LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-SR-609815

heXRD: Modular, Open Source Software for the Analysis of High Energy X-Ray Diffraction Data

D. E. Boyce, J. V. Bernier

January 9, 2013

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Final Report

Progress

During the contract period we had regular face-to-face meetings and worked with the contract PI, continually updating project priorities as the project proceeded. The project focus shifted primarily to the user interface intending to expand the user base. Notable accomplishments include:

- release of an open source version of hexrd on github
- presentation of the hexrd GUI at the high energy X-ray workshop at APS in October, 2011

GUI:
The GUI was expanded and now includes pages for

- material: crystallographic parameters,
- readers: parameters for GE deter
- detector: interactive detector calibration
- spots: setting parameters for spot finding
- indexing: grain indexing parameters
- grains: more detailed grain data
- graphics canvas: generic canvas for viewing detector images with various overlays

The pages for the materials, readers, detector and graphic canvas are mature. Pages for spots and indexing are relatively new; they have a basic functionality but need refinement. The grain details page is laid out (see *grain-panel* branch on github) but is not fully integrated.

Core
In terms of the underlying python core code base, improvements were made. The *experiment* module was developed as a focal point for the user interface. The intention was to make the hexrd package API accessible through this module. It would be source for scripted data processing as well as the primary callback for GUI development.

An essential performance issue was resolved during one of our meetings at APS. The mapping between pixel coordinates and polar coordinates had been using far too memory, so much that the application would crash on laptops and machines without large memory. This was resolved by breaking the calculation into smaller

components that could be performed independently, thus limiting the memory in use at any given time.

The sglite package was interfaced with hexrd. The sglite package provides information about space groups. The package had an existing, but old, python interface, which needed some minor updates. Another python module, *spacegroup*, was added to wrap the capabilities in a form appropriate for hexrd.

Documentation

We have set up a sphinx documentation system (see *start-doc* branch on github). Currently, the system is set up to automatically document the API of the Experiment class of the *experiment* module, which was intended to be the primary interface module.

Recommendations

Most of the recommendations are directed toward software management with the goal of gaining a larger user base for HEXRD. Since HEXRD is now publicly available, priorities include establishing and documenting an API (application programming interface) and making it easy for new users to get started.

Standardization

The first recommendation is to standardize the source code according to the python style guide (see Links section). The guide documents the conventions for source code and docstring style. The main advantage of following the style guide is to provide consistency both internally and with standard python packages. Additionally, software that processes the code to format or to produce documentation may expect these guidelines to be followed; for example, naming conventions are used to determine which variables and methods are considered public. The one exception to the style guide is that we recommend the use of relative path names instead of absolute paths for intra-package references. We believe that relative path names will be easier to use if subpackages need to be moved within a package.

Documentation

For python, the API of a module specifies the publicly accessible classes, functions and data. Sphinx is currently the standard documentation system for python. It can be used to automatically generate the API documentation for modules, but it can also process manually generated files. For HEXRD, we recommend:

- Follow the python style guide for naming public entities.
- Use Sphinx as the primary documentation tool.
- Manually document the *experiment* module as the primary API.
- Document the lower level modules as they are modified.

Testing

There needs to be standard tests and examples available. A new user needs to have a sample problem that he/she can run. Developers need existing tests to confirm that their changes do not break existing code, and they need to contribute new tests along with new features. Python provides tools for developing, managing and documenting tests.

- *unittest*: a standard python module for organizing and automating test code
- *doctest*: a standard python module for testing and documenting interactive tests

In addition, longer examples need to be provided for HEXRD since many of the capabilities act on large data sets in potentially complex sequences of actions. New users need simple cases to get started. Developers need examples that exercise all code capabilities.

Data Format

Because some postprocessing operations can take a long time, we need to be able to store the state of processing so that we can resume work later. Currently, HEXRD primarily uses the python pickle module to serialize certain data structures and write them to files. The advantage is that it is easy to program and works reasonably well. The disadvantage is that if the underlying modules and classes change too much, the pickled data files can become incompatible and need to be modified.

Standard data formats need to be developed and documented. The *numpy* module provides tools for numerical and image data. The

python module *ConfigParser* handles configuration style files. These are useful for text-based data formats and for organization.

Other Recommendations

GUI: Because there may be issues moving forward with wxPython as the GUI interface, we recommend porting the GUI capabilities to PyQt.

Distribution: Use PyPI for distribution.

Performance: Use cython for bottlenecks when possible.

Links

Python style guide	Describes conventions for python source code and docstrings
PyPI	Python Package Index, a repository for python software
PyQt	Python bindings for Qt application framework
cython	Interfacing C with python