Comparison of different Data Driven Approaches for Li-Ion Battery Modeling

Aman Tiwary
Roshan Kaanth M K S
Sai Sourabh Burela

UNIVERSITY *of* WASHINGTON

Battery Modeling

- Li-Ion batteries are common in Electric vehicle battery packs nowadays
- Like many other dynamical systems, the electrochemical reactions are governed by multiple high dimensional differential equations
- This makes deriving a model of the battery a cumbersome task
- The advent of data-driven approaches for dynamical systems comes as a huge relief

Goal

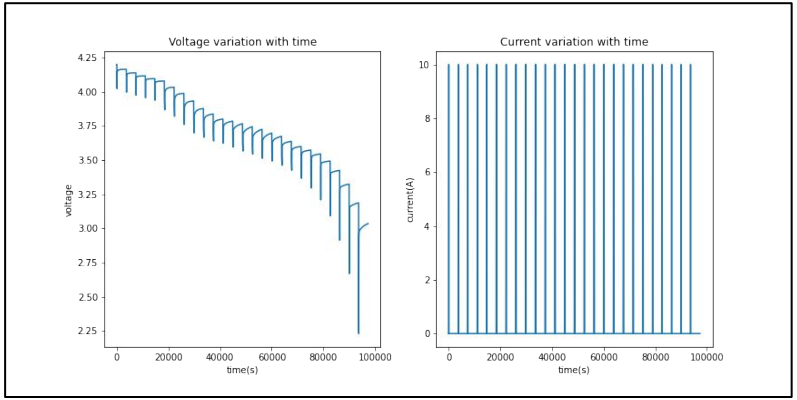
The goal of the project is model discovery of Li-Ion cells using both black and grey box modeling approaches

We are comparing and analysing

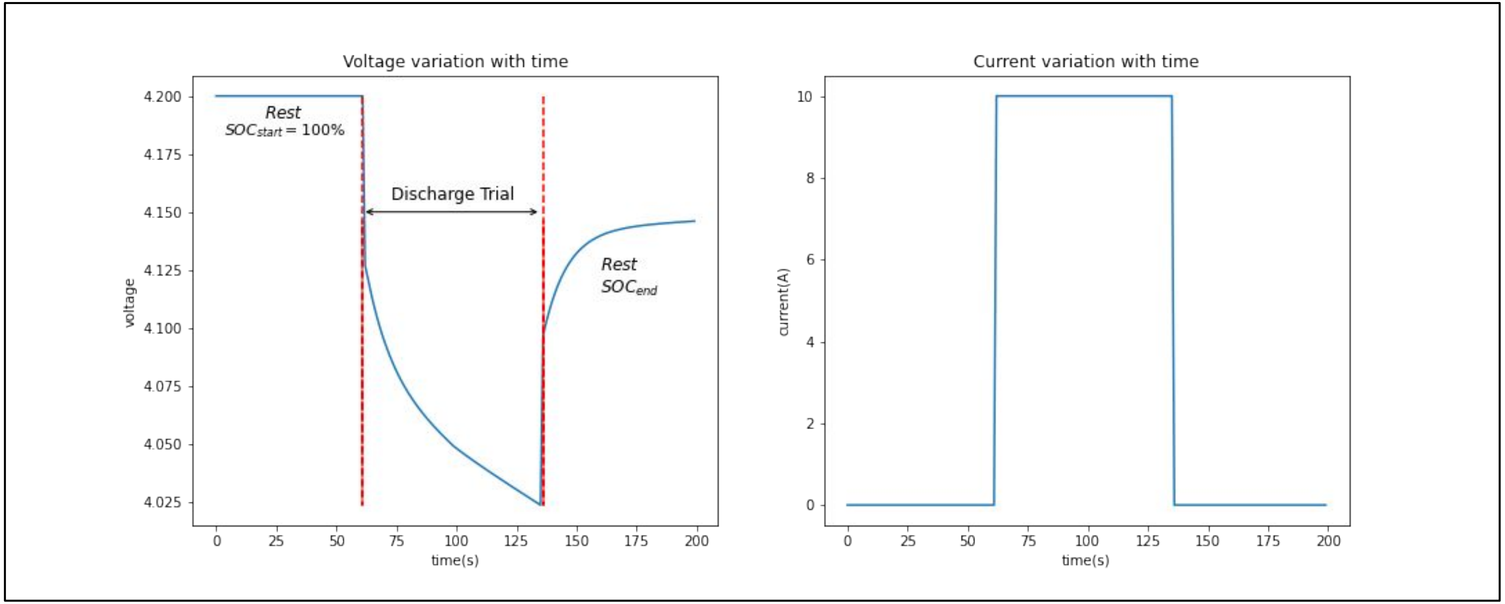
- SINDy - black box modeling
- Gaussian Process Regression - Grey box
- Genetic Algorithms - Grey box
- Regression - Grey box

Pulse Discharge Experiment

- In this project, we are taking a data set of a Li-ion battery model which is obtained using a Pulse discharge experiment. The experiment involves applying a current discharge pulse for a certain period of time on a cell at regular intervals with a resting period in-between for measuring the cell voltage with a predefined sampling rate. The test is performed from a fully charged to completely drained battery state.



Experiment



Voltage variation with time
Rest
$SOC_{start} = 100\%$
Discharge Trial
Rest
SOC_{end}
voltage
4.200
4.175
4.150
4.125
4.100
4.075
4.050
4.025
0 25 50 75 100 125 150 175 200
time(s)
Current variation with time
current(A)
10
8
6
4
2
0
0 25 50 75 100 125 150 175 200
time(s)

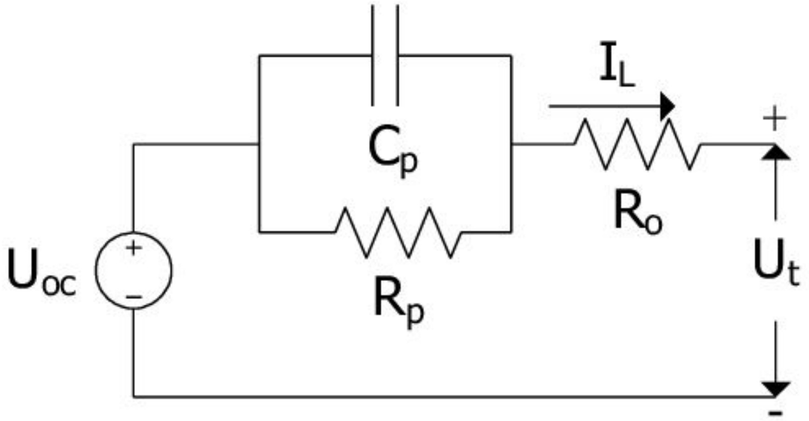
Equivalent Circuit Modeling

The governing equation

$$U_t = U_{oc} - I_L R_0 - (I_L\ R_P(1 - e^{-t/(RPCP)}))$$

where,

- U_{oc} = Open Circuit Voltage
- R_0, R_P = Resistance
- C_P = Capacitance
- U_t = terminal voltage
- I_L = input current



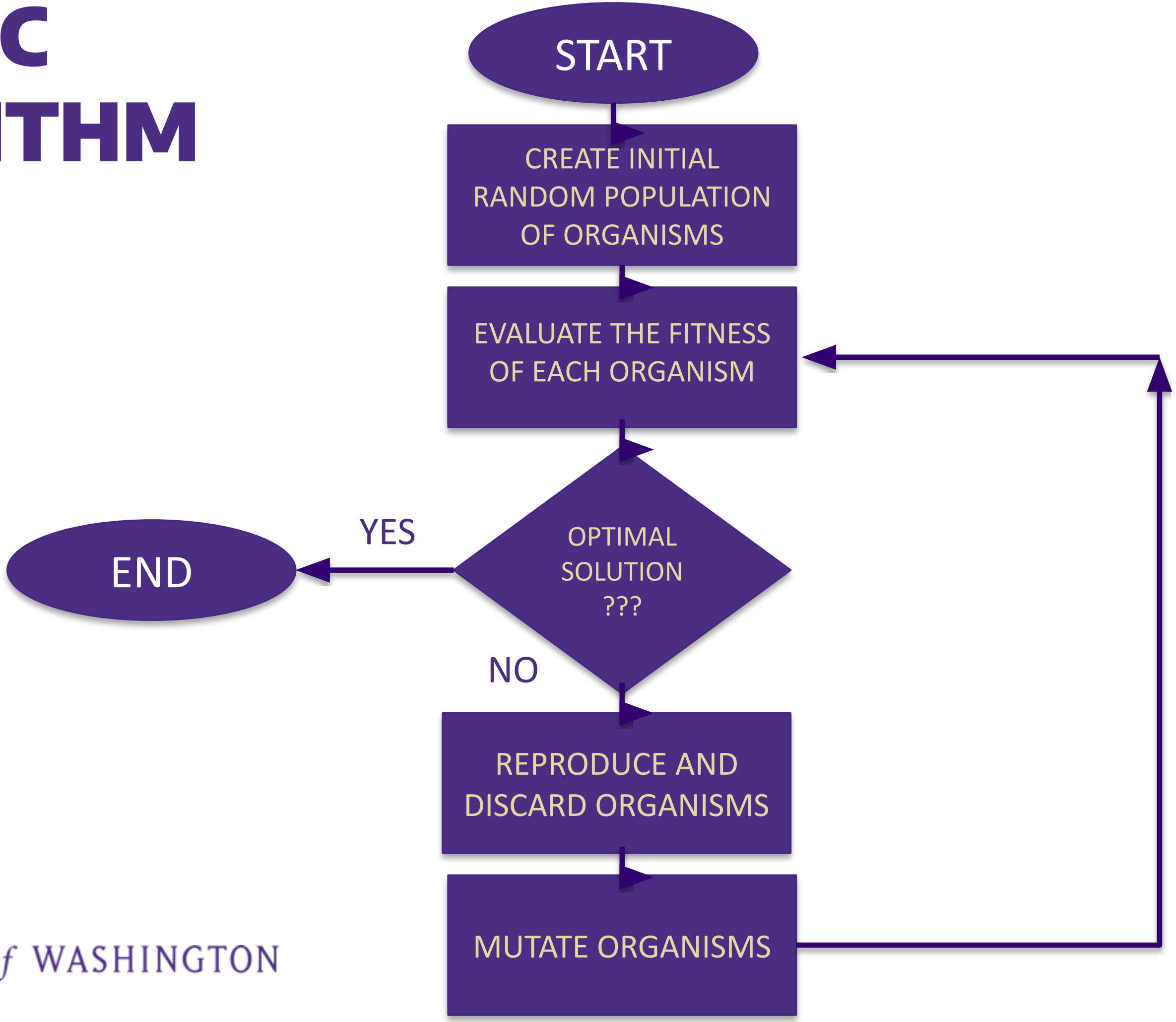
GENETIC ALGORITHM

- Genetic Algorithm (GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems
- Class of Evolutionary algorithms, that use techniques inspired from evolutionary biology
- Developed to:
  - Understand the adaptive processes of natural systems
  - Design Artificial system to retain the robustness of natural system

GENETIC ALGORITHM

- Basic idea is based on “Survival of the fittest”
- Provides effective techniques for optimization and machine learning applications
- Useful when search space is too large or complex
- Examples in real world GA applications:
  - Antenna Design, Turbine engine design, Network Design, Data Mining, Control systems design

GENETIC ALGORITHM



GENETIC ALGORITHM

```
def Genetic_Algorithm():
    initialize population
    evaluate the fitness of each individual in the population
    while optimal solution not found do:
        select fittest (best ranking) parents for reproduction
        perform mutation/crossover to breed new generation
        re-evaluate the fitness of the new population
    return (optimal solution)
```

GENETIC ALGORITHM

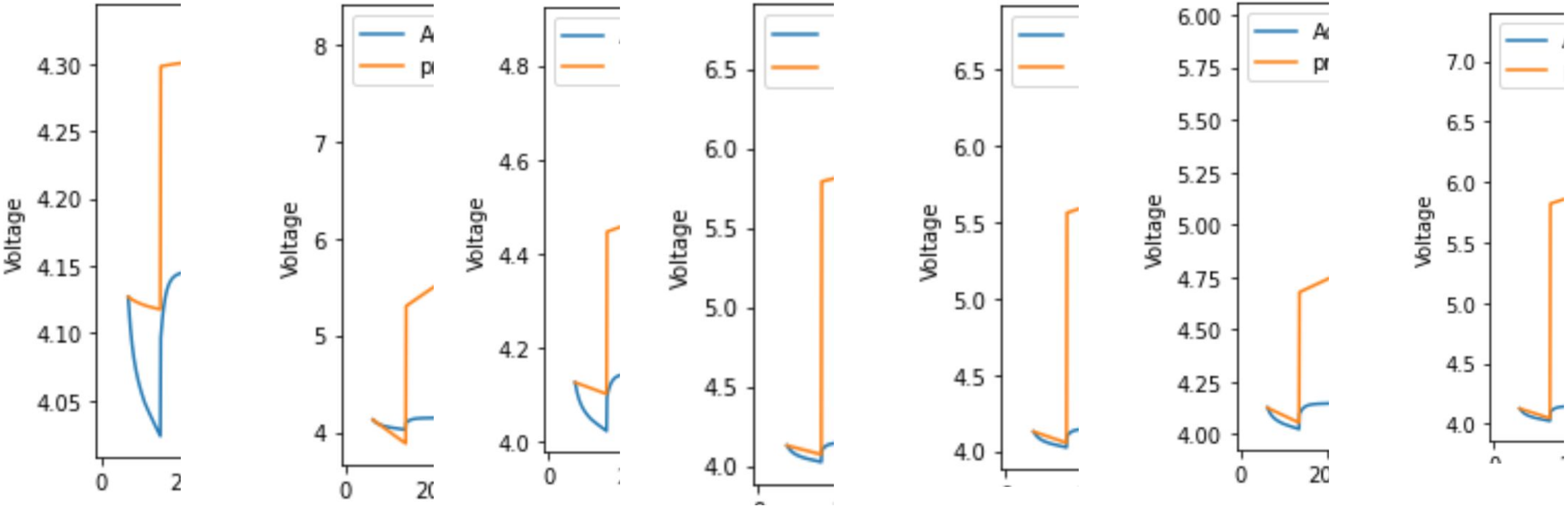
> **Battery Model**

```python
def battery(Em,R_0,R_1,C_1,R_2,C_2,R_3,C_3):
    '''Equation of the model for the voltage is

        Em - I*R_0-(+(I*R_1*(1- np.exp(-t/(R_1*C_1))))
                    +(I*R_2*(1- np.exp(-t/(R_2*C_2))))
                    +(I*R_3*(1- np.exp(-t/(R_3*C_3))))) - v_data[t]
    '''
    for t in range(62,1500):
        if 61<t<136:
            # for the time between t=62 and t=136 , the current I is 10 Amperes
            return Em -10*R_0- (+(10*R_1*(1- np.exp(-t/(R_1*C_1))))
                                +(10*R_2*(1- np.exp(-t/(R_2*C_2))))
                                +(10*R_3*(1- np.exp(-t/(R_3*C_3))))) - v_data[t]
        else:
            # for all other time instances , the current I is -10 Amperes
            return Em +10*R_0- (-(10*R_1*(1- np.exp(-t/(R_1*C_1))))
                                -(10*R_2*(1- np.exp(-t/(R_2*C_2))))
                                -(10*R_3*(1- np.exp(-t/(R_3*C_3))))) - v_data[t]
```

GENETIC ALGORITHM

Solutions of each generation



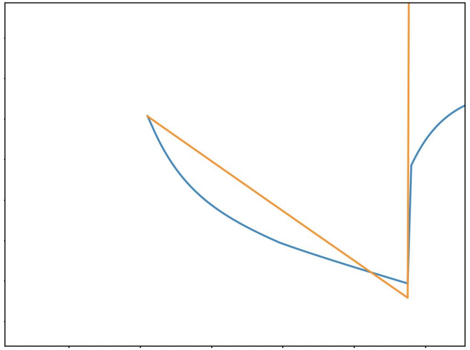
GENETIC ALGORITHM

Results

E_m R_0 R_1 C_1

```
====gen0 best solutions===
(array([5795.37993376]), (4.312862233015471, 0.008998123942075331, 0.8242315014070988, 47234.95057604644,
0.5715385482453891, 8952.559374074575, 0.12797498520881034, 43311.63512766412))
```

R_2 C_2 R_3 C_3



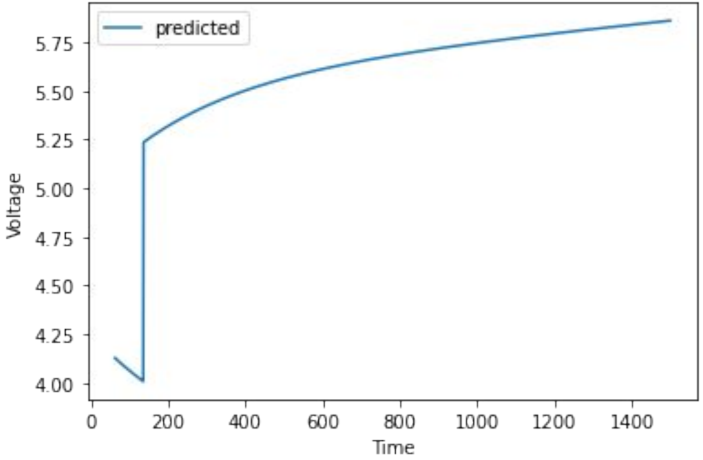
GENETIC ALGORITHM

> Generations of solutions

E_m R_0 R_1 C_1

```
====gen23 best solutions===
(array([795.03570759]), (4.569385234542004, 0.03637249235303941, 0.6183143238353149, 33092.443575661164, 0.
5816582703919699, 28911.1598408088, 0.8903371089946931, 16532.519996790288))
```

R_2 C_2 R_3 C_3



UNIVERSITY of WASHINGTON

GENETIC ALGORITHM

- Able to get approximate solutions using Genetic Algorithm
- Best solution tricky to obtain but not impossible
- Genetic Algorithm combined with Neural Networks will give best optimal solution but very slow in computation
- Efficient optimization algorithm with endless opportunities to solve difficult problems

Sparse Identification of Non-Linear Dynamics (SINDy)

A model discovery technique which identifies non-linear relationships in dynamical systems

<u>Algorithm is as follows</u>:

Input: library of functions of our choice

Output: weights of the functions we gave as input

A typical SINDy model finds the relationship between a state vector (with its combinations) and its derivative

In our model, we tried to equate different combinations of input features to an output vector

SINDy Algorithm

- Define the current input and true voltage output
- Define the sample time interval
- Input the library of functions
- Call the function used to calculate the parameter vector
- Calculate the predicted voltage value using the parameter vector and library of functions
- Plot the true and predicted voltage vs time
- Calculate the mean squared error

```
l=1439;
b=zeros(l,1)+10; % input current
t=1:1:l; % sample time interval
t=t';
theta = [b b.*(1-exp(-t/80)) b.*(1-exp(-t/800)) exp(-t/500) log(t)]; % library
lambda=0.023; % regularization parameter
n=1;
Xi = sparsifyDynamics(theta,volt,lambda,n); % calculating weights
syms ti I;
Thetasym=[I I*(1-exp(-ti/80)) I*(1-exp(-ti/800)) exp(-ti/500) log(ti)];
Thetasym=Thetasym';
vpredi=transpose(Xi)*Thetasym;
J=0;
vpredic=1:1:l;
vpredic=vpredic';
for i=1:l
    vpredic(i)=subs(vpredi,[ti,I],[i,b(i)]); % calculating predicted voltage
    J= J + ((vpredic(i)-volt(i))^2/l);  % calculating mean-squared error
end
```

Results

Here is the governing equation we obtained from the algorithm,

$$\text{Voltage} = w_0 + w_1 * I *(1-e^{-t/80}) + w_2 * I*(1-\exp{-t/800}) + w3\exp{-t/500} + w4\log(t)$$

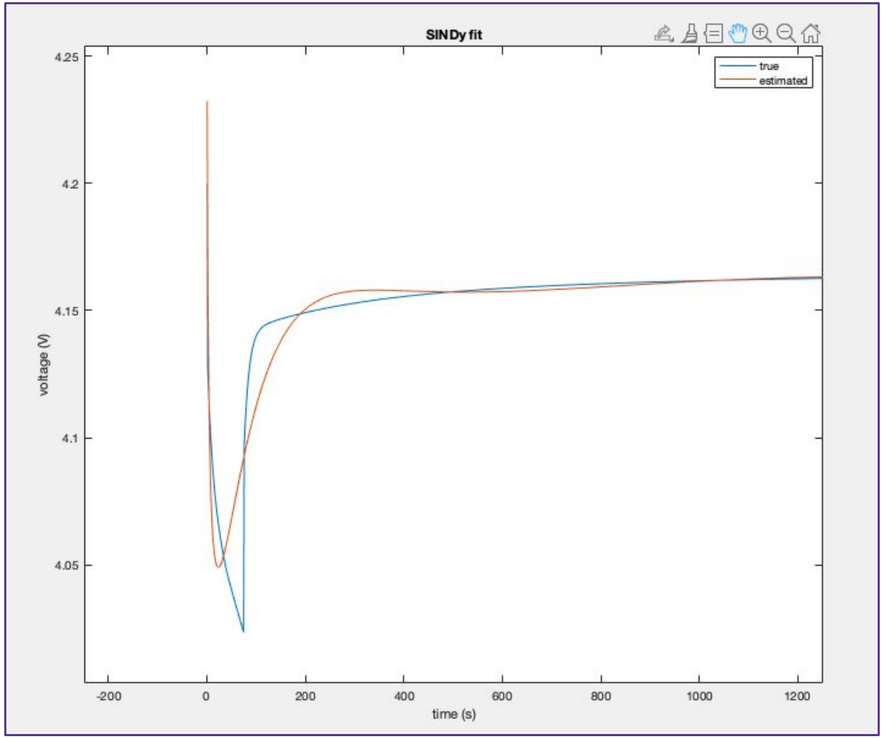
where w0 , w1 , w2 , w3 , w4 are the parameters obtained from SINDy algorithm and t denotes sample time.

The values obtained are

w0 = 0.41 , w1 = 0.038 , w2 = 0.03 , w3 = 0.08 , w4 = - 0.089

Model fit and Observations

- The figure on the right shows the fit of the SINDy model
- We can see that the fit is not perfect but follows the pattern and the MSE = 8.26e-05 sq volts
- SINDy is not able to capture the entire dynamics probably due to the fact that the input is a constant in this case

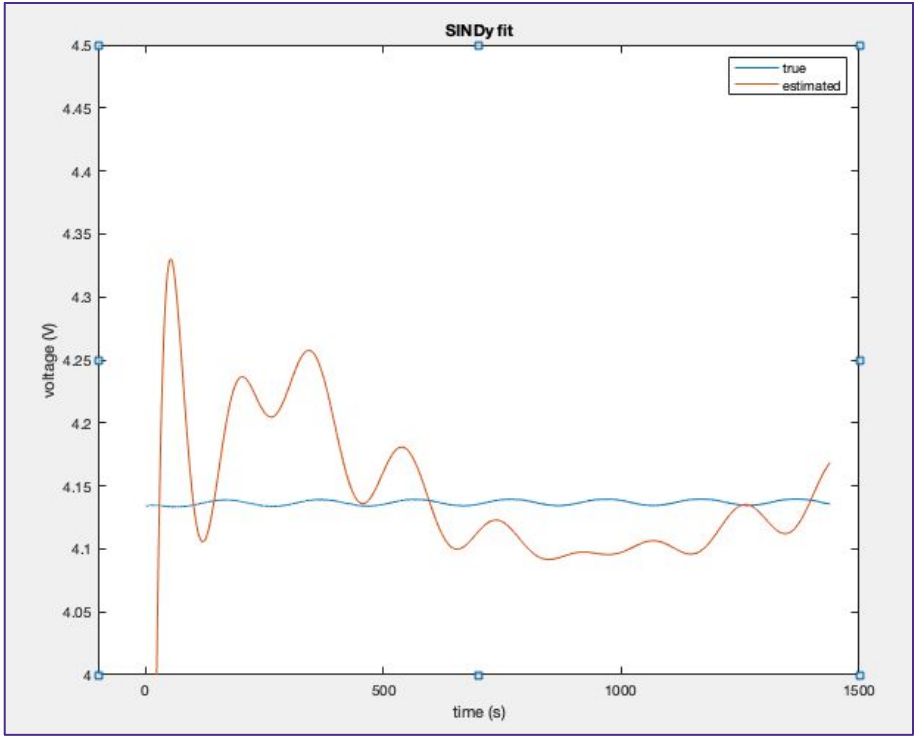


Validation

- We couldn't obtain the test data for the problem in hand
- So we used MATLAB's System Identification toolbox to generate an 87% fit Hammerstein - Wiener Non-linear model to generate the test data using a sinusoidal current profile of frequency 200 Hz

SINDy validation

- The figure on the right shows the fit of the SINDy model for test data
- We can see that the fit is not looking great, but tries to follow the pattern and the <u>MSE = 0.0168 sq volts</u>



Regression with SGD and Levenberg - Marquardt Algorithm

- Stochastic Gradient Descent is an extension of Gradient Descent which computes gradient of random input data points, thus reducing computational expense
- Levenberg - Marquardt algorithm is a combination of gradient descent with Newton - Raphson method which calculates the hessian matrix in addition to the gradient matrix
- Unlike SINDy which finds hidden dynamics, this algorithm requires a governing equation to be input and finds weights

Algorithm

This is our governing equation (equivalent circuit model):

$$V_{batt} = OCV - I*R0 - I*R1*(1-e^{-t/(R1*C1)}) - I*R2*(1-e^{-t/(R2*C2)}) - I*R3*(1-e^{-t/(R3*C3)})$$

- Here is the basic equation of Levenberg Marquardt algorithm we used with SGD:

$$x_{n+1} = x_n - [\nabla^2 f(x_n) + \lambda * I]^{-1} * \nabla f(x_n)$$

where

x_{n+1} - vector of parameters at the next time step

x_n - vector of parameters at the current time step

$\nabla^2 f(x_n)$ - hessian matrix

λ - Learning rate

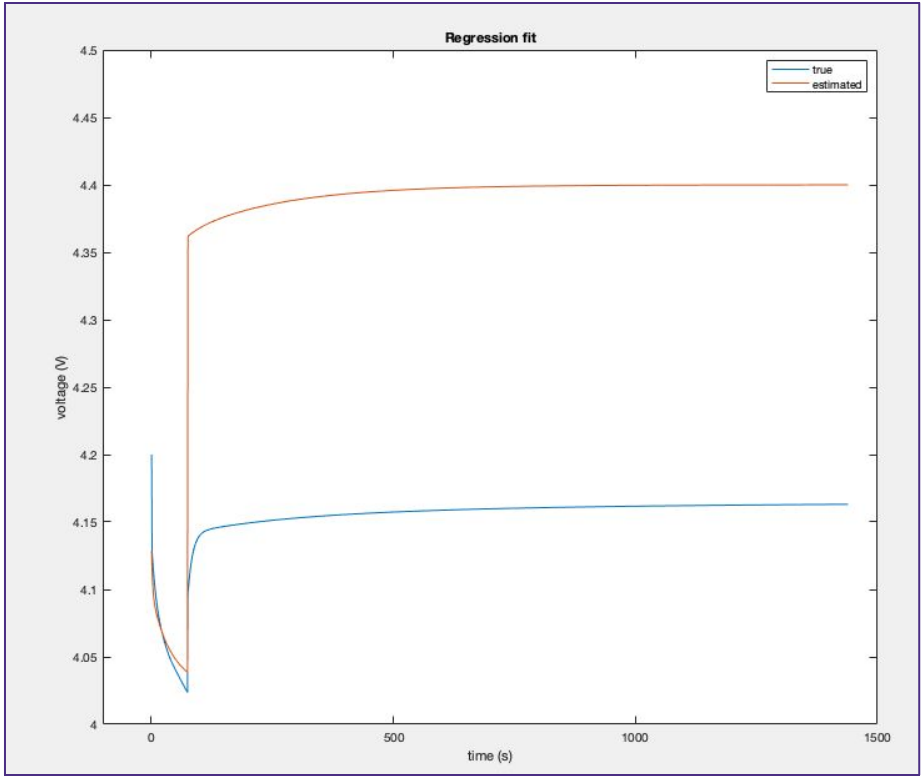
$\nabla f(x_n)$ - gradient

Algorithm

- Define the governing equation, current input and true voltage output
- Define the cost function
- Define the mini-batch of random points for stochastic gradient descent algorithm
- Calculate the Hessian and gradient
- Execute the Levenberg-Marquardt algorithm
- Use the obtained parameters to calculate the predicted voltage output
- Plot the true and predicted output with time
- calculate the mean squared error

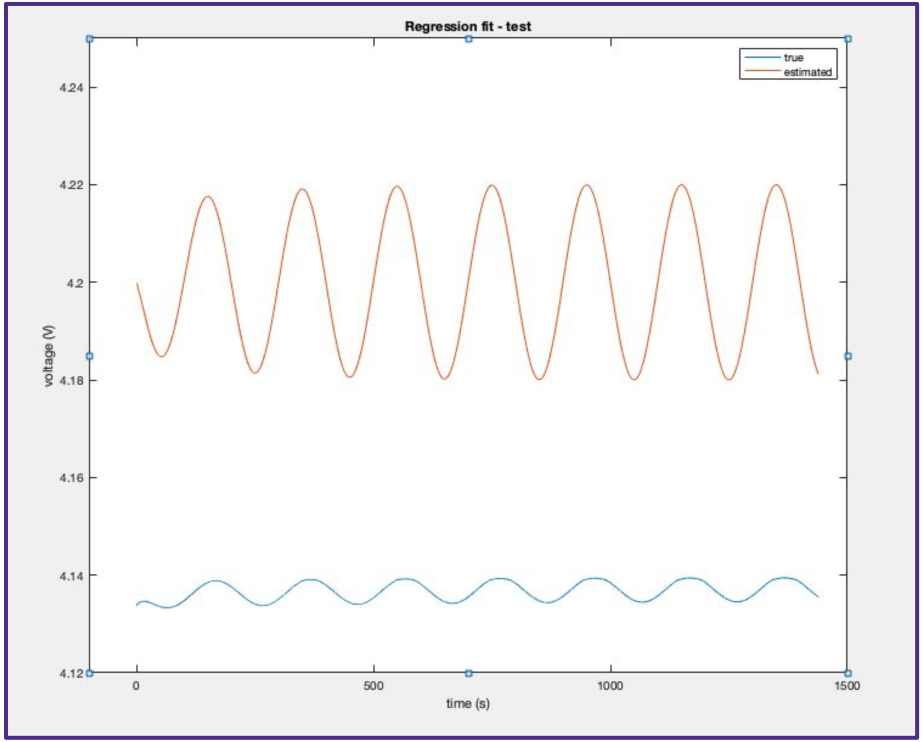
Results and Inference

- This is the best fit we could get with MSE = 0.0533 sq volts
- Fit largely depends on the initial guess of weights
- Initial guess is done using known physics of the system
- Even a full gradient descent doesn't provide better results because of the local minima issue



Regression validation

- The figure on the right shows the fit of the Regression model for test data
- The fit follows the same sinusoidal pattern as the true test data with an offset and <u>MSE = 0.0041 sq volts</u>



Gaussian Process Regression

What is the objective?

- It allows us to make predictions about our data by incorporating prior knowledge.
- Fully Specified by training data, mean and covariance functions
- Covariance is given by “kernel” which measures distance of inputs in kernel space

Gaussian Process Regression

Definition

- **Given Inputs (X) and output (y):**

$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} = (\mathbf{X}, y)$$

- **We presume that there is some structure in our data D. GPs model the output as a noisy function of the inputs:**

$$y_j = f(X_j) + \varepsilon \text{ ; where } \varepsilon \sim N(0, \sigma_n^2)$$

- **Given a set of inputs (X), GP models the outputs (y) as jointly gaussian:**

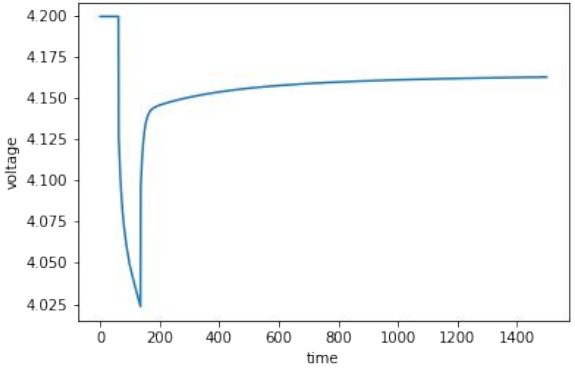
$$P(y \mid X) = N(\underbrace{m(X)}_{\text{Mean}}, \underbrace{K(X, X)}_{\text{Kernel}} + \underbrace{\sigma_n^2 I}_{\text{Noise}})$$

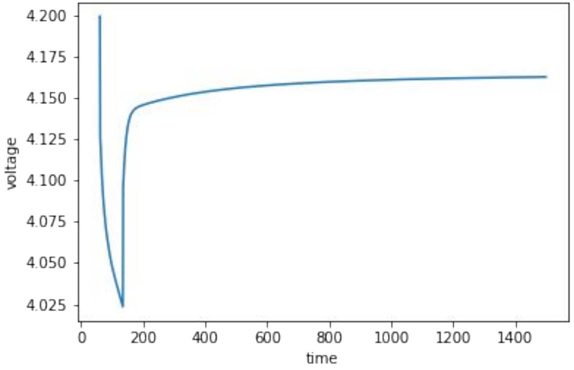
- **Usually we assume prior mean to be zero**
- **Covariance matrix *K* is defined through kernel function. Example: Squared exponential or RBF, Marten, Periodic exponential**

Gaussian Process Regression

Training Data

- For our system, current and time are the input parameters. Our Input matrix $X \in \mathbb{R}^{n \times 2}$ and our output $V \in \mathbb{R}^{n}$ where n is the size of the dataset.
- We are working on the first discharge pulse, n = 1500. As from t = 0s to t = 61s the battery is left ideal, we remove this portion from our analysis.

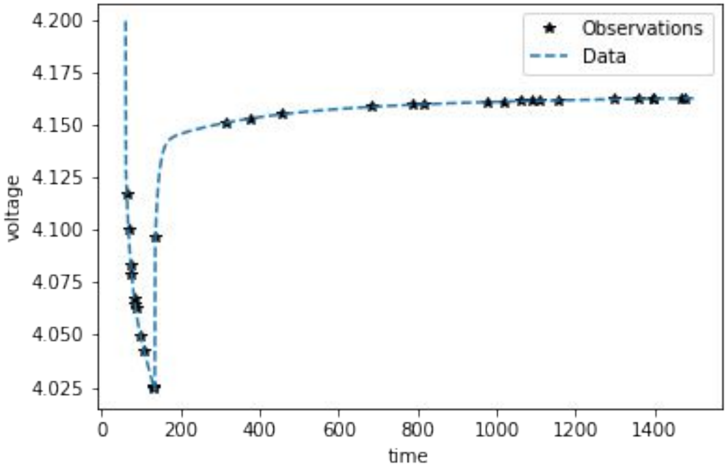




Gaussian Process Regression

Training Data

- First, Sample 30 points from the data randomly.
- Added a bias so that there are some points sampled from the region when the battery is being discharged
- These samples are our observations.
- We take all of the data as our test points
- Setup up input matrix X_{train} and X_{test} by vertically stacking [i(t), t]



Training Data

Gaussian Process Regression

Setting up Kernel Function

- To generate Covariance matrix, we evaluate kernel pairwise on all the points

$$\mathbf{K} = \begin{pmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & & \\ \vdots & k(x_i,x_i) & \vdots \\ & & \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{pmatrix}$$ **

Radial Basis Function kernel (RBF kernel)

- $\mathbf{K}$ = **σ** exp(- || x - x' ||^2 / 2 ℓ^2), where
    - ℓ is length scale
    - **σ** is signal variance

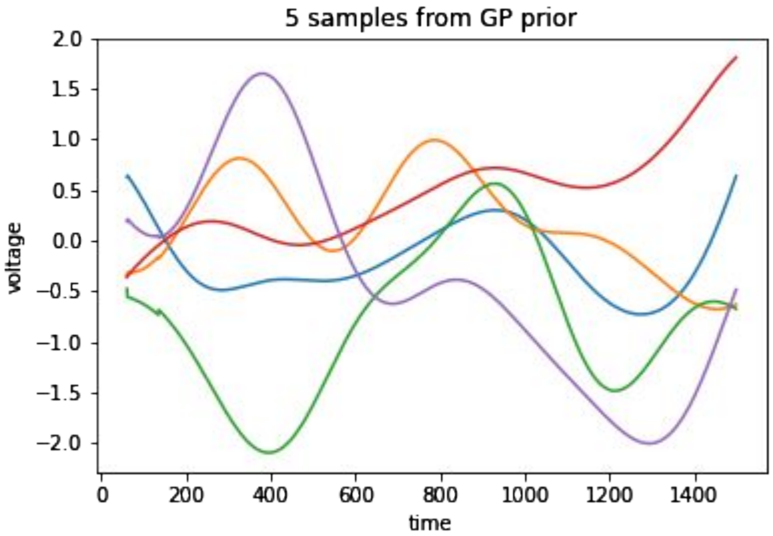
Marten Kernel

- $\mathbf{K}_{v=3/2}$ = (1 + √3 r/ ℓ) exp(- √3 r/ ℓ)
    - r is euclidean distance
    - ℓ is length scale

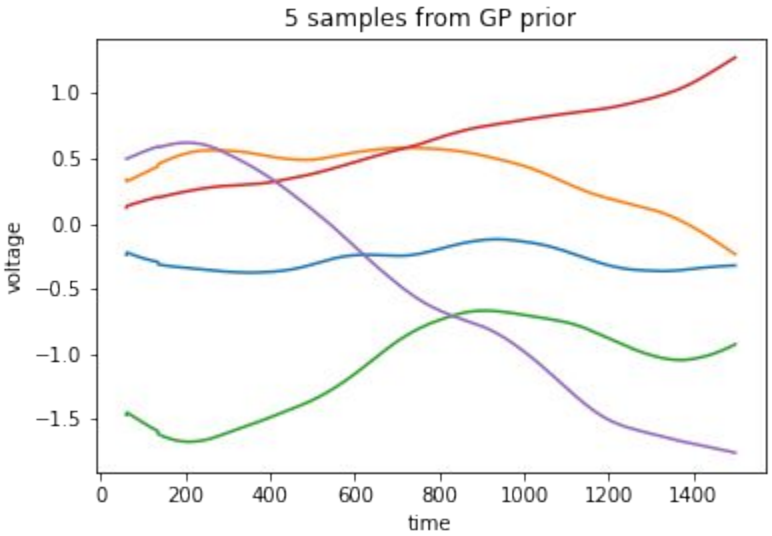
** taken from Probabilistic Robotics text book

Gaussian Process Regression

Functions Sampled from Prior



Prior using RBF kernel



Prior using marten kernel

Gaussian Process Regression

Prediction

- Recall, GP models the output y as jointly gaussian.
- Given training data set D = $\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), . . . , (\mathbf{x_n}, y_n)\} = (\mathbf{X}_{train}, y_{train})$ and test data set (x_*, y_*),

$$P(y_{train}, y_* \mid \mathbf{X}_{train}, x_*) = N(\mu, \mathbf{\Sigma})$$

- Condition on y: As the outputs are jointly gaussian, marginalisation on the known outputs would also result in a gaussian.

$$P(y_* \mid x_*, y_{train}, \mathbf{X}_{train}) = N(\mu_*, \mathbf{\Sigma}_*),$$

where μ_*, $\mathbf{\Sigma}_*$ are the mean and covariance of the functions and are given by:-

$\mu_* = K^T_* (K + \sigma^2_n I)^{-1} y_{train}$ *mean of predicted functions*

$\mathbf{\Sigma}_* = K_{**} - K^T_* (K + \sigma^2_n I)^{-1} K_*$ where, *Covariance of predicted functions*

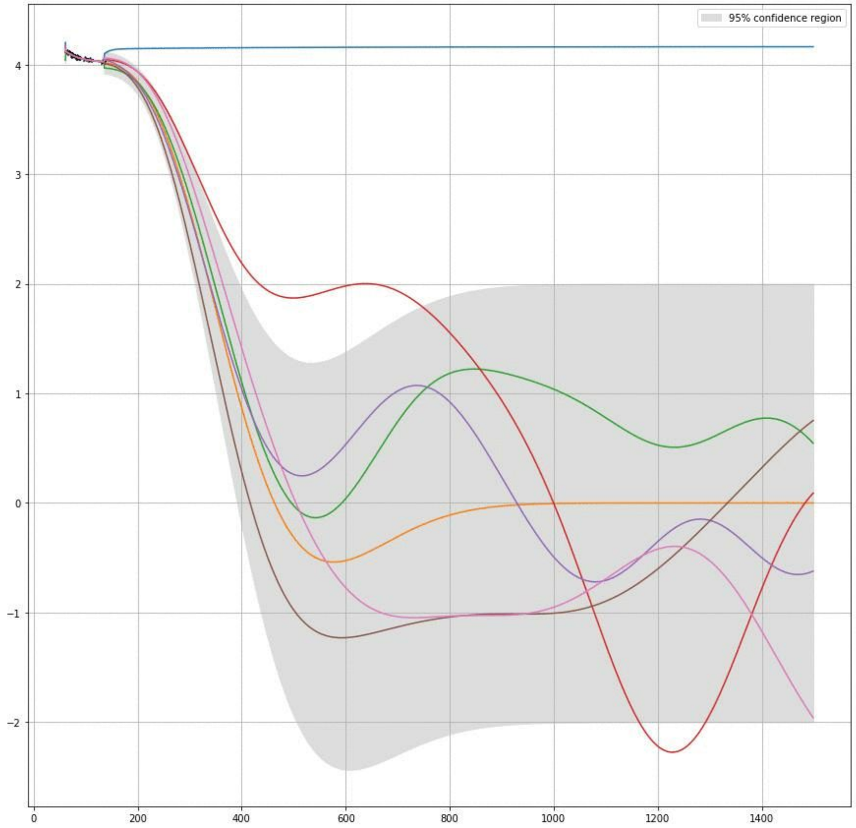
K_* = captures relation between training points and test points

K_{**} = captures relation among test points

Gaussian Process Regression

Prediction

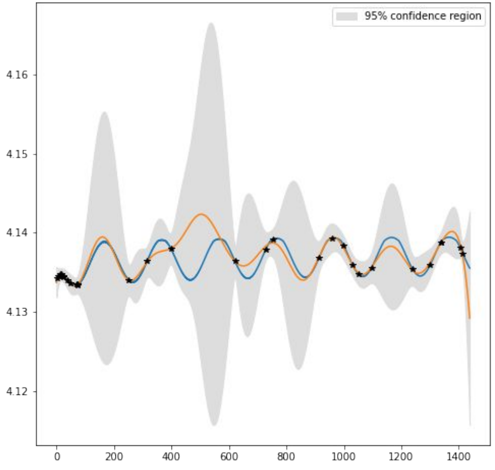
- Prediction using RBF kernel



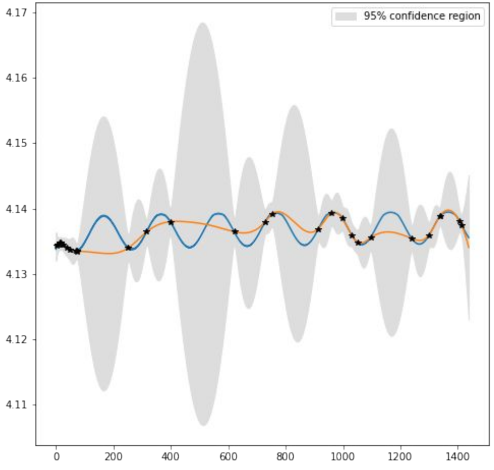
UNIVERSITY of WASHINGTON

Gaussian Process Regression

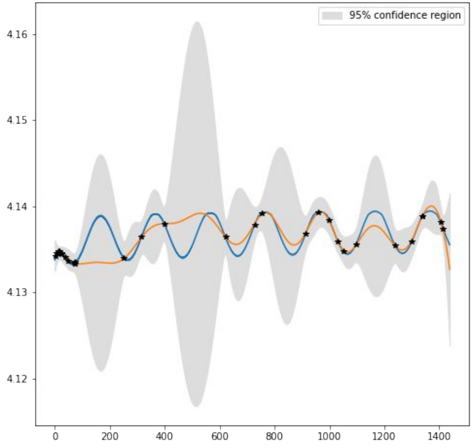
Validation



RBF kernel



Marten kernel v = 1.5



Marten kernel v =2.5

Conclusion

- In this project we explored 4 different data driven techniques for system identification or model discovery
- **SINDy** gives information about the system dynamics and is helpful to find the nonlinear dynamics
- **Genetic Algorithm** can be used to solve search and optimization problems efficiently
- **Regression Stochastic Gradient** is a powerful method for gray box model discovery
- **Gaussian Process Regression** gives a distribution over a set of functions. They allow us to make predictions about our data using prior knowledge.

References

1. Li-Ion Battery Pack Characterization and Equivalent Electrical Circuit Model Development – Ohuz H. Dagci and Ram Chadrasekaran
2. Genetic Algorithms for real parameter optimization- Alden H Wright
3. Data Driven Health Estimation and lifetime prediction of Lithium- Ion batteries- A review – Yi Li et al.
4. Multiobjective optimization using genetic algorithms- Abdullah Konak, David. W. Colt, Alice. E . Smith
5. High fidelity electrical model with thermal dependence for characterization and simulation of high power lithium battery cells- Tarun Huria, Massimo Ceraolo, Javier Gazzarri; Robyn Jackey

THANK YOU !!