Background

- LUME-EPICS and LUME-Model are members of the LUME project
- LUME project aims to wrap standard, developed electron/photon simulation codes with a common Python interface
- Surrogate models fall under this purview: ML surrogates trained on inputs and outputs of physics simulations lead to fast executing models, which may be used for tuning etc.
- NEED: Integration of surrogate models with the control system- execution on live variables, ability to surface outputs

LUME-EPICS (and LUME-model) Overview

**LUME-Epics**
- EPICS server (default both Channel Access and pvAccess, but configurable)
- Callbacks on input process variable update
- EPICS-based bokeh widgets for interface development
- Templated generation of displays

**LUME-Model**
- Base classes for guiding standardized development of surrogate model execution classes
- Variables classes with attribute type validation to enforce minimum data requirements

Distributed on conda-forge

**Tech Stack**
- Python >= 3.7
- bokeh
- p4p
- NumPy
- pyyaml
- pydantic
- pyepics

Distributed on conda-forge
Components

**Design features:**
- Separable server and client tools
- Compatibility enforced by LUME-Model variable validation
- Extensible SurrogateModel execution class for high levels of customizability
- Templated class for Keras models

**LUME-EPICS server**
- Input and output variables built with LUME-Model
- Configurable protocol (pvAccess/Channel Access)

**LUME-EPICS client**
- Bokeh widget tools initialized with LUME-Model variables
- Tiered controls:
  - Controller: Access process variables
  - Monitor: Formats data
  - Widgets: Development objects
LUME-EPICS Application Structure

- Synchronization of process variables between pvAccess and Channel Access processes
- Scalable with respect to number of variables
- Variable store for slow executing models

- Variable type based monitors for continued widget development
- Some bokeh widgets adapted for easy integration
Applications: Neural network surrogate models

- Packaged neural network surrogate model of the LCLS cu injector and served using LUME-EPICS toolkit (credit: Lipi Gupta)

Rendered in a Jupyter Notebook:

control sliders

Image plot widget

Striptools
Applications: Neural network surrogate models

Served bokeh dashboard with controls:

LCLS Cu Injector

Available [here](#)
Applications: Bmad model execution with PyTao

LCLS copper HXR beamline model dashboard rendered locally with bridge to live accelerator PVs:

Array plot widget

Value table monitoring live model inputs
Development roadmap

- Very much in Beta development
- Immediate goals:
  - User acquisition
  - Varied applications
  - Stress test of documentation, etc.
Learn more

- LUME: https://www.lume.science/
- Surrogate model of the cu injector (Auralee Edelen): https://www.youtube.com/watch?v=1f42uRNfx18
- Dockerized LCLS cu injector model, served with Binder: (here)
- LUME-Model documentation: (https://slaclab.github.io/lume-model/)
- LUME-EPICS documentation: (https://slaclab.github.io/lume-epics/)
Questions/Comments?