PaNOSC

Photon and Neutron Open Science Cloud

H2020-INFRAEOSC-04-2018

Grant Agreement Number: 823852

MS5.2 Demonstration of Simulation Services

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:	MS5.2 Demonstration of simulation services
Deliverable Type:	Report
Dissemination Level:	Public
Contractual Delivery Date:	31/11/2020
Actual Delivery Date:	09/11/2020
EC project Officer:	René Martins

Document Control Sheet

Document	Title: MS5.2 Demonstration of simulation services
	Version: 1
	Available at:
	Files: 1
Authorship	Written by: Carsten Fortmann-Grote
	Contributors: Juncheng E, Aljosa Hafner, Mads Bertelsen, Daniel Webster
	Reviewed by: Jordi Bodera Sempere
	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	Extreme Light Infrastructure Delivery Consortium (ELI-DC)	Belgium
6	Central European Research Infrastructure Consortium (CERIC-ERIC)	Italy
7	EGI Foundation (EGI.eu)	The Netherlands

Table of Content

1 Introduction

This report outlines the documented simulation APIs released under PaNOSC work package 5 and constitutes our second milestone. The work package is internally named the Virtual Neutron and X–Ray Laboratory "ViNYL", and is responsible for exposing simulation packages in a cloud environment with a unified user interface. Here we demonstrate the underlying APIs that will provide users access to the simulation packages through a documented python interface.

The major simulation packages exposed in ViNYL are SimEx, McStas and Oasys. SimEx is a python library for simulation of X-ray beamline experiments, and connects to a large number of simulation backengines. In some ways, SimEx serves as the template for how ViNYL can later harmonize access to more simulation packages.

McStas is a neutron instrumentation simulation package that uses a meta-language built on C as a user interface, which is more difficult to expose to a cloud based solution. Work package 5 is providing a python API for McStas called McStasScript which is more suitable for exposing through a cloud service such as a Jupyter Hub.

The last major project in ViNYL is Oasys which provides a graphical user interface to simulation of X-ray beamlines. This project will be made available through remote desktop software due to the need for the graphical user interface.

With the released APIs, ViNYL is now ready to expose them through a harmonized top-level API, named libpyvinyl. Libpyvinyl will provide an interface to simulation of both X-ray and neutron experiments.

This report includes an overview of each independent package including links to their documentation and information on the cloud deployment of the software. Examples and tutorials in the form of Jupyter Notebooks are available to showcase the functionality of the software included in ViNYL.

2 SimEx

SimEx[1] is a platform for simulation of experiments at advanced laser and X-ray light sources. As an essential part work package 5, SimEx is responsible for providing a start-to-end simulation API to the users of the photon facilities. It exposes all aspects of typical experiments at light source infrastructures from the source to the experiment allowing to build a simulation chain.

2.1 Capabilities

SimEx provides the harmonized APIs for the essential parts of the photon experiments in the large facilities, as well as the analysis tools for simulation output analysis. The simulations of photon source, light transport through optics elements in the beamline, interaction with a target or sample, scattering from the latter, photon detection, and data analysis can be done with various modules within the framework. This simulation framework has been actively used for single particle imaging studies [2, 3].

2.2 Documentation

SimEx provides a user manual including the API reference through this link: https://panosc-vinyl.github.io/SimEx/

More comprehensive documentations have been created and will be improved in the future:

- A more detailed user manual with a more user-friendly style on https://simex.readthedocs.io/en/latest/
- A repository of example Jupyter notebooks: https://github.com/PaNOSC-ViNYL/SimEx-notebooks

2.3 Code availability

The source code of SimEx is hosted on GitHub: https://github.com/PaNOSC-ViNYL/SimEx

2.4 Demonstration of the photon experiment simulation service

The SimEx API can be easily accessed after being deployed on a cloud Jupyter server. Take the Jupyter Hub cloud service on the DESY Maxwell cluster as an example, we will show how to execute the simulation through the Jupyter notebook interface.

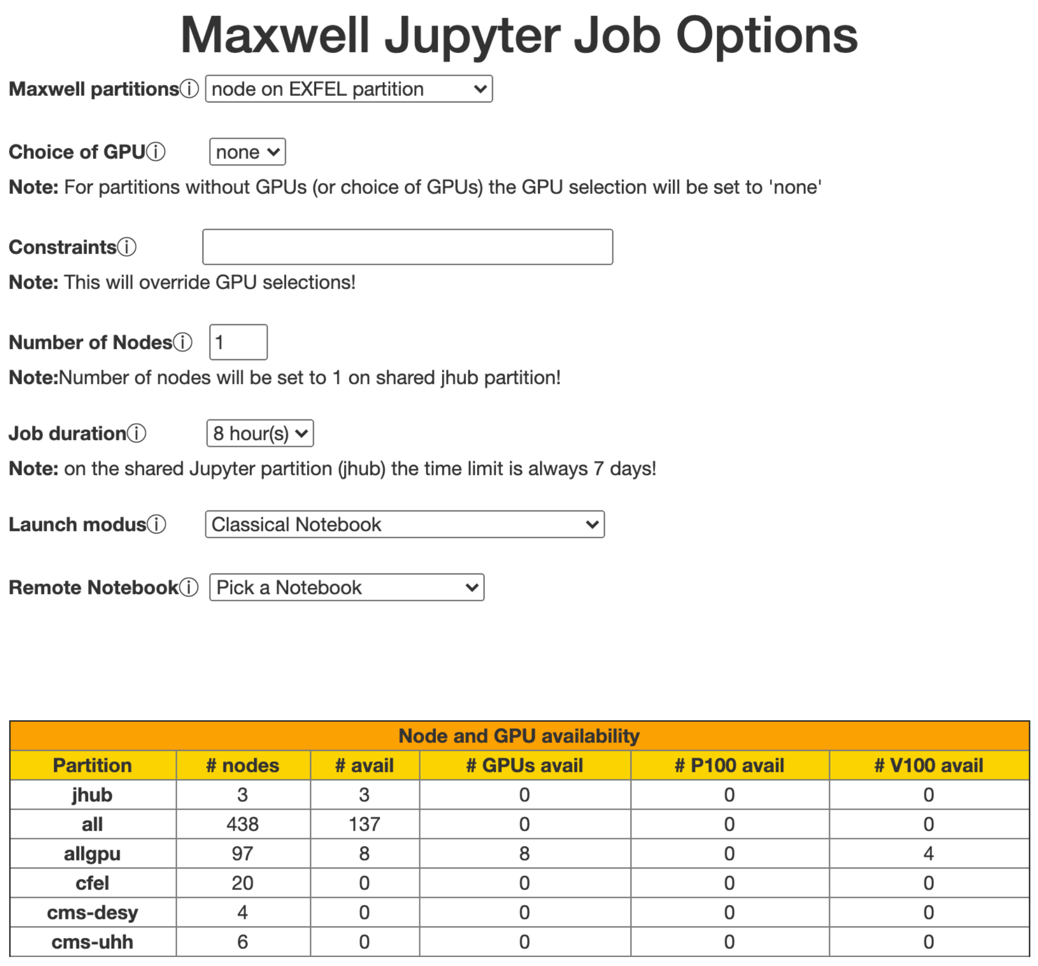
By logging into https://max-jhub.desy.de/, a Jupyter notebook job can be started as shown in Fig. 1.

After the Jupyter notebook job started, one can run any Jupyter notebooks uploaded to their cloud space. For example, the user can upload a notebook hosted on the SimEx-notebooks repository (https://github.com/PaNOSC-ViNYL/SimEx-notebooks) and open it.

Once the Jupyter notebook is opened, one can choose the deployed SimEx ipython kernel and execute the script as shown in Fig. 2.

3 Oasys

Oasys is a cross-platform multi-code environment for X-ray optics simulations. It is used in all stages of optical design of the beamlines: while prototyping, commissioning and upgrading. The GUI is provided by Orange (https://orange.biolab.si/), a framework developed specifically with visual programming in mind. A variety of simulation codes have by now been ported into it, e.g. Shadow (ray tracing), SRW (wavefront propagation). WP5 has been providing means to integrate Oasys with the rest of the simulation software mentioned in the report and to develop a new wavefront propagation code called Wiser.


Maxwell Jupyter Job Options

Maxwell partitions ⓘ node on EXFEL partition

Choice of GPU ⓘ none

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

Constraints ⓘ

Note: This will override GPU selections!

Number of Nodes ⓘ 1

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

Job duration ⓘ 8 hour(s)

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

Launch modus ⓘ Classical Notebook

Remote Notebook ⓘ Pick a Notebook

Node and GPU availability					
Partition	**# nodes**	**# avail**	**# GPUs avail**	**# P100 avail**	**# V100 avail**
jhub	3	3	0	0	0
all	438	137	0	0	0
allgpu	97	8	8	0	4
cfel	20	0	0	0	0
cms-desy	4	0	0	0	0
cms-uhh	6	0	0	0	0

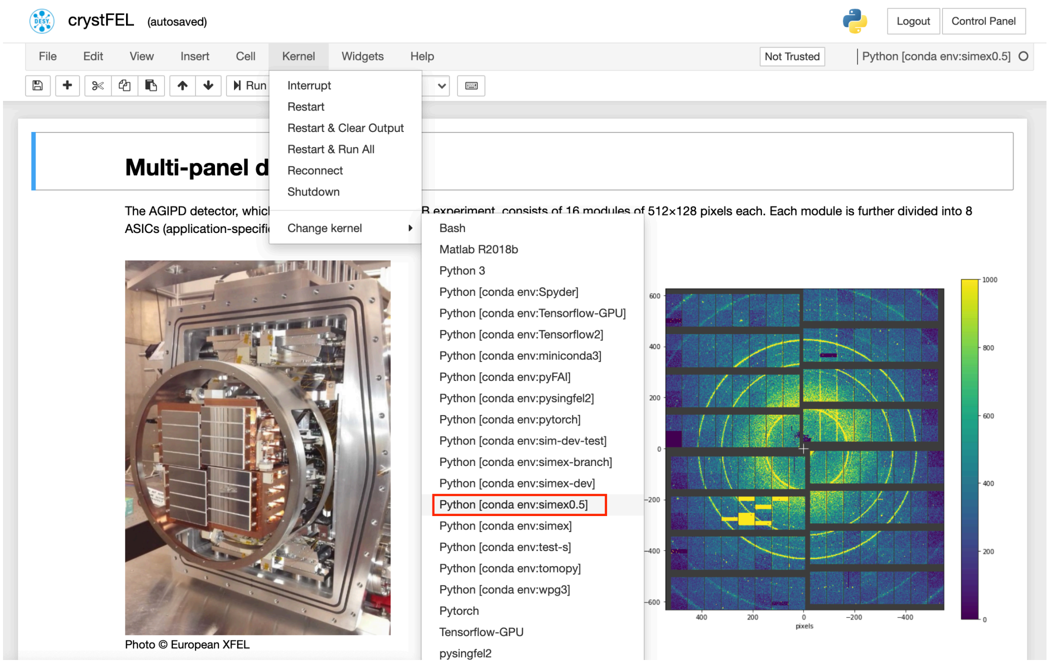
Figure 1: Start a Juypyter notebook from the Maxwell Jupyter Hub

3.1 Capabilities

Wiser addresses an important gap in X-ray optics simulations, namely the combined simulations of both the figure error and roughness. While the first introduces deviations from the perfect surface on large distances, the second represents the tiny fluctuations about the mean on atomic length scales. In reality, both are indistinguishable from each other, as the total position is a sum of both contributions. For the sake of easier understanding and easier simulations, the two have been decoupled. With Wiser it is possible to numerically simulate wavefront propagation through the combined profile.

As solving the Frensel-Huyghens integral is in general an expensive process, the code has been optimized with the use of Numba library. This not only enables the program to automatically parallelize and pre-compile, but also to scale over all the available CPUs. Running in cloud environment, such scaling makes the program as efficient as possible, based on the constraints introduced from the outside (available resources due to restricted user profile, etc.).

Another repository, containing features stemming specifically from Panosc collaboration, is located here: `https://github.com/PaNOSC-ViNYL/OASYS1-PaNOSC`. It aims to bring standardized file input/output to Oasys. Remote workspace access and Jupyter notebook support in Oasys is


crystFEL (autosaved)
Logout
Control Panel
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python [conda env:simex0.5]
Run
Interrupt
Restart
Restart & Clear Output
Restart & Run All
Reconnect
Shutdown
Change kernel
Bash
Matlab R2018b
Python 3
Python [conda env:Spyder]
Python [conda env:Tensorflow-GPU]
Python [conda env:Tensorflow2]
Python [conda env:miniconda3]
Python [conda env:pyFAI]
Python [conda env:pysingfel2]
Python [conda env:pytorch]
Python [conda env:sim-dev-test]
Python [conda env:simex-branch]
Python [conda env:simex-dev]
Python [conda env:simex0.5]
Python [conda env:simex]
Python [conda env:test-s]
Python [conda env:tomopy]
Python [conda env:wpg3]
Pytorch
Tensorflow-GPU
pysingfel2
pixels

Multi-panel d

The AGIPD detector, whic ... B experiment, consists of 16 modules of 512×128 pixels each. Each module is further divided into 8 ASICs (application-specifi

Photo © European XFEL

Figure 2: Start a Juypyter notebook from the Maxwell Jupyter Hub

planned in the next months.

3.2 Documentation

Wiser consists of three layers, the underlying computational library, an intermediate API link and the GUI part. Current situation with documentation is the following:

- LibWiser: all the functions have been documented with proper docstrings. A series of files, demonstrating the common use cases is in preparation.
- wofrywiser: functions are still being equipped with docstrings, but a series of demonstrator .py files, showing the scripting capabilities are available here: `https://github.com/oasys-elettra-kit/wofrywiser/tree/master/wofrywiser/_test`
- Wiser GUI: demonstration workspaces and instruction video will be provided as part of M5.2 and will be made available in the Panosc-Vinyl github repository.

General Panosc-supported Oasys school has been organised at ESRF in May 2019. Training material publicly available in the following github repository: `https://github.com/oasys-kit/oasys_school`.

3.3 Code availability

All the code for Oasys and its packages is open source and developed on GitHub under the GPL 3.0 licence. It can be installed using the Python package manager pip from the PyPI repository. Furthermore, Oasys/Orange has its own package manager through which the add-ons (from PyPI) are also available for automatic installation.

Description	Link
Oasys GitHub repository	https://github.com/oasys-kit/OASYS1
Wiser GitHub repository	https://github.com/oasys-elettra-kit/OASYS1-Wiser
WofryWiser GitHub repository	https://github.com/oasys-elettra-kit/wofrywiser
LibWiser GitHub repository	https://github.com/oasys-elettra-kit/WISEr/
Oasys1 pypi page	https://pypi.org/project/oasys1/
Oasys1-Wiser pypi page	Dev: https://pypi.org/project/oasys1-oasyswiser/
	Final: https://pypi.org/project/oasys1-wiser/
wofrywiser pypi page	https://pypi.org/project/wofrywiser/
LibWiser pypi page	https://pypi.org/project/libwiser/
Oasys1-Panosc GitHub repository	https://github.com/PaNOSC-ViNYL/OASYS1-PaNOSC
Oasys1-Panosc pypi page	https://pypi.org/project/oasys1-panosc/

3.4 Demonstration of X-ray optics simulation service

A video demonstration has been published on Zenodo and showcases the usage of OASYS and Wiser package as a remote application in CERIC-ERIC user portal, VUO. The demonstration is available here: https://zenodo.org/record/4250966.

4 McStasScript

McStasScript is a python API to the McStas package which is capable of simulating neutron scattering instrumentation. Using McStasScript, McStas simulations can be built and executed directly from a python interface. The resulting data from the simulations are made available as python data structures. These tasks are appropriate for Jupyter Notebooks which are more suitable to being exposed as a cloud service than the traditional McStas workflow.

The McStas package has a sister project named McXtrace that simulate X-ray insrumenation. These packages share the majority of the code base and have similar syntax, so with minor modifications, McStasScript can support both McXtrace and McStas. A github branch with these modifications is available, but it has yet to be incorparated into the master branch and main documentation.

4.1 Capabilities

McStasScript is an API to McStas, and covers nearly the entire set of McStas features. McStas is widely used for design of neutron scattering instrumentation, usually used for simulation from the neutron moderator to detector including sample physics. The package includes the tools necessary for simulating optics, samples of different types, sample environment and detectors. McStasScript can also be used as an API to McXtrace that has similar capabilities for X-ray instrumentation, yet is less widely used.

4.2 Documentation

Documentation for McStasScript is provided in several forms that suits different kinds of users and developers. Some understanding of the underlying McStas package is necessary, the basics of which are available through the Jupyter Notebook tutorials.

- Classic manual in pdf format outlining all userfacing classes and methods
- Developer documentation in pdf format explaining development principles and class diagram
- Set of 11 Jupyter notebooks that demonstrates and teaches McStas and McStasScript
- Documentation strings in python classes and methods that provide help during use

4.3 Code availability

The McStasScript API is a pure python package, and is distributed with pypi making installation easy. McStas is to be installed separately, and McStasScript needs to be configured to find the underlying McStas installation. McStas and McStasScript are also available together in a single docker container where configuration is already performed. Running the code with support from McXtrace requires obtaning McStasScript from a different GitHub branch.

Description	Link
McStas webpage	`https://mcstas.org`
McStas GitHub repository	`https://github.com/McStasMcXtrace/McCode`
McStasScript GitHub repository	`https://github.com/PaNOSC-ViNYL/McStasScript`
McStasScript pypi page	`https://pypi.org/project/McStasScript/`
McStas + McStasScript docker	`https://hub.docker.com/r/mccode/mcstas-2.6.1-mcstasscript`
McXtrace webpage	`https://mcxtrace.org`
McXtrace support branch	`https://github.com/PaNOSC-ViNYL/McStasScript/tree/McXtrace_support`

4.4 Demonstration of neutron raytracing simulation service

The European Spallation Source Data Management and Software Centre has a Jupyter Hub service where a docker image with the software from the Data Reduction and Analysis group is used for deployment. This Jupyter Hub is not yet available outside the office network due to outstanding concerns about security which are expected to be solved within the year. Instead a short video showcasing part of a McStasScript tutorial notebook is available `http://project.esss.dk/owncloud/index.php/s/a03kEdmf0tMEKG5/download`.

5 libpyvinyl

5.1 Introduction

Since X-ray and neutron scattering are complementary techniques, it would be convenient for users to access simulation services for both through a single python API. Furthermore infrastructure can be shared and standardized. The libpyvinyl package developed under PaNOSC work package is a first step in this direction, providing a flexible framework with access to both X-ray and neutron backengines for simulation services. The structure of the SimEx package is used as a template for the framework. We have successfully demonstrated that McStasScript can act as a backengine to facilitate neutron simulations, which underlines the versatility of the API.

5.2 Capabilities

The specifications of the libpyvinyl API are documented at https://github.com/PaNOSC-ViNYL/ViNYL-project/blob/master/Docs/Reports/TDR/API/specifications.md. In summary, libpyvinyl defines a BaseCalculator that defines methods for configuring the data input and output streams or files of the simulation, modifying the simulation parameters, running the simulation (including on HPC resources), snapshooting a simulation, restarting a simulation, and writing simulation data to the configured output stream or file with support for the openpmd formats developed in D5.1. Secondly, libpyvinyl defines a Parameters class as the top level object through which to define the physical, computational, and numerical parameters of a simulation.

5.3 Documentation

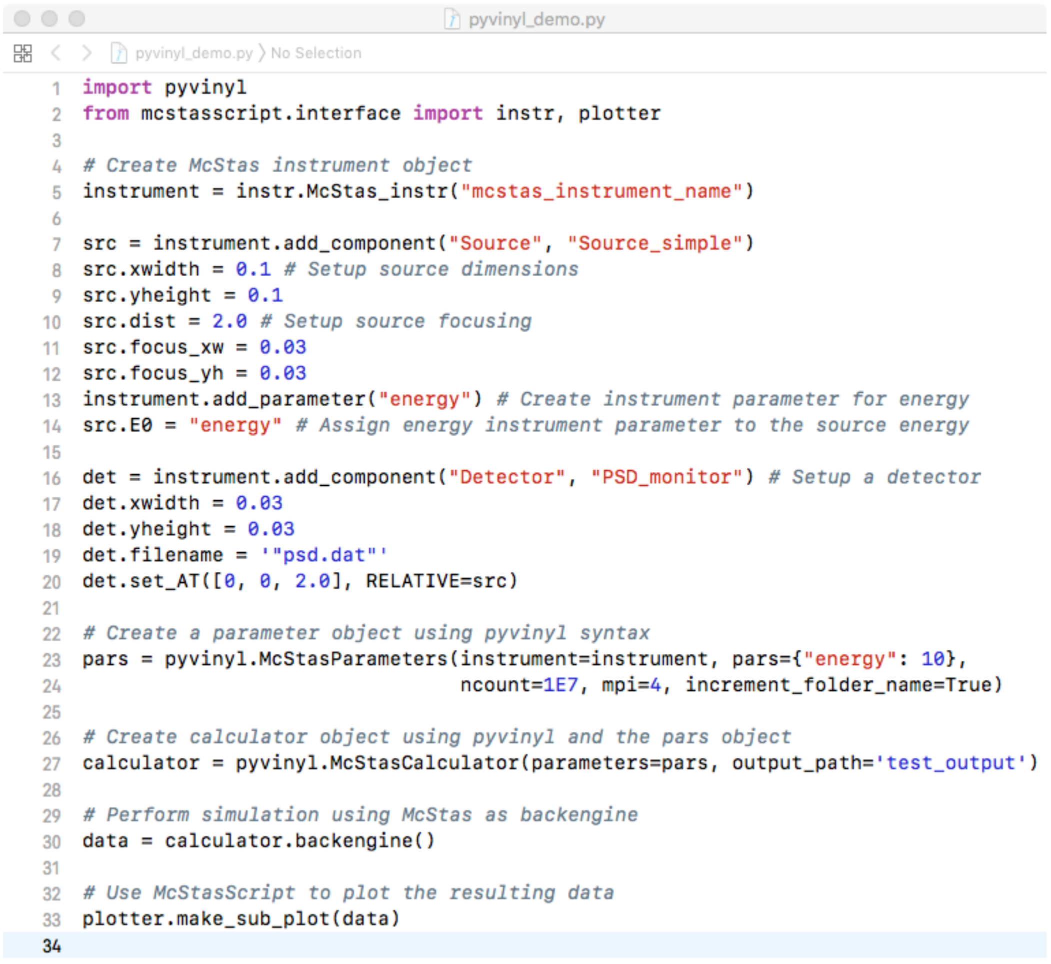
The API reference manual for libpyvinyl is published at https://libpyvinyl.readthedocs.io/en/latest. The syntax for using McStasScript with a libpyvinyl interface is shown in figure 3. The underlying jupyter notebook is available for download https://github.com/PaNOSC-ViNYL/McStas_ViNYL_concept.

Rollout of libpyvinyl in SimEx will be part of the upcoming work in WP5.

5.4 Code availability

The first release of libpyvinyl is available on the python package index (PYPI). The link and further resources are summarized in the following table:

Description	Link
libpyvinyl repository	https://github.com/PaNOSC-ViNYL/libpyvinyl
libpyvinyl on PYPI	https://pypi.org/project/libpyvinyl/
libpyvinyl reference manual	https://libpyvinyl.readthedocs.io/en/latest
McStasScript using libpyvinyl	https://github.com/PaNOSC-ViNYL/McStas_ViNYL_concept



```python
import pyvinyl
from mcstasscript.interface import instr, plotter

# Create McStas instrument object
instrument = instr.McStas_instr("mcstas_instrument_name")

src = instrument.add_component("Source", "Source_simple")
src.xwidth = 0.1 # Setup source dimensions
src.yheight = 0.1
src.dist = 2.0 # Setup source focusing
src.focus_xw = 0.03
src.focus_yh = 0.03
instrument.add_parameter("energy") # Create instrument parameter for energy
src.E0 = "energy" # Assign energy instrument parameter to the source energy

det = instrument.add_component("Detector", "PSD_monitor") # Setup a detector
det.xwidth = 0.03
det.yheight = 0.03
det.filename = '"psd.dat"'
det.set_AT([0, 0, 2.0], RELATIVE=src)

# Create a parameter object using pyvinyl syntax
pars = pyvinyl.McStasParameters(instrument=instrument, pars={"energy": 10},
                                ncount=1E7, mpi=4, increment_folder_name=True)

# Create calculator object using pyvinyl and the pars object
calculator = pyvinyl.McStasCalculator(parameters=pars, output_path='test_output')

# Perform simulation using McStas as backengine
data = calculator.backengine()

# Use McStasScript to plot the resulting data
plotter.make_sub_plot(data)

```

Figure 3: Small example of McStasScript using pyvinyl as an interface.

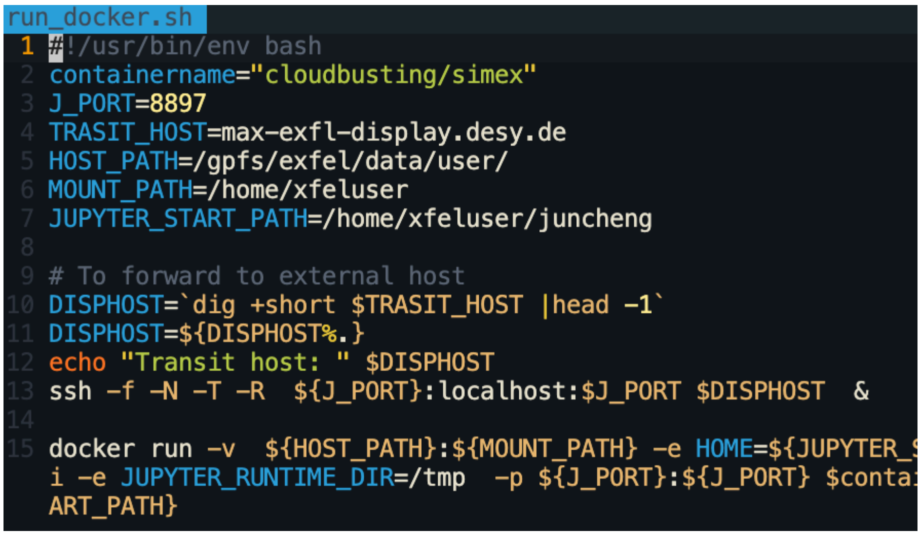
6 Cloud deployment

Access to these simulation services and computing resources is provided through deployment in cloud based environments. This is primarily foreseen to be in Jupyter Hubs providing the user with Jupyter Notebooks through which the simulations can be performed. Such a Jupyter Hub is a natural place for providing tutorials, examples and templates for common tasks.

It is important that the software suite is easy to deploy on new cloud infrastructure. By providing the software developed by ViNYL through docker containers, setup is less dependent on the particular software environment, and thus easier to generalize. Using containers also makes the deployment modular, allowing support for only a subset of the backengines if required. Omitting certain backengines can be necessary as some require licenses, and thus may not be available in all cloud environments.

6.1 Demonstration

A custom Jupyter notebook-enabled container with software included in ViNYL has been deployed for demonstration purposes, and to explore the different possibilities for deployment.

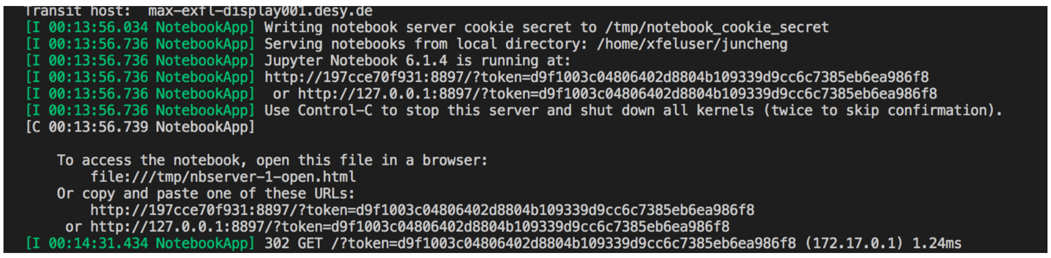


```
run_docker.sh
#!/usr/bin/env bash
containername="cloudbusting/simex"
J_PORT=8897
TRASIT_HOST=max-exfl-display.desy.de
HOST_PATH=/gpfs/exfel/data/user/
MOUNT_PATH=/home/xfeluser
JUPYTER_START_PATH=/home/xfeluser/juncheng

# To forward to external host
DISPHOST=`dig +short $TRASIT_HOST |head -1`
DISPHOST=${DISPHOST%.}
echo "Transit host: " $DISPHOST
ssh -f -N -T -R  ${J_PORT}:localhost:$J_PORT $DISPHOST  &

docker run -v  ${HOST_PATH}:${MOUNT_PATH} -e HOME=${JUPYTER_S
i -e JUPYTER_RUNTIME_DIR=/tmp  -p ${J_PORT}:${J_PORT} $contai
ART_PATH}
```

Figure 4: To edit the configuration for docker running.



```
Transit host:  max-exfl-display001.desy.de
[I 00:13:56.034 NotebookApp] Writing notebook server cookie secret to /tmp/notebook_cookie_secret
[I 00:13:56.736 NotebookApp] Serving notebooks from local directory: /home/xfeluser/juncheng
[I 00:13:56.736 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 00:13:56.736 NotebookApp] http://197cce70f931:8897/?token=d9f1003c04806402d8804b109339d9cc6c7385eb6ea986f8
[I 00:13:56.736 NotebookApp]  or http://127.0.0.1:8897/?token=d9f1003c04806402d8804b109339d9cc6c7385eb6ea986f8
[I 00:13:56.736 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 00:13:56.739 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///tmp/nbserver-1-open.html
    Or copy and paste one of these URLs:
        http://197cce70f931:8897/?token=d9f1003c04806402d8804b109339d9cc6c7385eb6ea986f8
     or http://127.0.0.1:8897/?token=d9f1003c04806402d8804b109339d9cc6c7385eb6ea986f8
[I 00:14:31.434 NotebookApp] 302 GET /?token=d9f1003c04806402d8804b109339d9cc6c7385eb6ea986f8 (172.17.0.1) 1.24ms
```

Figure 5: A successfully running Jupyter server in a docker image.

The scripts to run and create docker image can be found at `https://github.com/PaNOSC-ViNYL/SimEx`. To run the run_docker.sh script at a cloud service, one has to edit the script as shown in Fig. 4 to define the mounting host path exchanging data between the host and the container. The JUPYTER_START_PATH should be specified to where the mapped host user has writing permission.

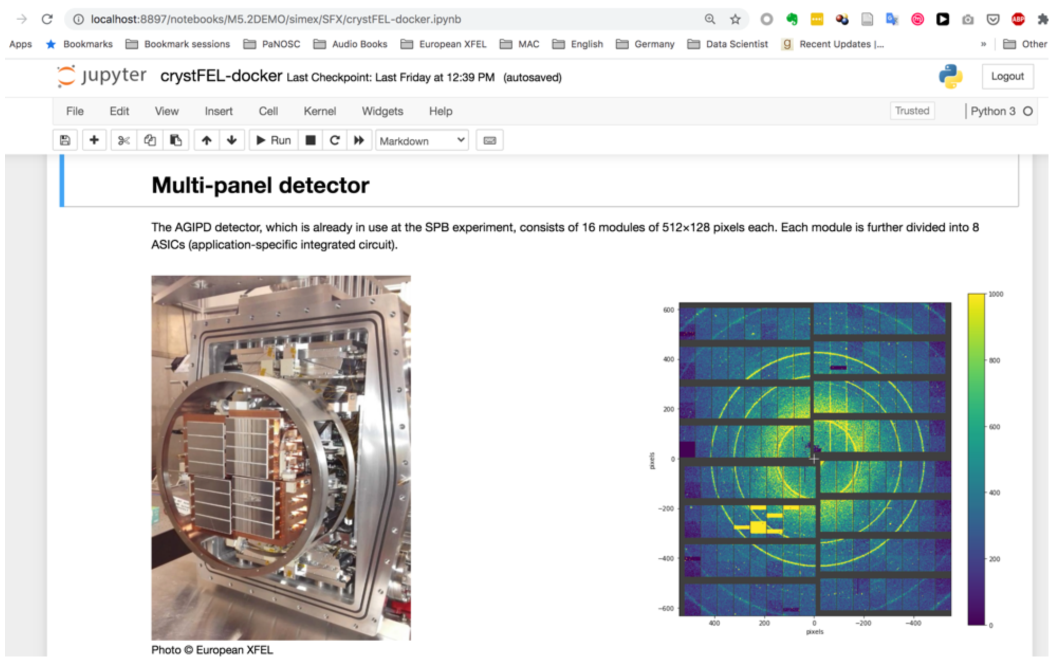
When the notebook runs successfully, one can copy the prompted URL into the web browser and use the Jupyter notebook server as in section 2.4. If the cloud service node running the docker is behind a firewall, the user needs to set the port forwarding per the instruction of the cloud service administrator.

References

[1] PaNOSC-ViNYL/SimEx. https://github.com/PaNOSC-ViNYL/SimEx (2020). URL `https://github.com/PaNOSC-ViNYL/SimEx`.

[2] Yoon, C. H. *et al.* A comprehensive simulation framework for imaging single particles and biomolecules at the European X-ray Free-Electron Laser. *Scientific Reports* **6** (2016).

[3] Fortmann-Grote, C. *et al.* Start-to-end simulation of single-particle imaging using ultra-short pulses at the European X-ray Free-Electron Laser. *IUCrJ* **4**, 560–568 (2017).



Multi-panel detector

The AGIPD detector, which is already in use at the SPB experiment, consists of 16 modules of 512×128 pixels each. Each module is further divided into 8 ASICs (application-specific integrated circuit).

Photo © European XFEL

Figure 6: The user can access the notebooks from their browser through the Jupyter server running in a docker image.