Contents

1 Overview

This project involves developing the Kalman filter for a Li-ion battery state of charge (SOC) application. The Kalman filter involves combining a process estimate with a measurement to estimate the value of a state of the system. The big assumption with the Kalman filter is that the system state is observable. A block diagram of the Kalman filter is shown in Figure 1.

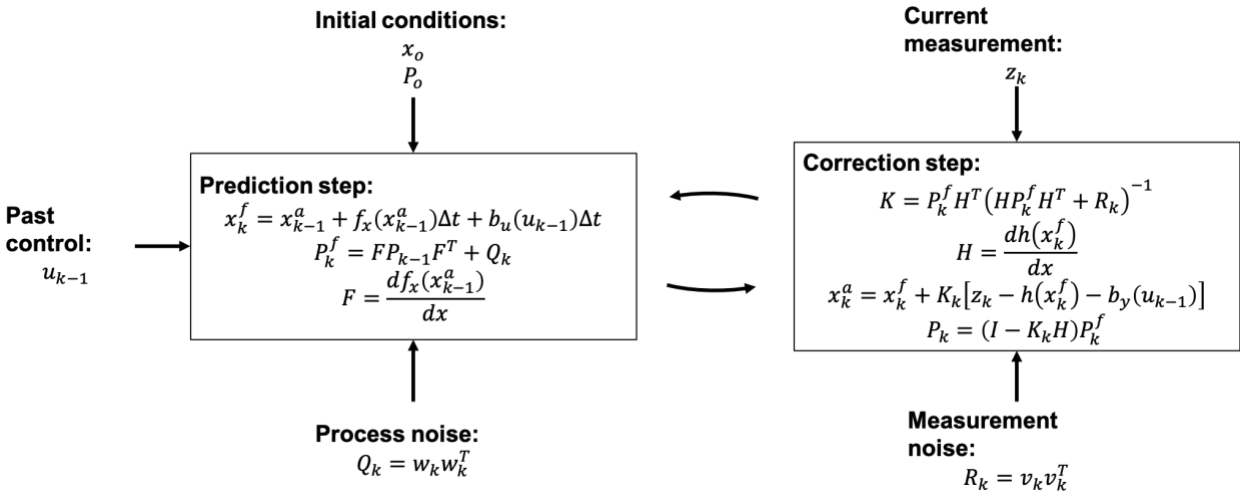

Figure 1: Kalman filter block diagram.

To implement the Kalman filter, we must have a model of the system. The modeling work was the most intense part of the project with plenty of opportunities for future improvements identified. Once the model was built, the derivation and implementation of the Kalman filter was fairly straightforward. The Kalman filter can always be further tuned to improve accuracy. Two of the big handles to tune the Kalman filter are the process and measurement covariance matrices. It can be non-trivial to find these. We take a very simplistic approach and assume everything is independent so that we don't have to worry about the coupling between states and errors.

For this application, we use a lithium ion battery with battery data supplied by the University of Maryland (https://calce.umd.edu/battery-data).

2 Model (Equivalent Circuit)

The first step of the project was to build an equivalent circuit model of the battery cell. The equivalent circuit model we use is shown in Figure 2.

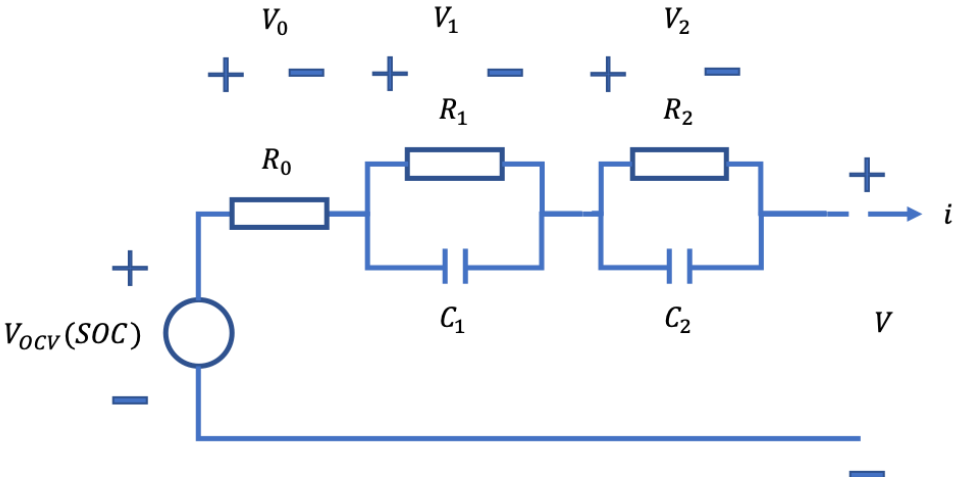

Figure 2: Li-ion equivalent circuit model.

The equivalent circuit model was chosen a little arbitrarily. From some basic research, we know that the open circuit voltage (OCV) of the cell is related to the SOC of the cell. This is our voltage source on the left of the schematic. We also expect the terminal voltage to differ from the OCV during operation. We include a resistance, R_0, to capture instant changes in terminal voltage due to an external current (current supplied by or to the battery). From the schematic, we see that our convention is positive current for when the cell is discharging and negative current for charging (maybe the reverse would have made more sense). We include two RC circuits to capture the dynamics of the cell: one RC circuit for fast dynamics and another for slow dynamics. This is capture with a resistance, R_i, and capacitance, C_i, where $i = 1,2$.

Figure 3 shows why we have a single resistance (R_0) and RC circuits (we would need at least one RC circuit). We see that when a current is applied, the terminal cell voltage instantaneously drops (the R_0 will capture this) with some dynamics, particulary when the current is unapplied and the voltage dynamics take some time to settle.

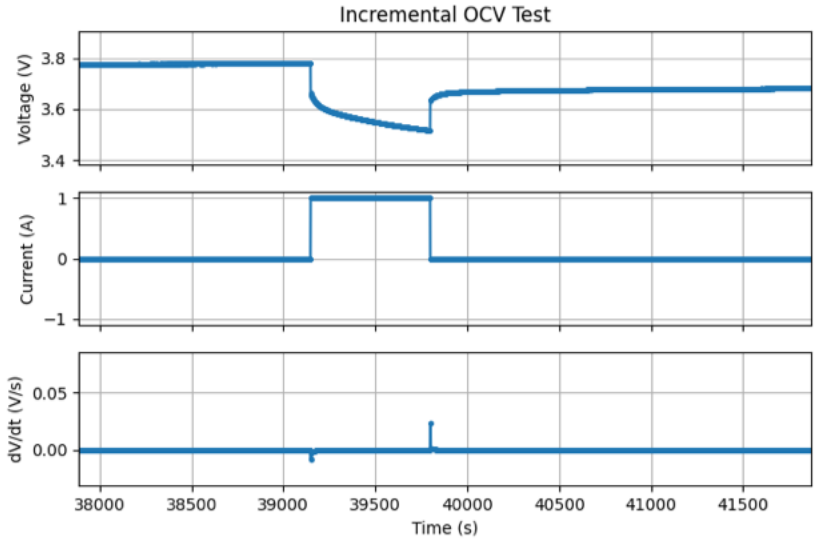

Figure 3: Part of an Incremental Current/OCV test.

In our approach, we break the modeling down into two steps:

1. We first want to find the OCV vs SOC relationship
2. We then want to find our other model parameters: $R_0, R_1, C_1, R_2, and\ C_2$

2.1 OCV vs SOC Relationship

The battery data gives two data sets for finding the OCV vs SOC relationship:

- Low current test – in this test, a low current is applied to cell to discharge and then charge, the current is low enough where you can assume that any internal dynamics of the cell are small and can be neglected
- Incremental current test – in this test, a current is applied incrementally to the cell to charge and discharge the cell where the cell voltage is allowed to settle during increments

We explore the OCV vs SOC relationship using both methods. The battery data also includes charge and discharge capacity data where we assume this gives the actual SOC of the battery. This is an assumption. I don't know exactly how the charge and discharge capacities are calculated.

Examples of the low current test and incremental current/OCV test are shown in Figure 4 and Figure 5, respectively.

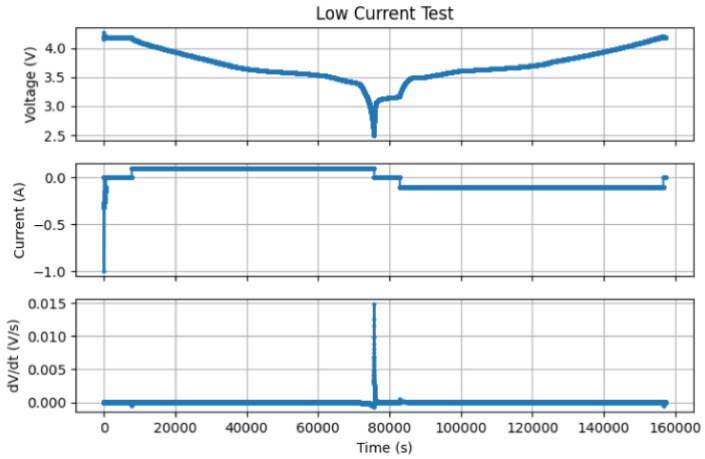

Figure 4: Low Current test.

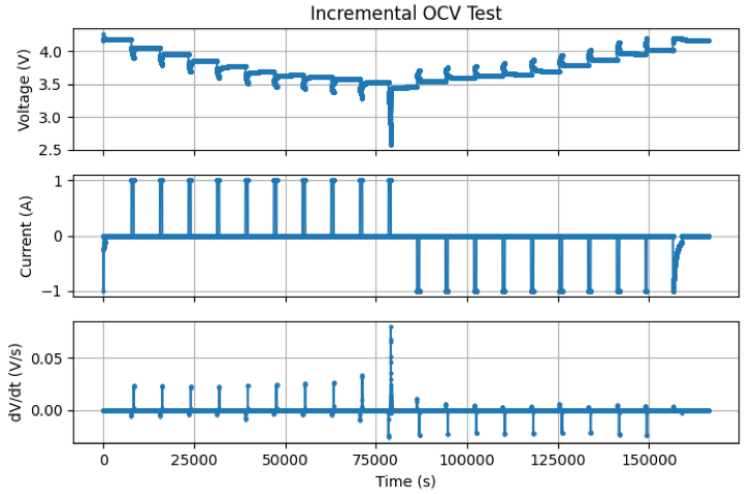

Figure 5: Incremental Current/OCV test.

Some of the data for the Low Current test was not usable. It's possible the data got corrupted somehow. All of the Incremental Current/OCV test data was good. This is summarized in Table 1.

Table 1: Data summary for Low Current and Incremental Current/OCV tests.

Temperature (deg. C)	Sample	Low Current test	Incremental Current/OCV test
0	1	Good	Good
	2	Good	Good
25	1	Bad	Good
	2	Good	Good
45	1	Bad	Good
	2	Bad	Good

For the Low Current test data, we find the OCV vs SOC relationship for the discharge part of the cycle and then find it again on the charge cycle of the test. We then take the average to cancel out any dynamics (think of how we would cancel out the effect of R_0). This is shown in Figure 6.

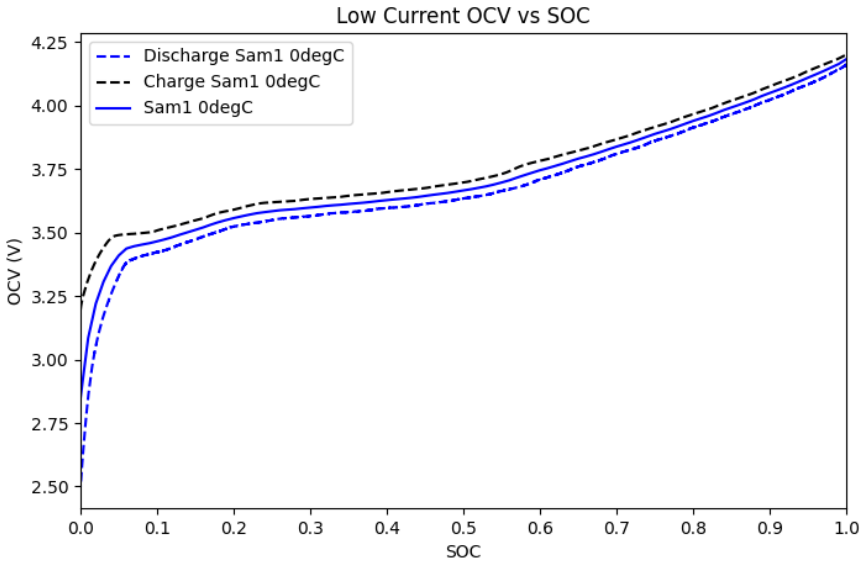

Figure 6: OCV vs SOC relationship from Low Current test for Sample 1 at 0 deg. C.

Figure 7 and Figure 8 show the OCV vs SOC and dOCV/dSOC vs SOC curves for the same data (sample 1 at 0 deg. C). Figure 8 trims the x-axis to start at 0.1 to better show shape from 0.1 to 1 SOC. We see that that the sharp drop off in OCV at low SOC leads to a sharp increase in dOCV/dSOC at those same low SOC values

We see that between 0.2 and 0.5 SOC, that dOCV/dSOC is small meaning that OCV doesn't change much in that region with a change in SOC which we also see when looking directly at the OCV vs SOC curve. This will be useful when using the Kalman filter which will be explained more later

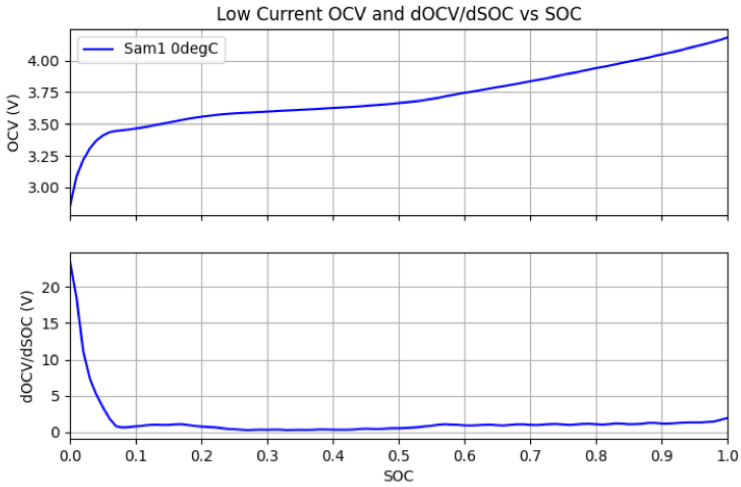

Figure 7: OCV and dOCV/dSOC vs SOC for Low Current test with Sample 1 at 0 deg. C.

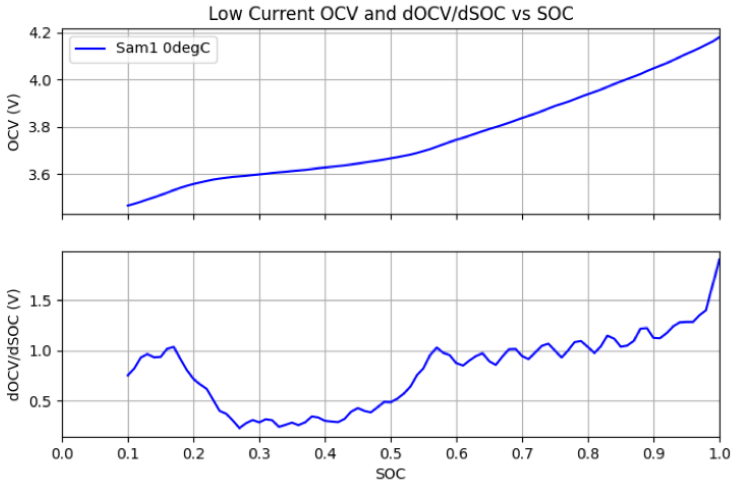

Figure 8: OCV and dOCV/dSOC vs SOC for Low Current test with Sample 1 at 0 deg. C (trimmed x-axis).

With the Incremental Current test data, the cell is discharged and charged incrementally. There is a long relaxation period where the cell voltage equilibrates and we use that measurement as the OCV. For the SOC, we again rely on the discharge and charge capacity measurements.

The Incremental Current test data doesn't give us a continuous relationship like the Low Current test (or at least a high-resolution relationship) and we use linear interpolation to fit the curve. This may not be totally appropriate as we can see that the relationship is not linear from the Low Current test data but we assume that linear interpolation is valid

We see that there is almost no difference between the charge and discharge OCV vs SOC curves for the Incremental Current tests. This is shown in Figure 9 for the same sample and temperature as shown in Figure 6.

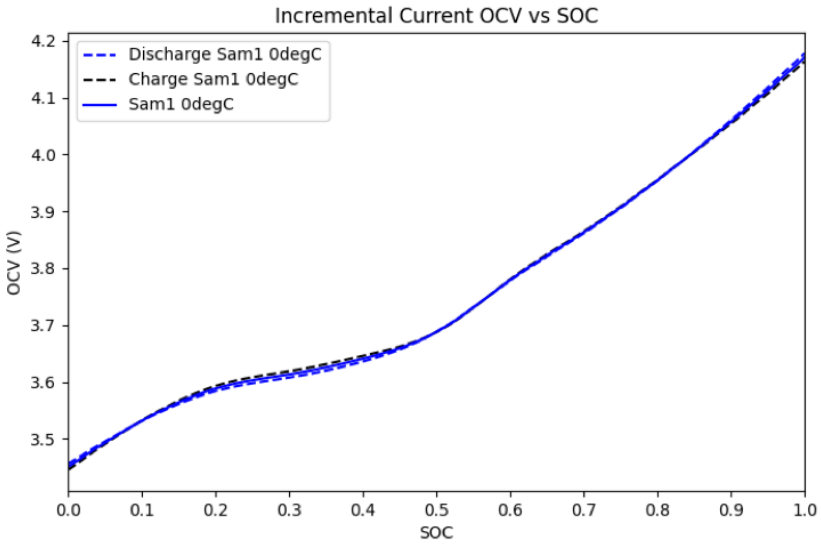

Figure 9: Incremental Current/OCV OCV vs SOC.

Compared to the Low Current OCV vs SOC curve, the Incremental Current OCV vs SOC curve doesn't result in the same sharp drop in OCV near 0 SOC. Looking at the Low Current and Incremental Current data, we do see a big recovery at the end of the discharge cycle. Since the Incremental Current data is

looking at the OCV after a sufficiently long time (long enough for the internal dynamics to settle), this can explain why the Incremental test data doesn't give the same shape at low SOC compared to the Low Current data. The OCV vs SOC and dOCV/dSOC relationship from the Incremental Current test for Sample 1 at 0 deg. C is shown in

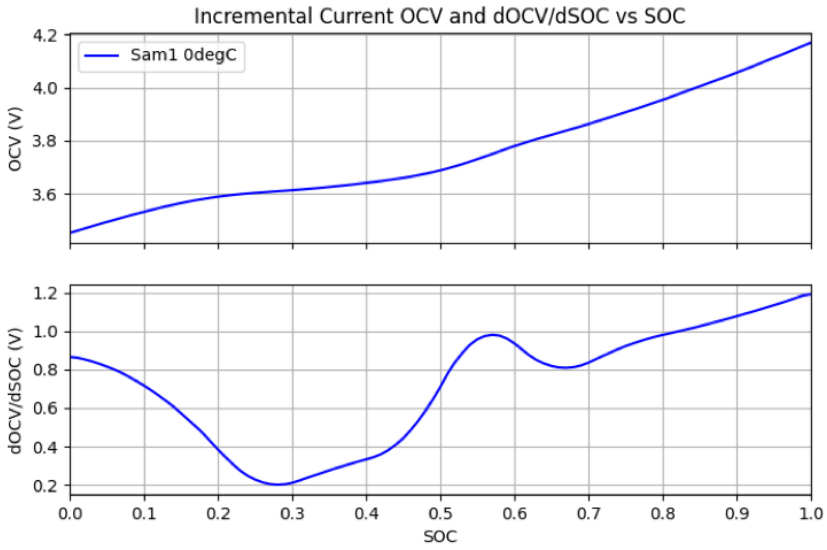

Figure 10: OCV and dOCV/dSOC vs SOC for Incremental Current test with Sample 1 at 0 deg. C.

In the SOC region between 0.2 and 0.4, we again see a small dOCV/dSOC region similar to the Low Current result which is due to the small OCV changes with respect to changes in SOC in that region

Figure 11shows all of the OCV vs SOC curves plotted together. Since the Low Current OCV vs SOC curves show a sharp drop at low SOC in OCV, we clip the curves to only show 0.1 < SOC < 0.9.

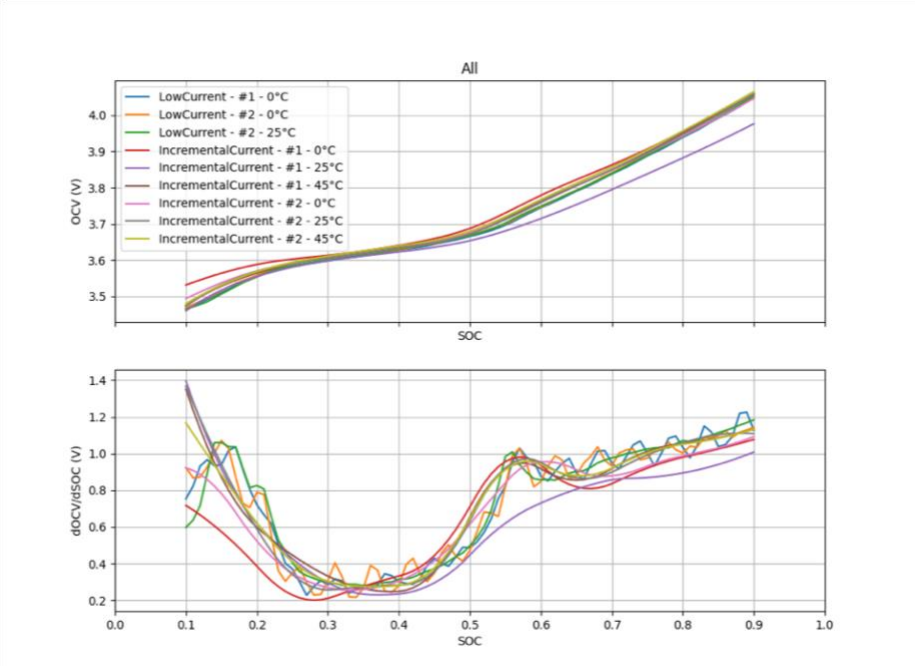

Figure 11: All OCV and dOCV/dSOC vs SOC data.

Generally, the curves are on top of each other. The Incremental Current Sample 1 0 deg. C is a bit of an outlier at low SOC and the Incremental Current Sample 1 25 deg. C is a bit of an outlier at high SOC

Looking at the dOCV/dSOC curve, it's more of the same with the Incremental giving a different shape at low SOC compared to the Low Current but again, that's most likely due to the sharp drop off in voltage for the Low Current test at low SOC.

We will look at a few more plots but the results here suggest that OCV vs SOC doesn't vary much with temperature or at least, that there is no noticeable trend with temperature.

Figure 12 shows the OCV and dOCV/dSOC relationship for the Low Current tests and Figure 13 shows the Incremental Current tests.

There isn't a big difference between the temperatures and the samples. We will take a closer look only at the Incremental Current results since we have more data for those tests but looking at the plots on the right, we don't see a big difference. We will also compare the difference among temperatures and samples to further evaluate how the OCV vs SOC relationship changes as a function of temperature

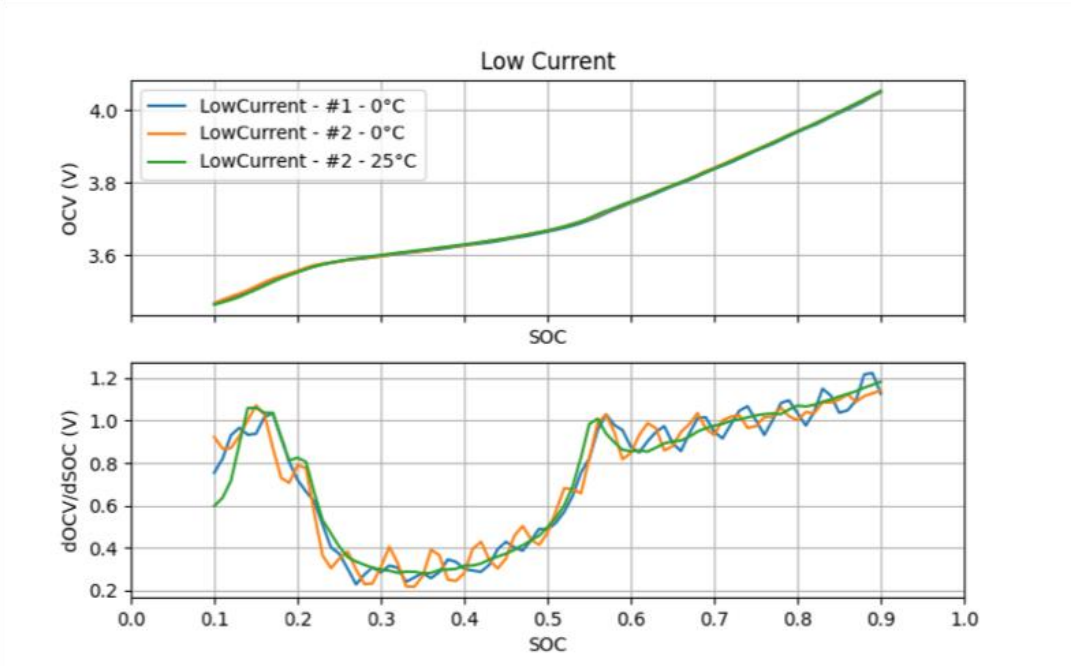

Figure 12: All Low Current test OCV and dOCV/dSOC vs OCV data.

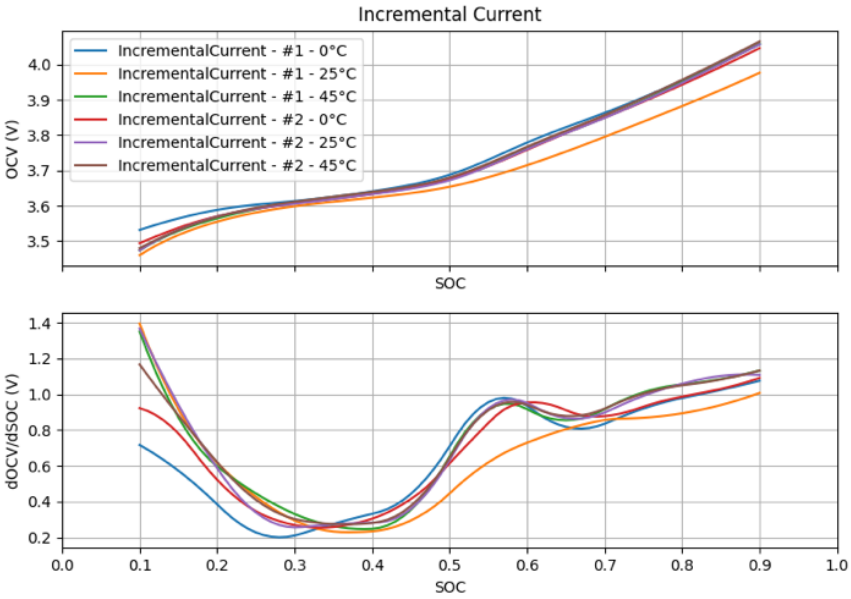

Figure 13: All Incremental Current test OCV and dOCV/dSOC vs OCV data.

Figure 14 shows the OCV and dOCV/dSOC data from the Incremental Current test for Sample 1 and Sample 2 while Figure 15 shows the same data but by temperature.

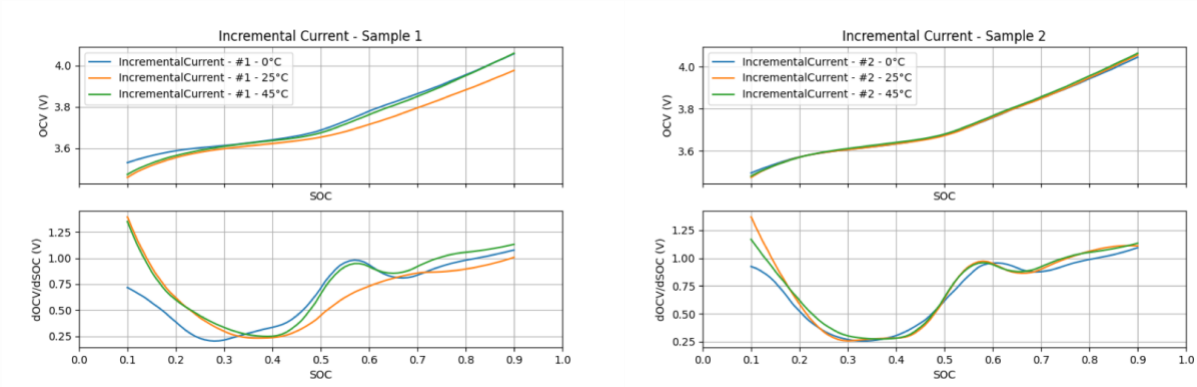

Figure 14: Incremental Current OCV and dOCV/dSOC vs OCV for Sample 1 (left) and Sample 2 (right).

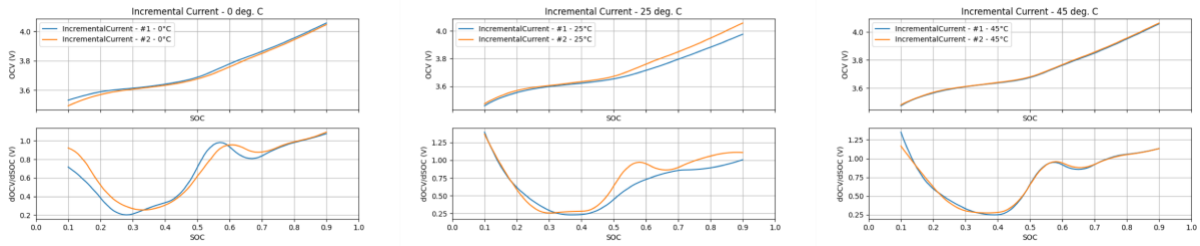

Figure 15: Incremental Current OCV and dOCV/dSOC vs OCV for 0 deg. C (left), 25 deg. C (middle), and 45 deg. C (right).

The plots show that there isn't a huge difference in OCV or dOCV/dSOC curves by temperature expect for the 25 deg. C curve for Sample 1. There is a similar difference between samples compared to differences by temperature.

Again, we have limited insight into the testing setup. It's also possible that the ambient temperature was held fairly constant but internal defects or due to differences between cells, there could be a significant temperature gradient across the cell which can affect the OCV vs SOC relationship. Either way, for simplicity and since we don't see a huge difference among temperatures (similar difference between samples), we assume that OCV vs SOC is not a function of temperature and use a constant OCV vs SOC relationship.

It's interesting to compare the OCV vs SOC relationship from the Low Current test to the Incremental Current test.

Figure 16 compares the OCV and dOCV/dSOC vs SOC relationships between the Low Current test and Incremental Current test. For each test, we take the average of all the OCV and dOCV/dSOC vs SOC plots (since we assumed the temperature dependence is weak and we neglect it). Figure 17 shows the same thing but with trimmed relationsihps on the x-axis to better see the comparison since the Low Current OCV vs SOC curve really falls off when the SOC gets low.

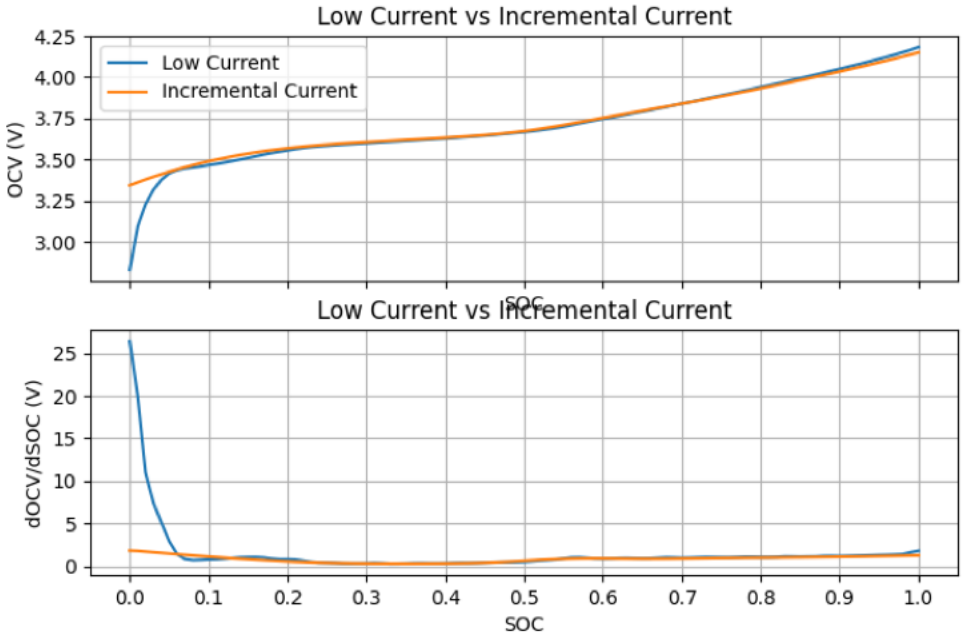

Figure 16: Low Current vs Incremental Current test - OCV and dOCV/dSOC vs OCV.

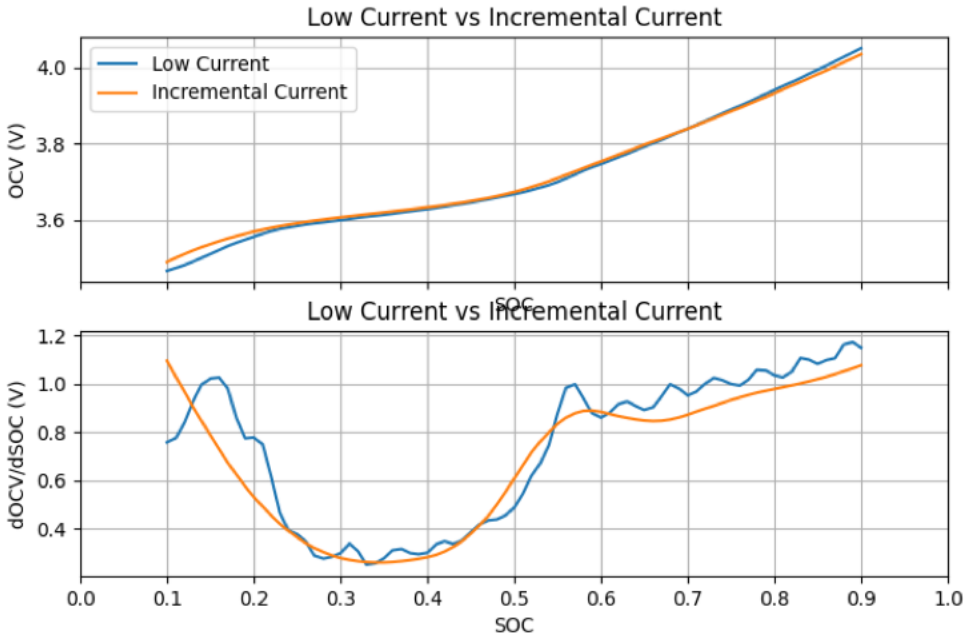

Figure 17: Low Current vs Incremental Current test - OCV and dOCV/dSOC vs OCV (trimmed x-axis).

There is fairly good agreement except near low SOC. For the Low Current curve, the slope increases sharply as you approach low SOC values. The Incremental Current curve has it's slope also increase but it isn't as sharp. There is a fairly smooth dOCV/dSOC vs SOC curve for the Incremental Current test compared to the Low Current test (which is probably partially due to the interpolation needed in the Incremental Current case). We also know that the OCV decreases sharply as you approach low SOC and that the battery dynamics are strong even for low currents. For these reasons, we will use the Incremental Current OCV vs SOC relationship for our analysis.

2.2 Equivalent Circuit Parameters

This section describes finding the parameters: $R_0, R_1, C_1, R_2, and\ C_2$. This work was actually fairly intensive but I had to draw the line somewhere so I didn't explore as much as I would have liked. I also considered changing the model but that wasn't the original focus of this project and I wanted to stay true to that. I copy the equivalent circuit model here again for reference in Figure 18.

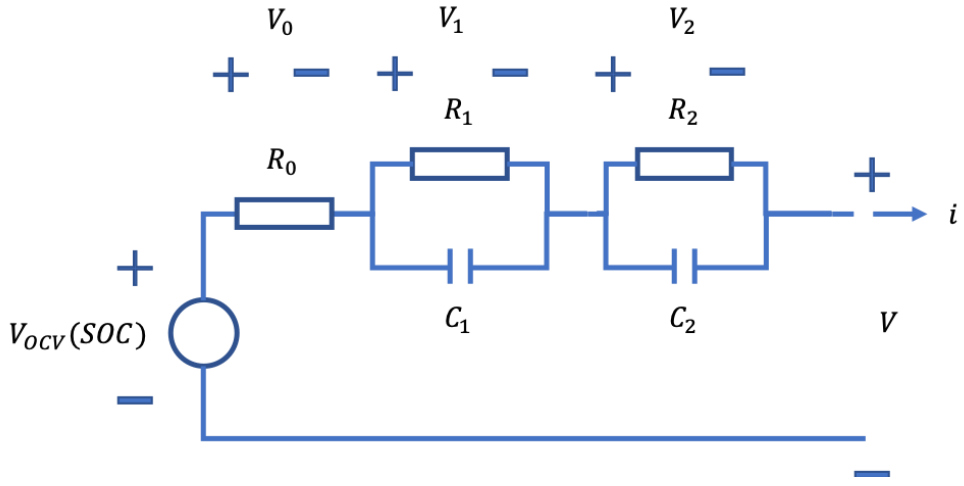

Figure 18: Equivalent circuit model.

The equivalent circuit model shown in Figure 18 is not novel (you find this extensively in the literature). We can put the following equation together:

$$V = V_{OCV} - V_o - V_1 - V_2$$

In this equation, $V_{OCV} = V_{OCV}(SOC)$, $V_0 = i * R_0$, and V_1 and V_2 are the RC circuit voltages. The RC circuit is a resistor and capacitor in series as shown in Figure 19

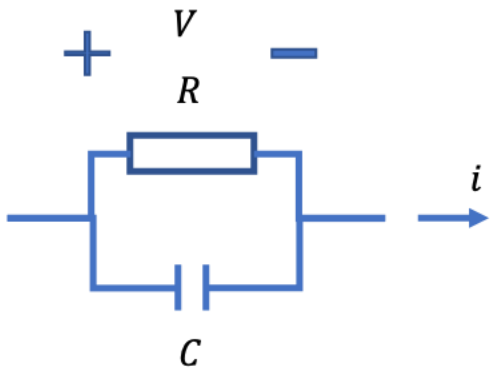

Figure 19: RC circuit.

We have for the RC circuit:

$$V_c = V_R = V$$

$$i = i_c + i_r$$

We also have:

$$V_R = i_R R$$

$$i_c = C\frac{dV}{dt}$$

We then have:

$$V = V_R = i_R R = (i - i_C)R$$

$$= \left(i - C\frac{dV}{dt}\right)R$$

$$\frac{dV}{dt} + \frac{1}{RC}V = \frac{1}{C}i$$

$$\frac{d}{dt}\left(Ve^{\frac{t}{RC}}\right) = \frac{1}{C}ie^{\frac{t}{RC}}$$

Solving this gives (assuming constant i):

$$\mathrm{Ve}^{\frac{\mathrm{t}}{\mathrm{RC}}} = \mathrm{iRe}^{\frac{\mathrm{t}}{\mathrm{RC}}} + \mathrm{A}$$

Where A is a constant

$$V = iR + Ae^{-\frac{t}{RC}}$$

At $\mathrm{t} = 0$, we have:

$$V(t = 0) = V_o = iR + A$$

$$\rightarrow A = V_o - iR$$

Then:

$$V = V_o e^{-\frac{t}{RC}} + iR\left(1 - e^{-\frac{t}{RC}}\right)$$

For $i = 0$,

$$V = V_o e^{-\frac{t}{RC}}$$

For $V_o = 0$,

$$V = iR\left(1 - e^{-\frac{t}{RC}}\right)$$

I first tried one approach using the Low Current and Incremental Current data to find R_0 and $\tau_i = R_i C_i$. This approach gave interesting results and then I ended up using DST data (Dynamic Stress Test) and used the least squares optimization method to find the parameters. Both of these approaches are described below:

2.2.1 First Approach – Recovery data

Both the Low Current and Incremental Currnet tests have recovery periods where I can use the $i = 0$ equation for the RC circuit. This is shown in Figure 20.

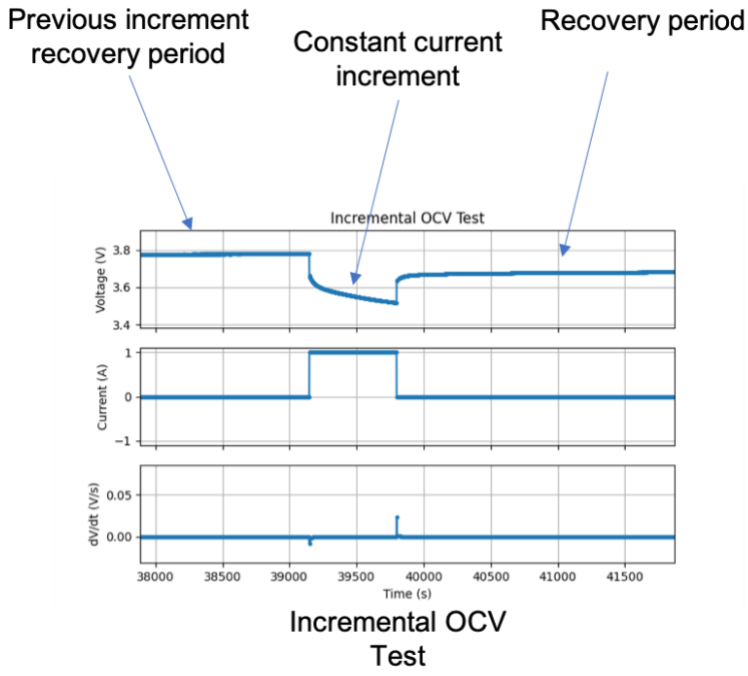

Figure 20: Recovery period in Incremental OCV test.

During the recovery period, the current is 0, my terminal voltage equation becomes:

$$V = V_{OCV} - V_o - V_1 - V_2$$

$$= V_{OCV}(SOC) - 0 - V_{1,o} e^{-\frac{t}{\tau_1}} - V_{2,o} e^{-\frac{t}{\tau_2}}$$

We find R_0 by using the voltage at the beginning of the recovery period and the voltage immediately before, take the difference, and then divide by the current:

$$R_0 = \frac{V_{recovery}(t = 0) - V_{charge\ or\ discharge}(t = end)}{i}$$

I use a curve fitting algorithm to find the parameters: $V_{1,o}, V_{2,o}, \tau_1, and\ \tau_2$. Two examples are shown below, Figure 21 shows a recovery period after a charge increment, and Figure 22 shows a recovery period after a discharge increment.

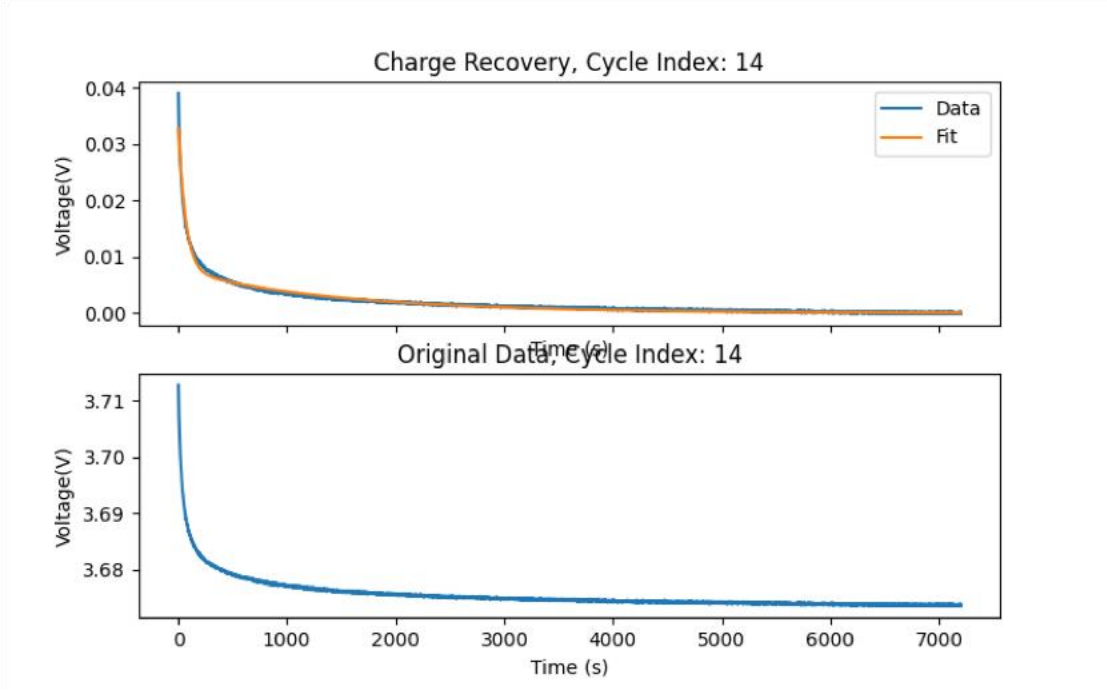

Figure 21: Incremental Current data and fit for Cycle Index 14 (Charge Increment).

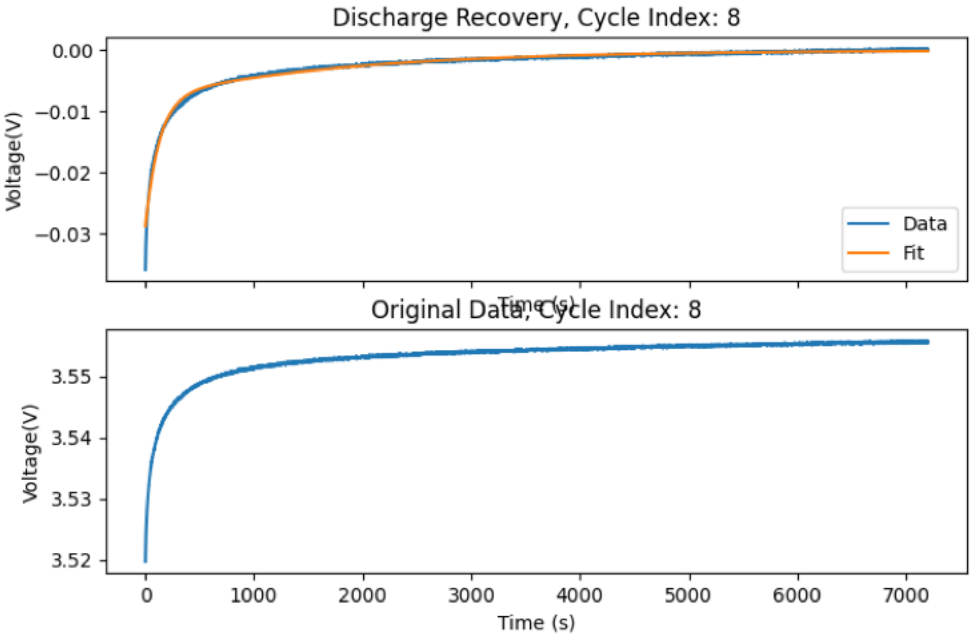

Figure 22: Incremental Current data and fit for Cycle Index 14 (Discharge Increment).

There are good fits with the data. When I look at the parameters, there was some significant variation that will be discussed next. I'll look at my R_0 fits which were not a result from the curve fitting shown above but

was a result of using the data looking at the voltage at the beginning of the recovery period, voltage prior to the recovery period, and the current. Figure 23 shows R_0 vs OCV for all the data (all the recovery period data).

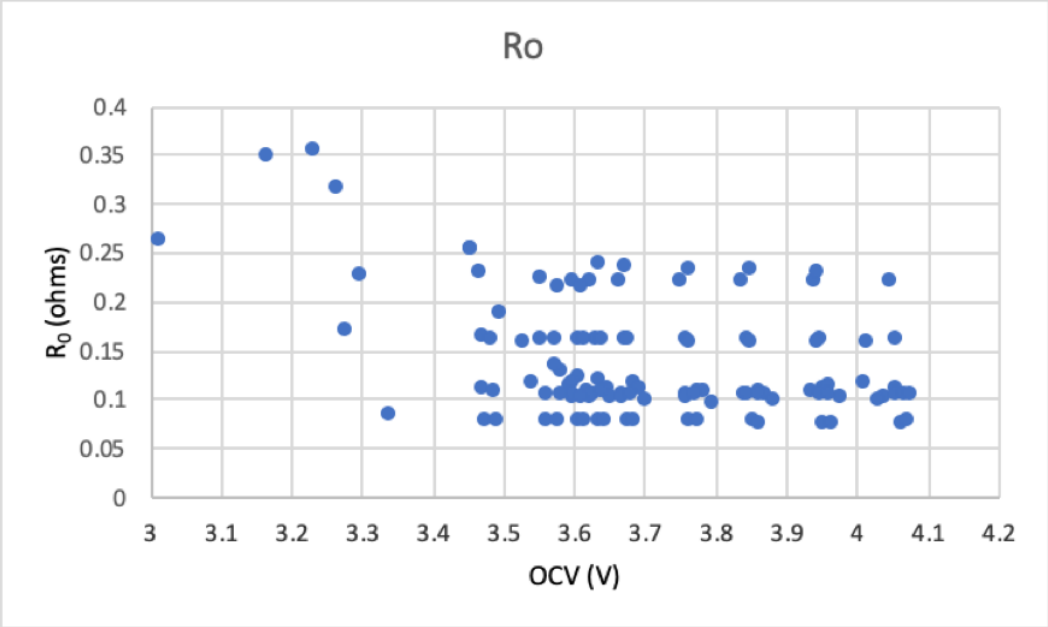

Figure 23: R_0 vs OCV for all data from the recovery periods of Low Current and Incremental Current tests.

At low OCV, the data gets a little messy which is not that surprising. When looking at the low OCV (or low SOC) behavior in the Low Current test, we see sharp changes in voltage which can complicate the data. We trim the low OCV values and this is shown in Figure 24.

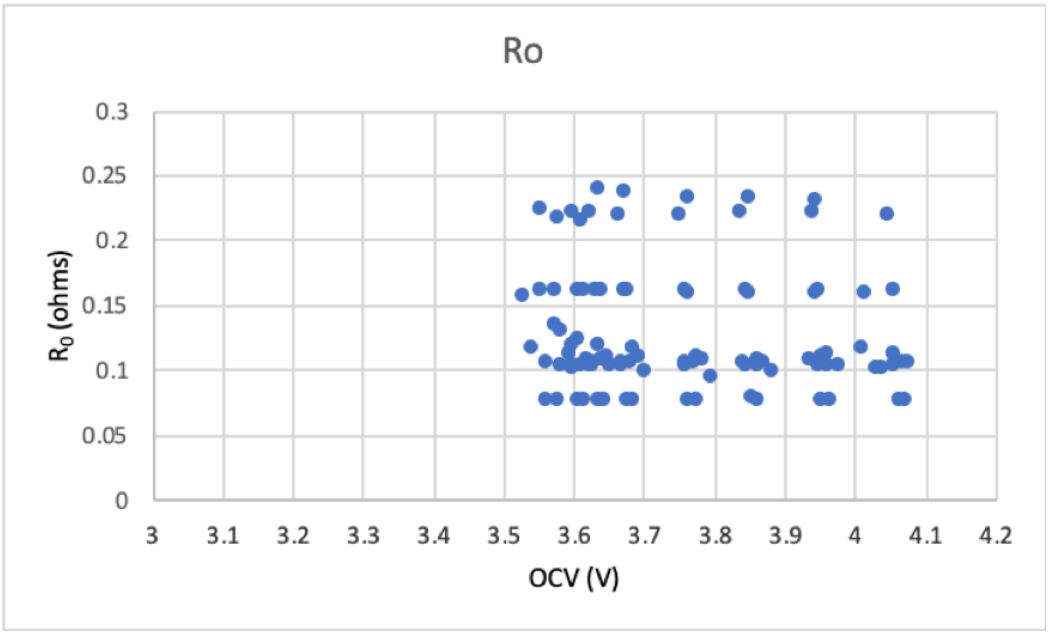

Figure 24: R_0 vs OCV with the low OCV values trimmed.

It's interesting to see different bands of values. There is quite the variation too with more than a factor of 2 separating the lowest band from the highest band. I wanted to further explore this and started breaking the data down by sample and temperature. I found that each band corresponds to a different sample at a

different temperature. One example for data from the Incremental Current test at 25 deg. C for Sample 1 and Sample 2 is shown below in Figure 25.

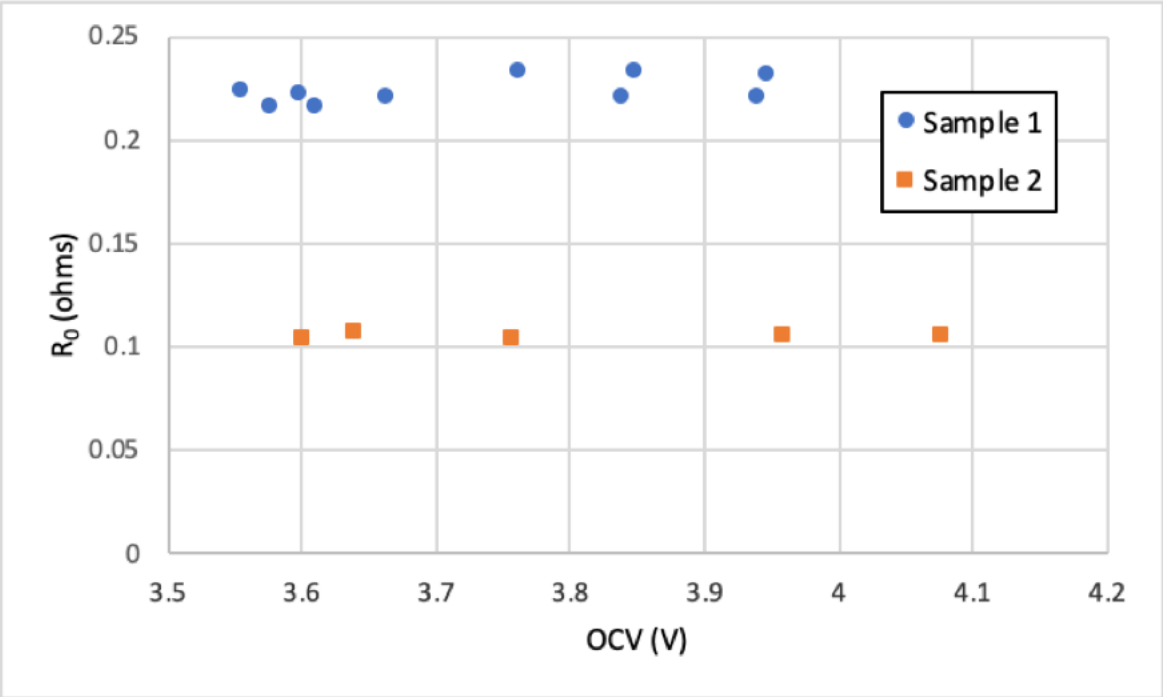

Figure 25: R_0 vs OCV for Sample 1 and Sample 2 at 25 deg. C from the Incremental Current data.

It's interesting to note how consistent the R_0 data is between samples. Without understanding the test setup and the data acquisition, it's difficult to explain the difference. I checked to see if the time difference between samples could explain it but it does not. It is possible that the R_0 value can vary cell to cell and it might just be a coincidence that Sample 1 is twice that of Sample 1 in this case.

Possibly the way the cell was instrumented was different and/or terminal resistance could explain the difference. Since, it's hard to explain the difference, this is one reason why we pivot to using the DST data.

Having a big variation in R_0 where it seems to vary by sample makes it difficult to trust the fits (it actually makes it hard to trust any of our fits) but we will still show the time constant fits. This will be interesting to compare to our second method which uses the DST data to fit the parameters. Figure 26, Figure 27, and Figure 28 show τ_1 vs OCV, τ_2 vs OCV, and τ_2/τ_1 vs OCV, respectively.

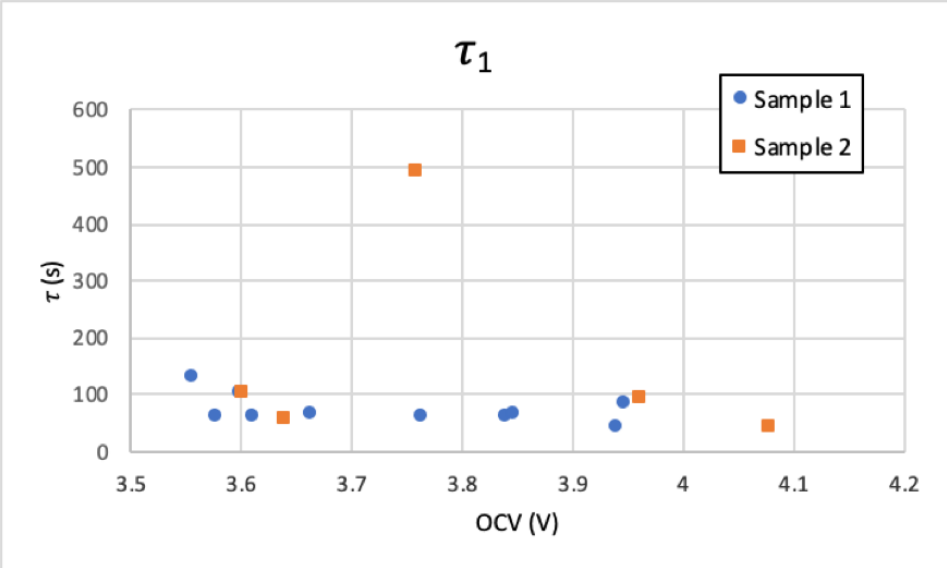

Figure 26: τ_1 vs OCV

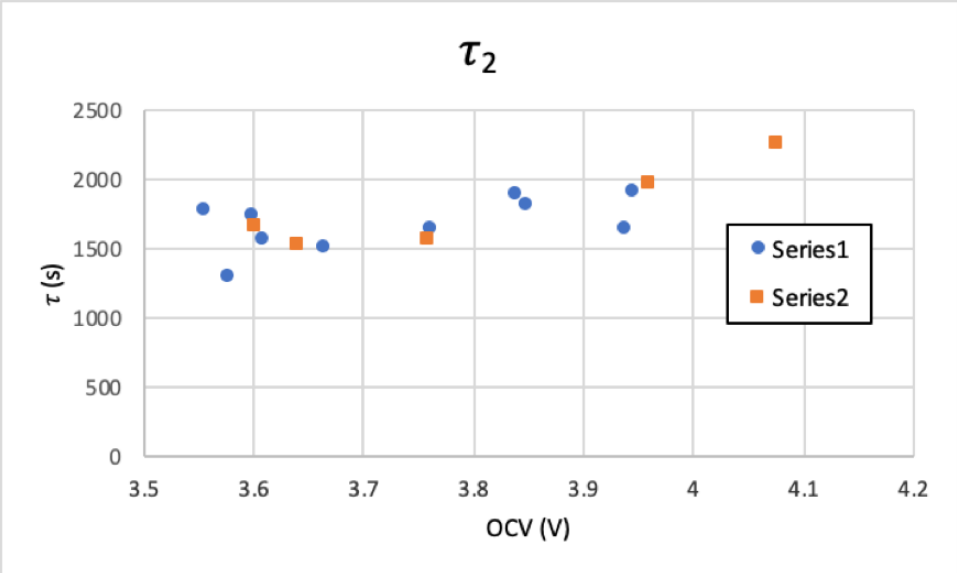

Figure 27: τ_2 vs OCV

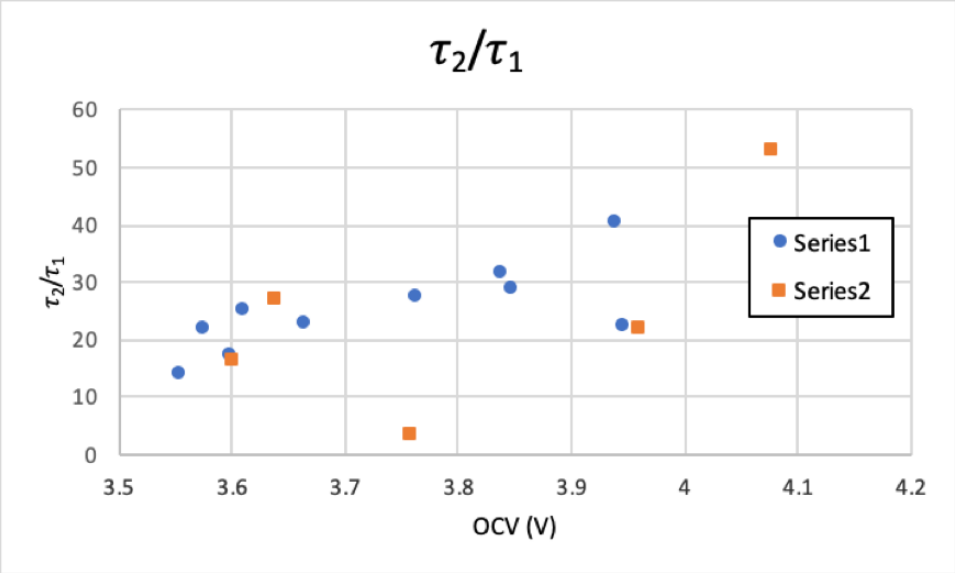

Figure 28: τ_2/τ_1 vs OCV.

The fits show a clear slow and fast dynamic. Since the recovery data is not really subjecting the cell to varying inputs, using that data to find the slow and fast dynamic response is not that appropriate. However, we do see an order of magnitude difference that helps validate our approach somewhat.

R_0 vs temperature is shown in Figure 29 and Figure 30 shows τ_1 and τ_2 vs temperature. The max. values are around room temperature for all values (curve is just the smooth line option for plotting, it's not any calculated fit). Since we had questions about the values and the validity of using the Low Current and Incremental Current recovery test data to fit the parameters, we want to use the DST data and compare those findings to these. We already know that our recovery time period might influence the results and the DST data gives us many changes in current that should really highlight the dynamics of the cell. We can compare our findings here to the findings using the second approach.

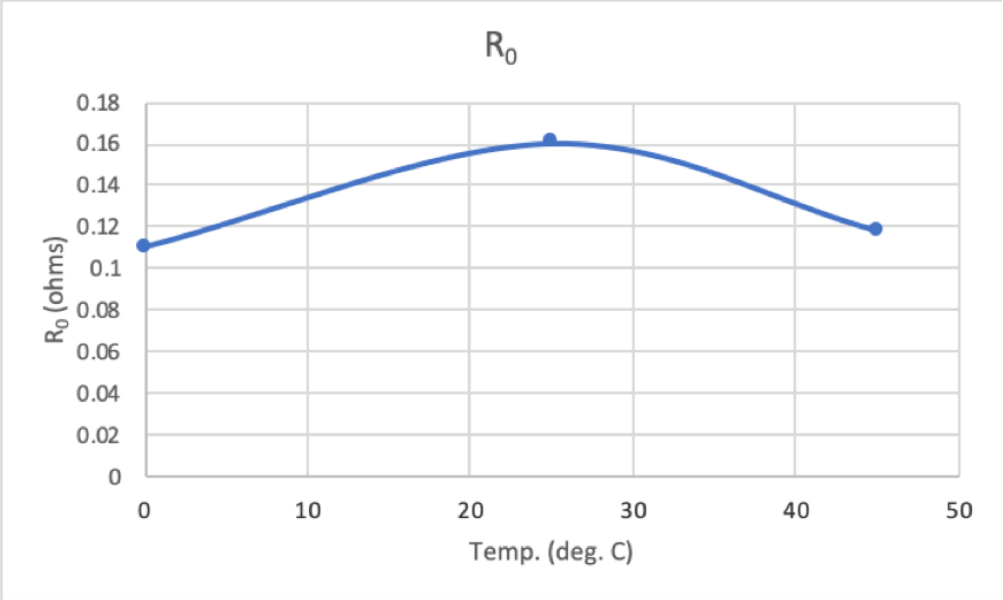

Figure 29: Recovery fit R_0 vs temperature.

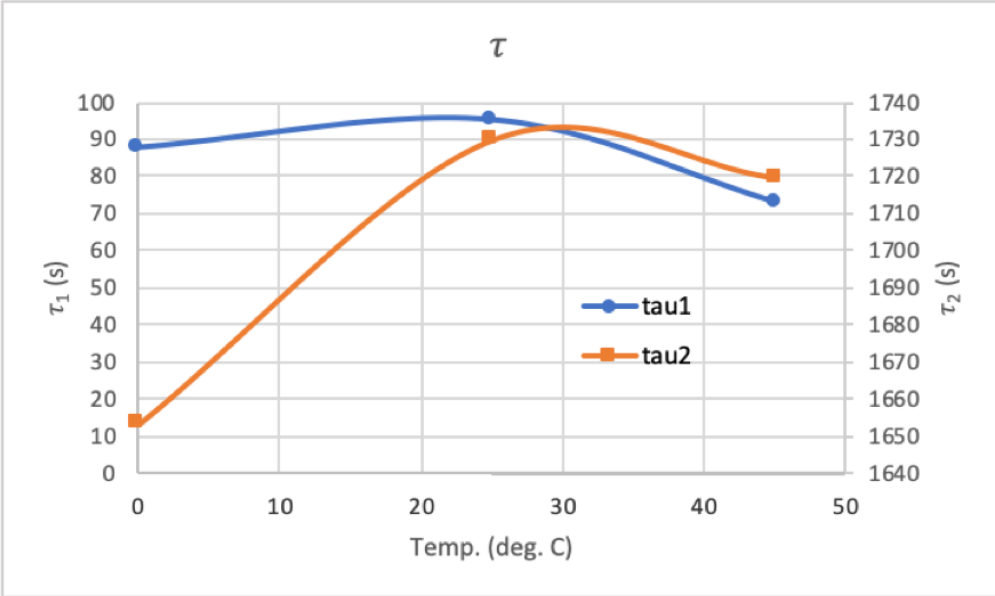

Figure 30: Recovery fit time constants vs temperature..

2.2.2 Second Approach – DST data

In the second approach, we use the DST data to find our parameters. Using this data has the advantage of capturing lots of dynamics due to the changes in current and the resulting changes in voltage. A plot of one DST data set is shown in Figure 31.

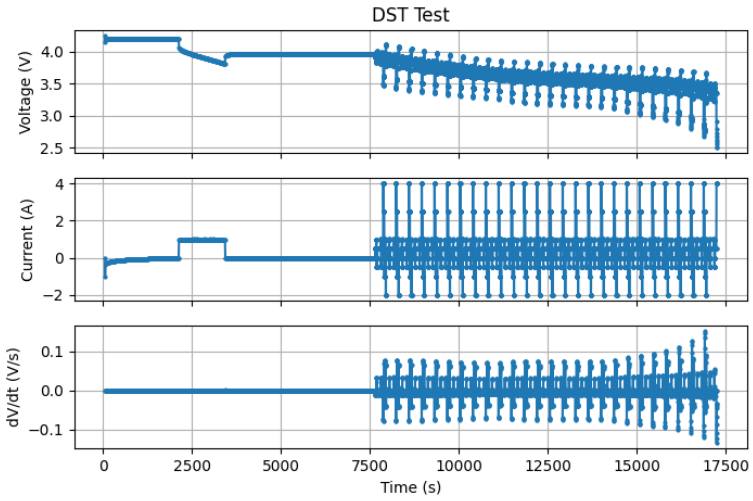

Figure 31: DST data at 0 deg. C starting at 80% SOC.

In this approach, we use the least squares optimization method to find our parameters. Again, we assume that we have the SOC data so that we know what the OCV is during the whole test using our OCV vs SOC relationship we built previously. We then have:

$$V(t) = V_{OCV}(SOC(t)) - V_0 - V_1 - V_2$$

Here $V_0 = i(t) * R_0$ and for V_1 and V_2, we have:

$$\frac{dV_i(t)}{dt} = -\frac{V_i(t)}{R_i C_i} + \frac{i(t)}{C_i}$$

Based on our current parameter guess, we use an ODE solver to solve for V_i based on the current data. We then have the residual:

$$V_{measure} - V_{fit}$$

I explored a few different curve fitting methods in python but found the best success with the least squares method (it was both faster and seemed to give fairly good results that made sense). I also explored a bit with the RC circuit voltage and how to integrate to solve for the RC circuit voltage. I'll discuss some on that too.

The way the fit program works is by starting with an initial guess of the parameters, calculating the OCV voltage from the SOC data, calculating V_0, and then calculating V_1 and V_2. I used an ODE solver in Python to find V_1 and V_2 so that I didn't solve the ODE directly but had the ODE tool in Python find the solution based on current data, initial conditions (V_1 and V_2 are both 0 initially), time data, and my current guess of $R_1, C_1, R_2, and\ C_2$. I also experimented with finding V_1 and V_2 by saying that the current is constant over the current time step and saying:

$$V_{i,k} = V_{i,k-1} e^{-\frac{\Delta t_k}{R_i C_i}} + i_k R_i \left(1 - e^{-\frac{\Delta t_k}{R_i C_i}}\right)$$

This is how the RC circuit voltage would be calculated in the controller in an actual application. One could also just use Euler forward integration too:

$$V_{i,k} = V_{i,k-1} - \frac{V_{i,k-1}}{R_i C_i}\Delta t + \frac{i_k}{C_i}\Delta t$$

I compare both methods when actually implementing the Kalman filter and will discuss this more in the Kalman filter section. When fitting parameters, I used the Python ODE solver instead of the exponential solution. I found that when I use the exponential solution, the fit doesn't look great.

I'll show one example of the fit for the DST data at 25 deg. C with the initial SOC at 80%. The parameter iteration is shown in

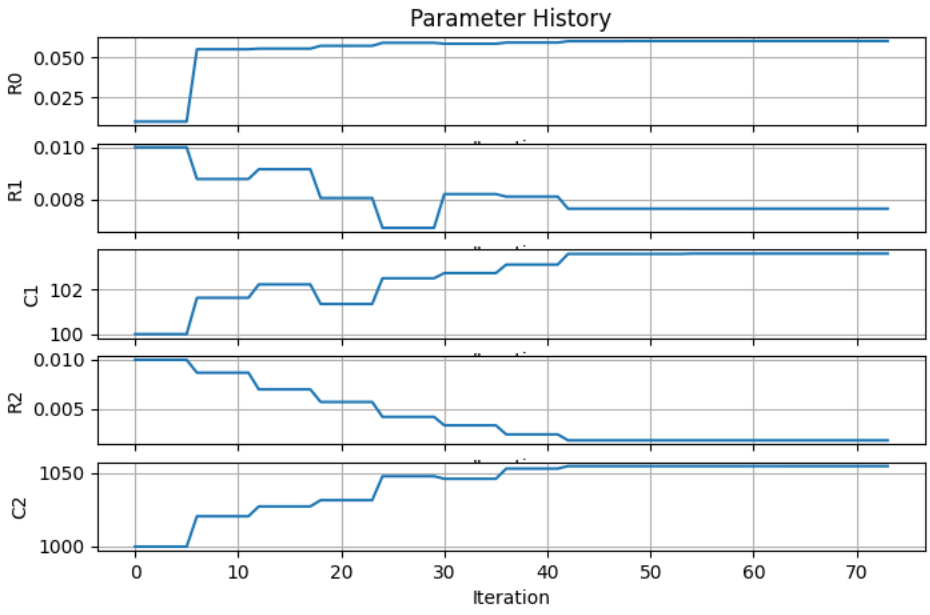

Figure 32: DST Parameter Fit (25 deg. C, 80% SOC).

The parameters adjust and settle after about 40 iterations. The code is sensitive to initial conditions and I didn't do too much exploring with this. I used initial conditions based off the values I found in the first

approach. The fit compared to the measured voltage is shown in Figure 33. I also compared the fit (using the Python ODE solver) to the exponential solution (online model) and this is shown in Figure 34.

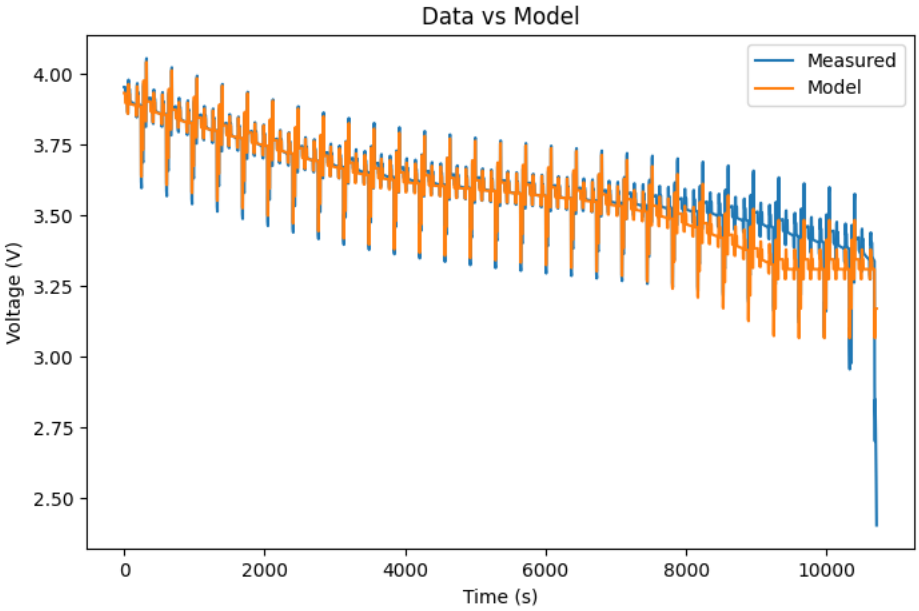

Figure 33: Model fit vs measured voltage.

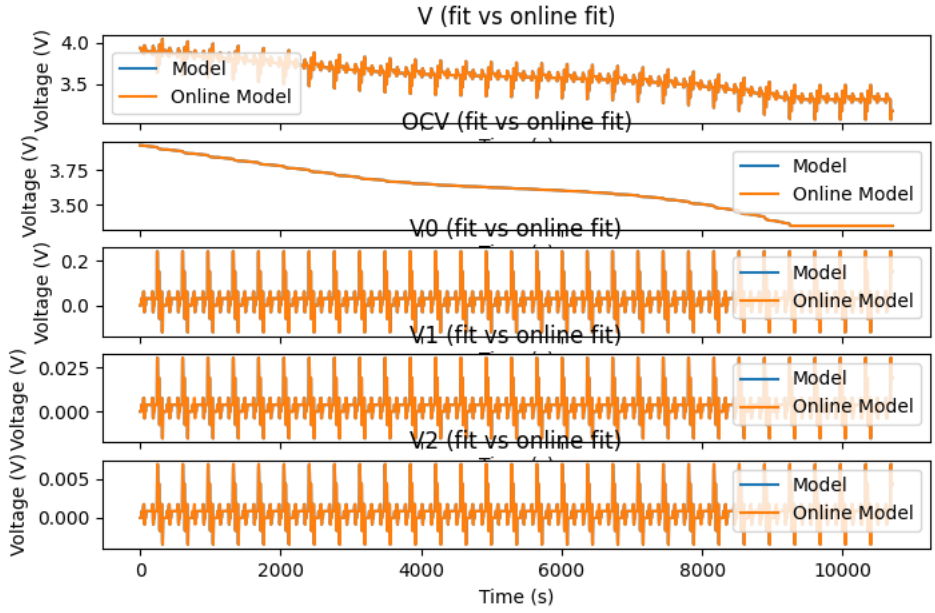

Figure 34: Comparison between fit using Python ODE tool (model) vs using exponential solution (online model).

I got a fairly good fit and both the model and online model agree fairly well too. The RMS error between the model and fit for this data set is 0.31 and the R^2 value is 0.88.

I evaluated using the exponential fit in the least squares algorithm for finding parameters. This didn't give me great results as shown in Figure 35. R_2 drops to almost 0 which shows that the second RC circuit is potentially not necessary. My time constants are approximately 0.7 s (700 ms) and 0.0005 (0.5 ms) which

don’t agree with the other fitting methods so I didn’t pursue this much more. Really I could evaluated a single RC circuit and dove into the parameter fitting details more (initial conditions, solver methods, etc.)

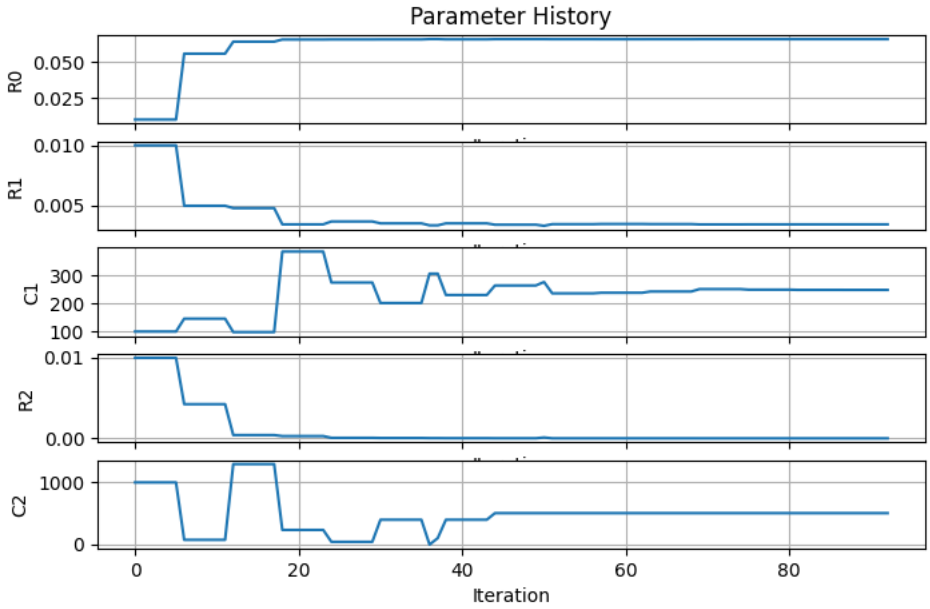

Figure 35: DST Parameter Fit (25 deg. C, 80% SOC) (using exponential RC solution).

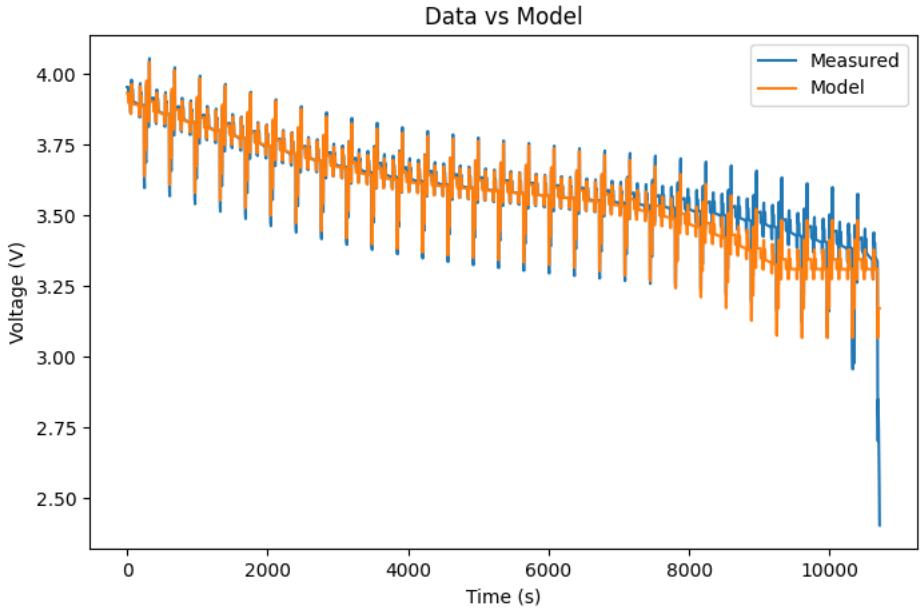

Figure 36: Model fit (online model) vs measured voltage.

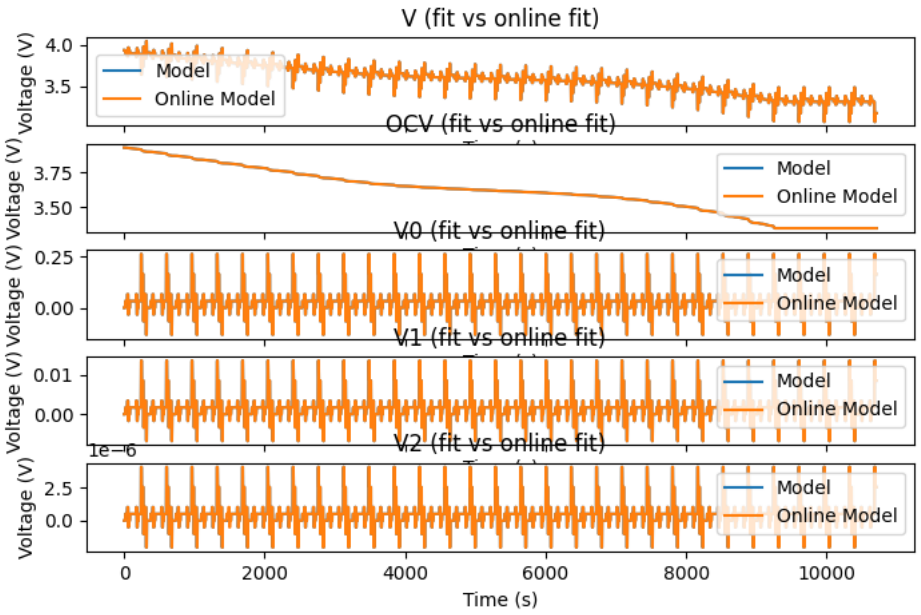

Figure 37: Comparison between fit using Python ODE tool (model) vs using exponential solution (online model).

The two above plots show that I still got a good fit. For all the fits, we do see that at low OCV (or low SOC), that the fit gets not good. The dynamics get very apparent at low OCV and possibly the Low Current OCV vs SOC relationship could better capture this. I don't really look into this much more either. It's also possible that the parameters could be functions of SOC (or OCV) too. Looking at the time scales, the tests were longer than 2.5 hours where internal heating could become significant. We would have to look at a different model (or look at Joule heating), assume that heat goes into the cell, use a cylindrical heat transfer model and look at the temperature gradient across the cell. A lot of assumptions could be made but it could give us some insight into the temperature gradient. Potentially at low temperatures, the convective cooling could keep the cell at fairly constant temperature while hotter temperatures lead to more of a gradient. This could potentially be useful in battery management systems where you are balancing battery performance and ambient conditions (do you use the batteries to run an A/C system to keep the cells cool – would that make any sense?)

The parameters I found are shown in Table 2. It's interesting to note how the fits are good (low RMS error and high R2) for low temperatures but the results get worse for higher temperatures. It's also interesting how the low temp. shows two distinct time constants (differing by an order of magnitude) while the warmer temperatures don't show this. They show comparable time constants. Potentially a double RC circuit would work for low temperatures and a single RC circuit would be more appropriate for warmer temperatures. It's clear that even the R_2 parameter gave me trouble with the ODE solver.

Table 2: Parameter fits from DST data.

Ro	**R1**	**C1**	**R2**	**C2**	**R2 fit**	**RMS**	**tau1**	**tau2**	**T (deg. C)**	**SOC (%)**
0.119	0.014	86.2	0.05	870.6	0.95	0.09	1.17	43.82	0	50
0.101	0.014	91.3	0.027	940	0.97	0.08	1.28	25.81	0	80
0.062	0.012	103.9	0.005	981.5	0.84	0.19	1.25	4.84	25	50
0.06	0.008	103.6	0.002	1054.8	0.89	0.31	0.79	1.85	25	80

0.077	0.01	100.3	0.006	1000.8	0.84	0.28	0.98	5.75	45	50
0.062	0.007	105.8	0.0003	1009.9	0.85	0.6	0.71	0.33	45	80

I ended up taking the average of the above values and using those in the Kalman filter. The final values are shown in Table 3.

Table 3: Parameter values to be used in Kalman filter.

Ro	R1	C1	R2	C2	tau1	tau2	T (deg. C)
0.11	0.014	88.7	0.039	905.3	1.23	35.21	0
0.061	0.01	103.7	0.003	1018.2	1.02	3.4	25
0.07	0.008	103.1	0.003	1005.3	0.85	3.05	45

The last thing to look at is the trends to compare to the first approach. Figure 38 shows the resistance (R_0, R_1, R_2) fits vs temperature. Figure 39 shows the time constant fits vs temperature. Figure 40 shows the capacitance (C_1, C_2) fits vs temperature.

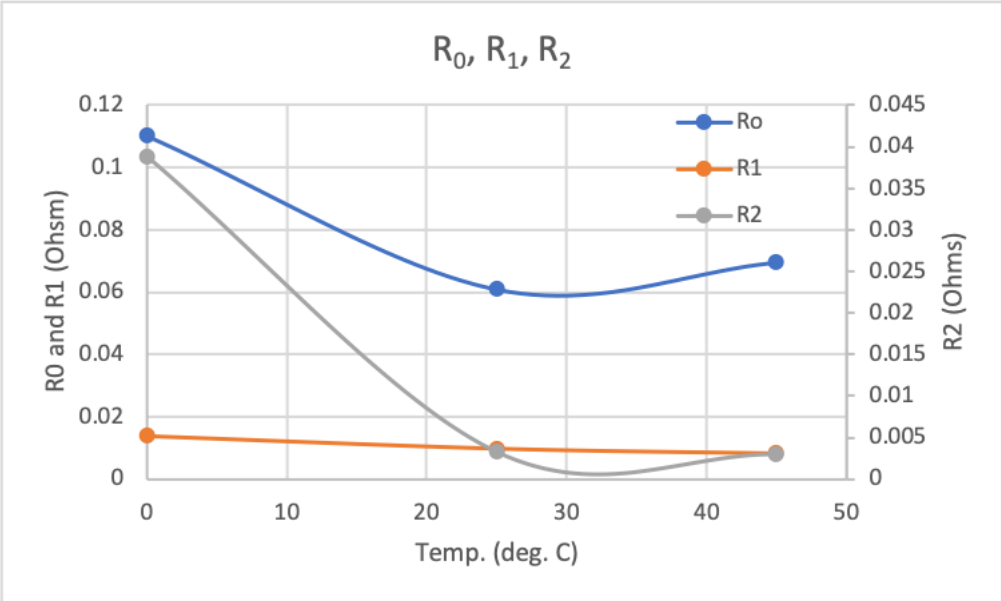

Figure 38: Resistance fits vs temperature.

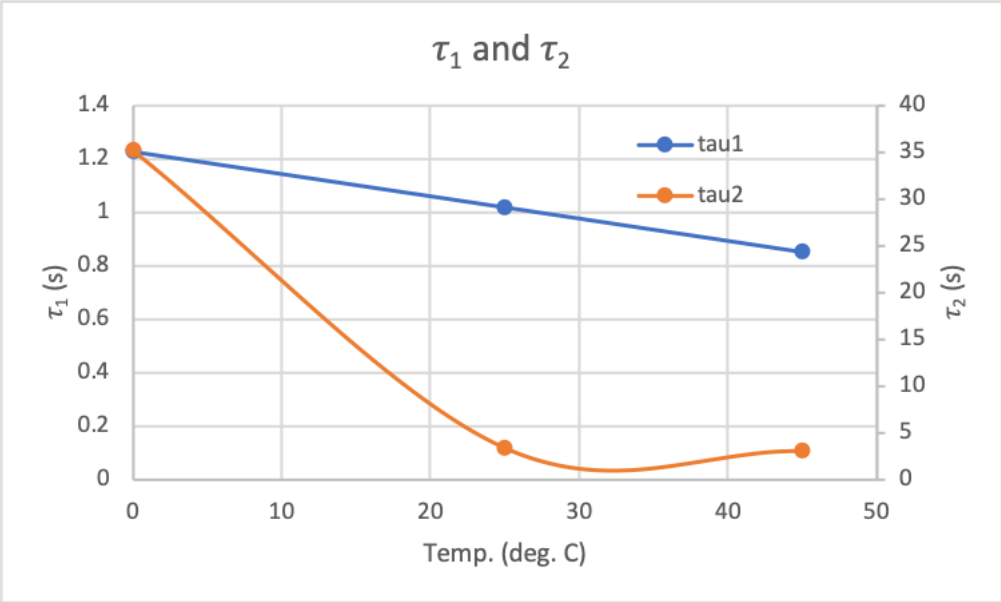

Figure 39: Time constant fits vs temperature.

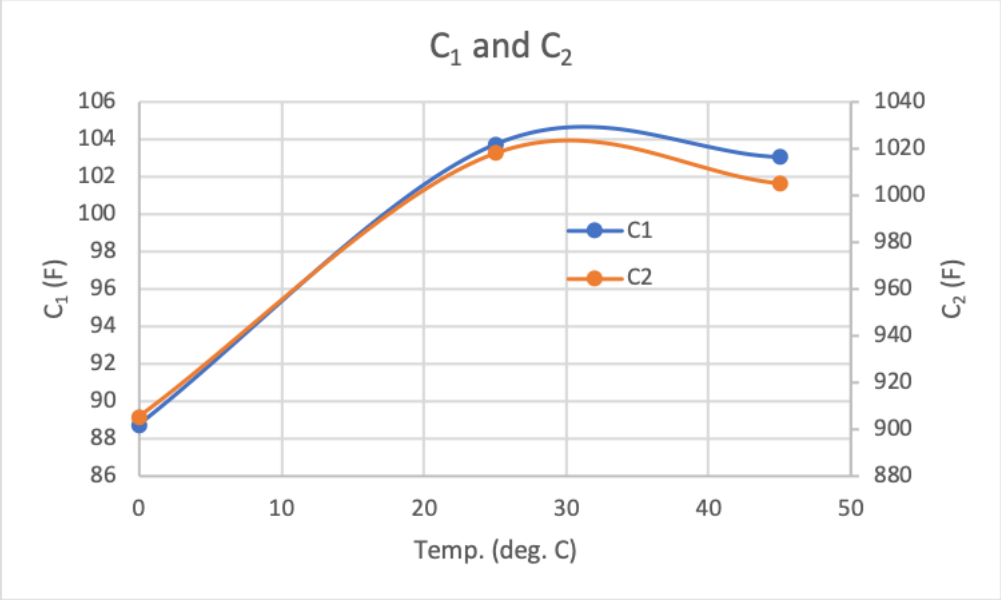

Figure 40: Capacitance fits vs temperature.

The curve shapes for the fits from the DST are almost the opposite of the first approach fit. Here, we have more concave shapes while the first approach showed convex.

This clearly shows that our battery model is, at least, a little suspect. I don't really look into this anymore. This has the anticipated effect of really influencing our Kalman filter SOC calculation results but it's decent enough to at least use to build the filter.

3 Kalman Filter

This section first has a quick discussion on other methods for calculating SOC, then shows the Kalman filter derivation, how I applied it to the lithium ion battery (there are multiple ways of doing this), and then showing some results.

3.1 Other SOC Calculation Methods

To help understand the results, we will actually compare three methods of tracking the SOC:

1. Coulomb counting (integrate current over time → SOC)
2. Voltage lookup (voltage → SOC)
3. Kalman filter (combining the above two)

The first two are fairly trivial in how they would be implemented. Clearly, the first is very susceptible to your initial value of SOC. The Kalman filter is a little more complicated but actually simple and elegant once we see the derivation and implementation. Here is a quick description of the first two methods:

3.1.1 Coulomb counting

This method is based on:

$$SOC(t) = SOC(t_o) + \int_{t_o}^{t} \frac{i(t)}{Q * 3600} dt$$

Here, Q is in units Ah. This method's accuracy is dependent on the initial value of the SOC but other logic could be introduced to help correct the initial guess. For most of our comparison we will assume that we know the initial SOC

3.1.2 OCV-SOC

This method is based on:

$$SOC(t) = SOC\big(OCV(t)\big)$$

Here, $OCV(t)$ is the voltage reading of the cell. As we have seen this method will result in measuring the cell voltage which is not necessarily equal to the OCV voltage

3.2 Derivation

One of the main motivations of using the Kalman filter is to improve our measurement of SOC. We can't directly measure the SOC but we assume that the state is observable based on current and voltage measurements which is supported by our SOC definition and our battery equivalent circuit model. This makes the Kalman filter well suited for our application

We will first look at the derivation of the Kalman filter and then apply it to our battery model. We will be technically using the Extended Kalman Filter (EKF) since the Kalman filter is for linear systems and our battery model is clearly non-linear but we linearize it around our point and then use the Kalman filter approach

We first start with state and observation functions:

$$\dot{x}_k = f_x(x_k) + b_u(u_k) + w_k$$
$$z_k = h(x_k) + b_y(u_k) + v_k$$

We have that $f_x(x)$ is a nonlinear function of x, b_u is a nonlinear function of the input u, w_k is process noise with zero mean, z_k is our measurement, $h(x)$ is our nonlinear observation function, b_y is a nonlinear function of the input u, and v_k is our measurement noise with zero mean.

We first need to get this in the form suitable for the Kalman filter so we use some sort of integration, forward Euler integration would be the most straightforward to get:

$$x_k = x_{k-1} + f_x(x_{k-1})\Delta t + b_u(u_{k-1})\Delta t + w_k$$
$$\rightarrow x_k = f_x(x_{k-1}) + b_u(u_{k-1}) + w_k$$

We assume that the initial state is x_o with known mean $\mu_o = E[x_o]$ and covariance $P_o = E[(x_o - \mu_o)(x_o - \mu_o)^T]$. If x is a nx1 column vector, then P_o is nxn.

For the derivation, we will assume we have for our state and measurement equations:

$$x_k = f_x(x_{k-1}) + b_u(u_{k-1}) + w_k$$
$$z_k = h(x_k) + b_y(u_k) + v_k$$

We then say that x_{k-1}^a is our previous estimate of the state at step k. We then say our forecast estimate of x_k is:

$$x_k^f = f_x(x_{k-1}^a) + b_u(u_{k-1})$$

We then have for our error estimate:

$$e_k^f = x_k - x_k^f = f_x(x_{k-1}) + b_u(u_{k-1}) + w_k - f_x(x_{k-1}^a) - b_u(u_{k-1})\Delta t$$
$$= f_x(x_{k-1}) - f_x(x_{k-1}^a) + w_k$$
$$\cong f_x(x_{k-1}^a) + \left(\frac{df_x(x_{k-1}^a)}{dx}\right)(x_{k-1} - x_{k-1}^a) - f_x(x_{k-1}^a) + w_k$$

In the last approximation, we expand $f_x(x_{k-1})$ in a Taylor Series about x_{k-1}^a. So we then have:

$$e_k^f = \left(\frac{df_x(x_{k-1}^a)}{dx}\right)(x_{k-1} - x_{k-1}^a) + w_k = Fe_{k-1} + w_k$$

Then we obtain:

$$P_k^f = E\left[e_k^f \left(e_k^f\right)^T\right] = E[(Fe_{k-1} + w_k)(Fe_{k-1} + w_k)^T]$$
$$= Fe_{k-1}e_{k-1}^T F^T + w_k w_k^T = FP_{k-1}F^T + Q_k$$

$Fe_{k-1}w_k^T$ and $w_k e_{k-1}^T F^T$ are both zero since the expected value of a vector or matrix multiplied by the noise is zero. Q_k is our process noise covariance matrix.

We then assume that the estimate of the state at the next time step is a linear function of our measurement, z_k:

$$x_k^a = a + K_k z_k$$

We want to find a. We first have:

$$e_k = x_k - x_k^a$$

Our expected error is:

$$E[e_k] = 0 = E[x_k - x_k^a] = E[x_k - a - K_k z_k]$$

We then use:

$$x_k = e_k^f + x_k^f$$
$$z_k = h(x_k) + b_y(u_k) + v_k$$
$$\cong h\left(x_k^f\right) + \frac{dh\left(x_k^f\right)}{dx}\left(x_k - x_k^f\right) + b_y(u_k) + v_k = h\left(x_k^f\right) + \frac{dh\left(x_k^f\right)}{dx}e_k^f + b_y(u_k) + v_k$$

We then plug this into our expected error equation and obtain:

$$E[e_k] = E[x_k - a - K_k z_k] \cong E\left[e_k^f + x_k^f - a - K_k\left(h\left(x_k^f\right) + \frac{dh\left(x_k^f\right)}{dx}e_k^f + b_y(u_k) + v_k\right)\right]$$

This then becomes:

$$E[e_k] \cong x_k^f - a - K_k h\left(x_k^f\right) - K_k b_y(u_k)$$
$$\rightarrow a = x_k^f - K_k\left[h\left(x_k^f\right) + b_y(u_k)\right]$$

$$x_k^a = a + K_k z_k = x_k^f + K_k\left[z_k - h\left(x_k^f\right) - b_y(u_k)\right]$$

Now we want to find what K_k is. First we plug our value of a back into our error equation:

$$e_k = x_k - x_k^a = x_k - a - K_k z_k = x_k - x_k^f - K_k\left[z_k - h\left(x_k^f\right) - b_y(u_k)\right]$$

$$e_k = f_x(x_{k-1}) + b_u(u_{k-1}) + w_k - x_k^f - K_k\left[h(x_k) + b_y(u_k) + v_k - h\left(x_k^f\right) - b_y(u_k)\right]$$

$$\cong f_x(x_{k-1}^a) + \left(\frac{df_x(x_{k-1}^a)}{dx}\right)(x_{k-1} - x_{k-1}^a) + b_u(u_{k-1}) + w_k - f_x(x_{k-1}^a) - b_u(u_{k-1})$$

$$- K_k\left[h\left(x_k^f\right) + \frac{dh(x_k^f)}{dx}\left(x_k - x_k^f\right) + b_y(u_k) + v_k - h\left(x_k^f\right) - b_y(u_k)\right]$$

$$= \left(\frac{df_x(x_{k-1}^a)}{dx}\right)e_{k-1} + w_k - K_k\left[\frac{dh\left(x_k^f\right)}{dx}e_k^f + v_k\right]$$

$$= Fe_{k-1} + w_k - K_k\left(He_k^f + v_k\right) = Fe_{k-1} + w_k - K_k[H(Fe_{k-1} + w_k) + v_k]$$

In the last expression, we use $F = \frac{df_x(x_{k-1}^a)}{dx}$, $H = \frac{dh\left(x_k^f\right)}{dx}$, and $e_k^f = Fe_{k-1} + w_k$. We then obtain:

$$e_k = (I - K_k H)Fe_{k-1} + (I - K_k H)w_k - K_k v_k$$

We then have:

$$P_k = E[e_k(e_k)^T] = (I - K_k H)Fe_{k-1}e_{k-1}^T F^T(I - K_k H)^T + (I - K_k H)w_k w_k^T(I - K_k H)^T + K_k v_k v_k^T K_k^T$$

$$= (I - K_k H)P_k^f(I - K_k H)^T + K_k R_k K_k^T$$

$$= P_k^f - P_k^f H^T K_k^T - K_k H P_k^f + K_k H P_k^f H^T K_k^T + K_k R_k K_k^T$$

R is our measurement noise covariance matrix. Again, the cross terms cancel since we assume the process noise and measurement noise are independent with a mean of 0. We want to minimize the trace of of P_k (e.g., minimize our mean square error) by taking the derivative with respect to K_k, setting that to zero, and solving for K_k.

We will have:

$$T(P_k) = T(P_k^f) - T\left(P_k^f H^T K_k^T\right) - T\left(K_k H P_k^f\right) + T\left(K_k H P_k^f H^T K_k^T\right) + T(K_k R_k K_k^T)$$

And we want to find $\frac{dT(P_k)}{dK}$. Before we do that, let's review the trace of a matrix and how we can take the derivative of the trace since the trace is a scalar. We start with comparing the trace of KA to $A^T K^T$. We have:

$$T(KA) = T\left(\sum K_{im}A_{mj}\right) = \sum K_{1j}A_{j1} + \sum K_{2j}A_{j2} + \cdots = \sum K_{ij}A_{ji}$$

$$T(A^T K^T) = T\left(\sum A_{im}^T K_{mj}^T\right) = \sum A_{1j}^T K_{j1}^T + \sum A_{2j}^T K_{j2}^T + \cdots = \sum A_{j1}K_{1j} + \sum A_{j2}K_{2j} + \cdots$$

$$= \sum A_{ji}K_{ij} = \sum A_{ij}K_{ji}$$

So we have that $T(KA) = T(A^T K^T)$. Now we can look at the derivative of the trace with respect to K

$$\frac{dT(KA)}{dK} = \begin{bmatrix} A_{11} & A_{21} & A_{31} & \cdots \\ A_{12} & A_{22} & A_{32} & \cdots \\ A_{13} & A_{23} & A_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = A^T$$

Since $T(KA) = T(A^T K^T)$, we have $\frac{dT(KA)}{dK} = \frac{dT(A^T K^T)}{dK} = A^T$. We also have something similar to $T(AK^T)$ so we want explore this to:

$$T(AK^T) = \sum A_{1j}K_{j1}^T + \sum A_{2j}K_{j2}^T + \cdots = \sum A_{ij}K_{ji}^T = \sum A_{ij}K_{ij}$$

$$\frac{dT(AK^T)}{dK} = A$$

We know can find $\frac{dT(P_k)}{dK}$. We have:

$$T(P_k) = T\left(P_k^f\right) - T\left(P_k^f H^T K_k^T\right) - T\left(K_k H P_k^f\right) + T\left(K_k H P_k^f H^T K_k^T\right) + T(K_k R_k K_k^T)$$

$$= T\left(P_k^f\right) - 2T(K_k H P_k^f) + T\left(K_k H P_k^f H^T K_k^T\right) + T(K_k R_k K_k^T)$$

Since $T\left(P_k^f H^T K_k^T\right) = T\left(K_k H P_k^f\right)$. Taking the derivative with respect to K_k gives:

$$\frac{dT(P_k)}{dK} = \frac{dT\left(P_k^f\right)}{dK_k} - 2\frac{dT\left(K_k H P_k^f\right)}{dK_k} + \frac{dT\left(K_k H P_k^f H^T K_k^T\right)}{dK_k} + \frac{dT(K_k R_k K_k^T)}{dK_k}$$

$$= -2\left(H P_k^f\right)^T + \left(H P_k^f H^T K_k^T\right)^T + \left(K_k H P_k^f H^T\right) + (R_k K_k^T)^T + (K_k R_k)$$

$$= -2\left(H P_k^f\right)^T + 2\left(K_k H P_k^f H^T\right) + 2(K_k R_k)$$

Setting this to zero and solving for K_k gives:

$$P_k^f H^T = K_k\left(H P_k^f H^T + R_k\right)$$

$$K_k = P_k^f H^T\left(H P_k^f H^T + R_k\right)^{-1}$$

We call this the Kalman gain and we can plug this back into our P_k equation. We can check the dimensions of the gain with what we would expect. If x is nx1, we expect P to be nxn. H is the derivative matrix of our measurement with respect to our state with dimension mxn where m is the number of measurements. R is our measurement covariance with dimension mxm. So we then have nxn * nxm (transpose of H) to give nxm for $P_k^f H^T$. For the term in parentheses, we have mxn * nxn * nxm + mxm which is mxm. The inverse is mxm then we have nxm * mxm which gives us nxm for the Kalman gain. In our lithium ion SOC application, we will have one measurement, the terminal voltage, so we expect our Kalman gain to be nx1.

Back to our P_k equation, we can rearrange:

$$P_k = P_k^f - P_k^f H^T K_k^T - K_k H P_k^f + K_k H P_k^f H^T K_k^T + K_k R_k K_k^T$$

$$P_k = (I - K_k H) P_k^f (I - K_k H)^T + K_k R_k K_k^T$$

We then can expand to get:

$$P_k = (I - K_k H) P_k^f - (I - K_k H) P_k^f H^T K_k^T + K_k R_k K_k^T = (I - K_k H) P_k^f + \left[-P_k^f H^T + K_k\left(H P_k^f H^T + R_k\right)\right] K_k^T$$

$$= (I - K_k H) P_k^f + \left[-P_k^f H^T + P_k^f H^T\right] K_k^T = (I - K_k H) P_k^f$$

$$\rightarrow P_k = (I - K_k H) P_k^f$$

We note that $K_k\left(H P_k^f H^T + R_k\right) = P_k^f H^T\left(H P_k^f H^T + R_k\right)^{-1}\left(H P_k^f H^T + R_k\right) = P_k^f H^T$.

We know have our understanding of the Kalman filter and summarize in Figure 41.

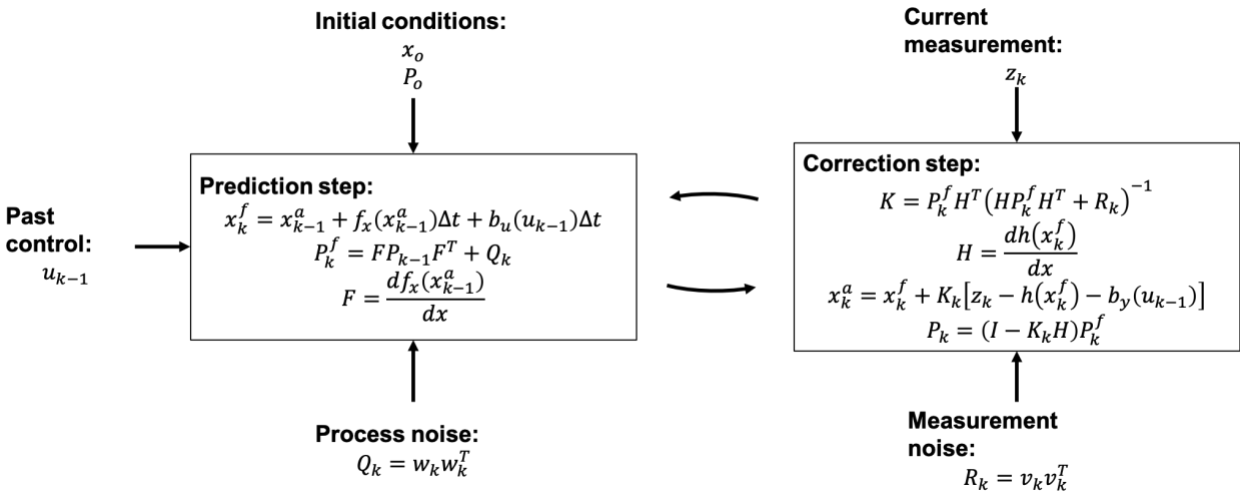

Figure 41: Kalman filter (really the Extended Kalman filter).

Initial conditions need to be defined for the Kalman filter. We will show how the Kalman filter can correct itself based on bad initial conditions. In the prediction step, we base our prediction on our understanding of the process (our model), we then update our error covariance matrix. In our correction step, we calculate our Kalman gain, and then use the Kalman gain and our measurement to correct our prediction. We then update our error covariance matrix using the Kalman gain as well.

An important note is that in Figure 41, we use u_{k-1} in our correction step instead of u_k. In our derivation we used u_k in our derivation (even though you can use u_{k-1} and the results stay the same). In practical applications, in our correction step we would have the measurement at z_k but only have knowledge of the past control output, u_{k-1}, since we are most likely using the Kalman filter result to (partly) decide on the next control action (e.g., current output).

Some of the challenges with implementing the Kalman filter are:

- Building an accurate model (in practice, the model may need to account for degradation effects. Imagine a battery in practice that after half its life, the battery capacity might reduce)
- Identifying the process and measurement noise covariance matrices. It's easiest to assume independence between noise variables to build a diagonal noise matrix but this may not be valid in some applications (for ours, we assume it is)

3.3 Application to Li-Ion Battery SOC

The Kalman filter combines a prediction (OCV estimate using our battery model with our OCV to SOC relationship) and a measurement (current measurement integrated over time to give SOC). In this section, we first find the state matrix (using two methods), the measurement equation, and then put the whole thing together in the Kalman filter.

3.3.1 Process Equation (State Matrix)

We first start with our basic equations:

$$V = V_{OCV} - V_0 - V_1 - V_2$$

$$V_{OCV} = f(SOC)$$

$$V_0 = i * R_0$$

$$\frac{dV_i(t)}{dt} = -\frac{V_i(t)}{R_i C_i} + \frac{i(t)}{C_i} \; (i = 1,2)$$

$$SOC(t) = SOC(t_0) - \int \frac{i(t)}{3600Q} dt$$

We want to derive our state matrix and then find $F = \frac{df(x)}{dx}$. There are two ways to do this (probably more but we'll look at two ways). We first start with the following equations for both methods:

$$\dot{x} = f(x) + b(u)$$

$$\frac{dSOC}{dt} = -\frac{i}{3600 * Q}$$

$$\frac{dV_1}{dt} = -\frac{V_1}{R_1 C_1} + \frac{i}{C_1}$$

$$\frac{dV_2}{dt} = -\frac{V_2}{R_2 C_2} + \frac{i}{C_2}$$

In my original approach, I didn't include V_0 as a state but I then added it and it gave me some more flexibility with specifying the process noise covariance matrix.

3.3.1.1 Euler Forward Integration

We can first use Euler forward integration and obtain:

$$SOC_k = SOC_{k-1} - \frac{i_{k-1}}{3600 * Q}\Delta t$$

$$V_{1,k} = V_{1,k-1} - \frac{V_{1,k-1}}{R_1 C_1}\Delta t + \frac{i_{k-1}}{C_1}\Delta t$$

$$V_{2,k} = V_{2,k-1} - \frac{V_{2,k-1}}{R_2 C_2}\Delta t + \frac{i_{k-1}}{C_2}\Delta t$$

$$V_{0,k} = i_{k-1} R_0$$

Using this method, we can say:

$$F = \frac{df}{dx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 - \frac{\Delta t}{R_1 C_1} & 0 & 0 \\ 0 & 0 & 1 - \frac{\Delta t}{R_2 C_2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Using this method, we can also say for our state estimate:

$$\begin{bmatrix} SOC \\ V_1 \\ V_2 \\ V_0 \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 - \frac{\Delta t}{R_1 C_1} & 0 & 0 \\ 0 & 0 & 1 - \frac{\Delta t}{R_2 C_2} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} SOC \\ V_1 \\ V_2 \\ V_0 \end{bmatrix}_{k-1} + \begin{bmatrix} -\frac{\Delta t}{3600 * Q} \\ \frac{\Delta t}{C_1} \\ \frac{\Delta t}{C_2} \\ R_0 \end{bmatrix} i_{k-1}$$

3.3.1.2 Analytical Integration

We can also assume that over our time increment, the current is constant and then we have:

$$SOC_k = SOC_{k-1} - \frac{i_{k-1}}{3600 * Q}\Delta t$$

$$V_{1,k} = V_{1,k-1} e^{-\frac{\Delta t}{R_1 C_1}} + i_{k-1} R_1 \left(1 - e^{-\frac{\Delta t}{R_1 C_1}}\right)$$

$$\mathrm{V}_{2,k} = \mathrm{V}_{2,k-1}\mathrm{e}^{-\frac{\Delta \mathrm{t}}{\mathrm{R}_2\mathrm{C}_2}} + i_{k-1}R_2\left(1 - e^{-\frac{\Delta t}{R_2C_2}}\right)$$

$$V_{0,k} = i_{k-1}R_0$$

Using this method, we then have:

$$F = \frac{df}{dx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-\frac{\Delta t}{R_1C_1}} & 0 & 0 \\ 0 & 0 & e^{-\frac{\Delta t}{R_2C_2}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Using this method, we can also say for our state estimate:

$$\begin{bmatrix} SOC \\ V_1 \\ V_2 \\ V_0 \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{-\frac{\Delta t}{R_1C_1}} & 0 & 0 \\ 0 & 0 & e^{-\frac{\Delta t}{R_2C_2}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} SOC \\ V_1 \\ V_2 \\ V_0 \end{bmatrix}_{k-1} + \begin{bmatrix} -\frac{\Delta \mathrm{t}}{3600 * Q} \\ R_1\left(1 - e^{-\frac{\Delta t}{R_1C_1}}\right) \\ R_2\left(1 - e^{-\frac{\Delta t}{R_2C_2}}\right) \\ R_0 \end{bmatrix} i_{k-1}$$

3.3.2 Measurement Equation (Observation Matrix)

For our measurement, we have:

$$z_k = h(x_k) + b_y(u_k) + v_k$$

From our equivalent circuit model, we have:

$$V = V_{OCV} - V_0 - V_1 - V_2$$

$$V = V_{OCV}(SOC) - i * R_0 - V_1 - V_2$$

$$\frac{dV_i}{dt} = -\frac{V_i}{R_iC_i} + \frac{i}{C_i}\ (i = 1,2)$$

$$V_0 = iR_0$$

The observation matrix is much more straightforward compared to our state matrix, we have that our measurement is the terminal voltage. We have:

$$H = \frac{dh(x_k)}{dx} = \begin{bmatrix} \frac{dV_{OCV}}{dSOC} & -1 & -1 & -1 \end{bmatrix}$$

This observation matrix holds for both state matrix approaches.

3.3.3 Covariance Matrices

Here we detail how we find our covariance matrices. I assume that all error sources are independent from each other so we end up with diagonal matrices. We take a sigma approach to finding the covariance matrices.

3.3.3.1 Process, Q

The process covariance matrix is the trickiest to find. We again start with the process equations for Euler forward integration:

$$SOC_k = SOC_{k-1} - \frac{i_{k-1}}{3600 * Q}\Delta t$$

$$V_{1,k} = V_{1,k-1} - \frac{V_{1,k-1}}{R_1C_1}\Delta t + \frac{i_{k-1}}{C_1}\Delta t$$

$$V_{2,k} = V_{2,k-1} - \frac{V_{2,k-1}}{R_2 C_2}\Delta t + \frac{i_{k-1}}{C_2}\Delta t$$

$$V_{0,k} = i_{k-1} R_0$$

And for analytical integration:

$$SOC_k = SOC_{k-1} - \frac{i_{k-1}}{3600 * Q}\Delta t$$

$$\mathrm{V}_{1,k} = \mathrm{V}_{1,k-1}\mathrm{e}^{-\frac{\Delta \mathrm{t}}{\mathrm{R_1 C_1}}} + i_{k-1} R_1 \left(1 - e^{-\frac{\Delta t}{R_1 C_1}}\right)$$

$$\mathrm{V}_{2,k} = \mathrm{V}_{2,k-1}\mathrm{e}^{-\frac{\Delta \mathrm{t}}{\mathrm{R_2 C_2}}} + i_{k-1} R_2 \left(1 - e^{-\frac{\Delta t}{R_2 C_2}}\right)$$

$$V_{0,k} = i_{k-1} R_0$$

In both methods, we have:

$$SOC_k = f_{SOC}(SOC_{k-1}, i_{k-1}, Q)$$

$$V_{i,k} = f_{V_i}\left(V_{i,k-1}, R_i, C_i, i_{k-1}\right)\ (i = 1,2)$$

$$V_{0,k} = f_{V_0}(R_0, i_{k-1})$$

To find the covariance matrices, we assume an error for all variables that the states are functions of except for the previous states (since we assume that the error is already baked into them). We can differentiate each state equation and obtain:

$$dSOC_k = \frac{\partial f_{SOC}}{\partial i_{k-1}} di + \frac{\partial f_{SOC}}{\partial Q} dQ$$

$$dV_{i,k} = \frac{\partial f_{V_i}}{\partial i_{k-1}} di + \frac{\partial f_{V_i}}{\partial R_i} dR_i + \frac{\partial f_{V_i}}{\partial C_i} dC_i$$

$$dV_{0,k} = \frac{\partial f_{V_0}}{\partial i_{k-1}} di + \frac{\partial f_{V_0}}{\partial R_0} dR_0$$

Our covariance matrix is $w_k w_k^T = Q_k$. We then want our errors squared so we have for the diagonals:

$$(dSOC_k)^2 = \left(\frac{\partial f_{SOC}}{\partial i_{k-1}} di\right)^2 + \left(\frac{\partial f_{SOC}}{\partial Q} dQ\right)^2$$

$$\left(dV_{i,k}\right)^2 = \left(\frac{\partial f_{V_i}}{\partial i_{k-1}} di\right)^2 + \left(\frac{\partial f_{V_i}}{\partial R_i} dR_i\right)^2 + \left(\frac{\partial f_{V_i}}{\partial C_i} dC_i\right)^2$$

$$\left(dV_{0,k}\right)^2 = \left(\frac{\partial f_{V_0}}{\partial i_{k-1}} di\right)^2 + \left(\frac{\partial f_{V_0}}{\partial R_0} dR_0\right)^2$$

We then have:

$$Q_k = \begin{bmatrix} (dSOC_k)^2 & 0 & 0 & 0 \\ 0 & \left(dV_{1,k}\right)^2 & 0 & 0 \\ 0 & 0 & \left(dV_{2,k}\right)^2 & 0 \\ 0 & 0 & 0 & \left(dV_{0,k}\right)^2 \end{bmatrix}$$

In this derivation, we assumed that our error sources were independent. This is probably not totally appropriate since we fitted our parameters together so any error in R_i $(i = 0,1,2)$ is probably related to the other resistances and the error in C_i $(i = 1,2)$. We could have included the time step error as well since that

might vary a bit. Since we found our parameters for V_0, V_1, and V_2 together, cross terms in the Q matrix could be applicable. However, for simplicity, we assume independent error sources.

3.3.3.2 Measurement, R

Since our measurement is the terminal voltage only, our error source is just our error in voltage. So we have:

$$R_k = [dV^2]$$

3.3.4 Some Closing Thoughts

One may notice in our equations for our measurement, we used i_k. In practice we will only have i_{k-1}. This introduces some more error. We also have that $V_{0,k} = i_{k-1}R_0$ when it actually is i_k. We have to make some adjustments to the Kalman filter to use online. We could introduce better estimate algorithms but we don't go into this too much.

In summary, we derived the Kalman filter and applied it to the lithium ion SOC application and we note the following:

- In the derivation, we assume for the measurement that we have u_k (i_k in our case) but we actually don't have this. This doesn't change the derivation too much but it does introduce some error
- For the state equations, we had to assume some method of integration. We introduce two methods (we will use the analytical integration one). To optimize this, we could explore different time increments but since we are relying on an external data source, we are limited here
- For the covariance matrices, we assume independence which might not be totally valid but it's easier to implement

3.4 Results

I'll show results for two scenarios. The first is where the initial SOC is found by the initial terminal voltage (when no current is applied). This would be an ideal situation where the initial SOC is known. The second case is where we say that the initial SOC is 0.75 of the SOC found from the initial terminal voltage so the initial SOC in the second scenario is 0.75 of the initial SOC used in the first scenario. This should illustrate the benefit of the Kalman filter better.

We call the first scenario the perfect initial SOC scenario and the second the unknown initial SOC. I don't dive too deep into interpreting the results but wanted to show how the Kalman filter can be beneficial compared to Coulomb counting (current integration) with not that much more complexity. I don't show the results when only relying on the terminal voltage and using the OCV – SOC relationship since those results are very messy and don't give any practical use.

It's also possible to only update using the Kalman filter every x iterations. I played around with this a bit but don't show this here. This would correspond to an application where we use the model and then every minute or at some other frequency, use a measurement to correct our calculation.

I also only look at the FUDS data. An example of the voltage, current, charge capacity, and SOC data is shown in

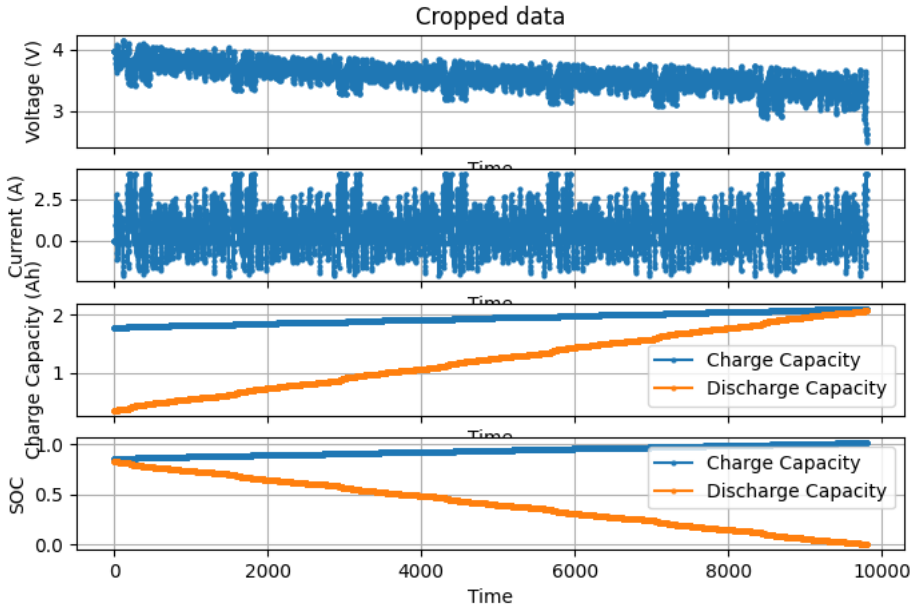

Figure 42: FUDS 0 deg. C 80 % SOC data.

The current changes fairly rapidly with voltage changes corresponding the changing current and changing SOC. We use the discharge capacity to calculate the actual SOC data.

3.4.1 Perfect Initial SOC

The results are shown in Figure 43 through Figure 48.

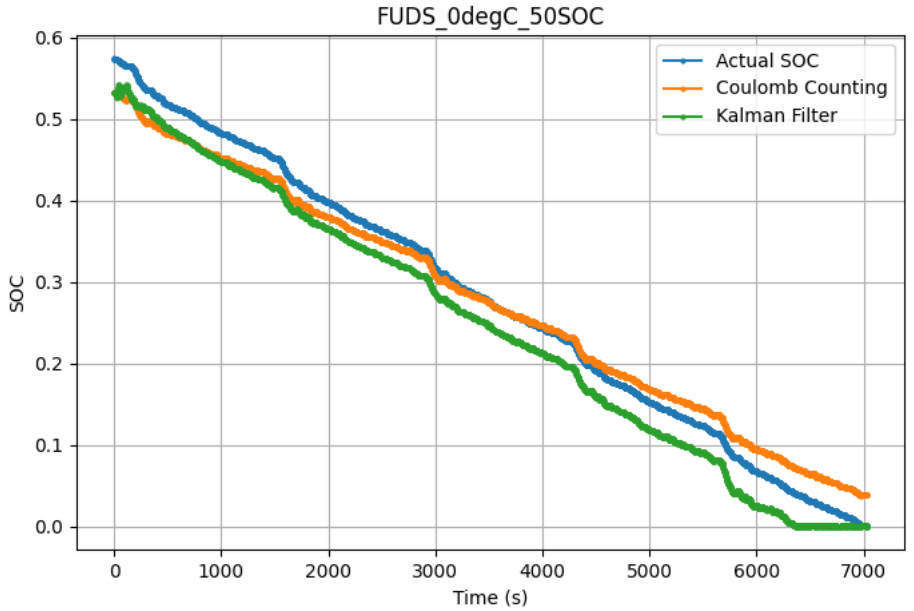

Figure 43: FUDS 0 deg. C 50 % SOC results.

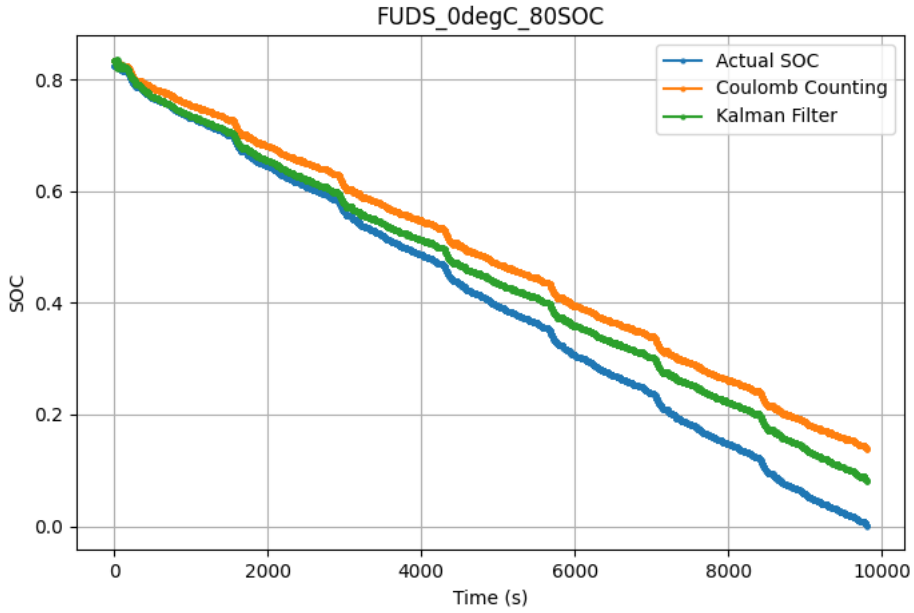

Figure 44: FUDS 0 deg. C 80 % SOC results.

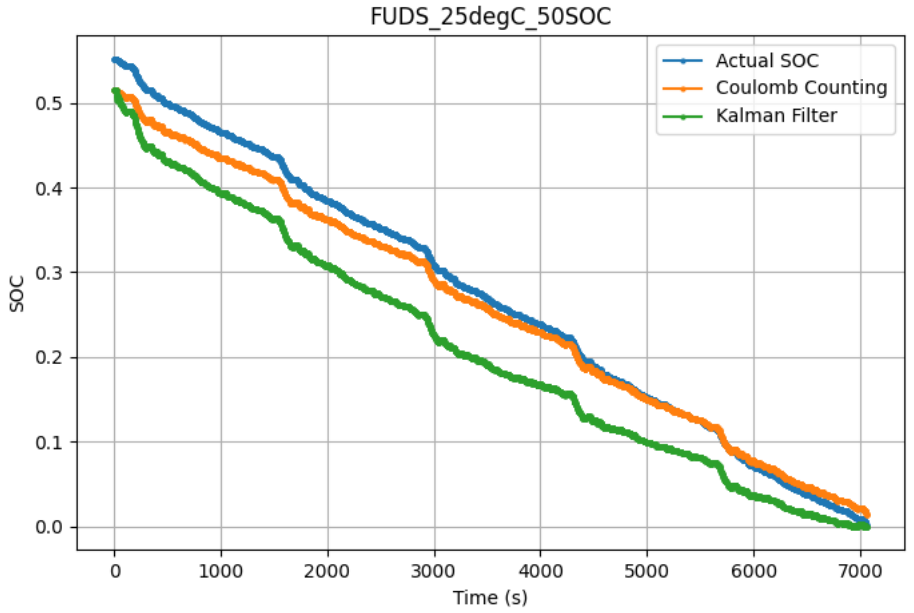

Figure 45: FUDS 25 deg. C 50 % SOC results.

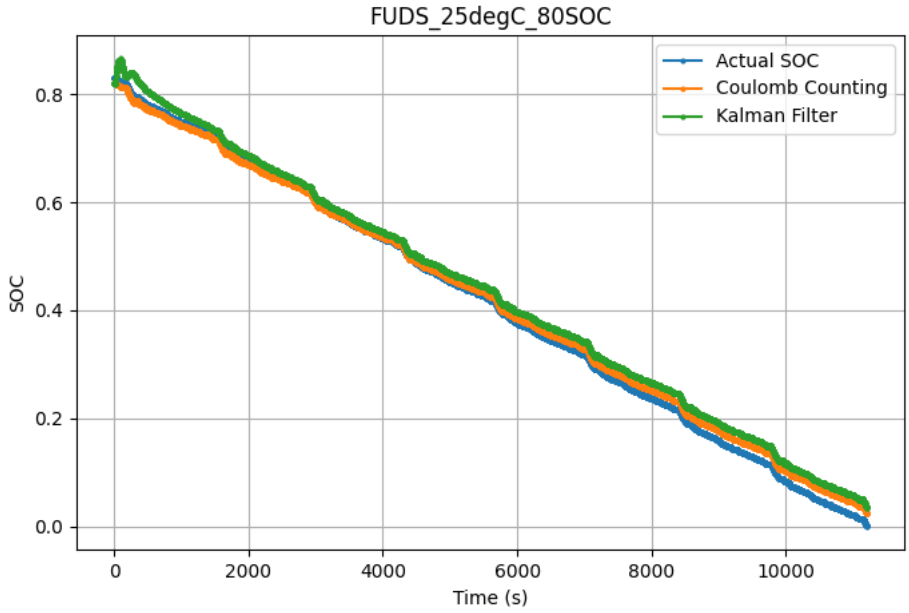

Figure 46: FUDS 25 deg. C 80 % SOC results.

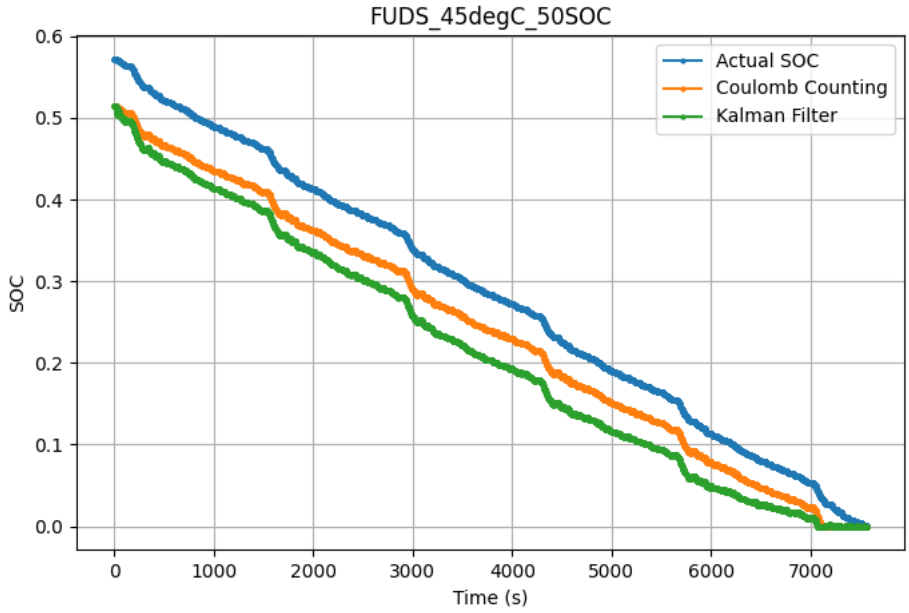

Figure 47: FUDS 45 deg. C 50 % SOC results.

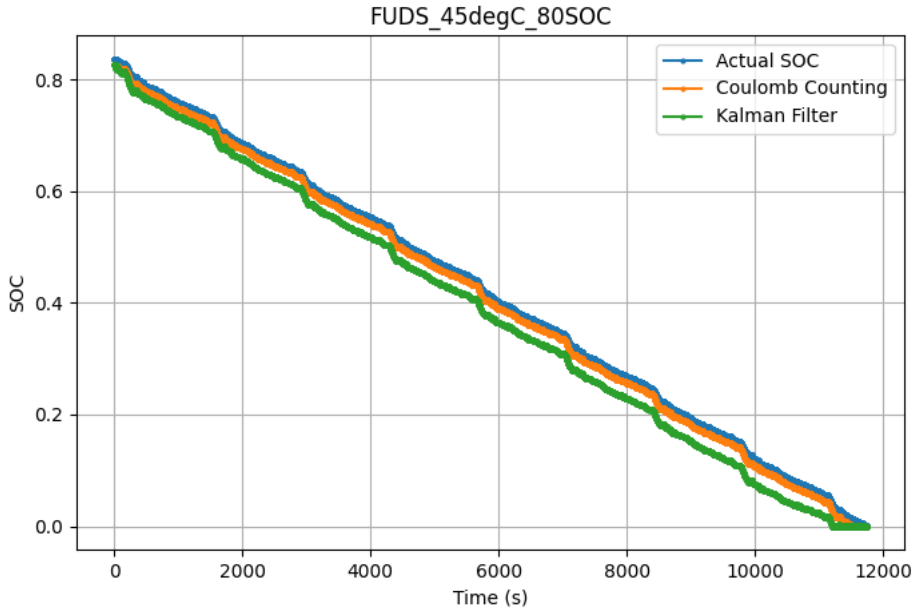

Figure 48: FUDS 45 deg. C 80 % SOC results.

Looking at these results, the Kalman filter doesn't outperform the Coulomb counting method and often underperforms. We see that the Kalman filter follows Coulomb counting pretty well as well suggesting that the Kalman gain in the filter is small.

We will look into a little more detail in the FUDS 0 deg. C 80% SOC test. We show the Kalman gain in Figure 49.

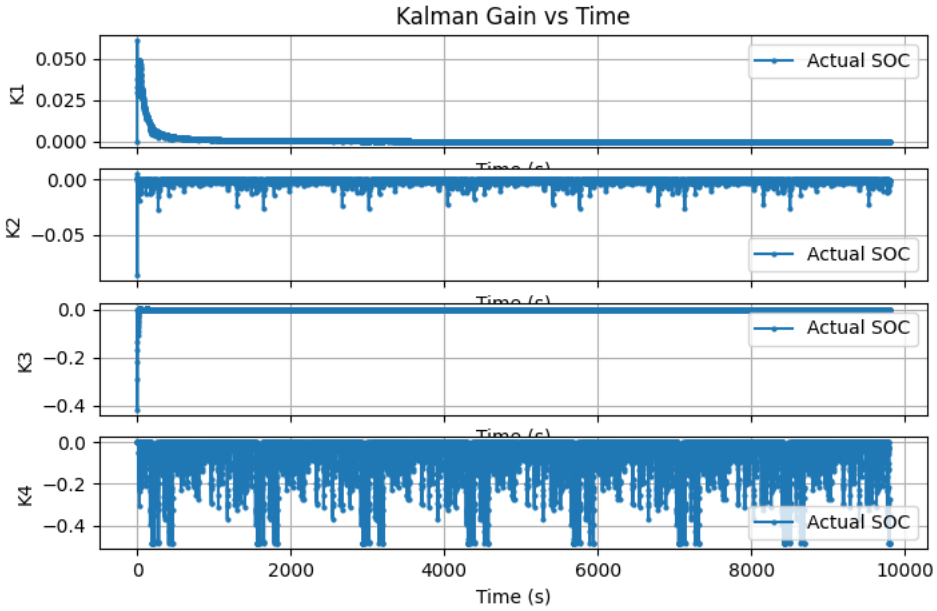

Figure 49: FUDS 0 deg. C 80% SOC Kalman gain.

It's interesting to see how the Kalman gains quickly approach 0 (or close to it) suggesting an initial correction but then mostly relying on the model. It's also important to note that our Kalman gain matrix is a 4x1 matrix

for our 4 states. We see that the Kalman gain for the 4th state (V_0) is actually quite large. This is not surprising since V_0 is more dependent on the current at the current state instead of the past state. This results in us a seeing a fairly large correction for that state.

In Figure 51, we plot the Kalman gain multiplied by the measurement minus the model measurement.

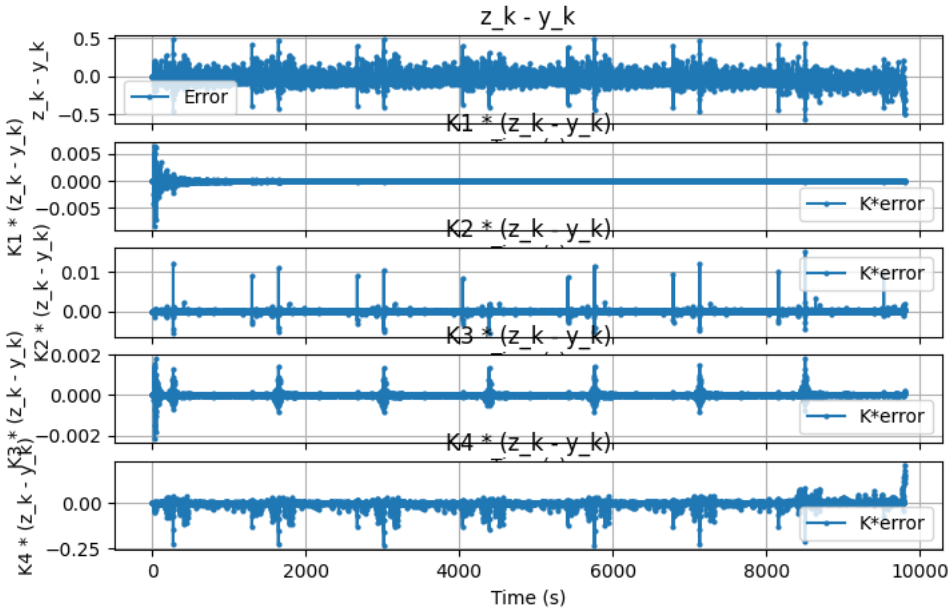

Figure 50: FUDS 0 deg. C 80% SOC Kalman gain * (measurement – model measurement).

We see that for our first state, SOC, that the correction term ($K_k[z_k - h(x_k^f) - b_y(u_{k-1})]$), is quite small suggesting we are not correcting our SOC state much. The correction also bounces around with it's average around 0 at the beginning of the test showing the it corrects SOC in one direction with a certain magnitude then almost re-corrects itself in the opposite direction with the same magnitude. We see that we have small changes for the other states as well except for the V_0 state which we already discussed.

We can also compare the terminal voltage (actual V) and the voltage predicted by the Kalman filter. This is shown in Figure 51.

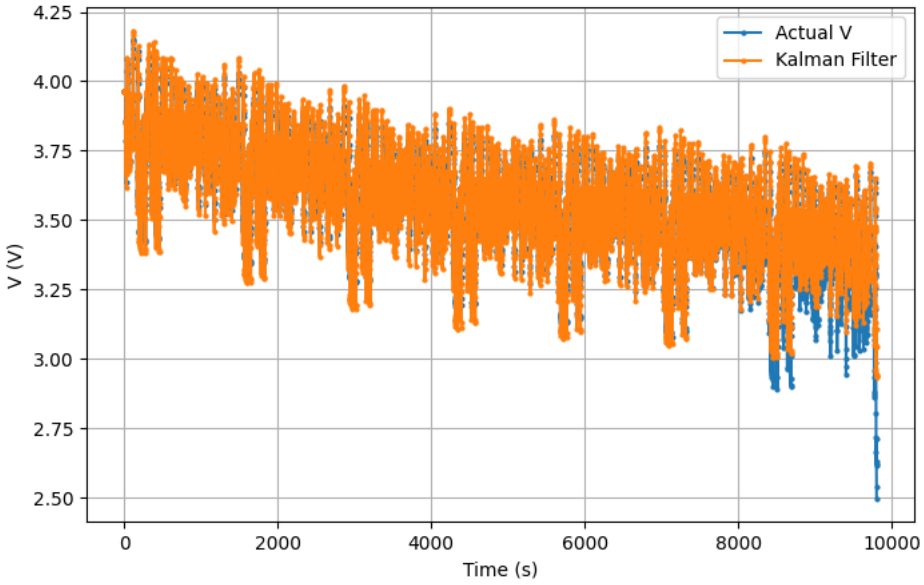

Figure 51: Voltage comparison for FUDS 0 deg. C 80 % SOC.

It's hard to see since the plot is crowded but we see a fairly good match except at low OCV and SOC values (end of test). This is expected since we saw that at low OCV, the dynamics can be significant and we ended up choosing to use the OCV-SOC curve from the Incremental method. Potentially, the OCV-SOC relationship from the Low Current test could provide better results. We can also look at the voltage for all the states. This is shown in Figure 52

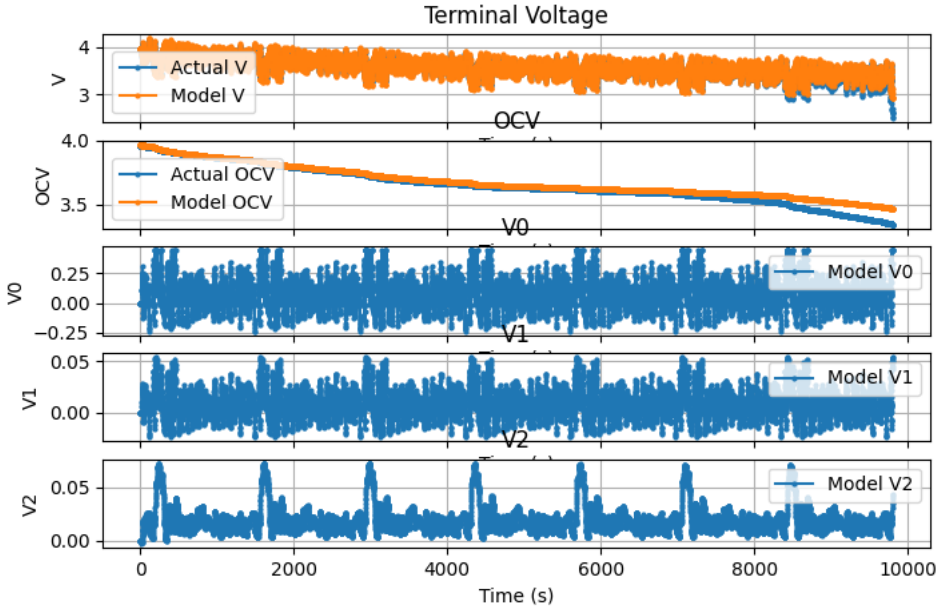

Figure 52: Voltage comparison for all states for FUDS 0 deg. C 80 % SOC.

We can compare the OCV prediction to the known OCV (based on the SOC so it's not actually known but we say it is). We see again that the OCV prediction deviates a bit at the end of the test. It's interesting to

note that V1 and V2 are similar magnitude suggesting that the FUDS test does a good job of capturing fast and slow dynamics.

I plot the average RMS error (average of 50% and 80% SOC) as a function of temperature in Figure 53

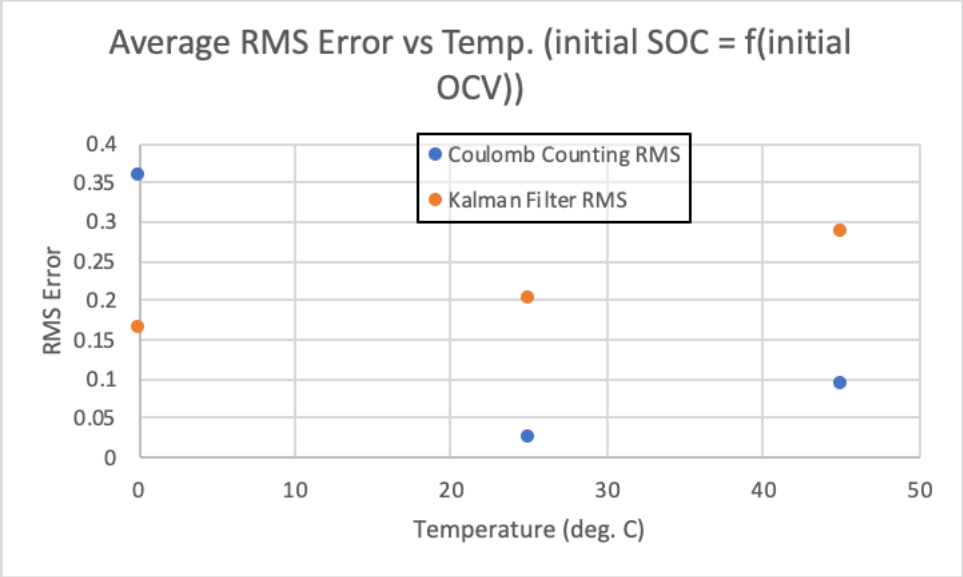

Figure 53: Average RMS Error vs Temperature (Perfect Initial SOC).

Figure 53 shows that Coulomb counting outperforms the Kalman filter except at 0 deg. C. Remember, at 0 deg. C, we had the best fit for our model parameters so this isn't too surprising. We still see that the RMS error from the Kalman filter and Coulomb counting are comparable being the same order of magnitude.

There's a lot more we could say but we'll move on to the next section now.

3.4.2 Unknown Initial SOC

In this section, we compare Coulomb counting to the Kalman filter with the initial SOC off by 25%. We already know that Coulomb counting has no way of correcting itself and we want to show the benefit of the Kalman filter when the initial state prediction is inaccurate.

The results are shown in Figure 54 through Figure 59.

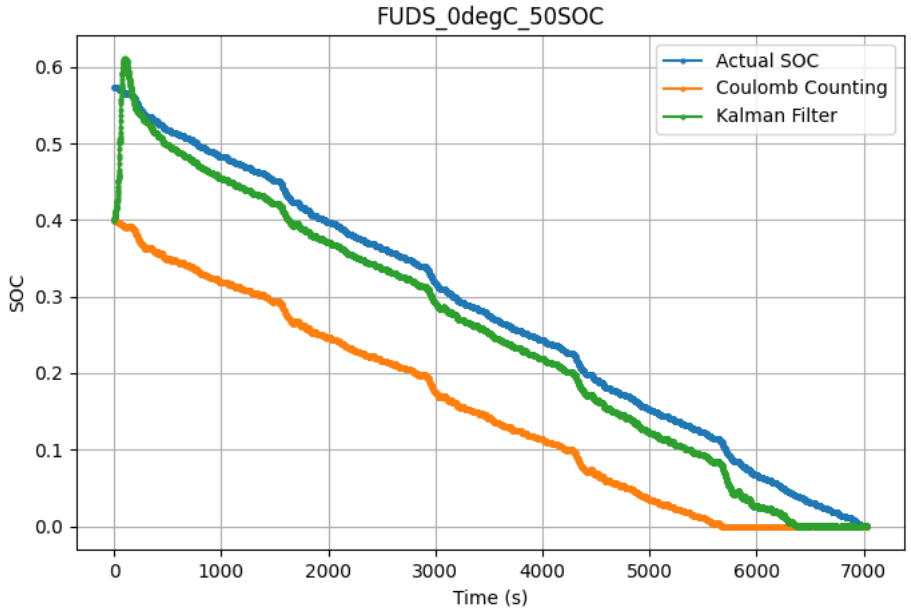

Figure 54; FUDS 0 deg. C 50 % SOC (Initial Guess Off).

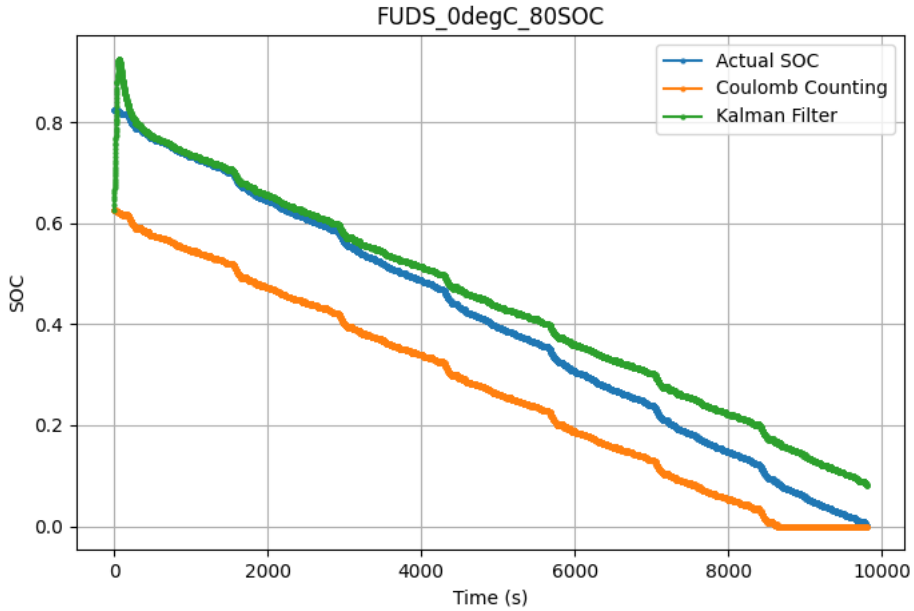

Figure 55: FUDS 0 deg. C 80 % SOC (Initial Guess Off).

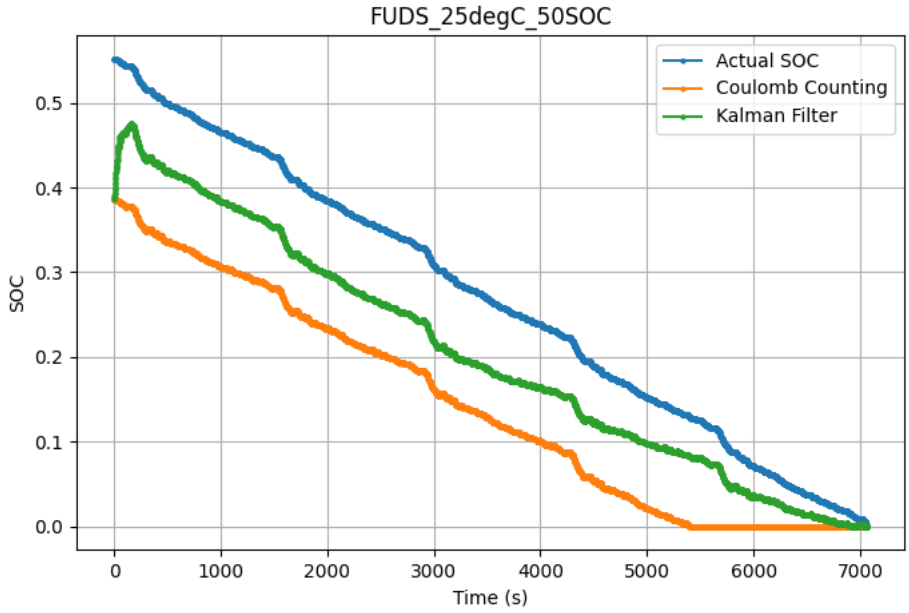

Figure 56: FUDS 25 deg. C 50 % SOC (Initial Guess Off).

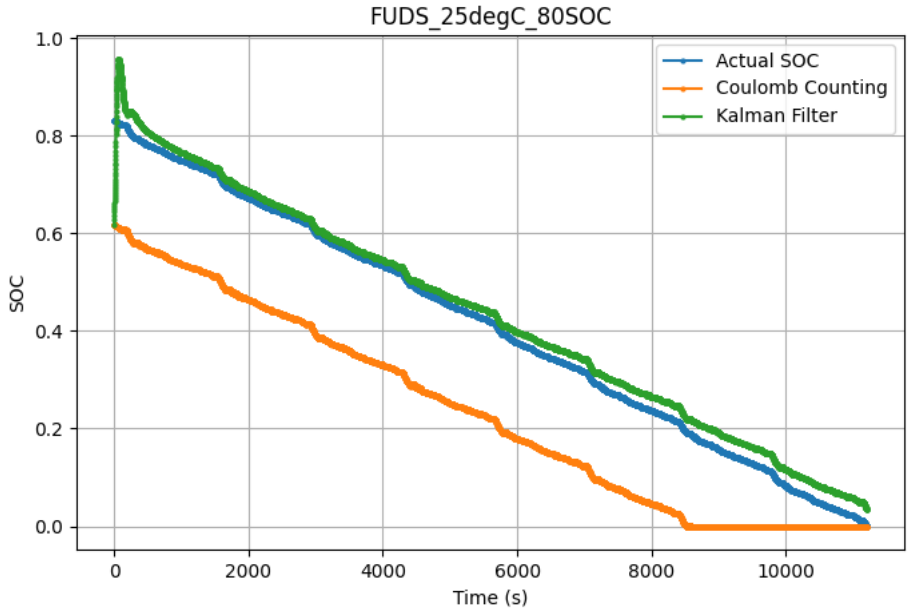

Figure 57: FUDS 25 deg. C 80 % SOC (Initial Guess Off).

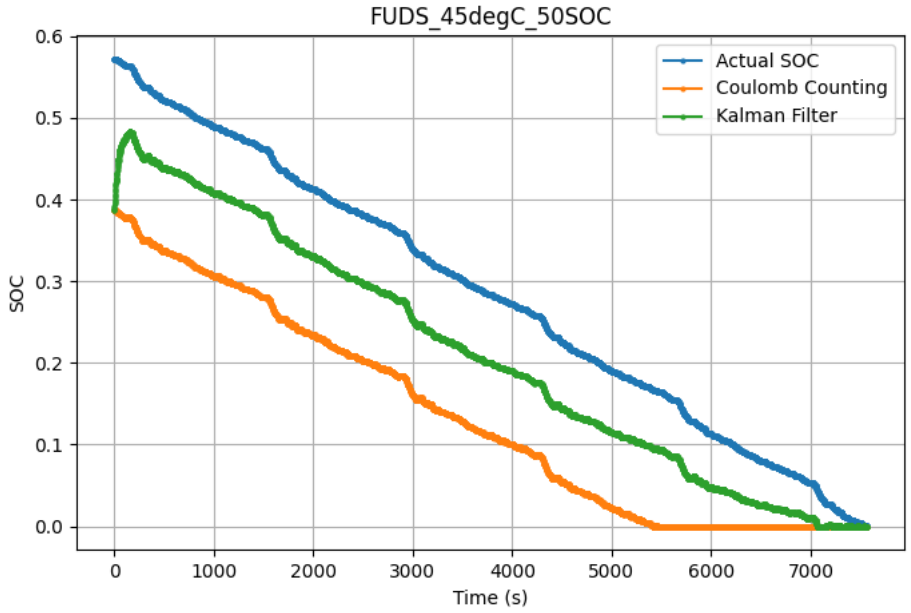

Figure 58: FUDS 45 deg. C 50 % SOC (Initial Guess Off).

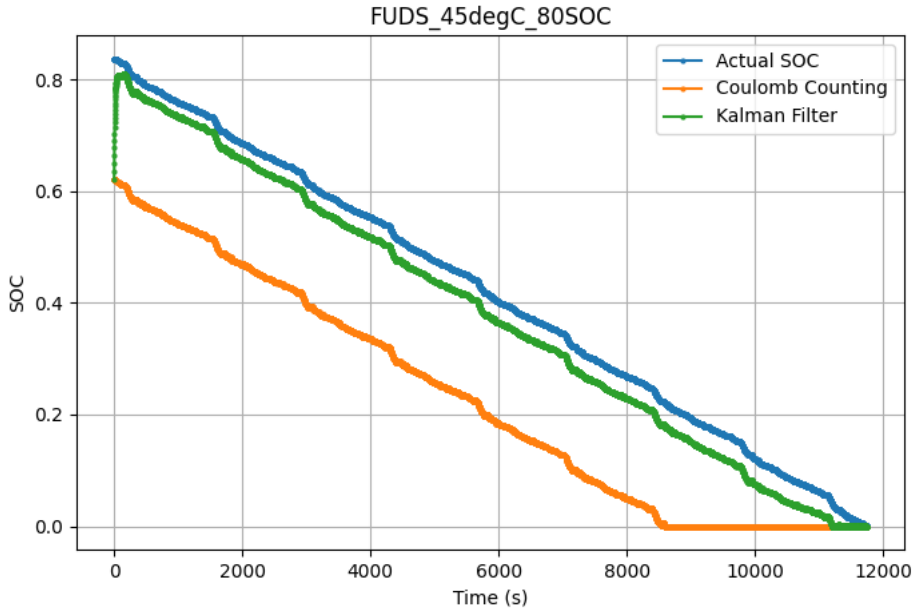

Figure 59: FUDS 45 deg. C 80 % SOC (Initial Guess Off).

It's cool to see how the Kalman filter quickly corrects itself. It's also interesting to see how the Kalman filter sometimes overshoots but sometimes undershoots when it corrects. We could draw a block diagram and look at the different ways of analyzing the Kalman filter to tune it better but I'll leave that for a future idea.

We'll look into more detail for the FUDS 0 deg. C 80 % SOC case again. We first show the Kalman gain in Figure 60.

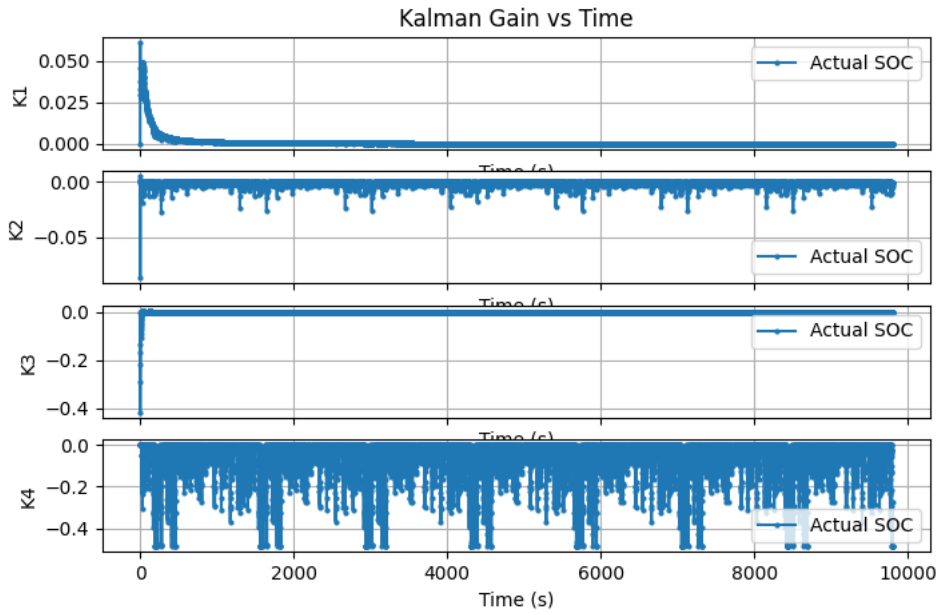

Figure 60: Kalman Gain for FUDS 0 deg. C 80 % SOC (Initial Guess Off).

A little surprisingly, we don't see a huge different between Figure 60 and Figure 49. It's apparent that the Kalman gain must be the reason why the Kalman filter corrects itself but perhaps since it corrects itself so quickly, we don't see the influence on the Kalman gain. Perhaps if we zoomed in on the x axis we would. Overall, the Kalman gain is still very close to what it was for the Perfect Initial Guess case. Once the Kalman filter corrects itself, it must go back to relying on the model. We show the Kalman gain multiplied by the difference between the measurement and the model measurement in Figure 61.

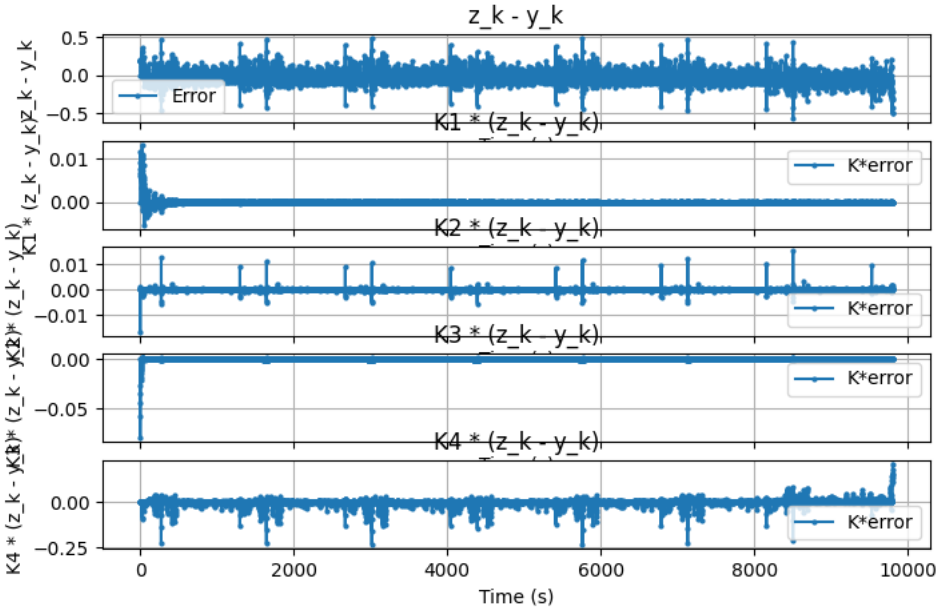

Figure 61: FUDS 0 deg. C 80% SOC Kalman gain * (measurement – model measurement) (Initial Guess Off).

We see that for our first state, SOC, that the correction term $(K_k[z_k - h(x_k^f) - b_y(u_{k-1})]$, is still small but that it is correcting in the positive direction compared to the result we got with the Perfect Initial Guess in Figure 61. It's cool to see how the Kalman gain pushes the SOC up to correct itself. We biased the SOC initial guess down so it makes sense to correct itself by pushing up. The other states must correct themselves to give the right terminal voltage with the OCV voltage being corrected also to align with the SOC estimate.

Figure 62 shows the terminal voltage comparison.

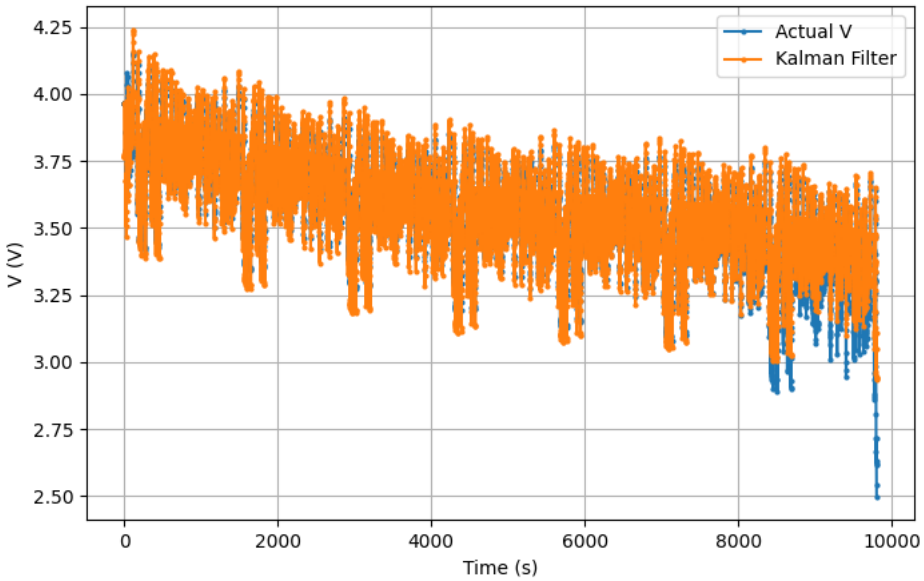

Figure 62: Terminal Voltage Comparison (Initial Guess Off).

We see a good match again with some deviation around low OCV values which we saw before. We also note that at the very beginning we see a deviation compared to the results in Figure 51. This must be the result of our initial guess being off but then we see that it quickly corrects and we get a good match for the rest of the test minus the end where we know we have problems.

Figure 63 shows the same for all states. We compare this to the results with the Perfect Initial Guess in Figure 52.

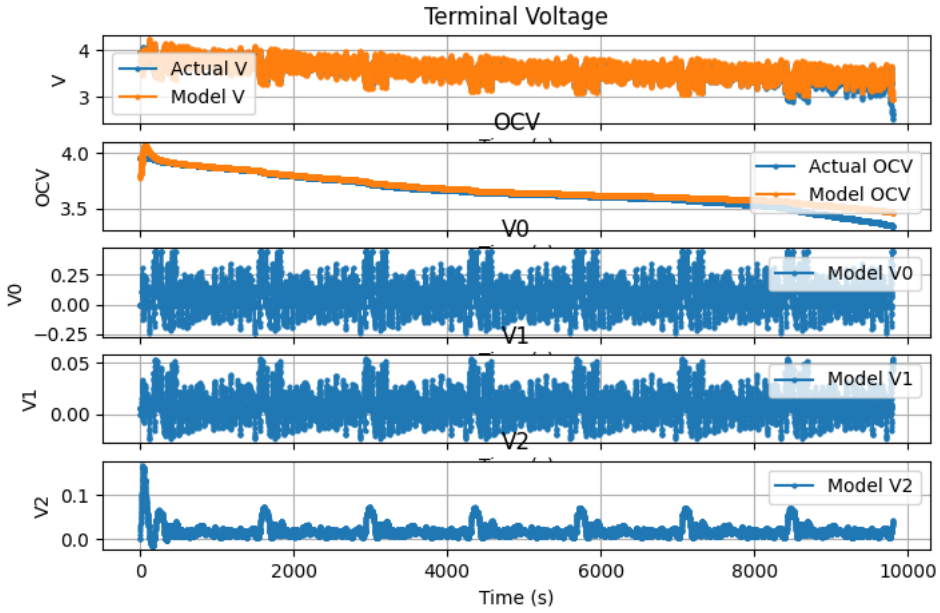

Figure 63: Voltage Comparison (Initial Guess Off).

We see at the beginning, that the OCV model is off and how it corrects itself. Also note that V2 spikes up initially also. This may be due to the Kalman filter trying to correct the voltage but that the time constant of V2 is slower so takes longer to settle.

Figure 64 shows the average RMS error vs temperature for the Initial Guess Off case.

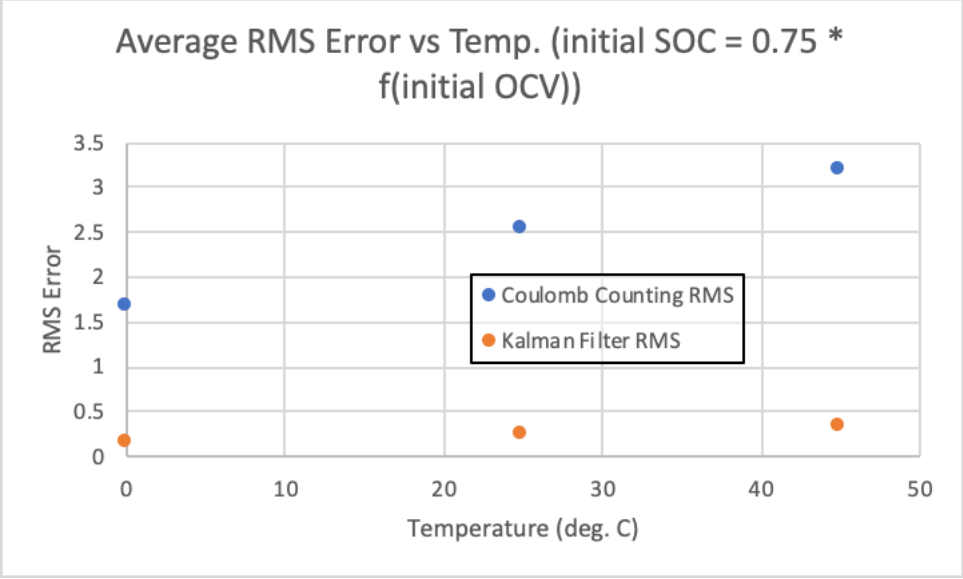

Figure 64: Average RMS Error vs Temperature (Initial Guess Off).

We see that the Kalman filter out performs the Coulomb counting method by an order of magnitude. We also see that, compared to the results in Figure 53, that the RMS error with the Kalman filter is comparable (in some cases, very similar) to the RMS error with the Perfect Initial guess. We plot the RMS error of the Initial Guess Off vs the Perfect Initial Guess for the Kalman filter in Figure 65 and for the Coulomb counting method in Figure 66.

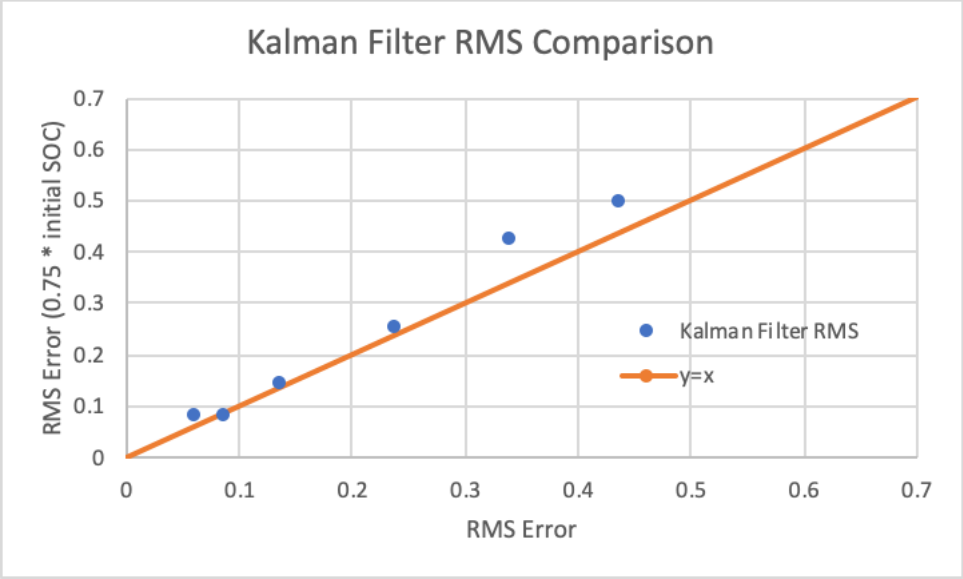

Figure 65: RMS Error (Initial Guess Off) vs RMS Error (Perfect Initial Guess) for the Kalman Filter.

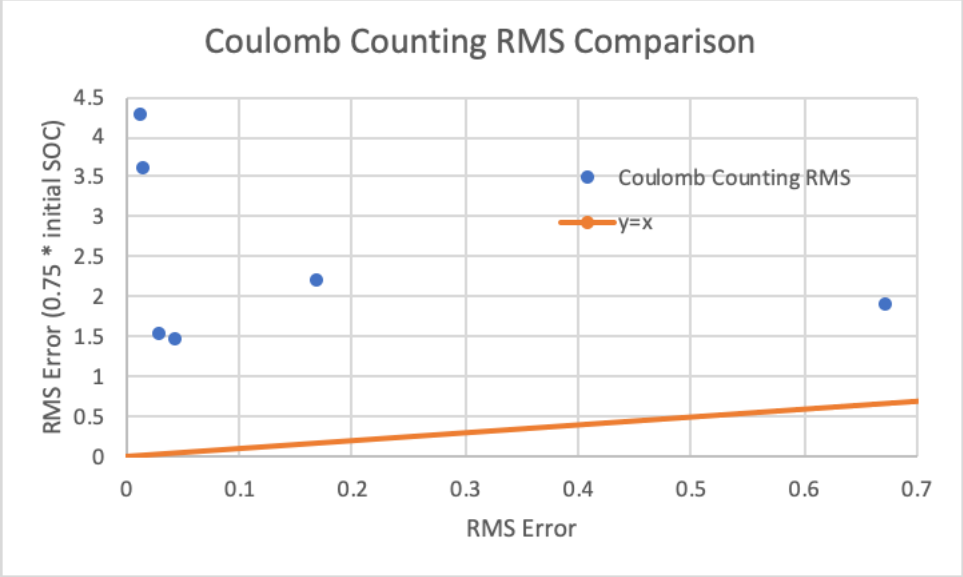

Figure 66: RMS Error (Initial Guess Off) vs RMS Error (Perfect Initial Guess) for Coulomb Counting.

Points above the y=x line correspond to the RMS error for the Initial Guess Off scenario being higher (worse) than the Perfect Initial Guess. We see that for the Kalman filter, the RMS error is worse for the Initial Guess Off scenario compared to the Perfect Initial Guess (except for one point which is interesting) but not that much worse. We see for the Coulomb counting method; the RMS error is way worse (order of magnitude higher) for the Initial Guess Off scenario compared to the Perfect Initial Guess scenario.

We showed how the Kalman filter can be applied to the lithium ion battery SOC estimation. While the Coulomb counting provided smaller error for the Perfect Initial Guess, we showed how this method is not robust against uncertainty in initial guesses and we can see how over time, the error may grow since any error integrates over time while the Kalman filter provides some robustness against uncertainty.

4 Some Closing Thoughts

I was able to show how to build the Kalman filter for a SOC application for lithium ion batteries. We saw how the filter can be derived and built and implemented while also some discussing some challenges. The

Kalman filter has obvious benefits in robustness that don't require too much complexity. There were some challenges that we discussed including:

- Developing an accurate model
    - Especially considering degradation over time
    - We also used fixed parameter values where we could potentially make them functions of SOC (in addition to temperature)
- Building the state and measurement equations and linearizing
    - We compared two methods: analytical integration and forward Euler
- Finding the process and noise covariance matrices
    - We implemented fairly simple covariance matrices where we assumed independence between the noise variables and constant values
- We didn't talk much about manual tuning but from the results, we see an obvious opportunity in how the Kalman filter can be tuned to better converge on the actual SOC
    - In some cases, the Kalman filter deviates and potentially could be corrected by scaling some variables influencing the Kalman gain
- In implementing the Kalman filter, one can choose how often the filter should correct itself using measurement data
    - In our results, we assumed that the filter corrects itself every time update but this is potentially unnecessary and potentially not beneficial
- We were constrained on our update rate but one can see how a study for determining the software update rate for the Kalman filter could be important
- There are probably others that I am forgetting

In this project, I aimed to develop a better understanding of the Kalman filter and practice implementing it. With respect to those goals, I would say I achieved them and also identified several other opportunities to expand my understanding that I may take advantage of in the future.