libsim

Release 1.0

A. Caldwell, A. Preston, A. Valkonen, J. Xiang, J. Yanez

Dec 20, 2021

CONTENTS:

libsim is a Python library that creates battery simulation models.

Check out the *Usage* section for further information, including how to *install* the project.

Note: This project is under active development.

CHAPTER

ONE

BACKGROUND

1.1 Models

One of the research challenges in the development of lithium-ion batteries (LIBs) is to predict their behavior under different operating modes, useful for estimating state of charge and state of health of batteries in electric vehicles (EV's). There exist empirical models, mostly equivalent circuit-based, widely used in the Battery Management Systems (BMS) of electronics and EV's. These types of models use past experimental data of a battery to anticipate its future states. Most of the experimental data used to find charge/discharge characteristics rely on the current or cell potential. On one hand, these empirical models are relatively fast and simple computationally, but they have drawbacks. For example, the physics-based parameters are not able to be predetermined. The battery characteristics aren't updated as the battery ages and a battery's model is unique to itself - it does not apply to all batteries but only a specific type.

Other major types of models are Pseudo-two-Dimensional (P2D) and Single Particle Model (SPM). Both popular battery models are widely used today. The P2D model is commonly used in lithium-ion battery studies, and the predicted behavior of this model matches experimental data quite accurately. A significant drawback with this model is the difficulty to use in real-time due to its computationally expensive nature. The SPM model simplifies anode/cathode interactions and reduces the dimensionality down to one dimension, which greatly enhances its computational capabilities. However, it places greater importance on the parameters of the anodes and cathodes.

CHAPTER

TWO

PROJECT GOALS

This project focuses on estimating and predicting the state of charge (SOC) and state of health (SOH) for lithium-ion batteries (LIBs) using a Single Particle Model (SPM) in order to reduce the computational cost and allow for the model to be implemented in real-time EV LIBs modeling. It accounts for the impact of complex parameters such as ion diffusivity, ion particle radius, and maximum ion concentration at the ion's surface on the performance of the battery.

CHAPTER

THREE

DESIGN PROCESS

In this code suite, abstracting battery behavior was not a trivial task. LIBs can be designed with many different types of anodes and cathodes which directly affect the electro-chemical properties and electrolyte interactions. To capture this variability, we decided to create a dictionary of different lithium-ion battery types that each have their own unique properties regarding diffusivity, particle radius, and ion concentration. This increases the versatility of the code suite to make the simulations widely applicable should an end user decide to test through various types of lithium-ion batteries of their choosing.

SPM model was used to simulate battery cycling behaviors. For generating solutions a finite element method was chosen. Architectural choices were made to allow for future implementation of various different model types.

3.1 UML DIAGRAM

INCLUDE NEW UML DIAGRAM

CHAPTER

FOUR

LIBSIM

4.1 arguments module

4.2 batterycell module

4.3 derivative module

Derivative

derivative.**first_derivative**(*Mesh*, *coefficient*, *timestep*)
Calculates the first derivative in Fick's Law

derivative.**second_derivative**(*Mesh*, *coefficient*, *timestep*)
Calculates the second derivative in Fick's Law The coefficient to be passed is a function Mesh is the mesh for which the derivative is to be evaluated.

4.4 electrode module

4.5 main module

4.6 mesh module

Mesh

class mesh.**Mesh1D**(*n_timestep*)
Bases: object

Mesh1D class

add_node(*x*)
Adds a node based on x location, returns new node

add_nodes(*length*, *n_elements*)
Add nodes of a length with n_elements

class mesh.**Mesh1D_SPM**(*n_timestep*)
Bases: *mesh.Mesh1D*

Mesh1D_SPM class, subset of Mesh1D

add_node(*x*, *initial_concentration*)
Adds a node based on x location, returns new node

add_nodes(*length*, *n_elements*, *initial_concentration*)
Add nodes of a length with n_elements

get_concentration_by_id(*node_id*, *timestep*)
Get concentration at a node, and return

4.7 node module

Node

class node.**Node**(*mesh*, *node_id*, *x*)
Bases: `object`

Node class

class node.**Node_SPM**(*mesh*, *node_id*, *x*, *initial_concentration*)
Bases: *node.Node*

Node_SPM class, subset of Node

4.8 plot module

4.9 solver module

CHAPTER

FIVE

USAGE

5.1 Installation

To use libsim, first install it using the following terminal command. This will pull the newest version of the libsim library.

```
$ git clone https://github.com/jerryzxiang/libsim.git
```

5.2 Modeling Batteries

TO DO

This is where we describe how to model batteries. How we use the libsim package.

CHAPTER

SIX

PROFILING

TO DO

CHAPTER

SEVEN

LESSONS LEARNED

Figure out good ways to verify tests when using a variety of parameters, as most of our work is looking at experimental data which is difficult to source.

TO DO…

CHAPTER

EIGHT

FUTURE WORK

8.1 Graphic User Interface (GUI)

In order to simplify the utilization of libsim, we hope to integrate a Graphic User Interface (GUI) to allow for an intuitive user experience as opposed to terminal commands.

8.2 Tests

With the number of models implemented into libsim, the expected number of edge cases and foreseeable issues will also grow. We hope to increase the number of tests built into the code in order to reduce the possibility for inaccuracies to be generated.

8.3 Expanding Models Library

The final result of this project allowed for a computationally inexpensive model of a reduced order that adapts well for real-time applications. In future versions of libsim, we will look to implement PSeudo 2 Dimensional (P2D) model aspects to improve accuracy while maintaining time cost in mind.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX