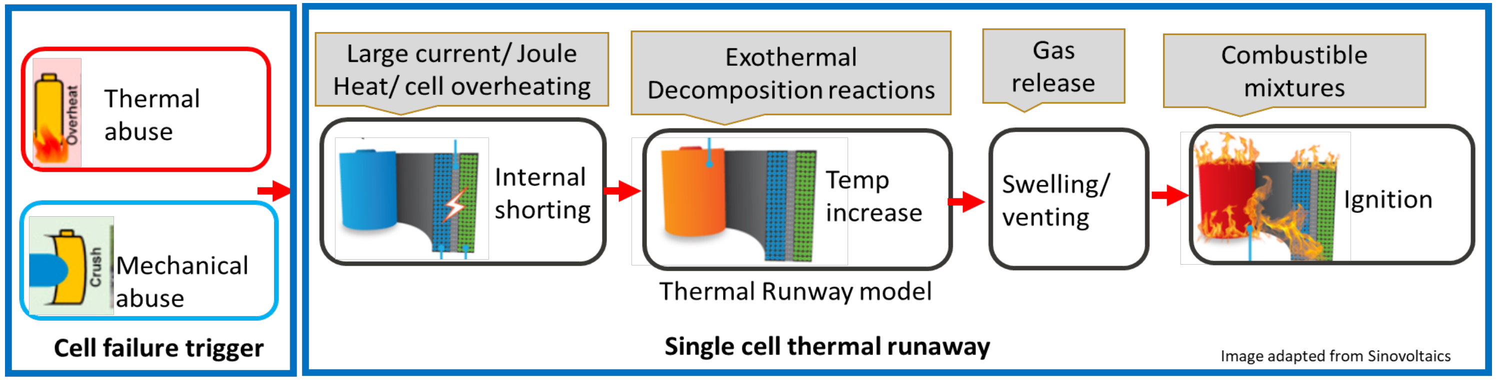
Battery Characterization App using PyDYNA

Srikanth Adya, Kevin Kong, Vidyu Challa, Iñaki Çaldichoury
Pierre L'Eplattenier, Mukul Attri, Dilip Bhalsod, Mike Howard

11/15/2023

Ansys

Battery Cell Thermal Runaway Modeling in LSDYNA

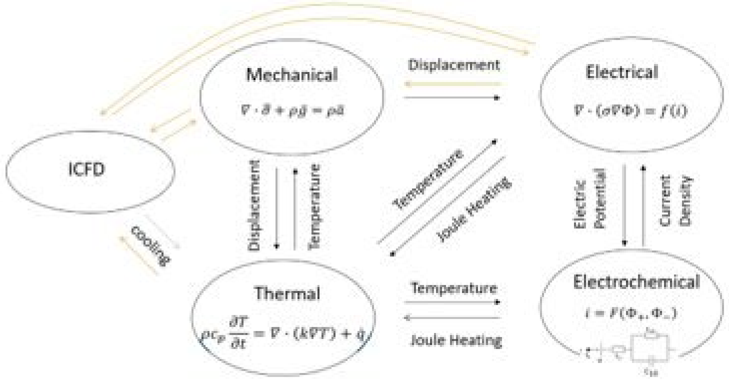


» Electro-thermal behavior represented using Distributed Randles Circuit models

» Mechanical deformations induces internal and external shorts

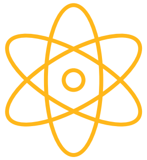
» Current flowing through the short creates joule heating

» Exothermal models for Thermal-Runaway

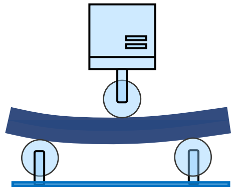


Why Battery Solution?

Automated model setup
Ease of converting HPPC test data to model input
Ease of converting mechanical test to model input

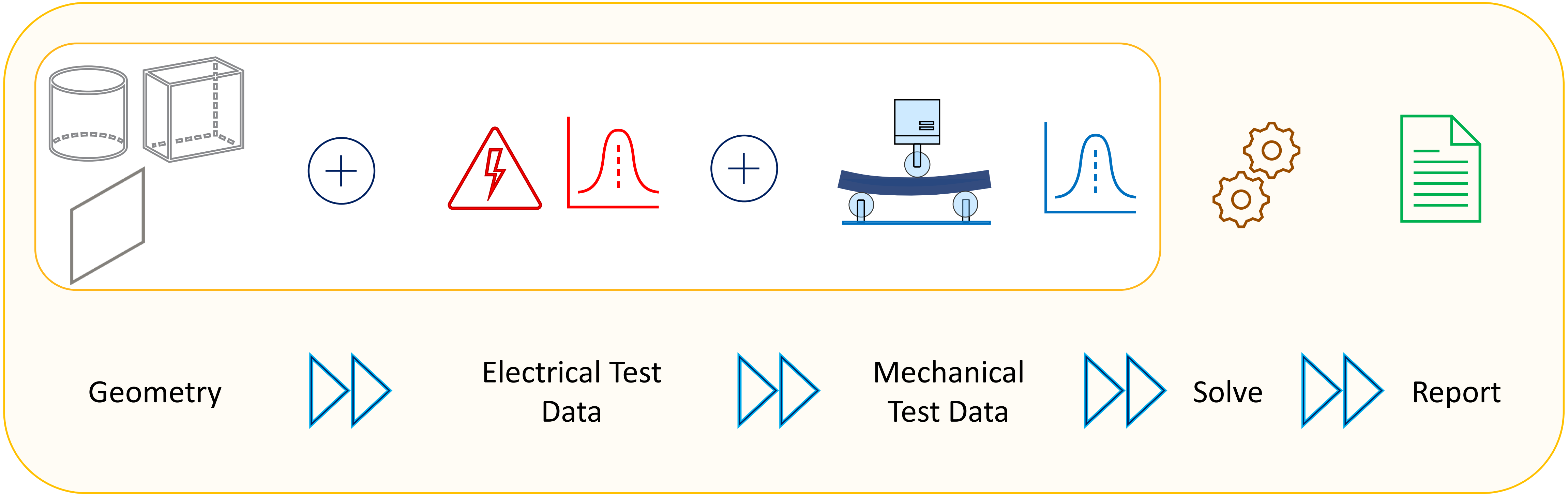


Tutorials on Multiphysics
Example models



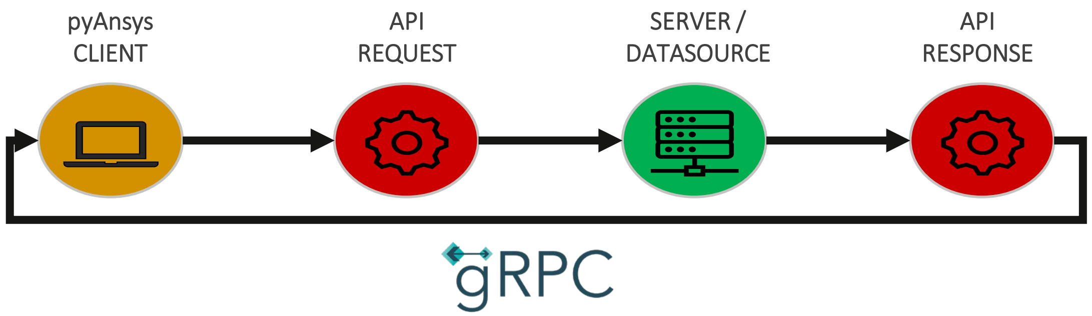
Test matrix suggestion
Generic material models
Stochastic behavior modeling

Our Solution for Single Cell



A fully automated , scripts driven workflow with custom User-Interface

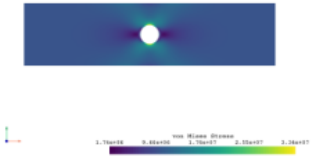
PyANSYS



Set of technologies that allow the user to interface with Ansys products pythonically

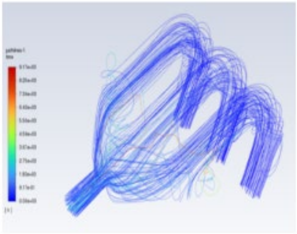
PyANSYS Ecosystem

Simulation librairies



PyDyna

Pythonic interface to build the Ansys DYNA input deck, submit it to the Ansys LS-DYNA solver and postprocess its results

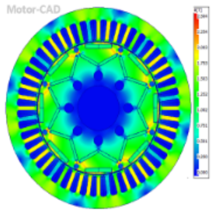


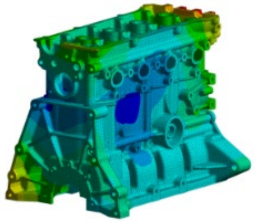
PyMAPDL

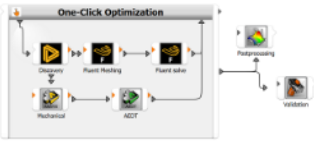
Pythonic interface to Ansys MAPDL (Mechanical APDL)

PyFluent

Pythonic interface to Ansys Fluent







PyMotorCAD

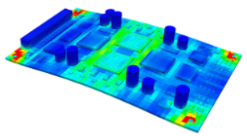
Pythonic interface to Ansys Motor-CAD.

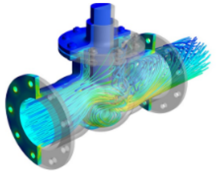
PyMechanical

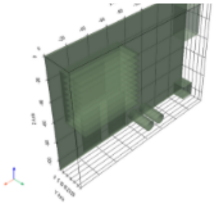
Pythonic interface to Ansys Mechanical

PyOptislang

Pythonic interface to Ansys Optislang







PySherlock

Pythonic interface to communicate with Ansys Sherlock

PySystem Coupling

Pythonic interface to communicate with Ansys System Coupling

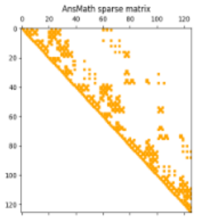
PyAEDT

Pythonic interface to AEDT (Ansys Electronic Desktop)

Utility libraries

PyPrimeMesh

Pythonic interface to Ansys Prime Server, which delivers core Ansys meshing technology



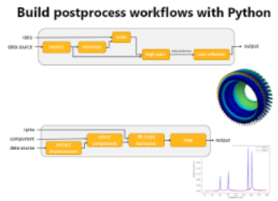
PyDynamicReporting

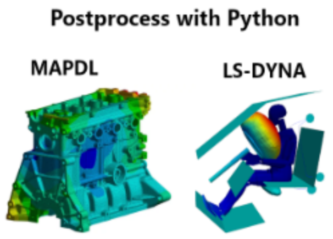
Pythonic interface to Ansys Dynamic Reporting for service and control of its database and reports

PyAnsys Math

Pythonic interface to PyAnsys Math libraries

Postprocessing librairies



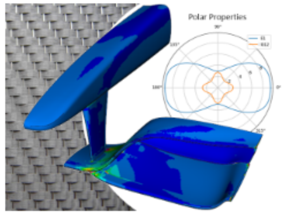


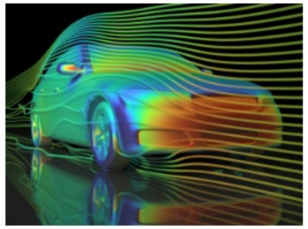
PyDPF - Core

Pythonic interface to DPF (Data Processing Framework) for building more advanced and customized workflows

PyDPF - Post

Pythonic interface to access and post process Ansys solver result files





PyDPF Composites

Pythonic interface to post-process layered and short-fiber composite models

PyEnSight

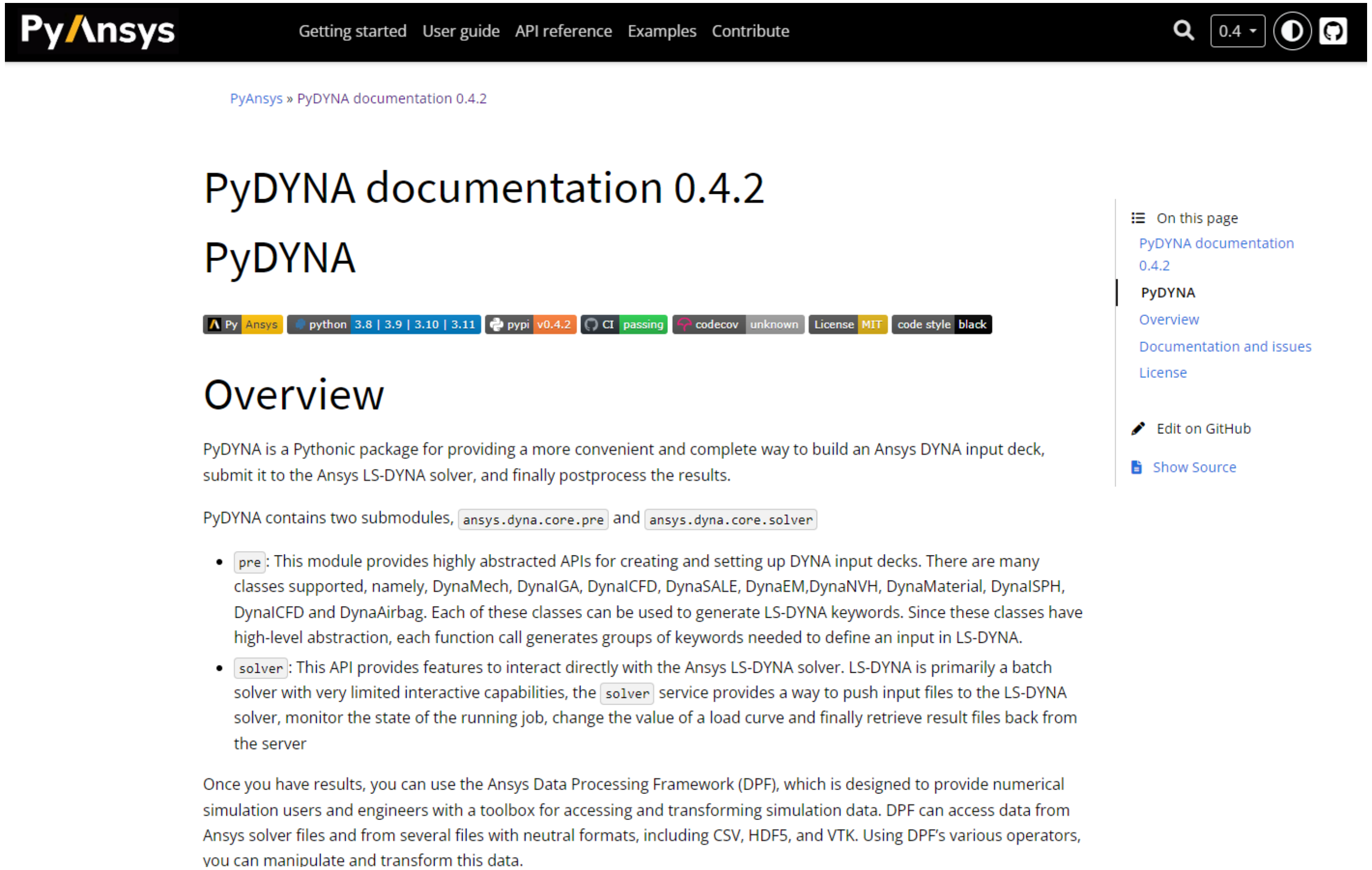
Pythonic interface to EnSight, the Ansys simulation postprocessor

Ansys

PyDYNA

PyDyna

Pythonic interface to build the Ansys DYNA input deck, submit it to the Ansys LS-DYNA solver and postprocess its results



PyAnsys Getting started User guide API reference Examples Contribute 0.4

PyAnsys » PyDYNA documentation 0.4.2

PyDYNA documentation 0.4.2

PyDYNA

Py Ansys | python 3.8 | 3.9 | 3.10 | 3.11 | pypi v0.4.2 | CI passing | codecov unknown | License MIT | code style black

Overview

PyDYNA is a Pythonic package for providing a more convenient and complete way to build an Ansys DYNA input deck, submit it to the Ansys LS-DYNA solver, and finally postprocess the results.

PyDYNA contains two submodules, `ansys.dyna.core.pre` and `ansys.dyna.core.solver`

- `pre`: This module provides highly abstracted APIs for creating and setting up DYNA input decks. There are many classes supported, namely, DynaMech, DynaIGA, DynaICFD, DynaSALE, DynaEM,DynaNVH, DynaMaterial, DynaISPH, DynaICFD and DynaAirbag. Each of these classes can be used to generate LS-DYNA keywords. Since these classes have high-level abstraction, each function call generates groups of keywords needed to define an input in LS-DYNA.
- `solver`: This API provides features to interact directly with the Ansys LS-DYNA solver. LS-DYNA is primarily a batch solver with very limited interactive capabilities, the `solver` service provides a way to push input files to the LS-DYNA solver, monitor the state of the running job, change the value of a load curve and finally retrieve result files back from the server

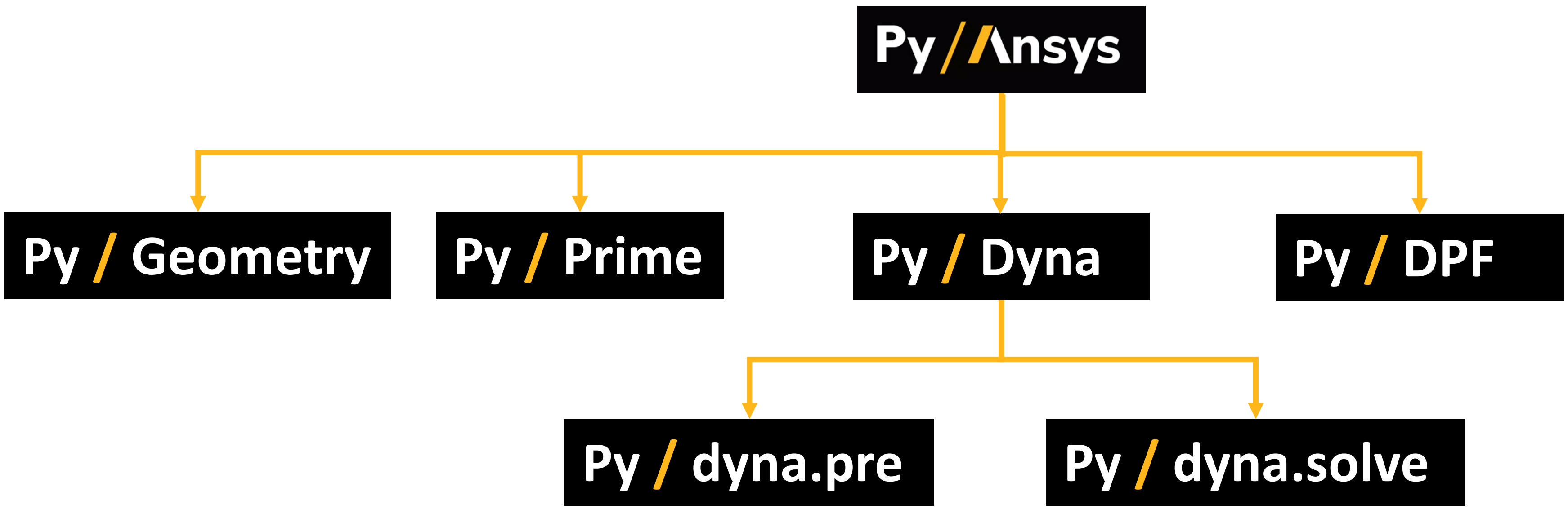
Once you have results, you can use the Ansys Data Processing Framework (DPF), which is designed to provide numerical simulation users and engineers with a toolbox for accessing and transforming simulation data. DPF can access data from Ansys solver files and from several files with neutral formats, including CSV, HDF5, and VTK. Using DPF's various operators, you can manipulate and transform this data.

On this page
PyDYNA documentation 0.4.2
PyDYNA
Overview
Documentation and issues
License
Edit on GitHub
Show Source

https://dyna.docs.pyansys.com/version/stable/

https://github.com/ansys/pydyna

Battery Characterization using PyAnsys



A Generic Battery APP

- Work for any Cell Format
- Automatically convert HPPC test curves to model input
- Define characterization test based on Cell Format
- Automatically convert mechanical indentation test to material input

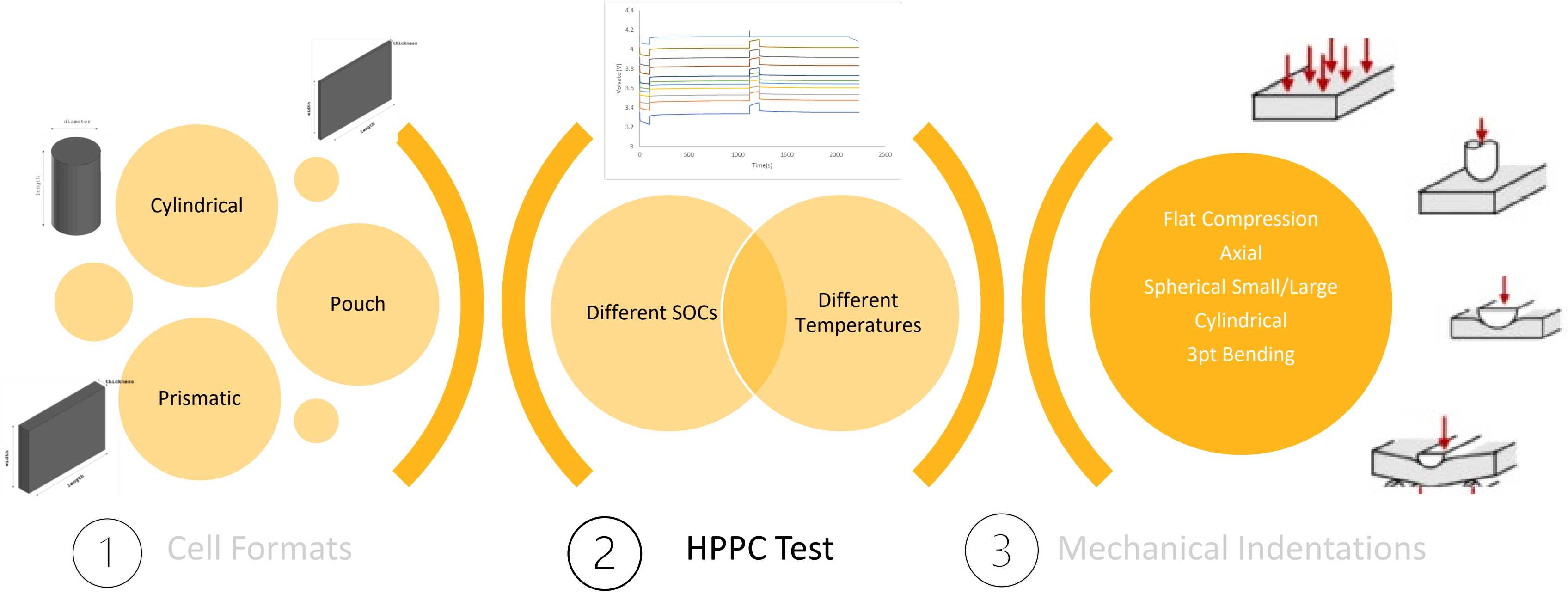
1 Cell Formats

2 HPPC Test

3 Mechanical Indentations

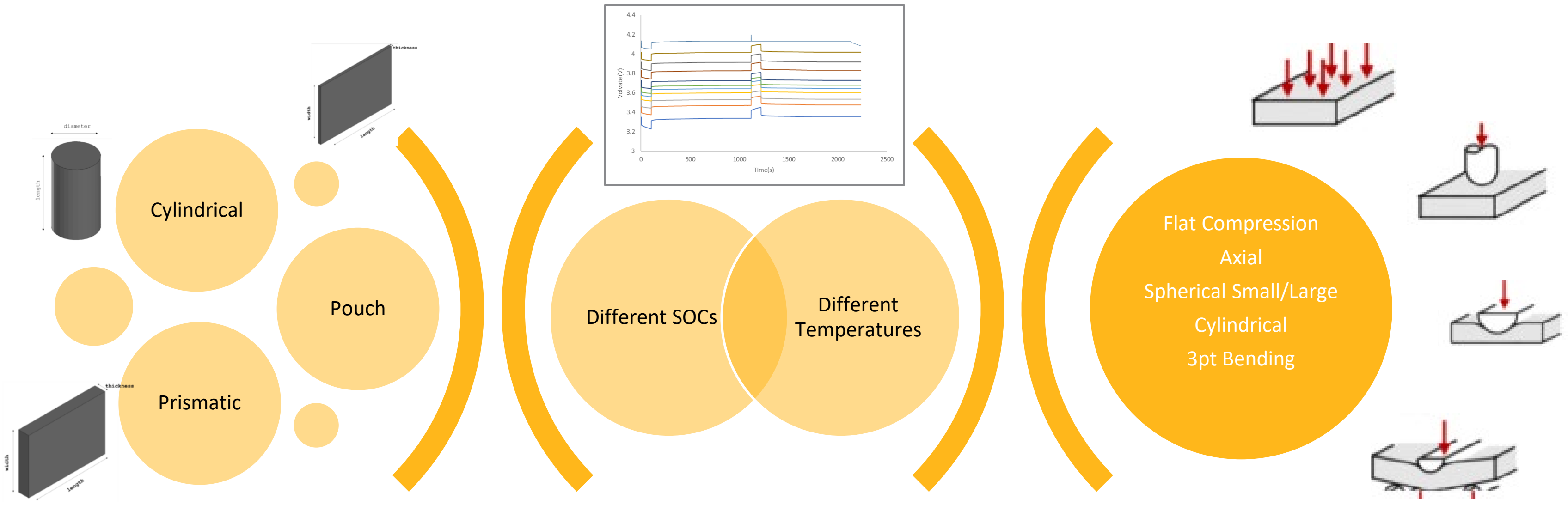
A Generic Battery APP

- Work for any Cell Format
- Automatically convert HPPC test curves to model input
- Define characterization test based on Cell Format
- Automatically convert mechanical indentation test to material input



A Generic Battery APP

- Work for any Cell Format
- Automatically convert HPPC test curves to model input
- Define characterization test based on Cell Format
- Automatically convert mechanical indentation test to material input



1 Cell Formats

2 HPPC Test

3 Mechanical Indentations

Battery Characterization App

Create Cell Geometry
pyGEOMETRY

Mesh Cell Geometry
pyPRIME

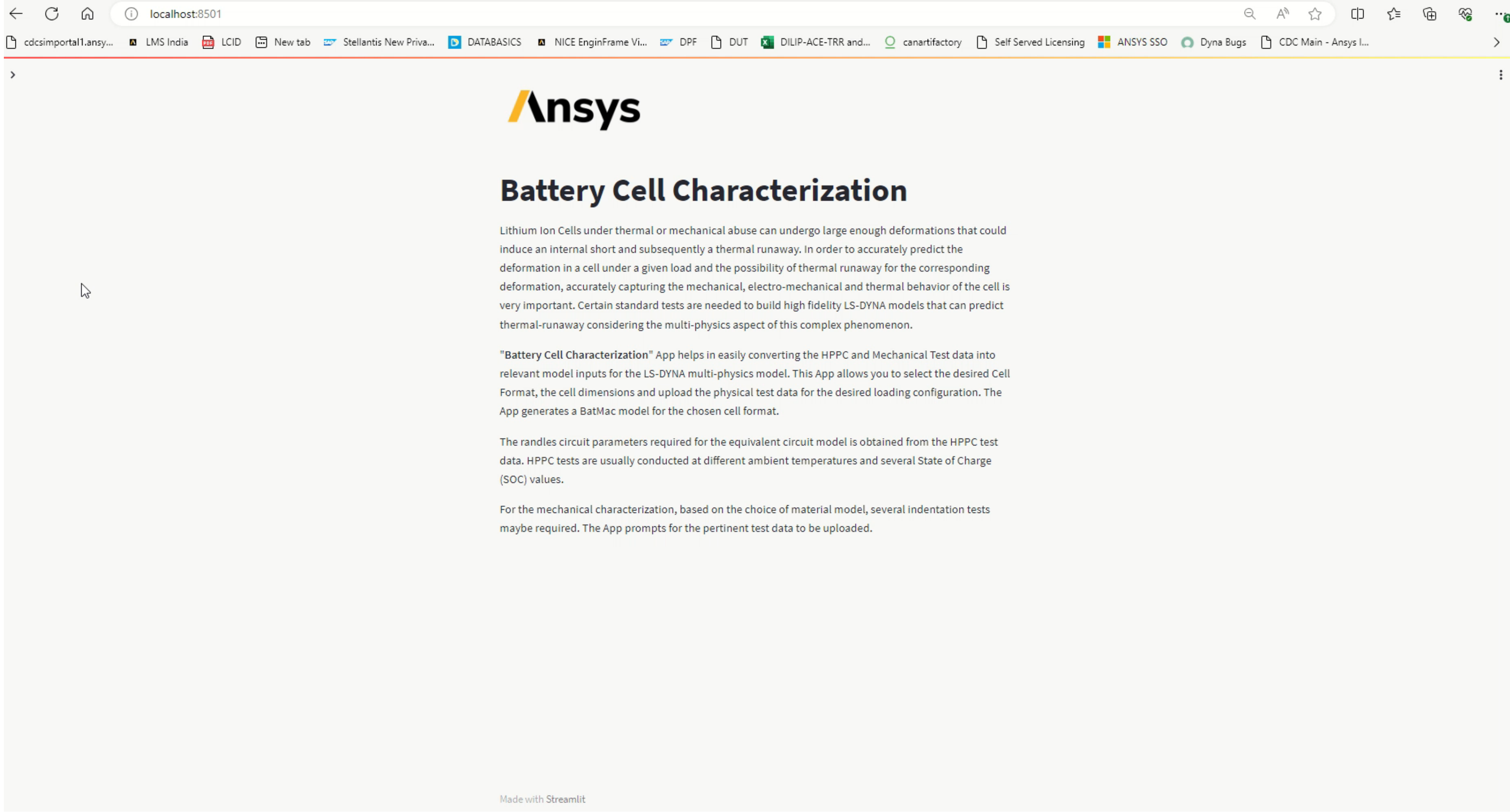
Extract Randles Circuit Parameters from HPPC Test Data
Standalone Python Script

Extract Stress Strain curve from Indentation Test
Standalone Python Script

Setup Dyna Input Deck
pyDYNA.pre

Solve
pyDYNA.solver

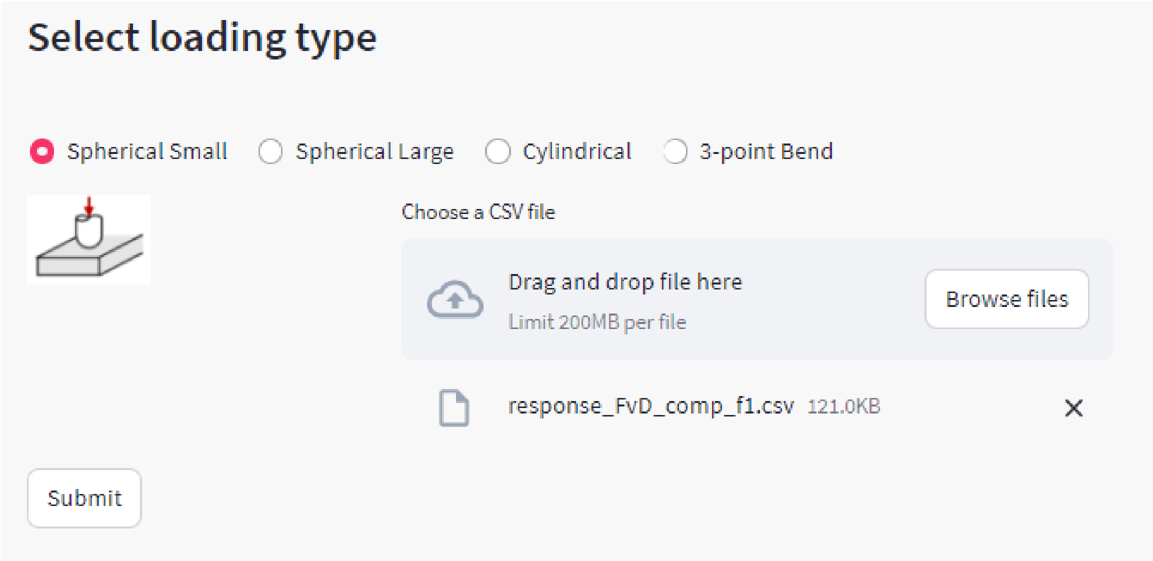
Post Process
pyDPF



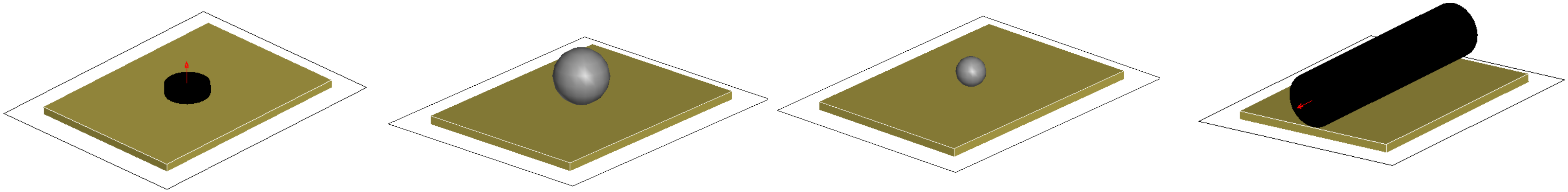
Thank You

Backup Slides

Indentation Model Setup

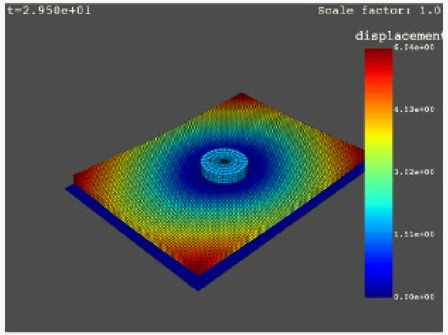


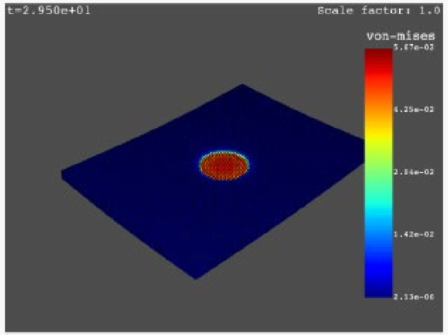
- Indenter specific RIGIDWALL
- Velocity defined for RIGIDWALL
- All necessary CONTROL cards defined
- MAT_THERMAL_ISOTROPIC
- EM_ISOPOTENTIAL and EM_ISOPOTENTIAL_CONNECT



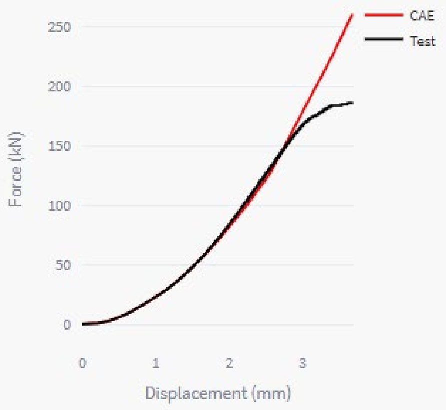
Battery Postprocessing

Mechanical Response



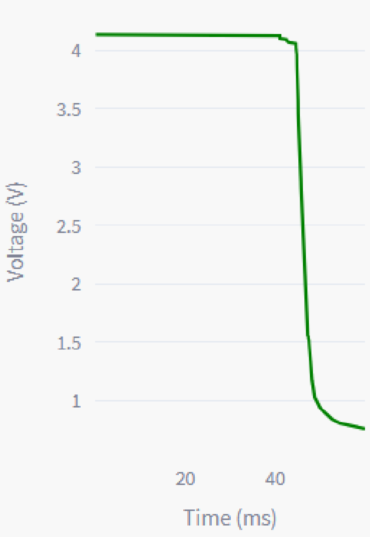


Force vs Displacement

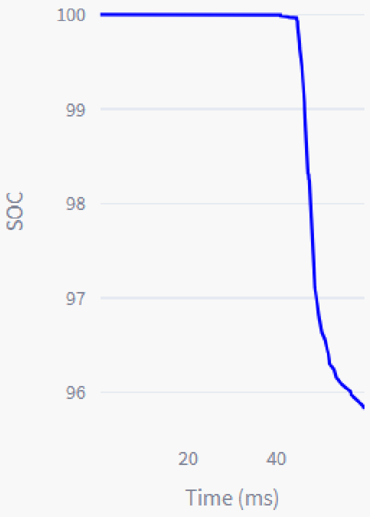


EM Response

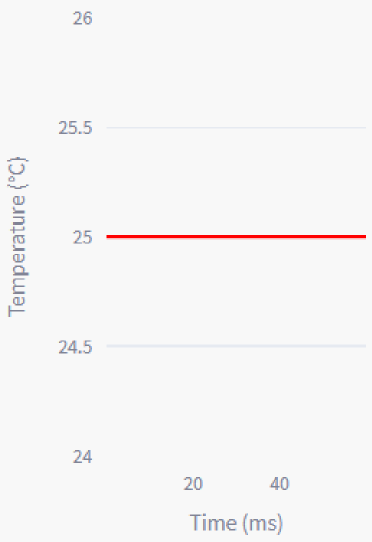
Voltage Response



SOC Response

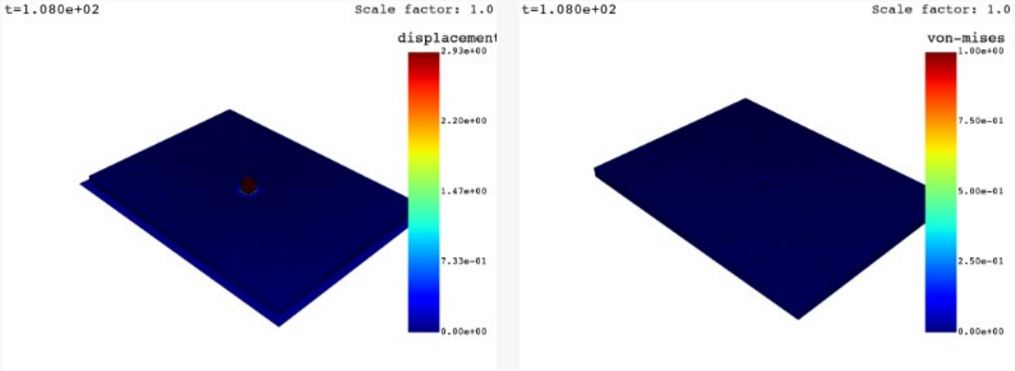


Temperature Response

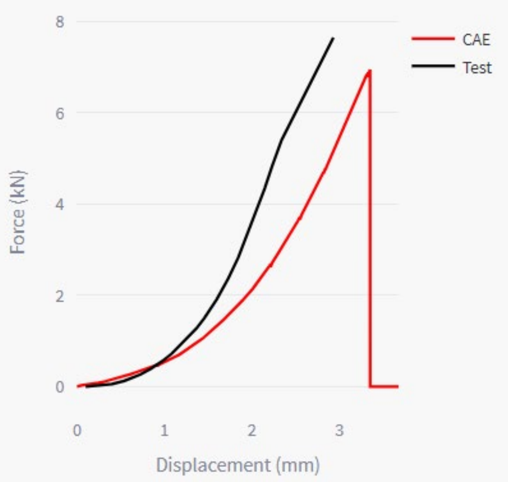


Small Spherical Indenter

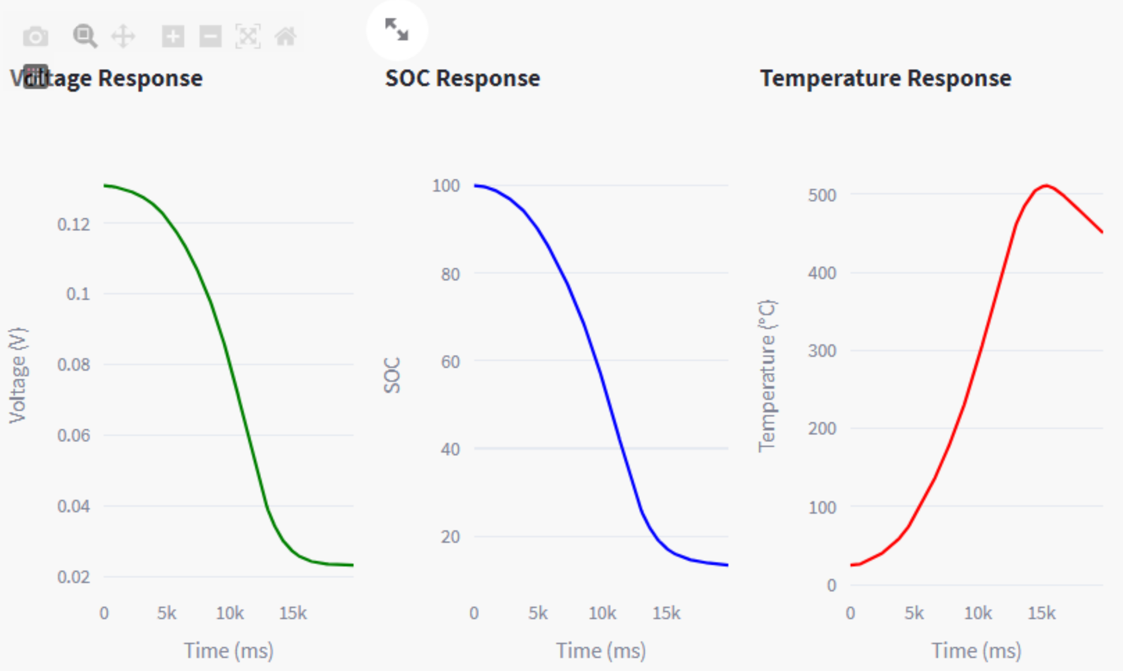
Mechanical Response



Force vs Displacement



EM Response



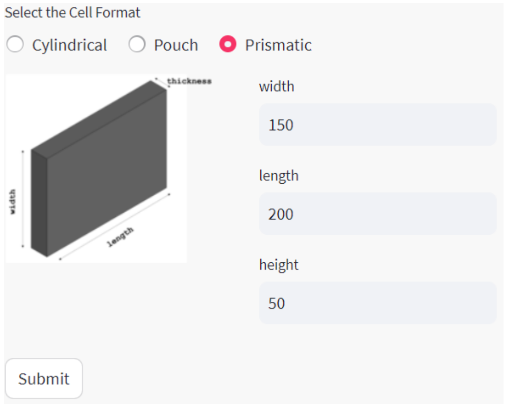
Solution Frameworks

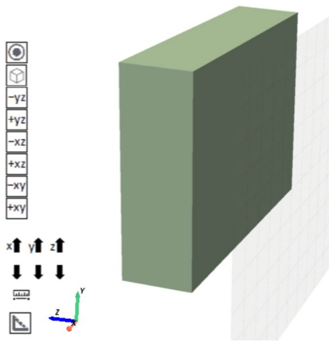
Technology Demonstrator
Training Tool

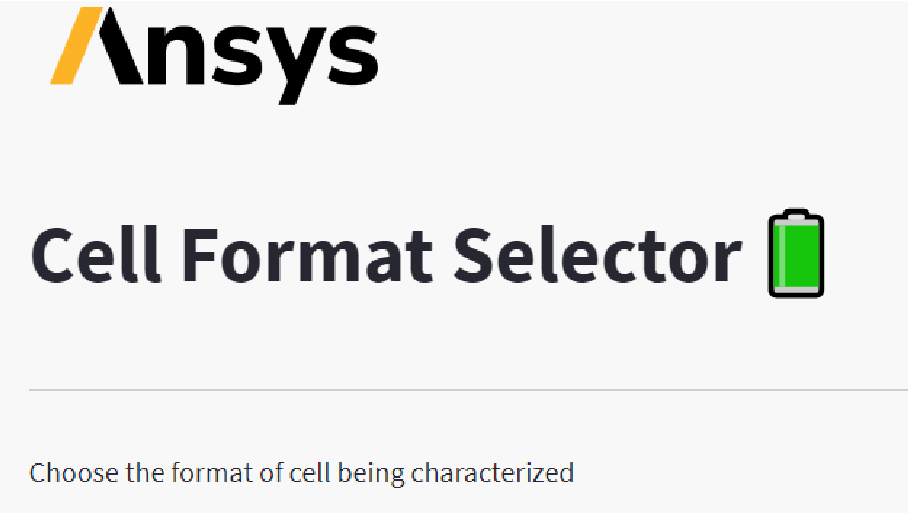
Streamlit

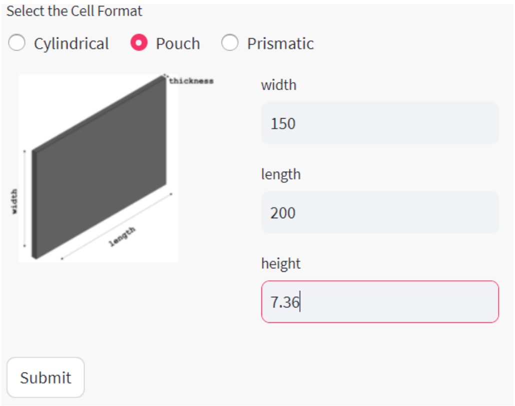
Dash
by plotly

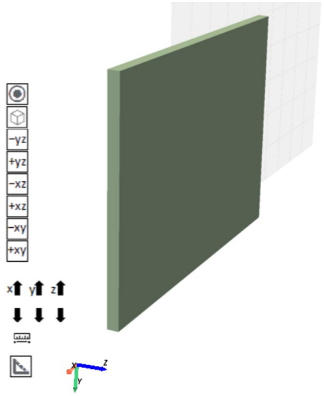
Cell Format Selector

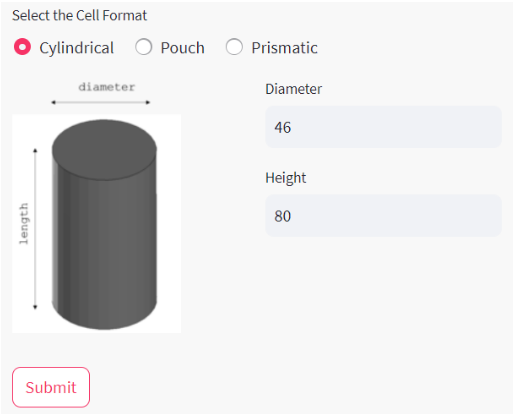


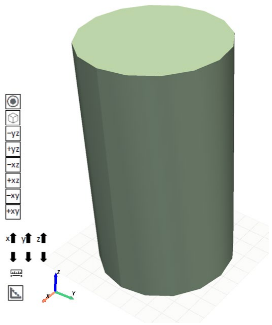








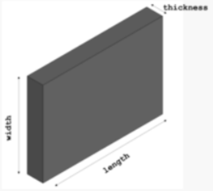




Cell Format Selector

Select the Cell Format

○ Cylindrical ○ Pouch ● Prismatic



width

150

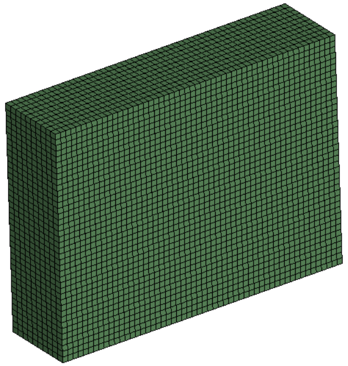
length

200

height

50

Submit



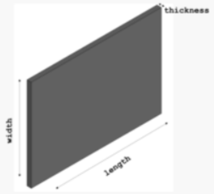
Ansys

Cell Format Selector 🔋

Choose the format of cell being characterized

Select the Cell Format

○ Cylindrical ● Pouch ○ Prismatic



width

150

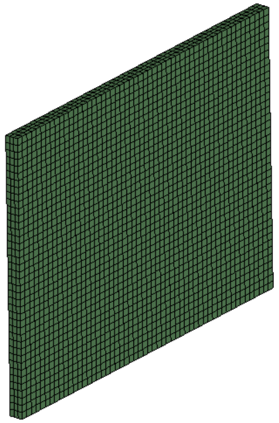
length

200

height

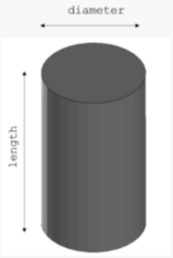
7.36

Submit



Select the Cell Format

● Cylindrical ○ Pouch ○ Prismatic



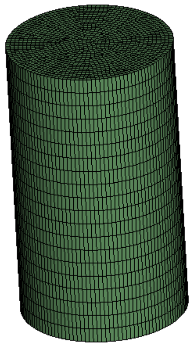
Diameter

46

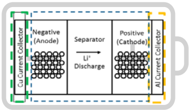
Height

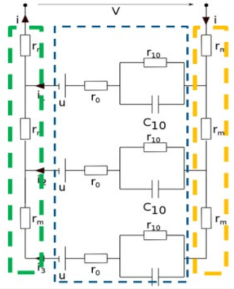
80

Submit



Calibrate R_0, R_{10}, C_{10} from HPPC test curves

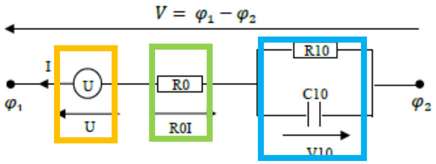


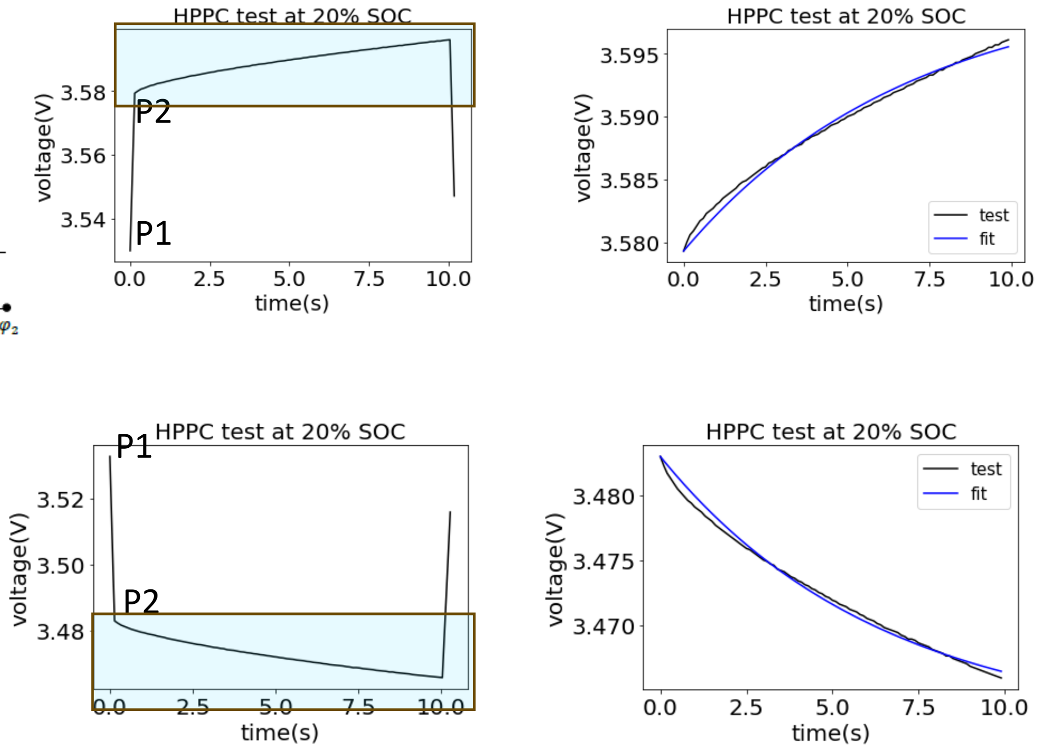


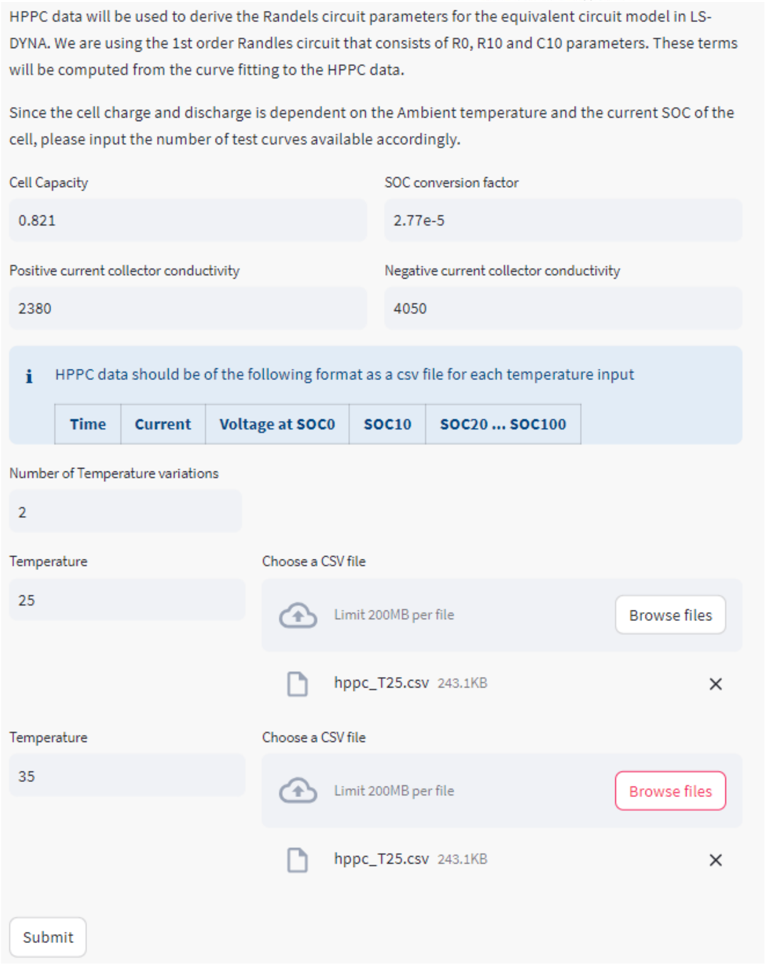
$$V_{total} = OCV - R_0 I - V_{10} \text{(same for discharge and charge)}$$

$$V_{10}(n+1) = V_{10}(n) + \left(t(n+1) - t(n)\right)\left(\frac{I(n)}{C_{10}} - \frac{V_{10}(n)}{R_{10}C_{10}}\right)$$

R_{10} and C_{10} are optimized.







PyDYNA Script Snippets

```
# Defining indenter based on input
if cell_type == "pouch":
    if indentor == "Flat Compress":
        rigidwall = RigidwallPlanar(Point(0, 0, 0), Point(0, 0, 1))
        compress.add(rigidwall)
        flat_indentor = RigidwallCylinder(Point(L/2.0,W/2.0,t+10.),Point(L/2.0,W/2.0,t+11.),radius=20,length=9.5)
        flat_indentor.set_motion(Curve(x=[0,10.0],y=[0,5.94]),motion=Motion.ACCELERATION,dir=Direction(0,0,-1))
        compress.add(flat_indentor)
# Setting EM control cards
em_solution = DynaEM()
em_solution.create_em_control(emsol=3)
em_solution.create_em_timestep(tstype=1,dtconst=1.0e-2)
cell_compression.add(em_solution)

# Defining NodeSets for the positive and the negative terminal
box1 = Box(xmin=L/4,xmax=(L/4+20),ymin=-1,ymax=1,zmin=-1,zmax=8)
box2 = Box(xmin=L*3/4,xmax=(L*3/4-20),ymin=-1,ymax=1,zmin=-1,zmax=8)

iso1 = Isopotential(NodeSetBox(boxes = [box1]),layer=EMRandlesLayer.CURRENT_COLLECTOR_POSITIVE)
iso2 = Isopotential(NodeSetBox(boxes= [box2]),layer=EMRandlesLayer.CURRENT_COLLECTOR_NEGATIVE)

# connecting the positive and the negative terminals
em_solution.connect_isopotential(contype=Isopotential_ConnType.RESISTANCE,isopotential1=iso1,isopotential2=iso2,val
ue=0.01)
```

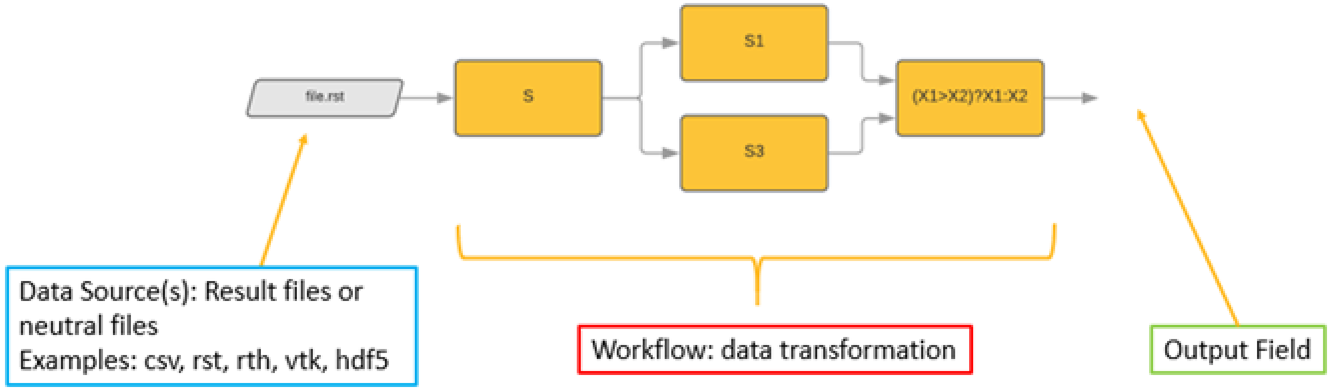
PyDYNA Script Snippets

```
# Defining RANGLES_BATMAC
cell = RandlesCell()
cell.set_batmac_model(cell_parts=PartSet([38]),
                      equilibrium_voltage=Curve(x=X, y=Y),
                   cell_capacity=randles_params[0]['Q'],soc_conversion_factor=randles_params[0]['cq'],charge_init_state=100,
                   circuit_parameter=[r0_tab_c,r0_tab_dc,r10_tab_c,r10_tab_dc,c10_tab_c,c10_tab_dc],constant_temperature=25,
                   temperature_from_thermal_solver=True,add_heating_to_thermal_solver=True)
compress.add(cell)


# Add thermal control cards
tanalysis = ThermalAnalysis()
tanalysis.set_timestep(initial_timestep=0.01)
tanalysis.set_solver(analysis_type=ThermalAnalysisType.TRANSIENT)
tanalysis.set_nonlinear(convergence_tol=0.01, divergence=0.5)
compress.add(tanalysis)
compress.initialconditions.create_temperature(nodeset=NodeSet(),temperature=25)
```

Post Processing with PyDPF

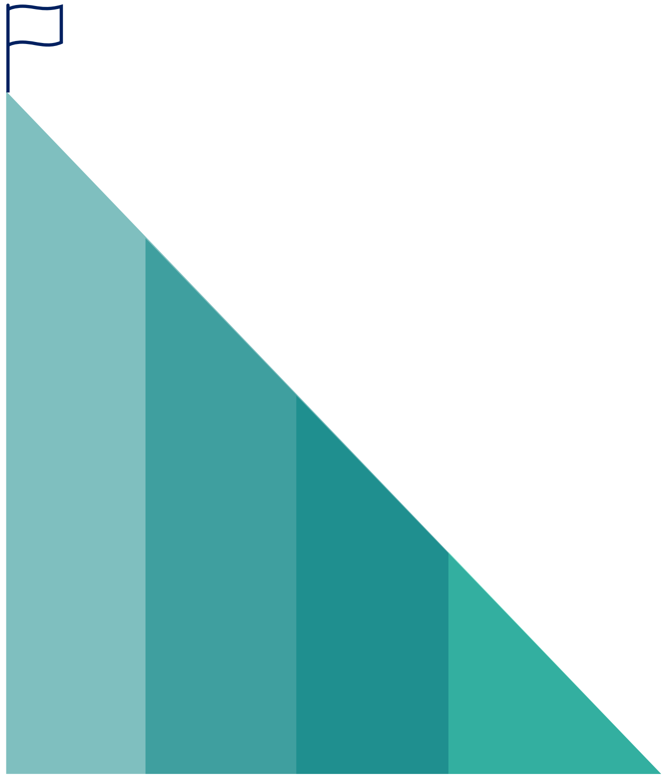
- PyDPF is a pythonic API used to post process all Ansys result formats
- With DPF, you can perform complex preprocessing or postprocessing of large amounts of simulation data within a simulation workflow



- For LSDYNA dpf supports reading d3plot and binout files

Further Improvements

- Support MAT_MODIFIED_HONEYCOMB
  - Anisotropy
  - Rate dependence
- Add Stochasticity to the model
- Job submission on multiple nodes



Ansys