PyBaMM machine learning parameter data parsing - Documentation

Elias Hohl

December 3, 2021

Abstract

This document explains how to supply PyBaMM with measured parameter data dependent of one or two variables.

0.1 Installation

The 2D parsing feature was included into the PyBaMM master branch. You will be able to use it when installing the latest version of PyBaMM from github:

```
# if you currently have another PyBaMM version installed:
python -m pip uninstall pybamm

# to install the newest development version from github:
python -m pip install --upgrade git+https://github.com/pybamm-team/
    pybamm
```

To be able to parse 2D measurement data, you will also need to download the tools for that:

```
git clone https://github.com/ehtec/pybamm-param-ml.git
```

0.2 Usage

0.2.1 Parsing 1D data

PyBaMM can parse 1D data using interpolation. A two-column CSV file called *graphite_volume_change_Ai2020.csv* can be supplied as a parameter as follows:

```
  parameter_values = pybamm.ParameterValues()
  parameter_values['Negative electrode volume change'] = "[data]
    graphite_volume_change_Ai2020"
```

The columns of the CSV file should be formatted like this:

```
independent variable 1, dependent variable
```

0.2.2 Parsing 2D data

PyBaMM was extended to parse 2D data using machine learning. Follow these steps to read data from a CSV file called *lico2_diffusivity_Dualfoil1998.csv*:

1. Launch the script *csv_model_trainer.py*. Supply the CSV path and leave all other settings at the default values. This will create a .pkl file containing the model, and a .json file containing the configuration, and another .json file containing the 2D data (the name of the second .json file will automatically be suffixed with _2D).

2. Supply the data to PyBaMM in a similar way as done for 1D data:

```
    parameter_values = pybamm.ParameterValues()
    parameter_values['Negative electrode volume change'] = "[
    ml data]lico2_diffusivity_Dualfoil1998_2D"

```

The columns of the CSV file should be formatted like this:

```
independent variable 1, independent variable 2, dependent variable
```

0.3 Working principles

Using the casadi 1D interpolant is pretty straightforward. However, this is not the case for 2D data. The 2D interpolant requires a perfect grid of measurement data, which is experimentally as good as impossible to generate. Therefore, a machine learning method (Random Forest Regressor) is used in a preprocessing step to create a model from the supplied data, which can be used to evaluate a grid of input data for the casadi 2D interpolant:

```
regressor = RandomForestRegressor(n_estimators=100, random_state=0)

regressor.fit(input_data[:, 0:-1], input_data[:, -1])
```

```
X = list(np.meshgrid(*x_))

x = np.column_stack([el.reshape(-1, 1) for el in X])

y = self.model.predict(x)
```