PyBluIce: Modular Data Acquisition Software for Long Duration, Scripted Automation, and High-Speed Protein Crystallography Experiments

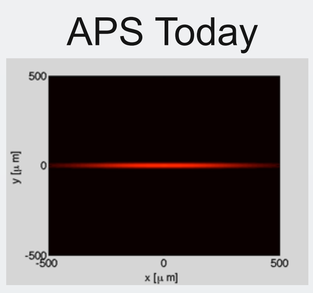
Mark C. Hilgart[1], Qingping Xu[1], Oleg Makarov[1], Sergey Stepanov[1], Michael Becker[1], Nagarajan Venugopalan[1], Craig Ogata[1], David Kissick[1], Robert F. Fischetti[1]

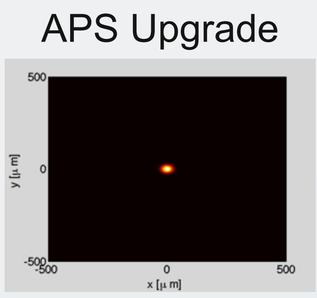
[1]GM/CA@APS, Argonne National Laboratory, Argonne, IL

Requirements

The PyBluIce project will make new methods for protein crystallography widely accessible to GM/CA users.

The software changes are designed to adapt to the APS upgrade, new GM/CA optics, fast detectors, high capacity automounters and serial crystallography methods.



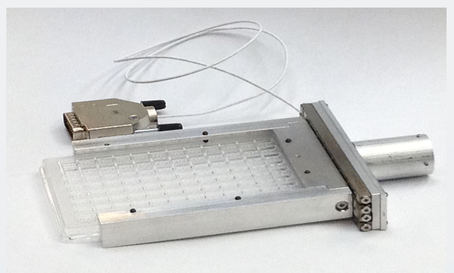


Microfocus Beam

The APS upgrade will deliver up to a 500x brighter beam. Along with GM/CA optics upgrades including new mirrors and compound refractive lenses (CRLs), this will produce an intense microfocus beam at the sample.

HDRMX

Faster detectors with streaming and multi-image files require software changes to reach their full potential.



Serial Crystallography

Fixed-target sample mounts such as the one above are optimally used with new tools to plan data collection.

Design

Modularity is the primary design goal for PyBluIce. Functions use the HTTP API and Redis to communicate with the rest of the application instead of traditional in-process memory access and method calls.

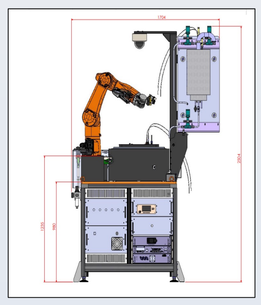
This allows single-use scripts and the well-refined UI to use the same API. It enables new types of complexity and automation. And it will make replacing core components in the future manageable.

```
import api.redis_api as ra
import api.http_api as ha

ra.run_b.delta_deg.set(0.2, id=1)
ha.collect_runs_py(runs=1).post()
```

All-Level API

From motor moves to full dewar data collection, the API is the complete and only way to access functionality.



HTTP Queue

A queue of HTTP commands allows for future automation of any functionality including vector and raster.

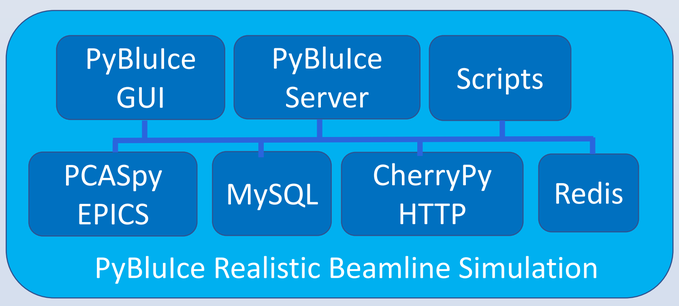
Data Processing

Streaming will improve local near-realtime processing, and supercomputer time will improve large compute task performance.

Implementation

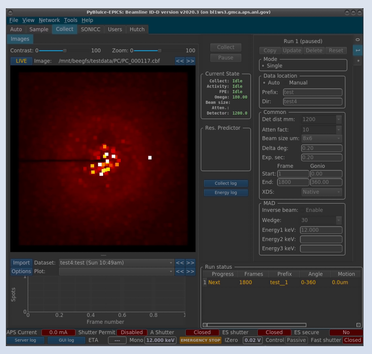
PyBluIce was initially implemented in 16 months using minimal access to real hardware.

New tools for automation and data collection are being developed alongside the initial port, and will be released in future updates.



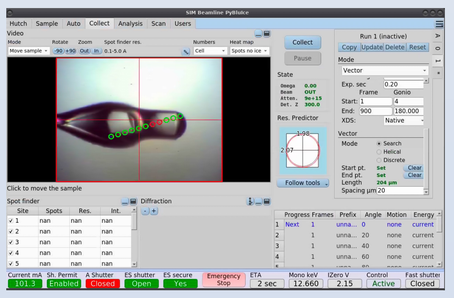
Offline Development

Real databases and servers are scripted for PyBluIce to connect to during offline development.



Initial UI

Shown here is dark mode, a new color scheme for PyBluIce.



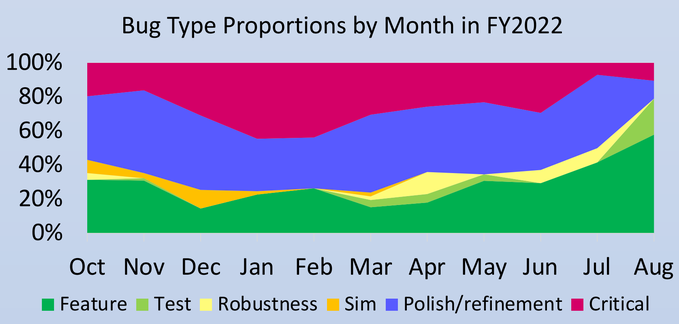
Revised UI

Crystallographer feedback revised the design to minimize training requirements for existing users.

Testing and Refinement

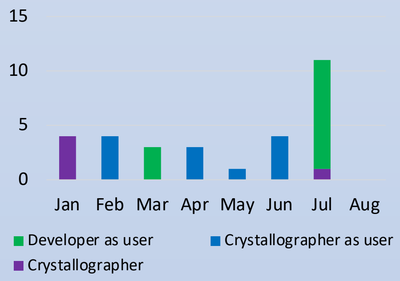
Crystallographer and developer testing progressed from simulation, to without beam, to with beam and crystals.

Initial testing involved studying the program, then later testing was focused on studying samples with the new program.



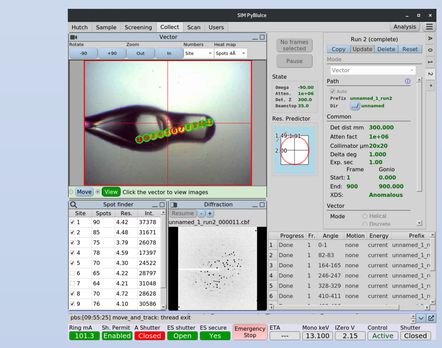
Zero Critical Bugs in August

Through sustained testing, the critical bug count peaked in January and has now gone to zero. Minor fixes have also greatly diminished.



Sustained Testing

Testing on the beamline has continued throughout 2022.



Refined UI

The last 12 months of refinement resulted in 586 bugs fixed and a program ready for real-world use.

Argonne NATIONAL LABORATORY