
SimEx-Lite	1.0.0	10.5281/zenodo.7371741
OASYS1-PaNOSC	0.3.2	10.5281/zenodo.7371215
OASYS1-OasysWiser	0.3.22	10.5281/zenodo.7371265

Table 6: Released versions and DOI for ViNYL software packages

References

[1] M. Manfredda et al. “WISER wavefront propagation simulation code: advances and applications”. In: *Advances in Computational Methods for X-Ray Optics V*. Ed. by Oleg Chubar and Kawal Sawhney. Vol. 11493. International Society for Optics and Photonics. SPIE, 2020, 114930B. DOI: 10.1117/12.2568574. URL: https://doi.org/10.1117/12.2568574.

[2] M. Glass and M. Sanchez del Rio. “Coherent modes of X-ray beams emitted by undulators in new storage rings”. In: *EPL (Europhysics Letters)* 119.3 (Aug. 2017), p. 34004. DOI: 10.1209/0295-5075/119/34004. URL: https://doi.org/10.1209/0295-5075/119/34004.

[3] M. Sanchez del Rio, R. Celestre, J. Reyes-Herrera, P. Brumund, and M. Cammarata. “A fast and lightweight tool for partially coherent beamline simulations in fourth-generation storage rings based on coherent mode decomposition”. In: *Journal of Synchrotron Radiation* 29.6 (Nov. 2022), pp. 1354–1367. DOI: 10.1107/S1600577522008736. URL: https://doi.org/10.1107/S1600577522008736.

[4] X. J. Yu et al. “Beamline simulations using monochromators with high d-spacing crystals”. In: *Journal of Synchrotron Radiation* 29.5 (Aug. 2022), pp. 1157–1166. DOI: 10.1107/s160057752200707x. URL: https://doi.org/10.1107/s160057752200707x.

[5] C. Fortmann-Grote et al. “SIMEX: Simulation of Experiments at Advanced Light Sources”. In: *arXiv:1610.05980 [physics]* (Nov. 2016). arXiv: 1610.05980 [physics].

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

[6] L. Samoylova, A. Buzmakov, O. Chubar, and H. Sinn. "WavePropaGator: Interactive Framework for X-ray Free-Electron Laser Optics Design and Simulations". In: *Journal of Applied Crystallography* 49.4 (Aug. 2016), pp. 1347–1355. ISSN: 1600-5767. DOI: 10.1107/S160057671600995X.

[7] Chun Hong Yoon. *Chuckie82/Pysingfel*. https://github.com/chuckie82/pysingfel.

[8] Thomas A. White et al. "*CrystFEL* : A Software Suite for Snapshot Serial Crystallography". In: *Journal of Applied Crystallography* 45.2 (Apr. 2012), pp. 335–341. ISSN: 0021-8898. DOI: 10.1107/S0021889812002312.

[9] Juncheng E et al. "Expected Resolution Limits of X-Ray Free-Electron Laser Single-Particle Imaging for Realistic Source and Detector Properties". In: *Structural Dynamics* 9.6 (Nov. 2022), p. 064101. DOI: 10.1063/4.0000169.

[10] C. Fortmann-Grote et al. "Start-to-End Simulation of Single-Particle Imaging Using Ultra-Short Pulses at the European X-Ray Free-Electron Laser". en. In: *IUCrJ* 4.5 (Sept. 2017), pp. 560–568. DOI: 10.1107/S2052252517009496.

[11] Juncheng E et al. "Effects of Radiation Damage and Inelastic Scattering on Single-Particle Imaging of Hydrated Proteins with an X-ray Free-Electron Laser". In: *Scientific Reports* 11.1 (Sept. 2021), p. 17976. ISSN: 2045-2322. DOI: 10.1038/s41598-021-97142-5.

[12] Oleksii Turkot. *Getting Started with EXtra-xwiz — User Documentation 1.0 Documentation*. https://rtd.xfel.eu/docs/data-analysis-user-documentation/en/latest/xwiz/xwiz.html.

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