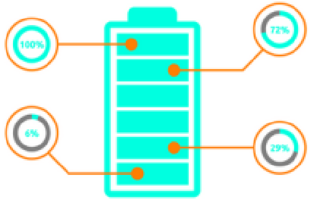
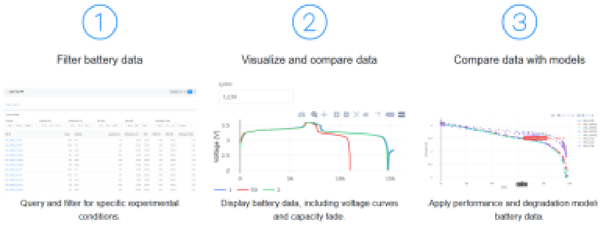
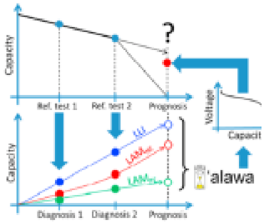
SAND2022-6109C

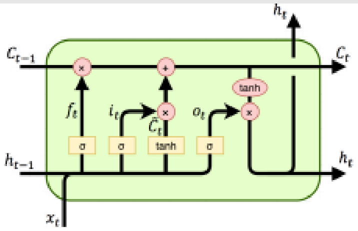
Sandia National Laboratories

Analyzing and Manipulating Data with the PyData Stack









PRESENTED BY

Gabe Hege and Valerio De Angelis

Tutorial EN05—Introduction to Data Science for Battery Degradation

2022 MRS Spring Meeting & Exhibit, May 8, 2022

CORE PROJECT MEMBERS

Yuliya Preger (SNL), Ammar Safdari (U Michigan), and Irving Derin (Form Labs).

U.S. DEPARTMENT OF ENERGY NNSA

SAND2022-XXXX PE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Growing set of open-source tools in Python

- PyData tools to import and manipulate battery data
  - NumPy: Optimized mathematical functions
  - Pandas: Data analysis and manipulations tool
  - Matplotlib: easy plotting
  - Plotly: plotting library with finer controls
  - Jupyter Notebook (JN): Present data and analysis in annotated notebook

- Web tools to visualize, compare and share data
  - Redash: Open source web app to visualize data from any source
  - Battery Lifecycle Framework: Battery-specific implementation of Redash and Data importers

- Data tools to consolidate and exchange data
  - JSON data exchange
  - SQL to extract data

- Approaches to compare experimental and model results (more details in the other sessions)
  - Degradation simulation packages: PyBaMM
  - Synthetic data
  - Machine learning tools

Outline

- Example 1: Simple manipulations
  - Upload Arbin data
  - Visualize key cell data
  - Export the data and generate additional statistics
  - Run degradation simulations with PyBaMM

- Example 2: Compare with synthetic data
  - Why synthetic data
  - Start from data in the dashboard
  - Match cell degradation with synthetic data

- Example 3: Use 3rd party models and data (if time)
  - Dataset description: Abuse data
  - Framework description
  - Load and use a machine learning model

Materials and environment

- All the data and materials used in the examples
  - https://github.com/battery-data-commons/mrs-sp22-tutorial
- Battery Life Cycle framework: Open-source packages used in the tutorial
  - https://github.com/battery-lcf
- AmpLabs: Example of the Battery Life Cycle framework in a hosted environment
  - https://tinyurl.com/mrs-amplab

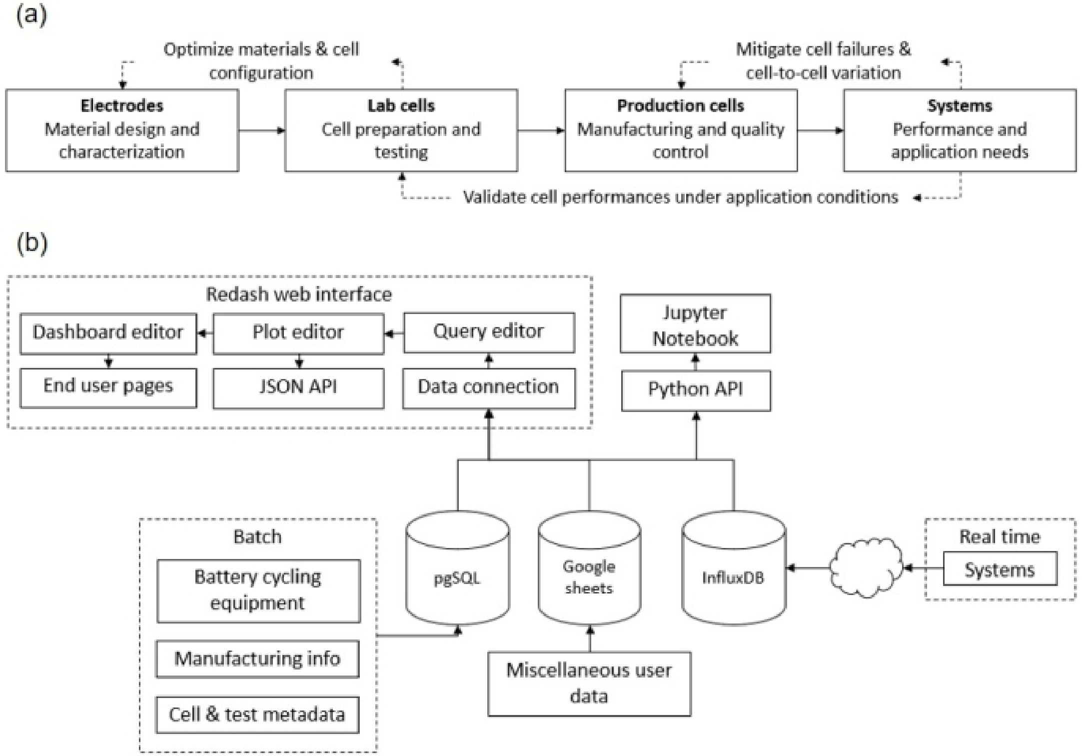
The slides provide background on the case studies and packages that will be used in the tutorial Python examples.

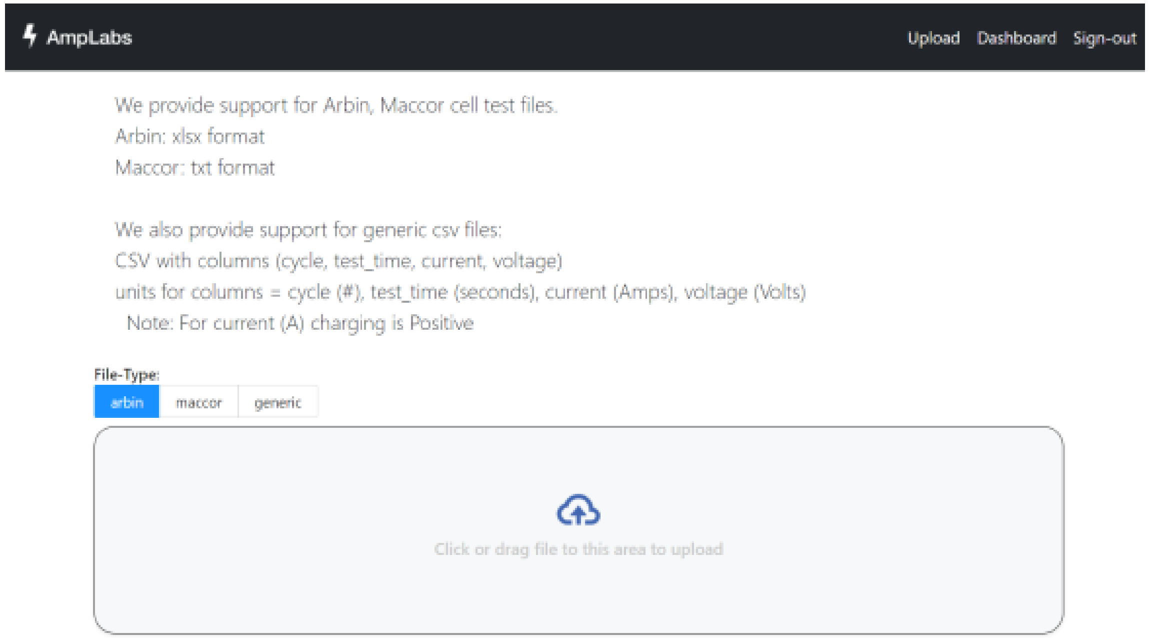
The full details of implementation are provided in the Jupyter Notebook

Example 1: Import and visualize the data

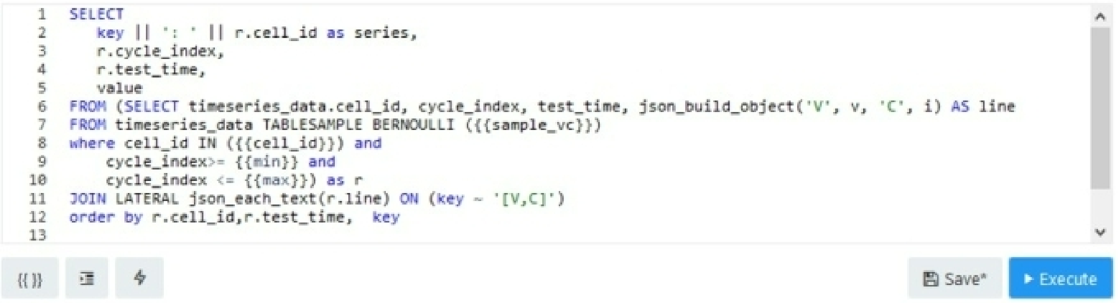
The Battery Lifecycle (BLC) Framework is an open-source platform that provides tools to visualize, analyze, and share battery data through the technology development cycle, including data from material characterization, cell testing, manufacturing, and field testing.

https://ecsarxiv.org/h7c24/
https://github.com/battery-lcf





Hosted example of the BLC Framework

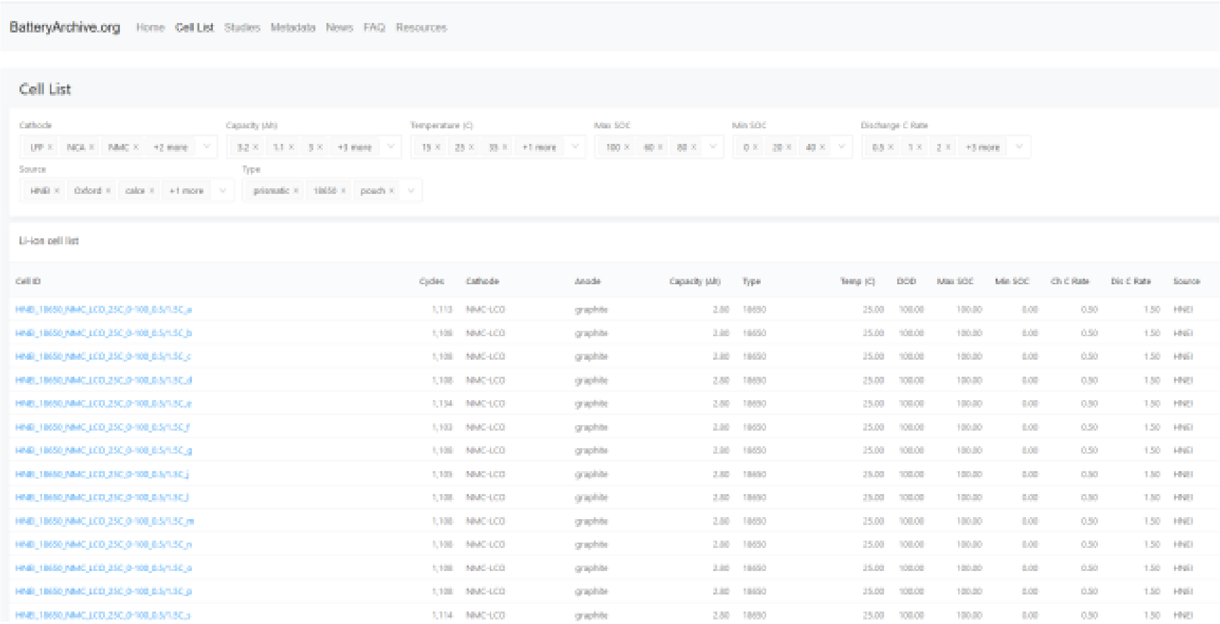


Data can be retrieved and organized using SQL

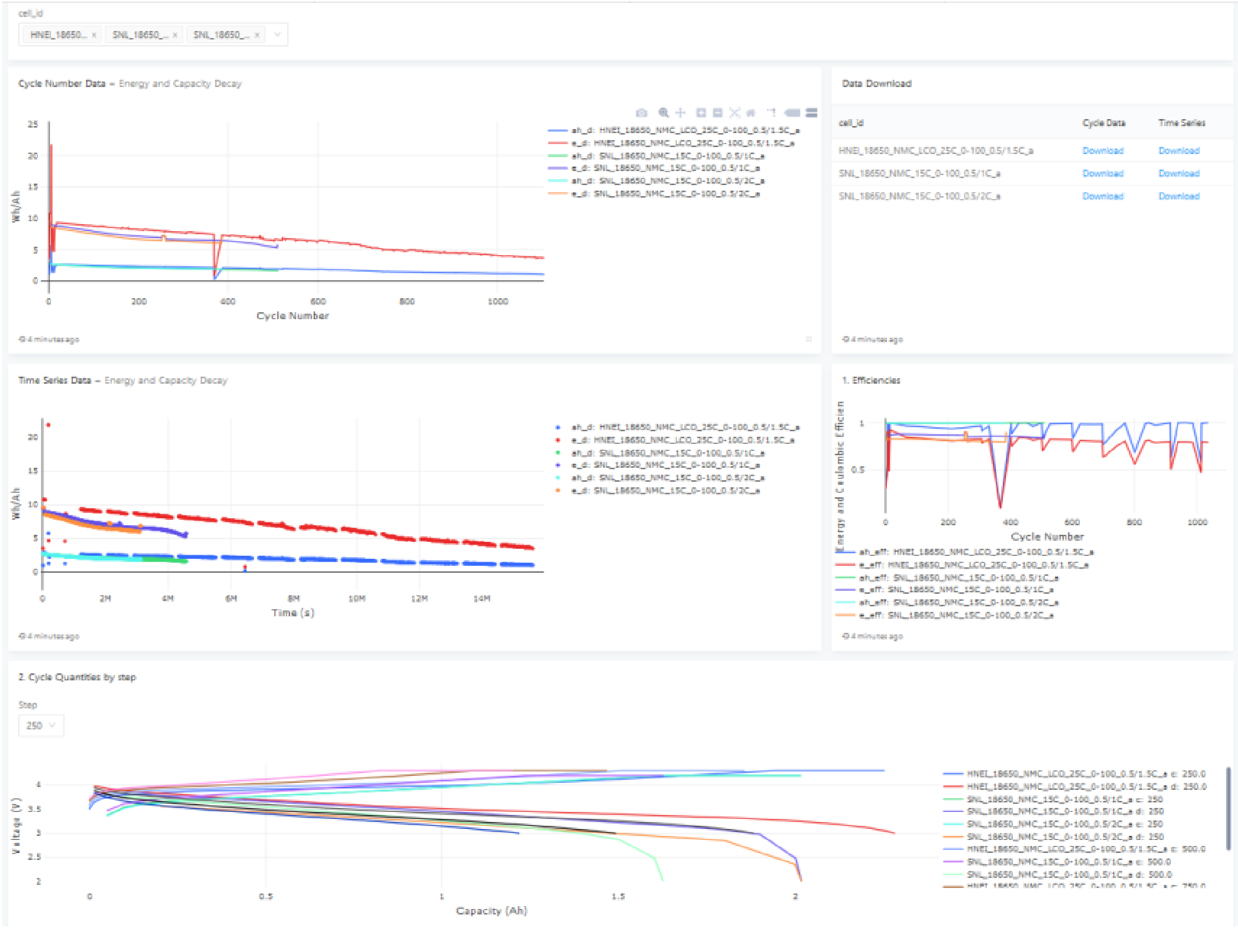
Example 1: Public data on batteryarchive.org

Another example application of the BLC Framework is Battery Archive, a public repository of large datasets of battery data.

https://www.batteryarchive.org



Cell and test metadata are displayed in a table format

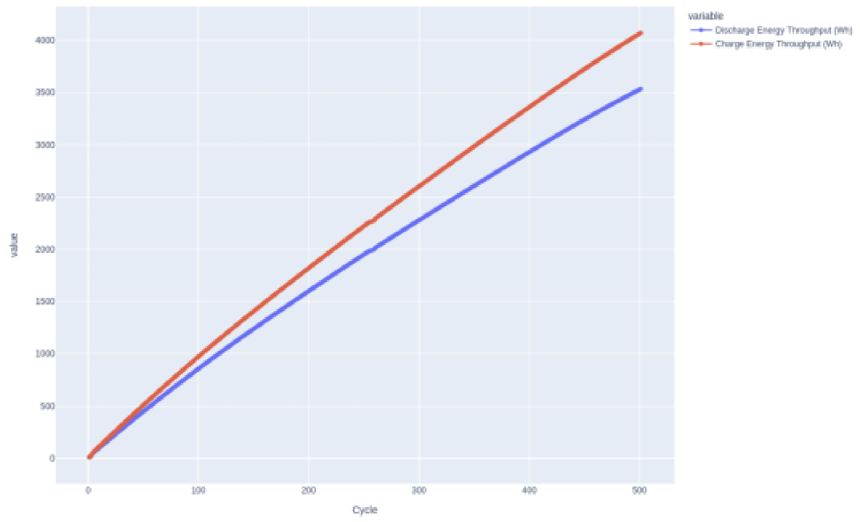


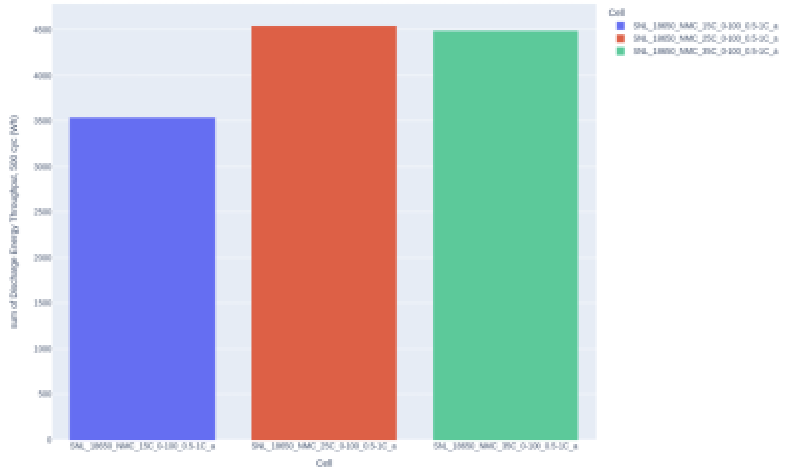
Standard battery plots generated automatically

Example 1.a: Download and use the data

The snippet below shows how to retrieve data from AmpLabs and manipulate it in a Jupyter Notebook to generate new graphs.

```python
def get_amplabs_cycledata(user, cell_id):
        cycle_data_url = 'https://www.amplabs.ai/download/cells/cycle_data_json/'
        url = "{}{}".format(cycle_data_url,cell_id)
        httprequest = urllib.request.Request(
            url, method="GET"
        )
        httprequest.add_header("Cookie", "userId={}".format(user))
        try:
            with urllib.request.urlopen(httprequest) as httpresponse:
              response = json.loads(httpresponse.read())
              return response, 1
        except urllib.error.HTTPError as e:
            print(e)
        return None, 0
```





Example 1.a: Jupyter Notebook live demo

Example 1.b: PyBaMM

PyBaMM (Python Battery Mathematical Modelling) is an open source Python library provided by the Faraday Institution that can solve physics-based electrochemical DAE models by using state-of-the-art automatic differentiation and numerical solvers (DFN/SPM). The software was developed by Valentin Sulzer (now at U Chicago) https://github.com/pybamm-team/PyBaMM

Sulzer, V., Marquis, S. G., Timms, R., Robinson, M., & Chapman, S. J. (2021). Python Battery Mathematical Modelling (PyBaMM). Journal of Open Research Software, 9(1).

PyBaMM can be set up to run cycling protocols as shown in the Jupyter Notebook example or application profiles as shown in the Figure. In the tutorial, simulation and experimental results will be compared.

Tutorial 5 - Run experiments

In Tutorial 4 we saw how to change the parameters, including the applied current. However, in some cases we might want to prescribe a given voltage, a given power or switch between different conditions to simulate experimental setups. We can use the Experiment class for these simulations.

```
[ ] %pip install pybamm -q    # install PyBaMM if it is not installed
    import pybamm

    Note: you may need to restart the kernel to use updated packages.
```

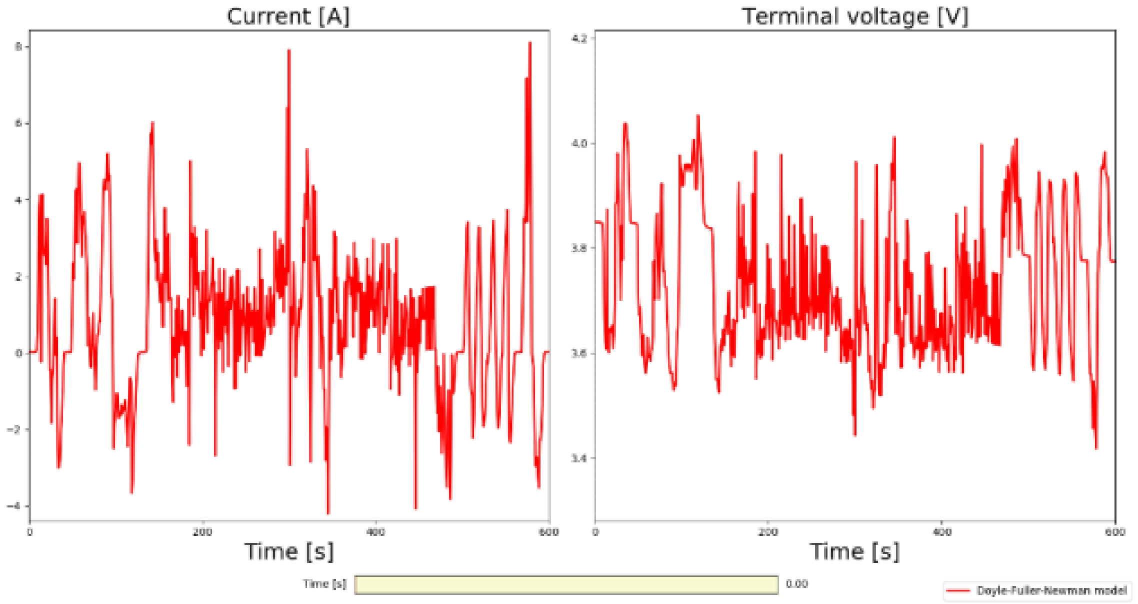
We start defining an experiment, which consists on a set of instructions on how to cycle the battery. For example, we can set the following experiment:

```
[ ] experiment = pybamm.Experiment(
        [
            ("Discharge at C/10 for 10 hours or until 3.3 V",
            "Rest for 1 hour",
            "Charge at 1 A until 4.1 V",
            "Hold at 4.1 V until 50 mA",
            "Rest for 1 hour"),
        ] * 3
    )
```

A cycle is defined by a tuple of operating instructions. In this case, the experiment consists of a cycle of constant current C/10 discharge, a one hour rest, a constant current (1 A) constant voltage (4.1 V) and another one hour rest, all of it repeated three times (notice the * 3).

Then we can choose our model

```
[ ] model = pybamm.lithium_ion.DFN()
```



Simulation of the DFN model using a drive cycle as input current.

Example 1.b: Jupyter Notebook live demo

Example 2: Degradation paths

Problem:
- Accurate lithium battery diagnosis and prognosis is complex
- Battery degradation is intricate, nonlinear, and path-dependent
- Data-driven models play a significant role but are limited by the amount of training data available

Synthetic datasets can be orders of magnitude larger than experimental datasets and provide significant opportunity for battery degradation analysis.

In the tutorial, we will show how Python tools can be used to combine synthetic and experimental degradation data to evaluate the underlying degradation pathways for the experimental data.

Synthetic Data Reference:
"Big data training data for artificial intelligence-based Li-ion diagnosis and prognosis"
Matthieu Dubarry, David Beck, University of Hawaii, Hawaii Natural Energy Institute, USA
https://doi.org/10.1016/j.jpowsour.2020.228806

MATLAB package:
- https://www.soest.hawaii.edu/HNEI/alawa/

Size of the dataset: 5000 different degradation paths
- https://data.mendeley.com/datasets/bs2j56pn7y/1
- https://data.mendeley.com/datasets/6s6ph9n8zg/1

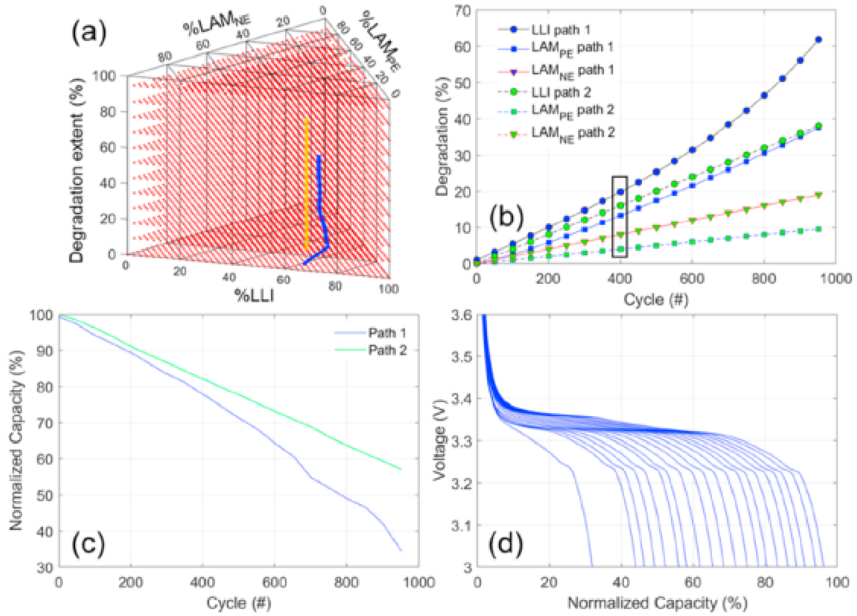
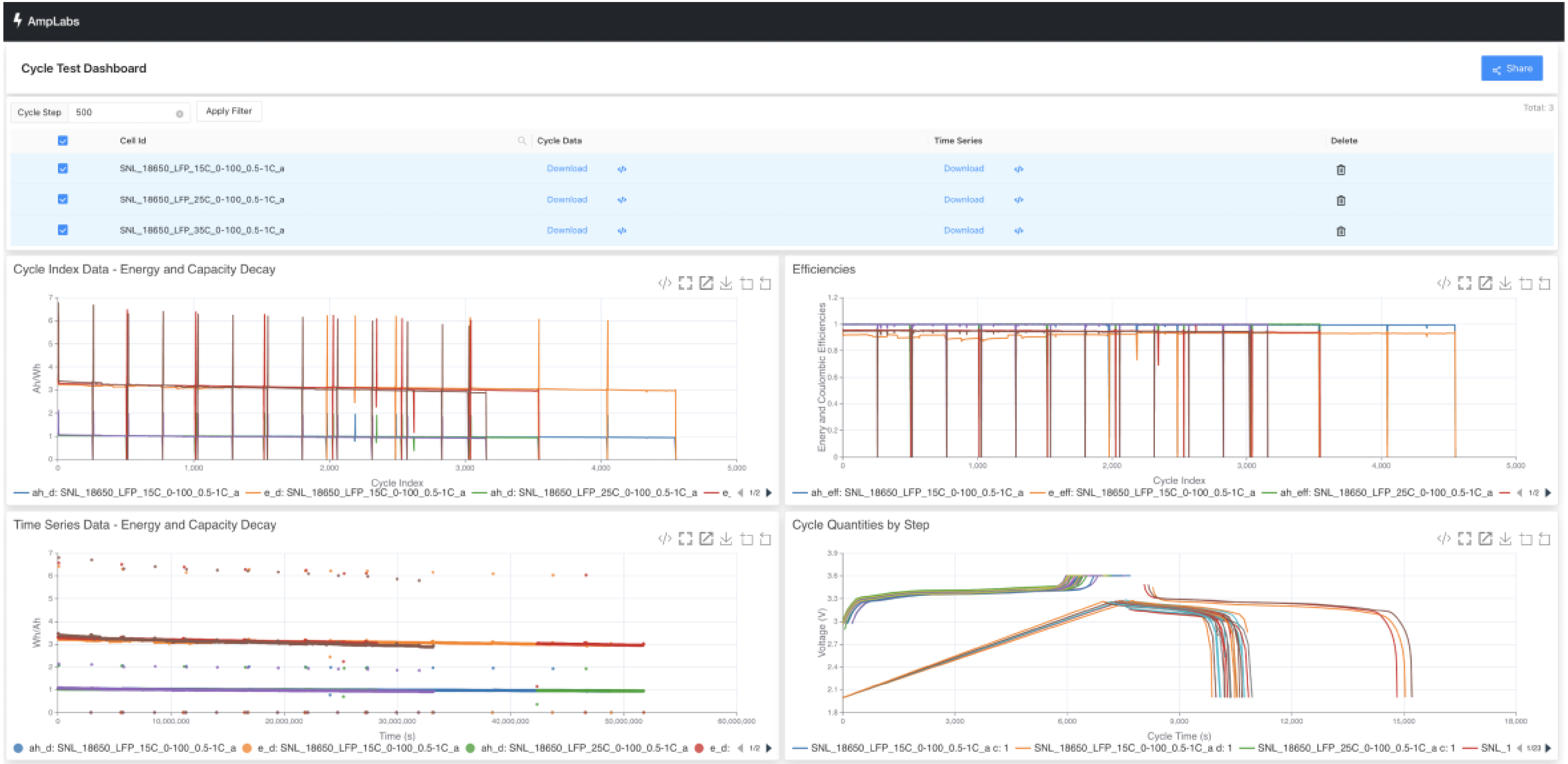


Fig. 4. (a) Mechanistic model principles to emulate every possible degradation with example of the prognosis dataset calculation for two different paths for a Gr// LFP cell with (b) the triplet variations, (c) the reconstructed capacity loss, and (d) the voltage variations for path 1. Only partial degradation calculations were represented in (a) for illustration purposes.

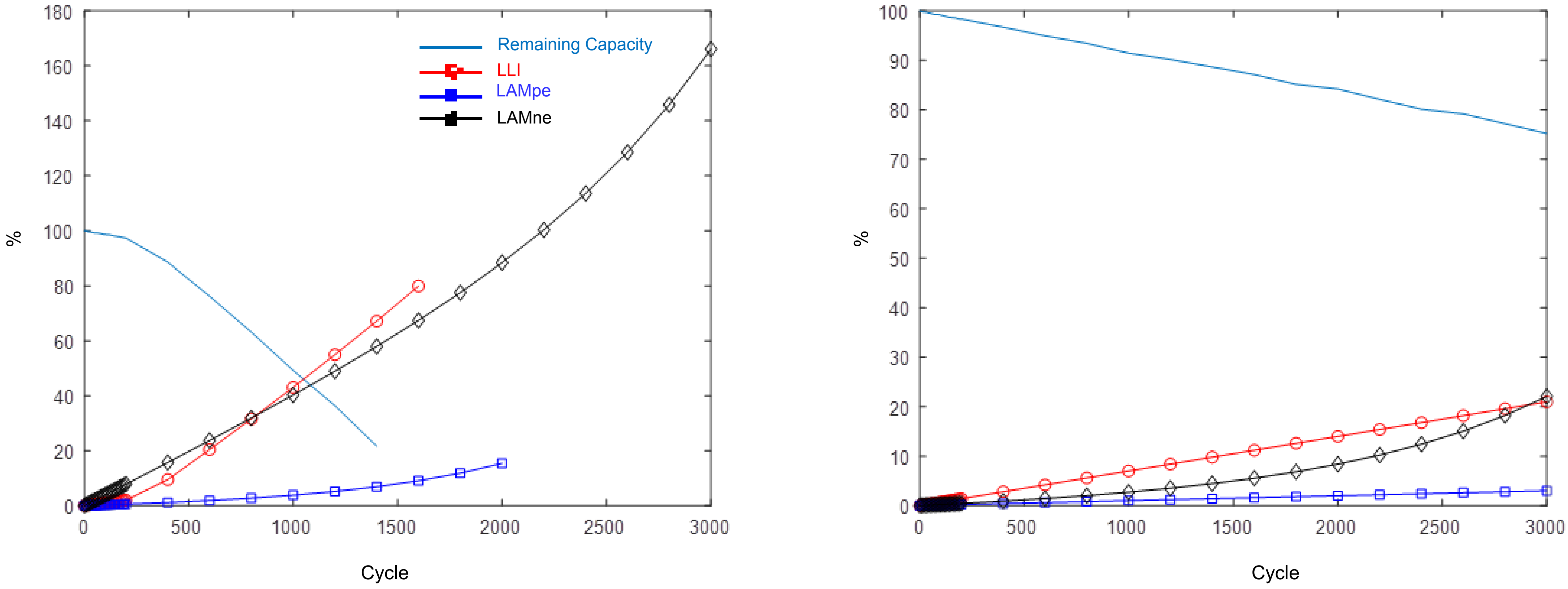
Example 2: Experimental data

Data for LFP experimental cells at 3 temperatures (15, 25, and 35 °C) for comparison with the synthetic data



Example 2: Synthetic data

Capacity fade curves can be presented as different combinations of LLI (loss of lithium inventory), LAMpe (loss of active material, positive electrode), and LAMne (loss of active material, negative electrode).



Example 2: Jupyter Notebook live demo

Example 3: Build compatible tools

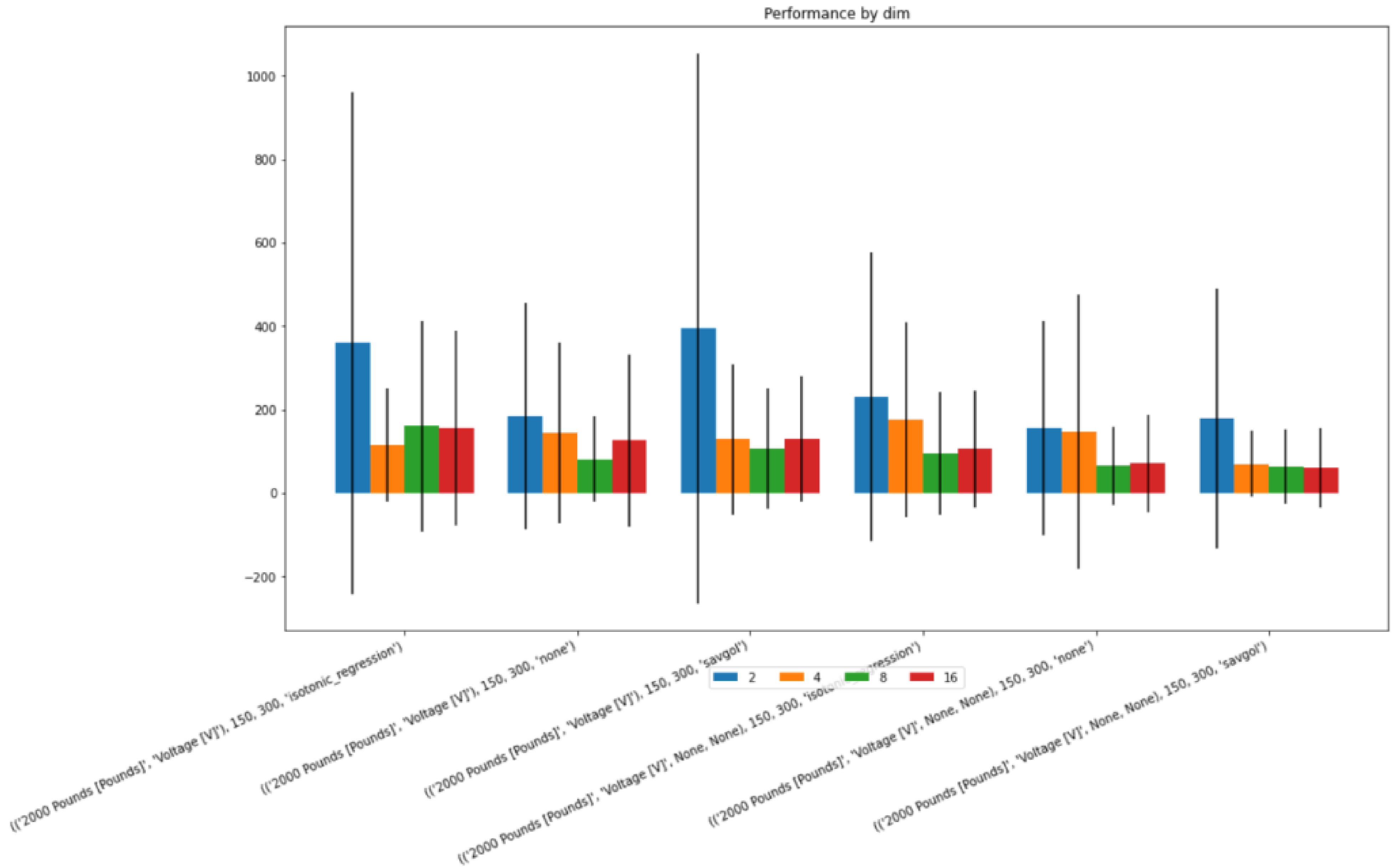
Data on AmpLabs

Models built by ORNL

JN to compare data and model predictions

Example 3: Mechanical abuse data

Example 3: Performance of the ML model



Example 3: Jupyter Notebook live demo

Hands-on session in the afternoon

- Download the required code from GitHub to reproduce any of the 3 examples
  - https://github.com/battery-lcf/mrs-tutorial-05-22
- Bring your data: CSV with time, cycle number, voltage, and current
- Bring requests for stats, graphs, analysis, or anything else

Acknowledgments

Work in collaboration with Matthieu Dubarry (HNEI), Srikanth Allu (ORNL), Irving Derin (Form Labs), Ammar Safdari (U Michigan), Hyungjin Choi (SNL), and Yuliya Preger (SNL).

We would like to thank Loraine Torres-Castro (SNL), Matthieu Dubarry (HNEI), and the MRS for hosting and promoting the tutorial, and Babu Chalamala for his encouragement and support.

To join the open-source projects, please email vdeange@sandia.gov

Partially funded by the U.S. Department of Energy, Office of Electricity, Energy Storage program. Dr. Imre Gyuk, Program Director.

We have job openings if you are interested in contributing to the work