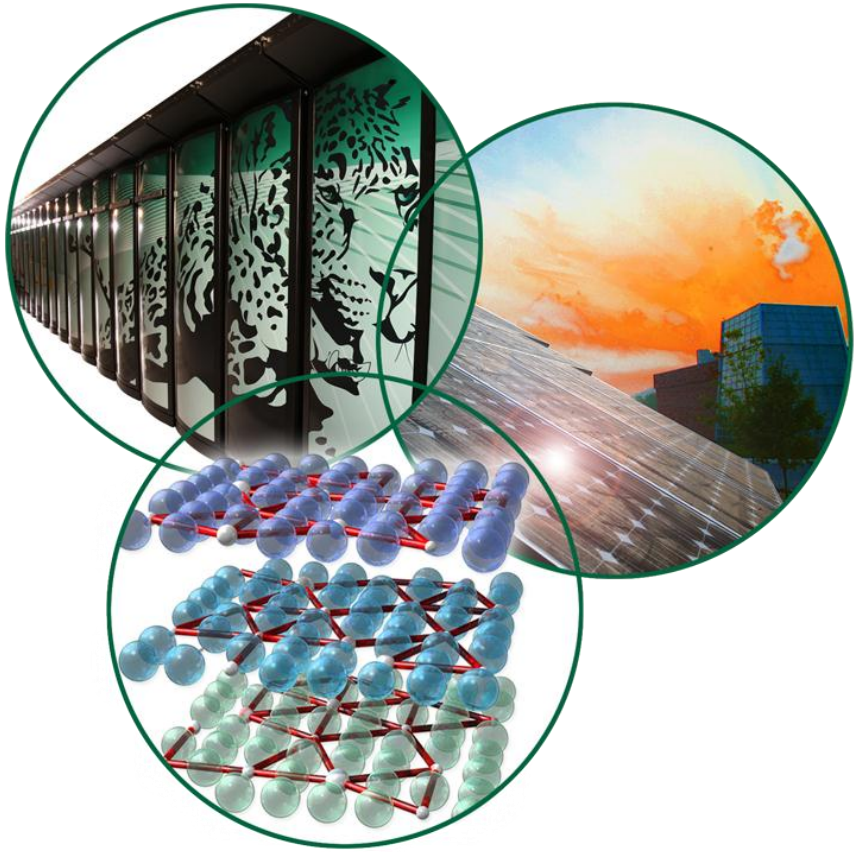
OAS (IPS-VIBE) Framework

Sreekanth Pannala & ORNL CAEBAT Team

OAS Kick-off Meeting

Aug. 2, 2011

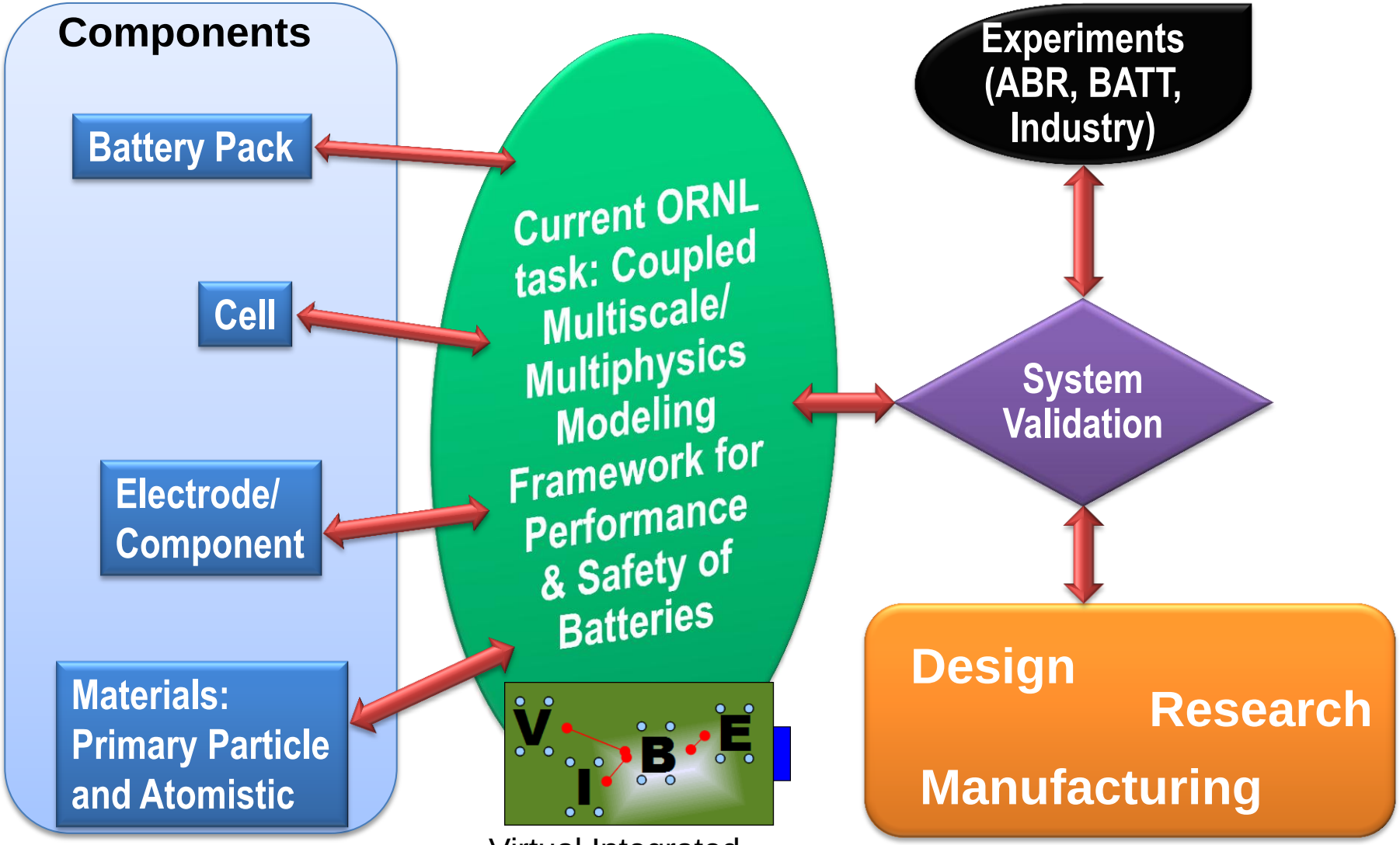
ORNL



U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

IPS-VIBE Simulation Framework



Virtual Integrated Battery Environment (VIBE)

CAEBAT OAS simulation platform has two aspects

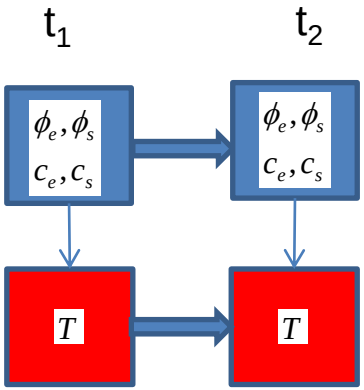
Software Infrastructure

- **flexible**
  - language-agnostic
  - multiple modeling approaches
  - combine appropriate component models for problem at hand
  - support integrated sensitivity analysis and uncertainty quantification
- **extensible**
  - ability to add and combine proprietary component models
- **scalable from desktop to HPC platforms**
  - hardware architecture-aware

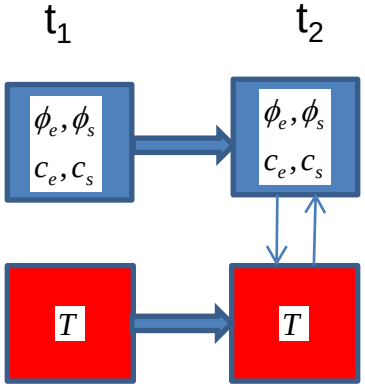
Numerical coupling/Scale-bridging approach(es)

- **flexible coupling strategy**
- **ability to transfer information across different models in a mathematically / physically consistent fashion**
- **similarly for bridging time-scales**

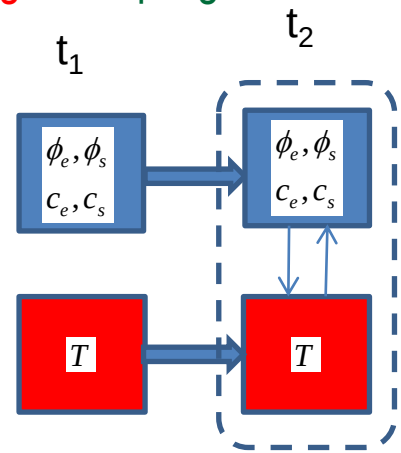
Coupling scenarios in battery modeling



One-way Coupling

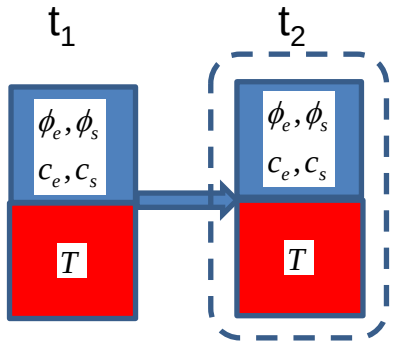


Two-way Loose Coupling



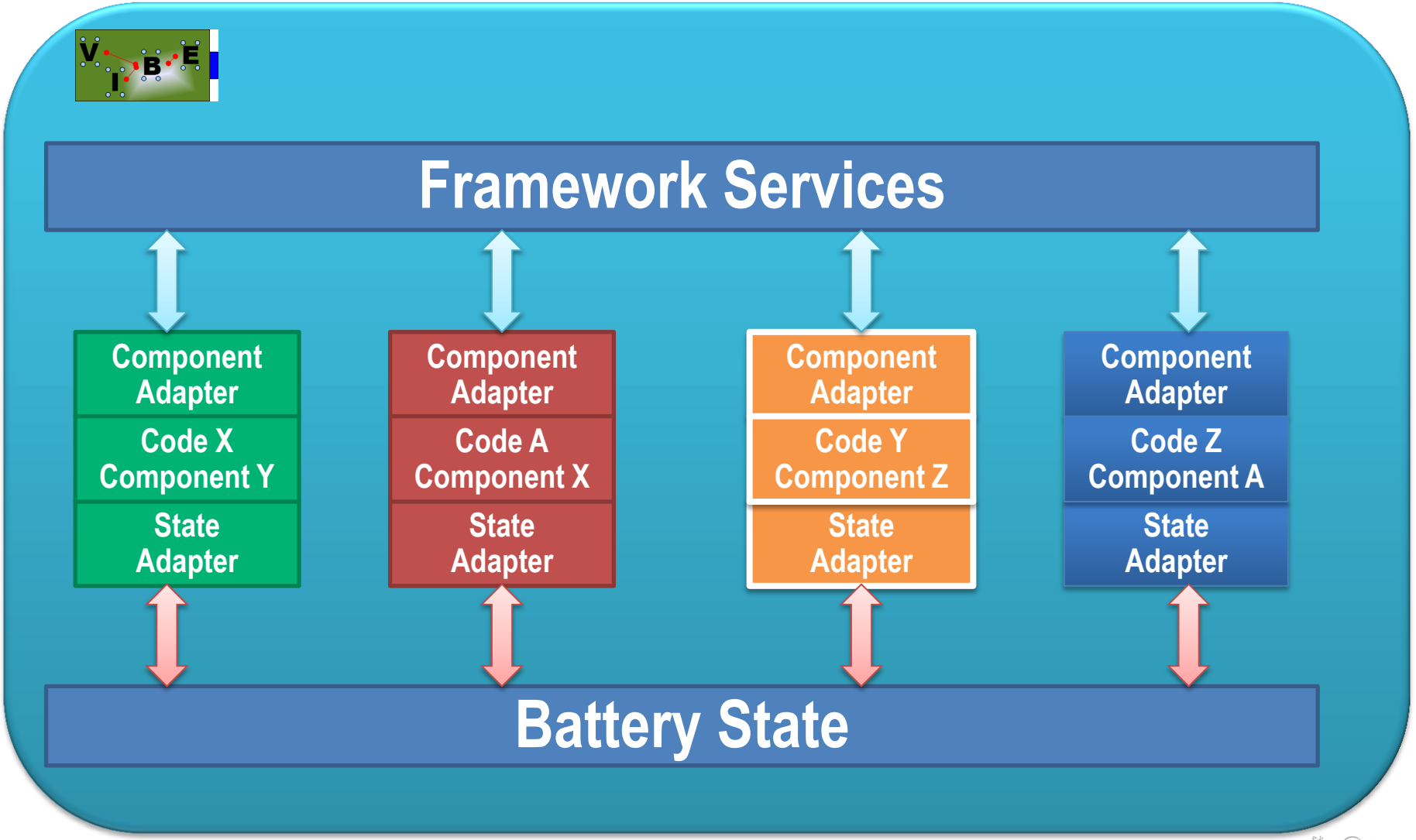
Two-way Tight Coupling

Picard
self-consistent iterations to some convergence criteria

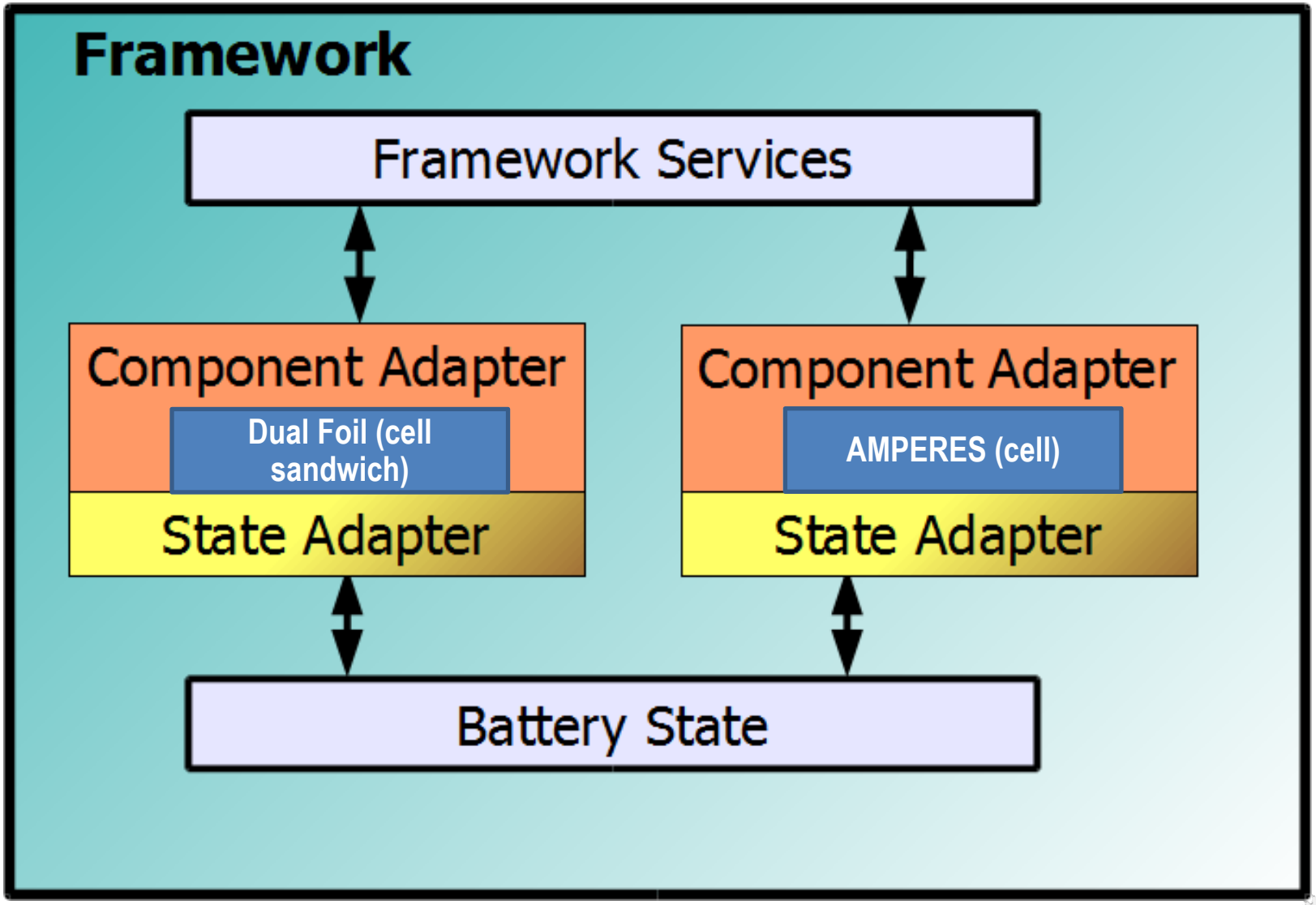


Fully Implicit
Consistency at each iteration across the physics in terms of full non-linear residual

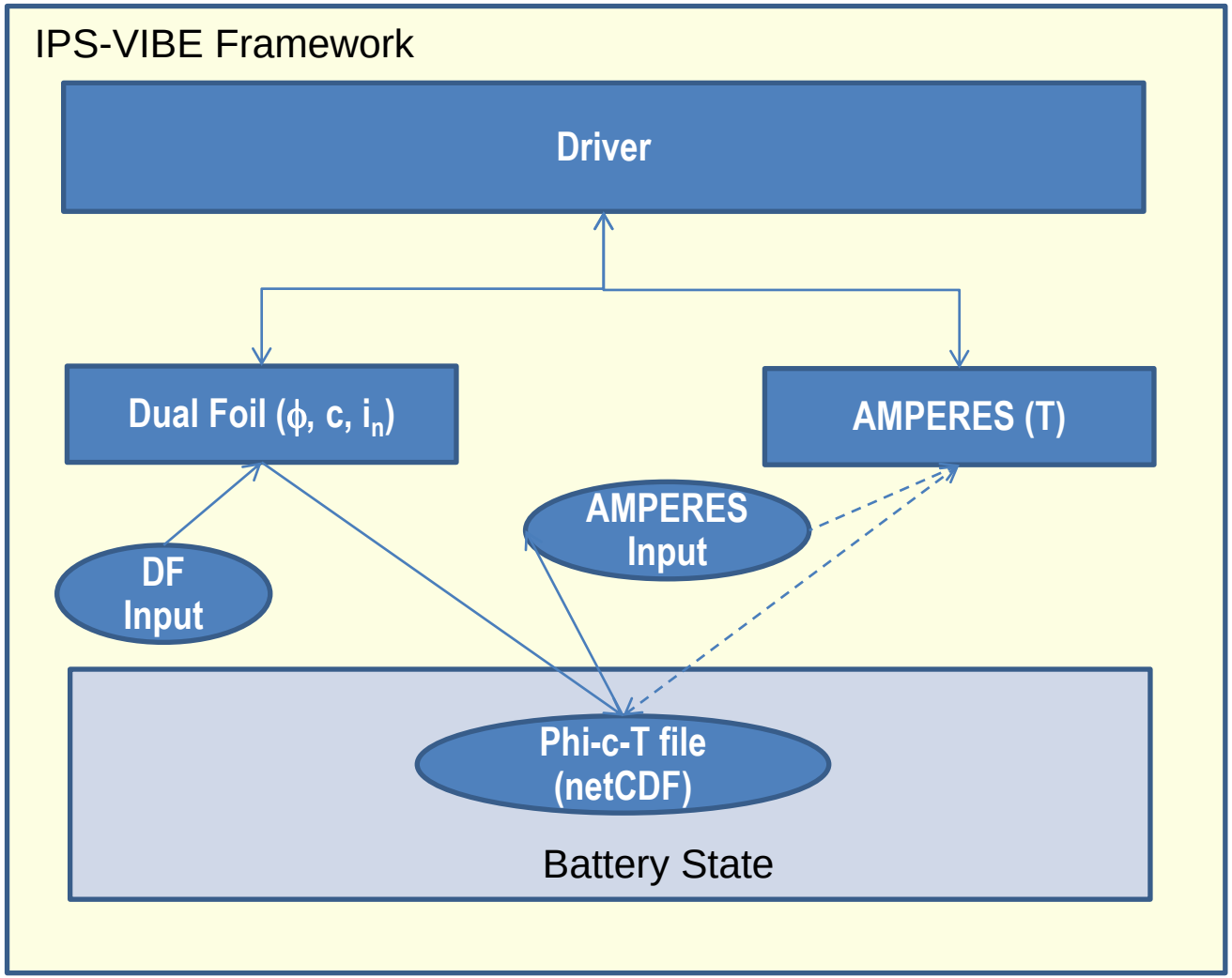
VIBE Software Platform for CAEBAT



IPS-VIBE Framework for CAEBAT (demo: Dualfoil/AMPERES)



VIBE Software Platform for CAEBAT (demo: Dualfoil/AMPERES)



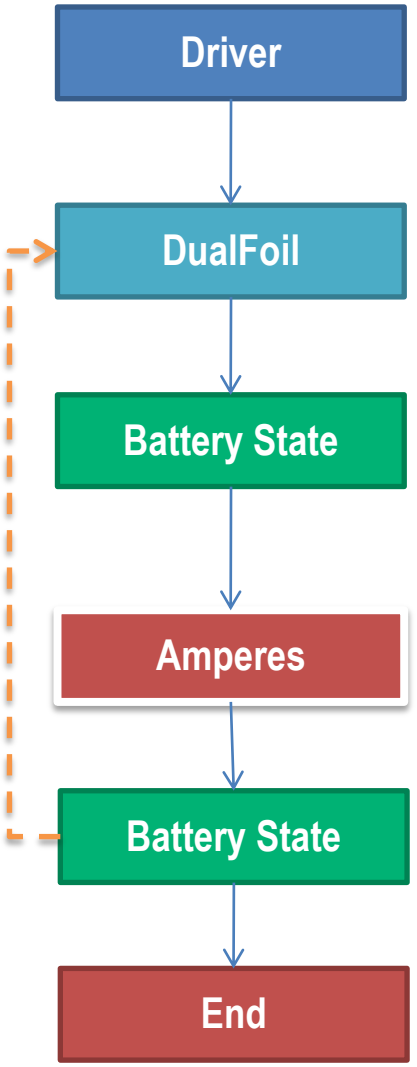
Possible Scenarios Explored:

a) DualFoil for entire duration followed by AMPERES (one-way)
b) Loosely coupled DualFoil and AMPERES
c) Automated parameter sweep

More discussion on generalized input this afternoon

VIBE Sequence (another view)

- The Dualfoil wrapper calls dualfoil executable – that reads dualfoil input and writes out native files and one additional ascii file with ϕ_1, ϕ_2, c_e, c_s, i_n, and reference T as a function of x and t.
- The output wrapper reads this file, maps to 3D and writes to the netCDF battery state (phi_c_T.nc)
- Amperes input preparation tools reads this file, runs other files to generate the input for Amperes
- Amperes wrapper runs amperes and the output wrapper updates the battery state with new T
- Iterate based on prescribed Δt for exchange of information (average T on the zones)



Battery state file

- **This file(s) will have minimal set of variables so that all the components can talk to each other and the state is completely defined**
- **We have, *for now*, chosen netCDF format for this file**
  - internal specifics for CAEBAT to be determined by community

Species Conservation

Electrolyte phase:
$$\frac{\partial(\varepsilon_e c_e)}{\partial t} = \nabla\cdot\left(D_e^{eff}\nabla c_e\right) + \frac{1-t_+^0}{F} j^{Li} - \frac{i_e\cdot\nabla t_+^0}{F}$$

Solid phase:
$$\frac{\partial(\varepsilon_s c_s)}{\partial t} = \nabla\cdot\left(D_s^{eff}\nabla c_s\right) - \frac{j^{Li}}{F}$$

Closures:
$$D_e^{eff} = D_e\varepsilon_e^{\xi} \qquad D_s^{eff} = D_s\varepsilon_s^{\zeta}$$

Charge Conservation

Electrolyte phase:
$$\nabla\cdot\left(\kappa^{eff}\nabla\phi_e\right) + \nabla\cdot\left(\kappa_D^{eff}\nabla\ln c_e\right) + j^{Li} = 0$$

Solid phase:
$$\nabla\cdot\left(\sigma^{eff}\nabla\phi_s\right) - j^{Li} = 0$$

Closures:
$$\kappa^{eff} = \kappa\varepsilon_e^{1.5} \qquad \kappa_D^{eff} = \frac{2RT\kappa^{eff}}{F}(t_+^0 - 1)\left(1 + \frac{d\ln f_{\pm}}{d\ln c_e}\right) \qquad \sigma^{eff} = \sigma\varepsilon_s^{m}$$

Electrode Kinetics
$$\bar{j} = ai_0\left[\exp\left(\frac{\alpha_a F}{RT}\eta\right) - \exp\left(-\frac{\alpha_c F}{RT}\eta\right)\right]$$

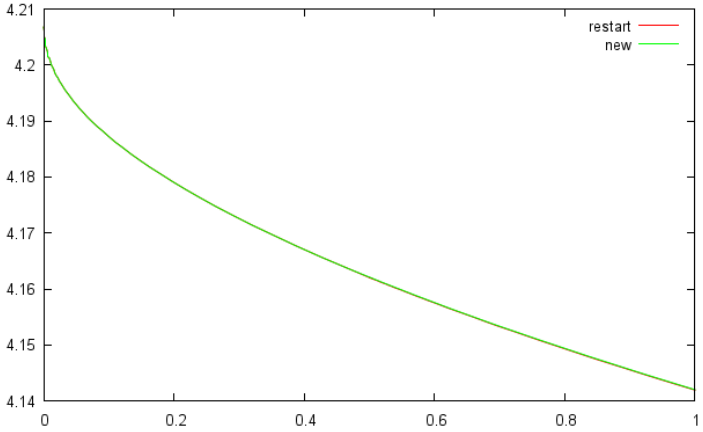
$$\frac{\partial(\rho c_p T)}{\partial t} = \nabla\cdot(\lambda\nabla T) + q$$

More discussion this afternoon

Modifications to software components for initial demonstration

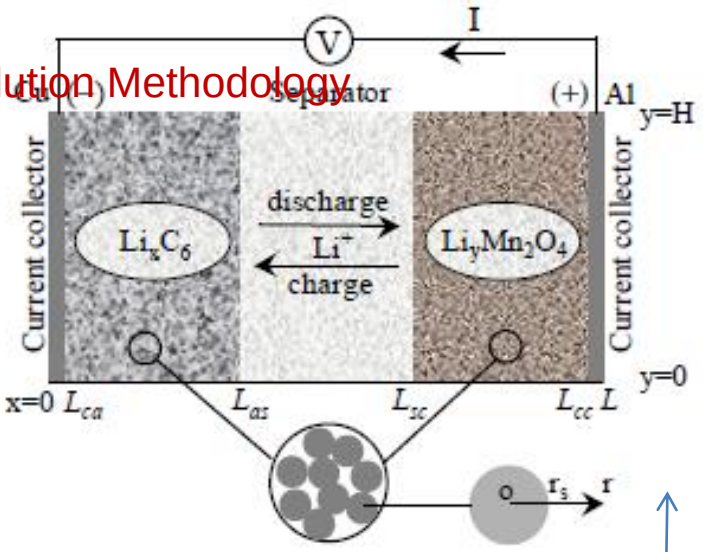
- **Changes to DualFoil**
  - very minimal changes for one way coupled
  - minor modifications to write the information needed for battery state as a function of time and space
  - **additional modifications to allow restart**
- **Changes to Amperes**
  - very minimal changes
  - minor modifications to read the input files generated by the prepare input wrapper
  - additional arguments for conducting the parametric sweeps through the IPS-VIBE framework

Verification of the restart capability



Thermo-Electrochemical Modeling in LIBs – Problem Definition

Solution Methodology

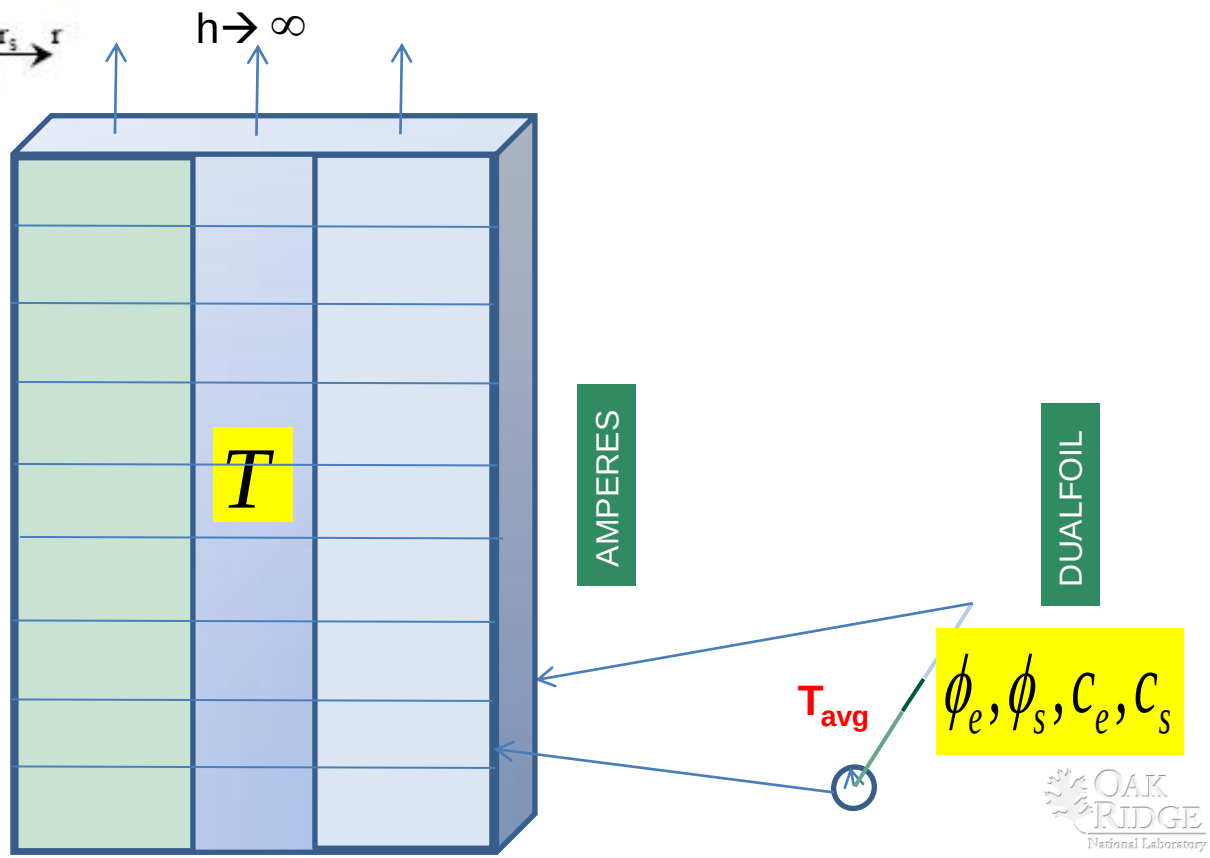


Composite positive electrode

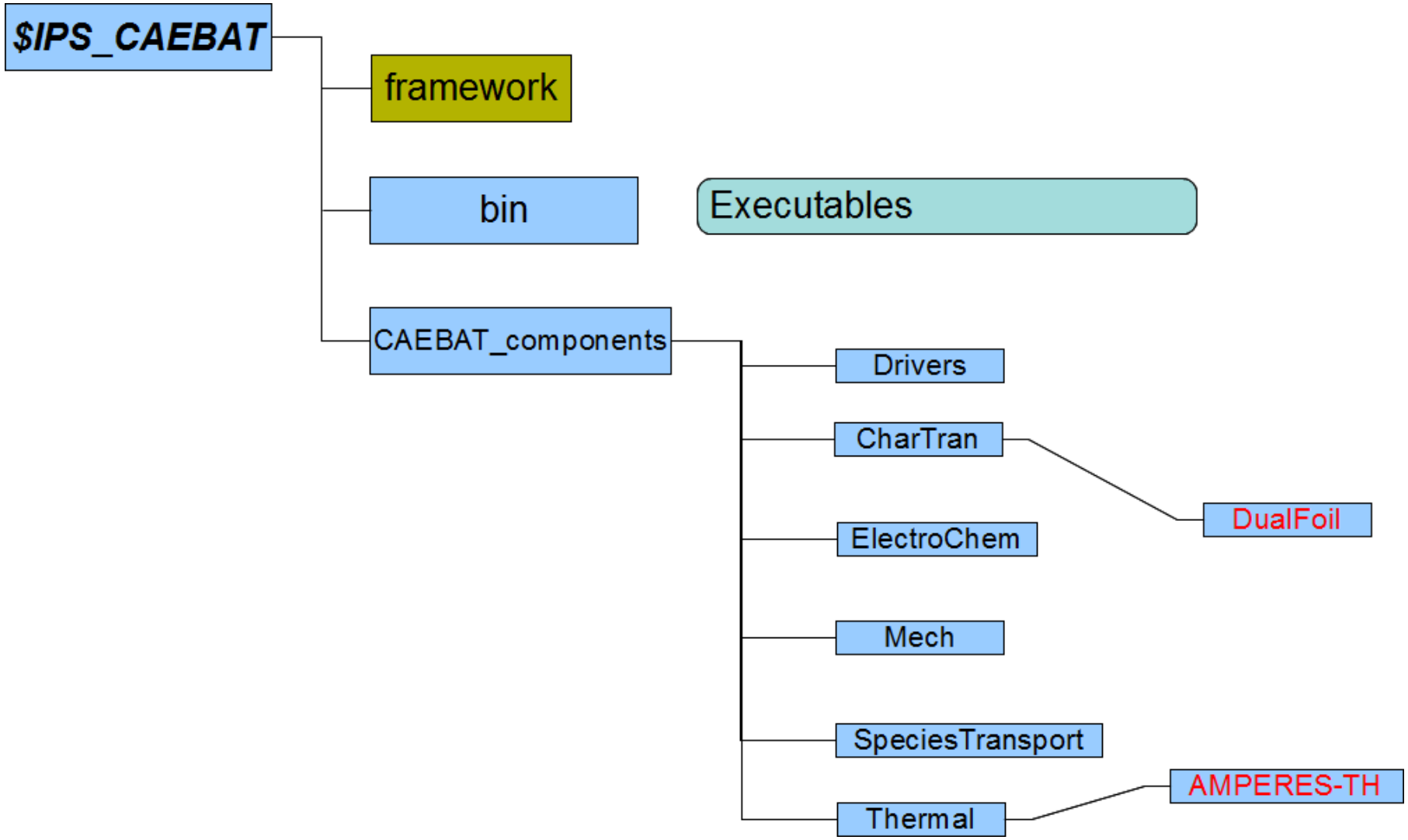
$$Li_{y-x}Mn_2O_4 + xLi^+ + xe^- \underset{\text{charge}}{\overset{\text{discharge}}{\rightleftharpoons}} Li_yMn_2O_4$$

Composite negative electrode

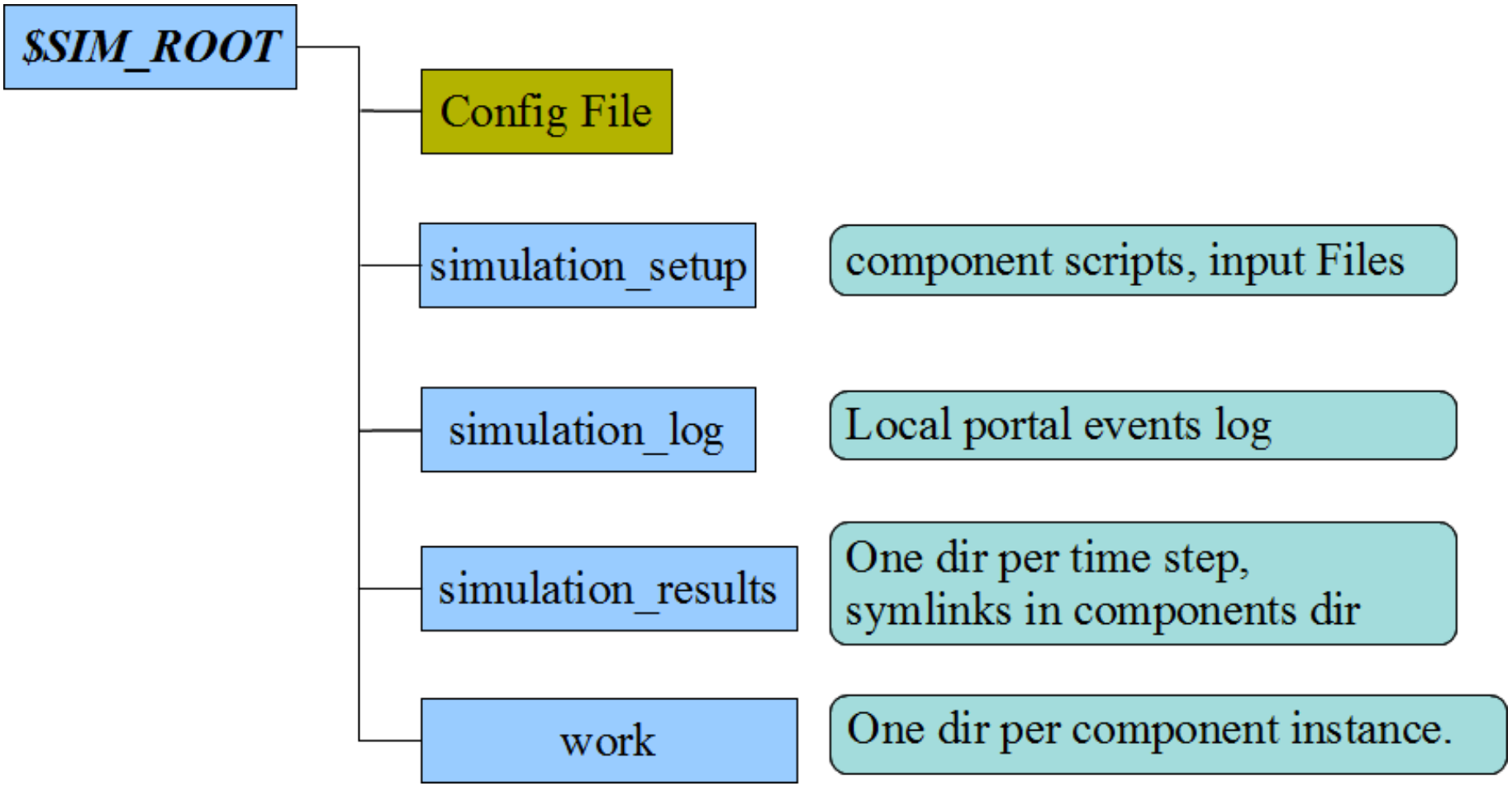
$$Li_xC_6 \underset{\text{charge}}{\overset{\text{discharge}}{\rightleftharpoons}} Li_0C_6 + xLi^+ + xe^-$$



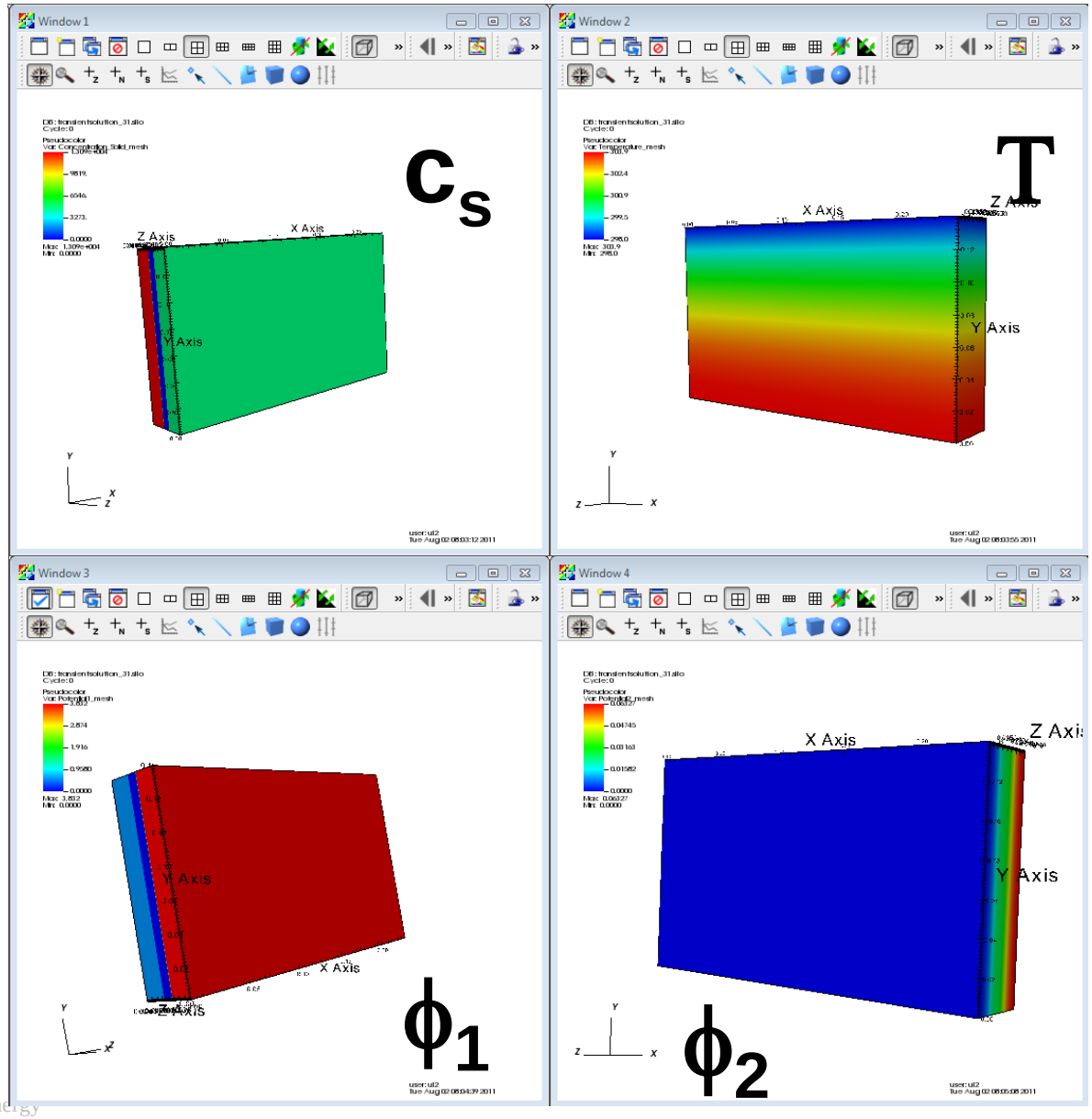
Directory structure



Data management



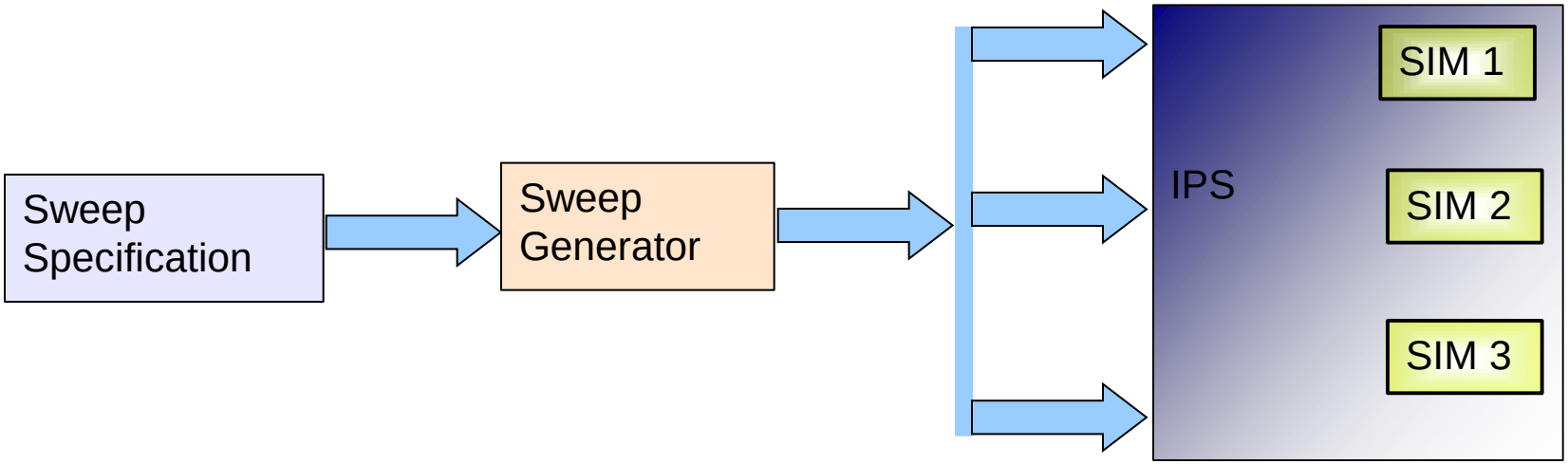
Sample results



What does this case demonstrate?

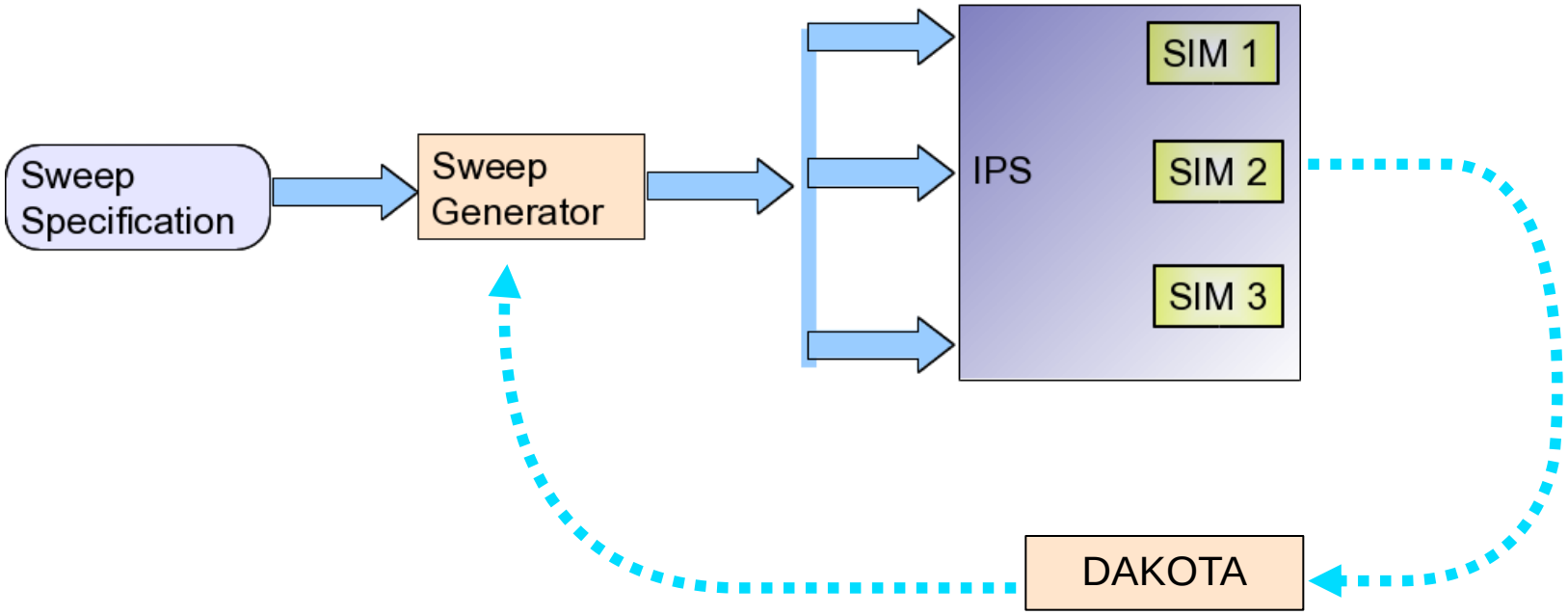
- **Loose coupling (one way and two way) between required physics**
  - charge transport + thermal
- **Mixing two common scientific programming languages**
  - Fortran and C++
- **Standardize the inputs and outputs to these two different components**
- **This initial demo should provide a template for other components in addition to serving as a starting point for standardization of the entire battery state**

Parameter Sweep using the IPS – Phase



- **Pre-defined parameter set that covers the parameter space for all components in the simulation.**

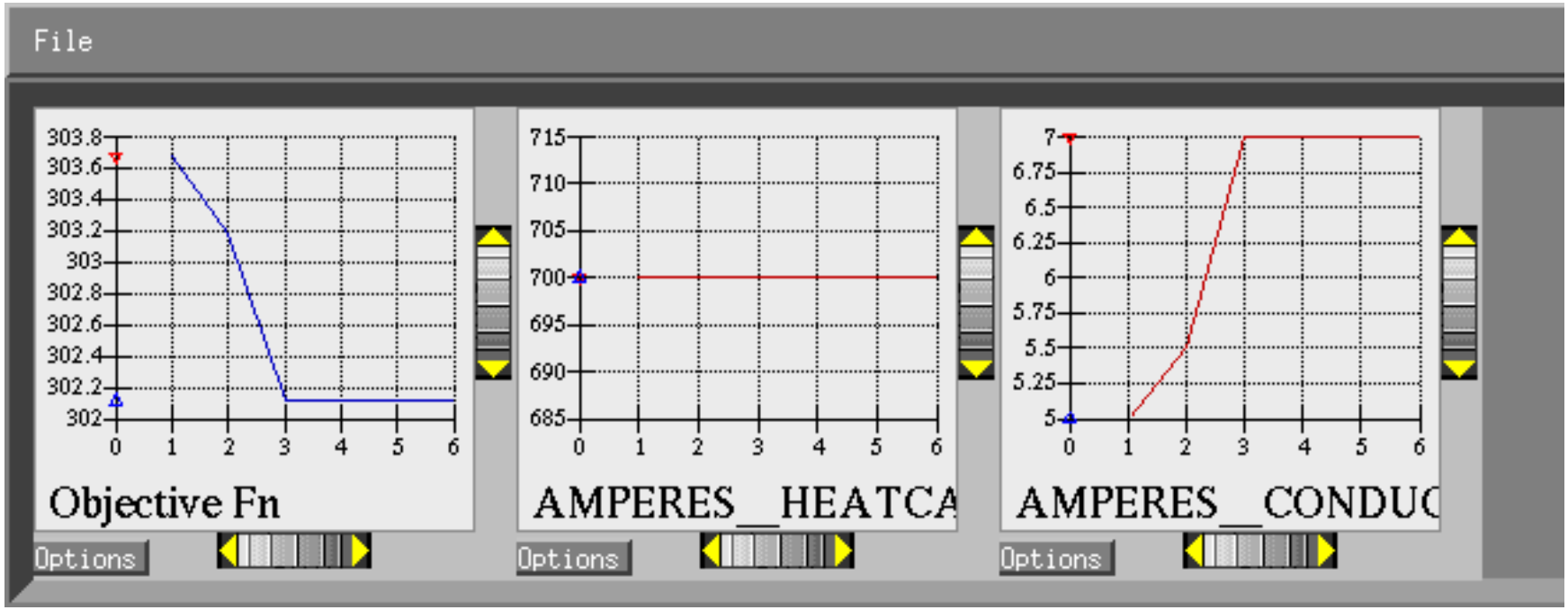
Parameter Sweep using the IPS – Phase 2



- Dynamic generation for design optimization using the DAKOTA tool kit (From Sandia National Lab)

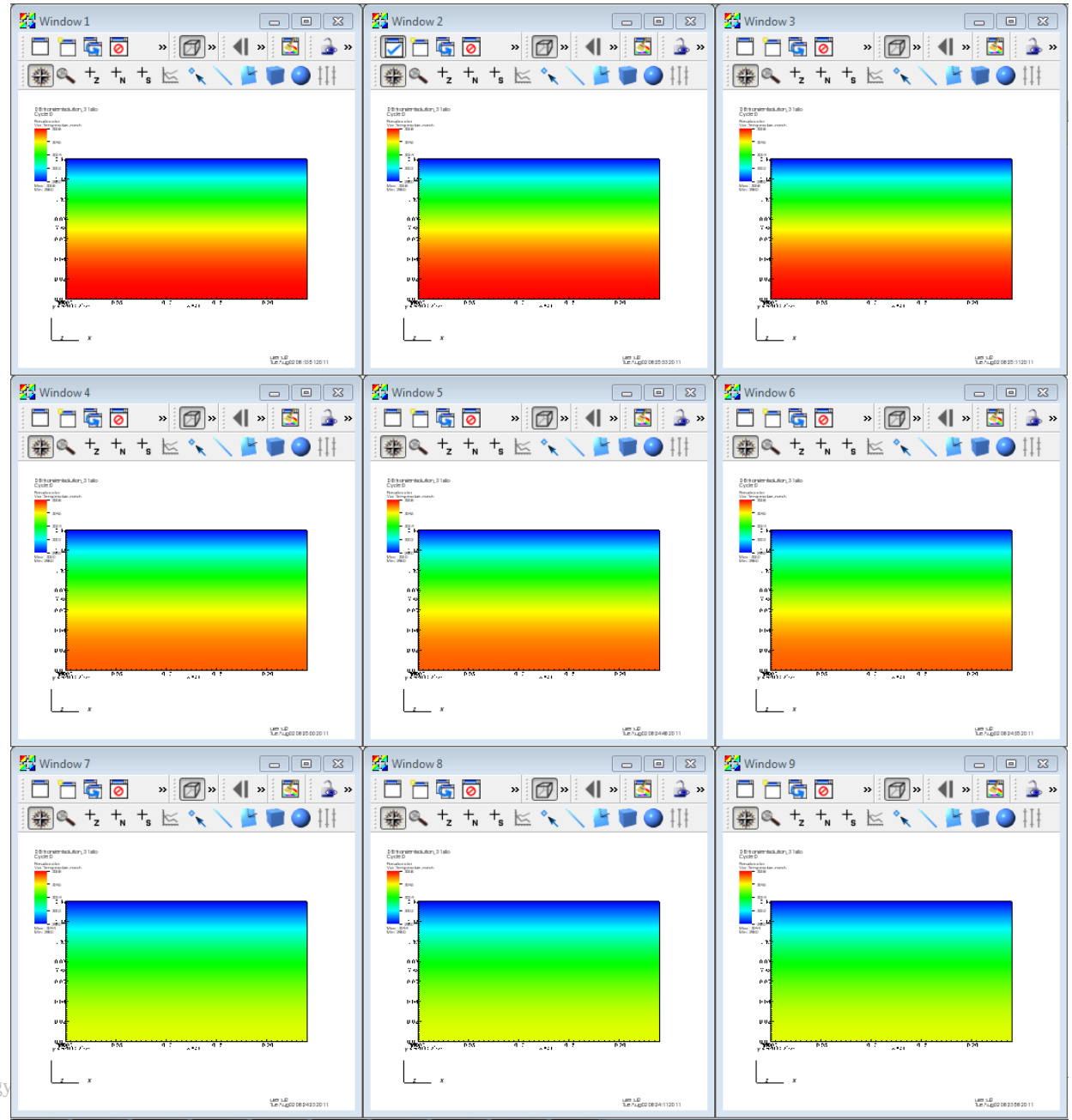
Dakota Optimization (Simple demo)

Objective = Minimize T_{avg}
Cp = 300, 700 (starting 700)
λ = 3,7 (starting 5)

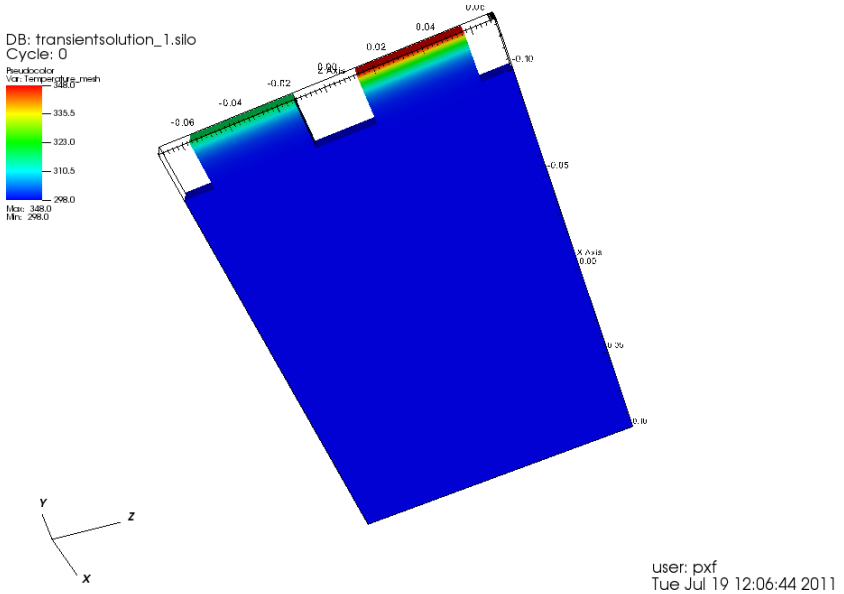


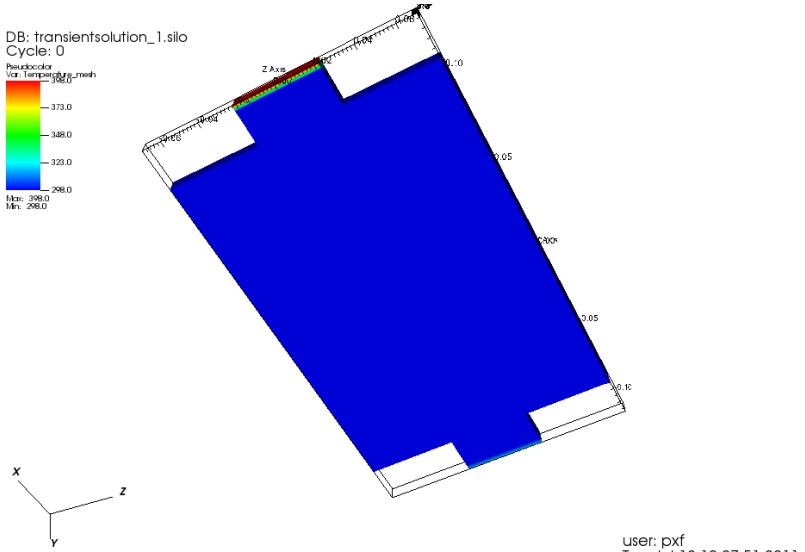
Critical as even temperature changes have huge impact on safety and life

Parameter Sweep Results



Realistic geometry case with tabs (still under works)





Advantages of this approach

- **Component-based approach**
  - *Extensibility*, V&V, independent development.
- **Common solution (battery) state layer**
  - Data repository.
  - Conduit for inter-component data exchange.
- **File-Based data exchange**
  - No change to underlying codes.
  - Simplify *"unit testing"*
- **Scripting Based Framework (Python)**
  - Rapid Application Development (RAD).
  - Adaptability, changeability, and flexibility.
- **Simple component connectivity pattern**
  - Driver/workers topology.
- **Codes as components:**
  - Focus on code-coupling vs physics-coupling as first step.
- **Simple unified component interface**
  - init(), step(), finalize().

Demonstration

- Walk through the directory structure
- Where are the dualfoil/amperes executables and data?
- Where the dualfoil/amperes wrappers are located?
- Config files
- Where are the drivers?
- Run a case with different scenarios
  - One-way coupled
  - Two-way coupled – one zone (issues?)
- Run an optimization through Dakota
- Results

Next steps

- **Finish, compare and contrast**
  - One-way coupled (with different Δt)
  - Two-way loosely coupled
  - Two-way coupled – multiple zones
- **Case with tabs**
- **Validation**
- **Explore Dakota Optimization for Tabs**
- **Standards – definition and implementation**
- **Documentation and Software Release**
- **Integration with NREL's MSMD model, the three partner's components, other National Labs contributions**
  - Working closely with you….