PYTHON-BASED HIGH-LEVEL APPLICATIONS DEVELOPMENT FOR SHANGHAI SOFT X-RAY FREE-ELECTRON LASER

T. Zhang*, J. Chen, B. Liu, D. Wang, SINAP, Shanghai 201800, China

Abstract

Shanghai soft x-ray free-electron laser is currently being built at SSRF campus of SINAP. At the same time, the development of the high-level applications are on-going, with the intention of building a fully open source and robust software ecosystem, Python has been chosen as the essential developing programming language. Up to now, the software framework has been readily established, multiple physics-related high-level applications are under development. Additionally, EPICS soft-IOC applications has been built for the software debugging. The development are taken in a distributed manner, i.e. git is used to organize the source code and for the ease of team collaboration, specific applications are built into Python modules and finally integrated into a single Python package named 'felapps' for deployment. In this paper, details about the Python-based software development at SXFEL and the future ideas are covered.

INTRODUCTION

Shanghai soft x-ray free-electron laser (SXFEL) facility is designed to be a two-staged seeded FEL, which baseline physics operation mode is cascaded high-gain harmonic generation (HGHG), as such the fully coherent radiations with ultra-high brilliance at the wavelength around 8.8 nm could be generated at the end of the second HGHG stage. It is now under construction at the northeast of SSRF campuse site of SINAP, CAS (see Fig. 1), the first lasing is expected to be at the end of 2017. SXFEL also has the potential to radiate at even high frequencies, e.g. after the beam energy upgrading up to 1.1 GeV, FEL radiations within the water window regime could be privided to the users [1].

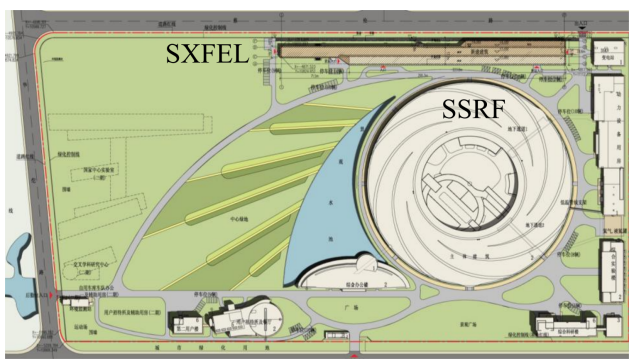

Figure 1: Buiding sketch of SXFEL and SSRF.

Meanwhile, the high-level applications for SXFEL are now under development, the mission is to build a user-friendly, extensible, maintainable system to make the usage of such kind of large scientific facilities be easier. To achieve such purposes, Python has been choosed to be the main developing programming language, since it is a fully object-oriented designed, could be much more powerful when incorporating third-party modules [2]. Numpy and Scipy Python packages are used to handle complex numerical computing issues, such as matrix manipulations, numerical integration as well as other general mathematical calculations [3]. The matplotlib package is used to handle the data visualization jobs, which provides matlab-like commands to deal tasks like figure plotting, but with a much more flexible and extensible manner [4]. So the core Python, together with Numpy, Scipy and matplotlib, actually makes an excellent alternative for matlab.

Since the control system of SXFEL relys on EPICS [5], all the raw data streaming come up to the high-level applications are from the lower EPICS levels. PyEpics is used to be the interface between the lower-level and high-level [6]. The hdf5 self-explantory scientific data format is used to be as the standard format throughout the whole high-level applications [7, 8].

Efficient and nice-looking graphical user interface (GUI) is of much great importance to imporve the user experience, here we choose wxPython as the main GUI builder toolkits, which is the python wrapper for C++ GUI class named wxWidget [9]. The fully open-sourced developing environment definitely make it quite flexible and also means could save a lot budget.

SOFTWARE FRAMEWORK OF FELAPPS

The python-based high-level applications for SXFEL is named felapps, the framework is shown in Fig. 2. The general view is divided into two parts, the left grey part is for applications developed for non-python, for example, legacy apps, and the right orange part is the incubated python world. There are apps with general purposes, like imageviewer, dataworkshop, etc., which could be invoked and served as universal task-handler; and the apps that designed for specific goals are basicly physics-related ones, which be used to solve the corresponding physics problem, such as laser-beam interaction, beam lattice matching, etc.; all these apps are behaving as subpackages or submodules in the python interpretation, integrating into felapps package.

Generally speaking, the design philosophy behind felapps is trying to make the complex suff into the packages, but the simply interface to the users, as well as the sustainable, maintainable development procudure. In order to debug and test the software, an intentionally designed EPICS soft IOC is deployed. Version control system git is utilized to manage the source code, and for the ease of team collaborations [10].

* zhangtong@sinap.ac.cn

ISBN 978-3-95450-136-6

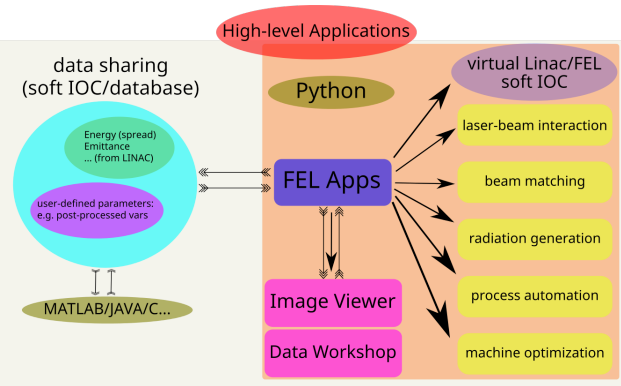

Figure 2: General framework of felapps.

DEVELOPMENT STATUS OF FELAPPS

Trying to make the development extensible and maintainable, Fig. 3 illustrates the project structure. The utils subpackage of felapps actually serves as the infrastructure, providing verious classes/functions, e.g. the parseutils module is responsible for the configuration file parsing tasks, and GUI components & data visulization classes/functions could be found in pltutils module, etc., newly added modules could be classified according to the functions.

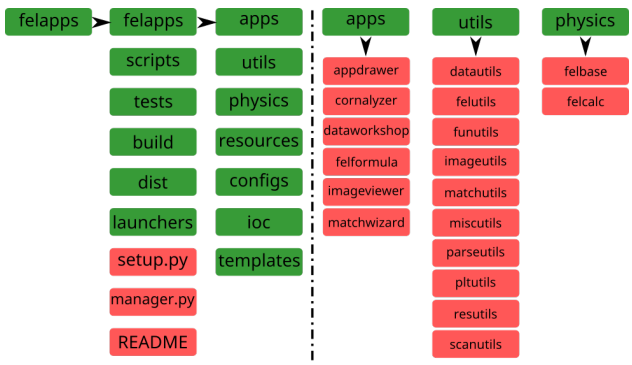

Figure 3: felapps project structure.

Up to now, multiple general purpose apps have built for the felapps package, alphabetically, they are appdrawer, cornalyzer, dataworkshop, felformula, imageviewer and matchwizard, please note that every app could be further feature-improved.

Appdrawer is also aliased as runfelpps, this is the main portal to the apps that felapps package contains. Figure 4 shows the graphical interface when appdrawer is invoked, from which other apps could be called by pushing the corresponding button, the alphabet list could be extended when new apps are created.

Imageviewer is a general purpose designed application for data or image acquisition (Fig. 5), rich properties could be user-defined in the Configurations menu item, once the input EPCIS PV name is connected, data acquisition routine could be started by clicking DAQ Start button, the acquisition frequency could also be configured, data and image could be saved, advanced saving function could be issued through

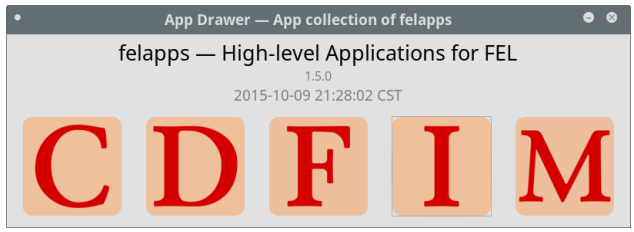

Figure 4: Subpackage of felapps: appdrawer.

Auto Save menu item in Operations menu, into which, saving data format e.g. hdf5, saving path and saving frequency could be defined. The saved data could be post-processed by Dataworkshop, which is a general purpose data post-processor for felapps.

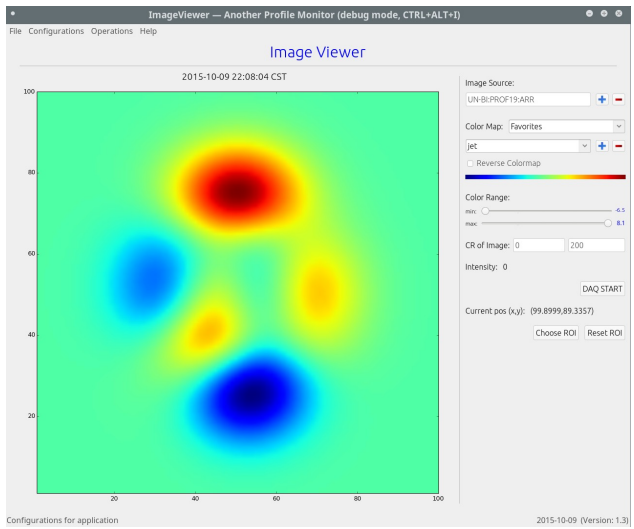

Figure 5: Subpackage of felapps: imageviewer.

After data files are imported into Dataworkshop (Fig. 6), the corresponding images are shown on the right image grid panel, sophisticated functions could be applied to the loaded images/data to get processed informations.

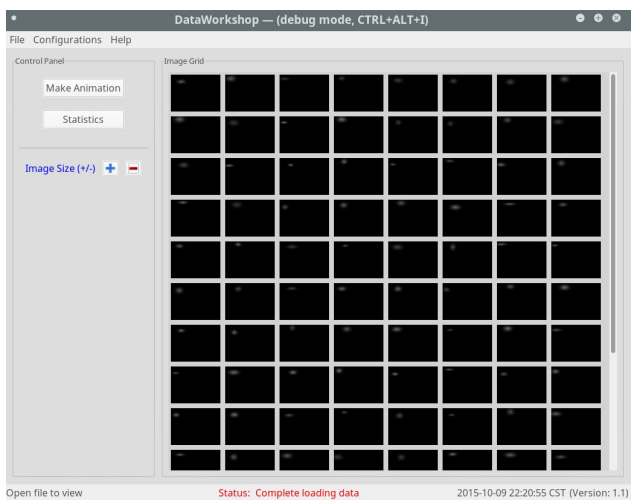

Figure 6: Subpackage of felapps: dataworkshop.

ISBN 978-3-95450-136-6

Felformula app is designed to handle FEL calculations (Fig. 7), after the user completes filling the electron beam parameters, felformula shows the primary properties of FEL radiations, parameter scan could be enabled when optimization is demanding.

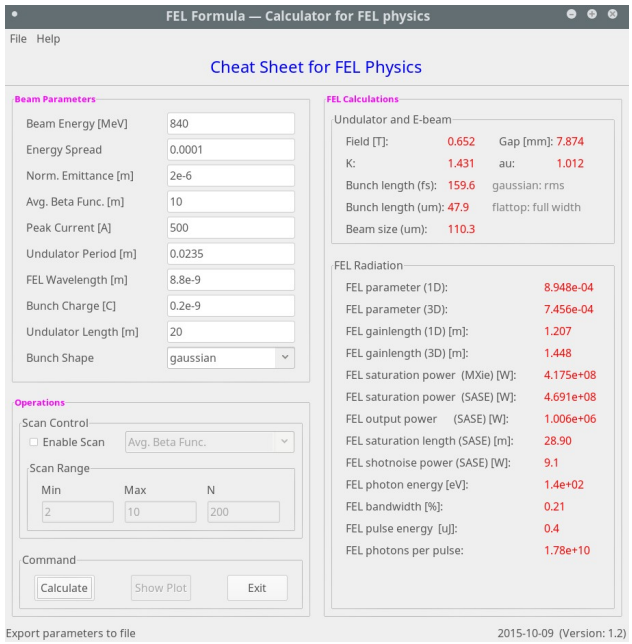

Figure 7: Subpackage of felapps: felformula.

The parameter correlation and scan feature could be handled by app cornalyzer, which is brief for correlation analyzer, also designed with a flexible configurable manner (Fig. 8).

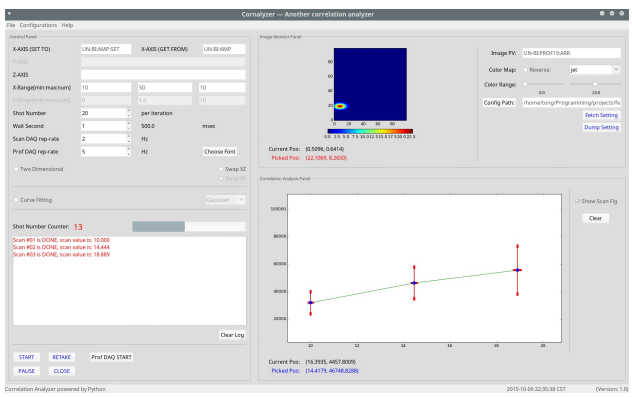

Figure 8: Subpackage of felapps: cornalyzer.

Matchwizard is the app for handling the lattice matching tasks, however it only has the ability to do lattice visulization, which incorporates the python package — beamline to parse

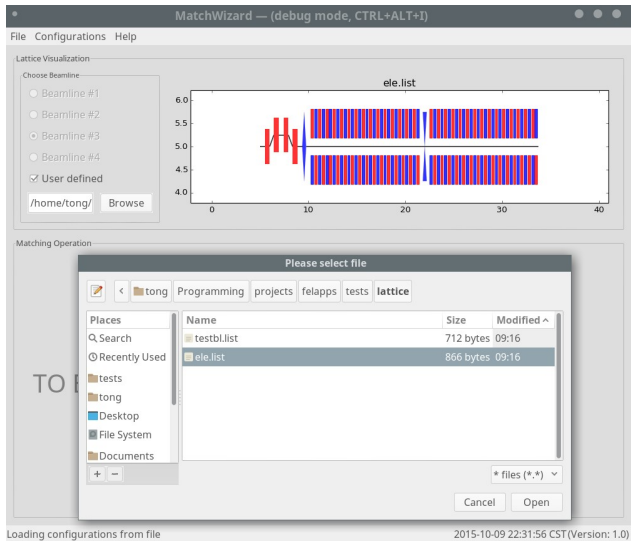

Figure 9: Subpackage of felapps: matchwizard.

and visualize the lattice file, e.g. MAD format lattice, see Fig. 9.

It is significant to note that the initial design consideration of felapps is trying to make the whole software easy to develop and maintain, as well as to delopment. Thanks to the great python module: setuptool, the deployment is really easy and clear now. We can build the .whl package to directly dictribute felapps, or upload onto PyPI web server [11] and deploy onto other system by issing 'pip install felapps'.

CONCLUSIONS

In general, the python-based high-level applications for SXFEL are under development, whole picture is on the way to build an fully-featured aigle python ecosystem, and it is envisioned that verious applications should be benefited from such kind of python-based infrastructure.

REFERENCES

[1] B. Liu et al., "Status of SXFEL and DCLS," FEL'15, Daejeon, August 2015, WEA02, (2015); http://www.JACoW.org
[2] https://docs.python.org/
[3] http://docs.scipy.org/doc/
[4] http://matplotlib.org/
[5] http://www.aps.anl.gov/epics/tech-talk/
[6] http://cars9.uchicago.edu/software/python/pyepics/
[7] https://www.hdfgroup.org/HDF5/
[8] http://docs.h5py.org/en/latest/
[9] http://www.wxpython.org/Phoenix/docs/html/main.html
[10] http://www.git-scm.com/
[11] https://pypi.python.org/

ISBN 978-3-95450-136-6