AE567 Final Project: State of Charge Estimation for Battery Management System

Soham Shirish Phanse
Graduate Student, Department of Mechanical Engineering
University of Michigan Ann Arbor
Ann Arbor, United States of America
ssphanse@umich.edu

***Abstract*—The estimation of State of Charge (SOC) is a critical function in Battery Management Systems (BMS) for electric vehicles (EVs), essential for ensuring optimal performance, range prediction, and safety. This study explores advanced Kalman filtering techniques—Extended Kalman Filter (EKF), Adaptive EKF (AEKF), Unscented Kalman Filter (UKF), and Gauss-Hermite Kalman Filter (GHKF)—applied to a second-order equivalent circuit battery model. The proposed methods leverage the model's non-linear dynamics and incorporate system uncertainties to enhance SOC prediction accuracy. Simulation results demonstrate the performance trade-offs between filters across tracking error and computational cost, with AEKF and GHKF showcasing significant improvements in accuracy and adaptability over traditional approaches. The findings highlight the importance of adaptive filtering in robust SOC estimation, offering insights for next-generation EV BMS design.**

***Index Terms*—State of charge (SOC), battery management system (BMS), Kalman filters, electric vehicles (EVs), nonlinear estimation, adaptive filtering**

I. Background

Electric vehicles (EVs) are rapidly gaining traction as a sustainable alternative to conventional internal combustion engine vehicles, driven by the global push to reduce greenhouse gas emissions and reliance on fossil fuels. Advancements in battery technology and energy management systems have been pivotal in making the EVs more efficient, reliable, and accessible to consumers. At the heart of these developments lies the battery management system (BMS), which ensures the safe and efficient operation of the battery pack—the core energy source of EVs. One of the most critical functions of the BMS is the estimation of the battery's state of charge (SOC), a parameter essential for range prediction, energy optimization, and preventing operational failures.

Various methods are available for SOC estimation, including data-driven techniques like machine learning, coulomb counting, and electrochemical impedance spectroscopy. In the Coulomb counting approach, SOC is evaluated as the ratio between the current capacity, as the integral of the battery current, and the nominal capacity of the battery [1]. However this method is very sensitive to the initial SOC condition and the integration can easily diverge in presence of noise.

Other methods like impedance spectroscopy, and using neural networks are also demonstrated [7], however, model-based methods such as Kalman filters stand out due to their ability to incorporate battery dynamics and provide accurate predictions even in the presence of noise and uncertainties, alsong with incurring comparatively low computational cost. By leveraging mathematical models of battery behavior, Kalman filters excel in scenarios where real-time adaptability and robust performance are required, making them a preferred choice for SOC estimation in EVs and other advanced applications.

Prof. Alex Gorodetsky, Mr. Hongyu Zhou

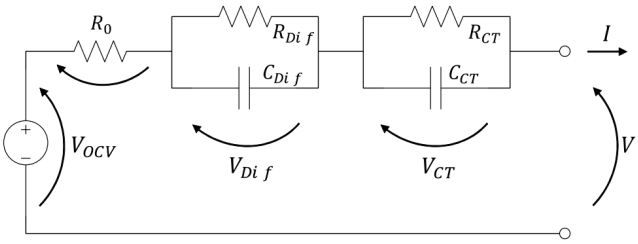

Fig. 1. Second Order Electric Circuit based model

II. Problem Setup

Here we provide a brief description of the battery mathematical model which becomes a basis for application of a variety of Kalman filters. In the literature, 2 different categories of models have been provided, electrochemical process based models, or equivalent circuit based models In the present study, a second order electric circuit based model as shown below: The symbols represent the following quantities:

1) V_{OCV} = open circuit voltage
2) R_0 = battery cell internal resistance
3) V = output voltage
4) V_{Dif} = voltage difference due to diffusion
    a) R_{Dif} = equivalent diffusion resistance
    b) C_{Dif} = equivalent diffusion capacitance
5) V_{CT} = voltage difference due to charge transfer
    a) R_{CT} = equivalent charge transfer resistance
    b) C_{CT} = equivalent charge transfer capacitance
6) I = current

A. State Space Model

An equivalent state space formulation in discrete time domain is as follows: (this is the dynamics model)

$$SOC(k+1) = SOC(k) - \frac{\Delta t}{Q_{nom}} I(k) \tag{1}$$

$$V_{CT}(k+1) = e^{-\frac{\Delta t}{\tau_{CT}}} V_{CT}(k) + R_{CT}\left(1 - e^{-\frac{\Delta t}{\tau_{CT}}}\right) I(k) \tag{2}$$

$$V_{Dif}(k+1) = e^{-\frac{\Delta t}{\tau_{Dif}}} V_{Dif}(k) + R_{Dif}\left(1 - e^{-\frac{\Delta t}{\tau_{Dif}}}\right) I(k) \tag{3}$$

where the symbols are defined as follows:

1) SOC = State of charge
2) Δt = time step
3) k = discrete time instant
4) Q_{nom} = nominal battery capacity
5) $\tau_{CT} = R_{CT}C_{CT}$ = time constant diffusion equivalent circuit
6) $\tau_{Dif} = R_{Dif}C_{Dif}$ = time constant charge transfer equivalent circuit

The first equation represents the SOC dynamic, and the current $I(k)$ is considered positive during discharging and negative during charging. The RC parallels (R_{CT}, C_{CT}) and (R_{Dif}, C_{Dif}) represent the charge transfer and diffusion phenomena inside the battery. We now present the output equation (or the so-called observation or measurement model)

$$V(k) = V_{OCV}(SOC(k)) - V_{CT}(k) - V_{Dif}(k) - R_0 I(k) \tag{4}$$

We define the state vector as

$$x(k) = [SOC(k), V_{CT}(k), V_{Dif}(k)]^T \tag{5}$$

, the model input is $u(k) = I(k)$, and output $y(k) = V(k)$, hence the discrete time non-linear state space model can be represented concisely as follows:

$$x(k+1) = Ax(k) + Bu(k) \tag{6}$$

$$y(k) = g(x(k), u(k)) \tag{7}$$

where,

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{-\frac{\Delta t}{\tau_{CT}}} & 0 \\ 0 & 0 & e^{-\frac{\Delta t}{\tau_{Dif}}} \end{bmatrix} \tag{8}$$

$$B = \begin{bmatrix} -\frac{\Delta t}{Q_{nom}} \\ R_{CT}\left(1 - e^{-\frac{\Delta t}{\tau_{CT}}}\right) \\ R_{Dif}\left(1 - e^{-\frac{\Delta t}{\tau_{Dif}}}\right) \end{bmatrix} \tag{9}$$

The parameter values are as follows: $R_{CT} = 1.6m\Omega$, $\tau_{CT} = R_{CT}C_{CT} = 3.68s$, $R_{Dif} = 7.7m\Omega$, $\tau_{Dif} = R_{Dif}C_{Dif} = 84.34s$

We extract the V_{OCV}, and Input current $I(k)$ using the figures given in the reference and using an online tool called hyperlinkWebplotdigitizerhttps://automeris.io/, and the data is as follows (plotted for visualization): Note that the above current profile is an excerpt, and the actual current profile will have a different time scale, however the nature of the plot and the periodicity will remain the same. Similarly the extracted V_{OCV} data is as follows: The range of SoC shown here is in percentage, which means that a 0 SOC refers to a fully discharged battery, whereas a 1 SOC indicates a fully charged battery.

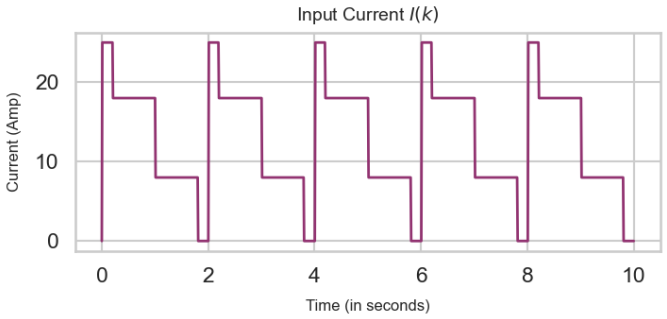

Fig. 2. Input Current Profile

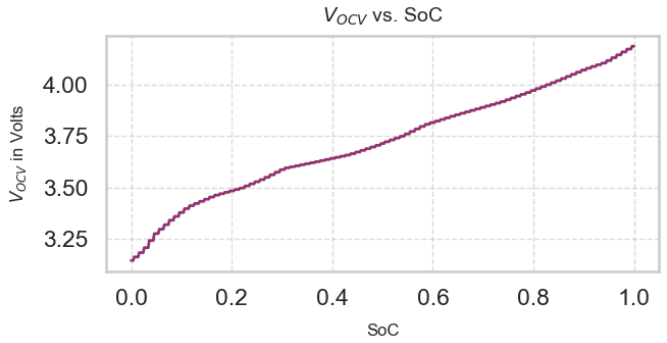

Fig. 3. V_{OCV} vs. SOC characteristic

III. Proposed Methods

We propose using a variety of different Kalman filtering techniques to perform State of charge estimation in our work which are as follows:

A. Extended Kalman Filter

The basic Kalman filter is not directly applicable since the model is highly non-linear. We don't want to linearize the system because of loss of accuracy. We now describe in detail the methodology by how which we implement:

1) The system dynamics are as follows: $k_1 = e^{-\frac{\Delta t}{\tau_{CT}}}$ and $k_2 = e^{-\frac{\Delta t}{\tau_{Dif}}}$

$$\begin{bmatrix} SOC(k+1) \\ V_{CT}(k+1) \\ V_{Dif}(k+1) \end{bmatrix} = \begin{bmatrix} SOC(k) - \frac{\Delta t}{Q_{nom}} I(k) \\ k_1 V_{CT}(k) + R_{CT}(1-k_1) I(k) \\ k_2 V_{Dif}(k) + R_{Dif}(1-k_2) I(k) \end{bmatrix} + q(k) \tag{10}$$

where q is the process model noise and is defined as follows: $q \sim \mathcal{N}(0, Q)$ where Q is defined as follows, and is the process model noise covariance matrix:

$$Q = 1000 \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \tag{11}$$

where R is the measurement model noise covariance value and is given to be 8.432×10^{-4}. Similarly the measurement model is given as follows:

$$V(k) = V_{OCV}(SOC(k)) - V_{CT}(k) - V_{Dif}(k) - R_0 I(k) + r(k) \tag{12}$$

where $r(k)$ is the measurement model noise, and is defined as $r \sim \mathcal{N}(0, R)$

2) The state $x(k) = [SOC(k), V_{CT}(k), V_{Dif}(k)]^T$, and hence we consider the prior to be centered at the initial condition and with a standard deviation of 1. Note that we assume that the states are not co-related, hence we initialize a covariance matrix as an identity matrix.
3) We now compute the Jacobian matrix of the system dynamics model as follows: (let the system dynamics be represented as $f(\bar{x})$, hence we get,

$$\nabla f(\bar{x}) = \begin{bmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \frac{df_1}{dx_3} \\ \frac{df_2}{dx_1} & \frac{df_2}{dx_2} & \frac{df_2}{dx_3} \\ \frac{df_3}{dx_1} & \frac{df_3}{dx_2} & \frac{df_3}{dx_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{-\frac{dt}{\tau_{CT}}} & 0 \\ 0 & 0 & e^{-\frac{dt}{\tau_{Dif}}} \end{bmatrix} \tag{13}$$

4) Compute the observation model Jacobian

$$\nabla h(\bar{x}) = \begin{bmatrix} \frac{dh}{dx_1} & \frac{dh}{dx_2} & \frac{dh}{dx_3} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \tag{14}$$

5) Prediction Update:
   a) Start with the Gaussian prior and propagate the means and covariances:

$$\mu_{k+1|k} = f(\mu_{k|k}) \tag{15}$$

   $\mu_{k|k}$ is the mean state estimate at time k after incorporating measurements.
   b)

$$\Sigma_{k+1|k} = F_k \Sigma_{k|k} F_k^T + Q \tag{16}$$

   where, $\Sigma_{k+1|k}$ is the covariance of the posterior obtained at time step k after incorporating measurement, F_k is the jacobian of the dynamics model.
6) Measurement Update:
   a) Compute Kalman gain:

$$K_k = \Sigma_{k+1|k} H_k^T \left(H_k \Sigma_{k+1|k} H_k^T + R \right)^{-1} \tag{17}$$

   H_k is the measurement model jacobian at time step k
   b) Update state after measurement:

$$\mu_{k+1|k+1} = \mu_{k+1|k} + K_k \left(y_k - h(\mu_{k+1|k}) \right) \tag{18}$$

   c) Update covariance after measurement:

$$\Sigma_{k+1|k+1} = \left(I - K_k H_k \right) \Sigma_{k+1|k} \tag{19}$$

In short the Extended Kalman filter algorithm can be depicted as:

1) Prediction Step:
   a) $\hat{x}^-(k) = Ax(k) + Bu(k) + v(k)$
   b) $P^-(k) = AP(k-1)A^T + Q$
2) Correction Step:
   a) $L(k) = P^-(k)C(k)^T \left(C(k)P^-(k)C(k)^T + R \right)^{-1}$
   b) $\hat{x}(k) = \hat{x}^-(k) + L(k) \left(V_{exp}(k) - g(\hat{x}^-(k), u(k)) \right)$
   c) $P(k) = \left(I - L(k)C(k) \right) P^-(k)$

B. Adaptive Extended Kalman Filter

Adaptive Extended Kalman Filter: An enhanced version of the Extended Kalman filter where the process covariance matrix is adaptively updated. The idea behind the adaptive-EKF is to update the statistical noises covariance parameters Q and R in order to improve the estimation performance. In adaptive algorithms the estimated performance is considered through the information represented by the innovation, which is the difference between the observed data and the predicted measurement as follows:

$$d(k) = V_{exp}(k) - g(\hat{x}^-(k), u(k)) \tag{20}$$

The innovation covariance matrix can be computed as:

$$\hat{D}(k) = \frac{1}{N} \Sigma_{i=i_0}^{N} d(i) d(i)^T \tag{21}$$

where N denotes a moving average window, and can be set by the user. The larger the value of N, it is essentially increasing the retention of the adaptation algorithm - the algorithm considers more of the history of innovations. Matrix $\hat{D}(k)$ represents the actual performance of the estimation process, and the choice of N : it must not be very small to correctly represent the estimation performance. From [1], the adaptive filter can be implemented as follows: (assuming the measurement model noise is constant)

$$\hat{Q}(k) = \frac{1}{N} \Sigma_{i=i_0}^{N} \Delta\hat{x}(i) \Delta\hat{x}(i)^T + P(k) - AP(k-1)A^T \tag{22}$$

where $\Delta\hat{x}$ is the state correction:

$$\Delta x(k) = \hat{x}(k) - \hat{x}^-(k) \tag{23}$$

From Eq [2b] we can write,

$$\Delta\hat{x}(k) = L(k) d(k) \tag{24}$$

$$\hat{Q}(k) = L(k) \hat{D}(k) L(k)^T \tag{25}$$

In conclusion, the AEKF algorithm for the SoC estimation uses the same equations of the EKF filter summarized in Aggregation [III-A] except Eq [1b] as follows:

$$P^-(k+1) = AP(k)A^T + \hat{Q}(k) \tag{26}$$

C. Unscented Kalman Filter

To improve on the results obtained by the extended and the adaptive extended Kalman filter we implement the Unscented Kalman filter which is a general Gaussian filtering technique based on the unscented transform. The UKF process at a high-level is described below:

1) Prediction Step:
   a) Generate σ-points and weights (different for means and covariances) using unscented transform:

$$x = \sigma - \text{function}(x, P) \tag{27}$$

$$w^m, w^c = \text{weight-function}(n, \text{params}) \tag{28}$$

   where x is the current state estimate and P is the motion model noise.
   b) Pass each σ-point x_i through the motion model: $f(x, \Delta t)$ to obtain the new prior (propagated sigma points through motion model):

$$y = f(x, \Delta t) \tag{29}$$

   c) Compute mean and covariance of the new prior:

$$\bar{x} = \Sigma_{i=0} w_i^m y_i \tag{30}$$

$$\bar{x} = \Sigma_{i=0} w_i^m y_i \tag{31}$$

$$\bar{P} = \Sigma w_i^c (y_i - \bar{x})(y_i - \bar{x})^T + Q \tag{32}$$

   where Q is the motion model covariance.
2) Update Step:
   a) Kalman filters perform the update in measurement space, hence propagate the σ-points through the observation model:

$$z = h(y) \tag{33}$$

   b) Compute mean and covariance:

$$\mu_z, P_z = UT(z, w_m, w_c, R) \tag{34}$$

$$\mu_z = \Sigma w_i^m z_i \tag{35}$$

$$P_z = \Sigma w_i^c (z_i - \mu_z)(z_i - \mu_z)^T + R \tag{36}$$

   where R is the observation model covariance.
   c) Compute residual and Kalman gain:

$$y = z - \mu_z \tag{37}$$

   d) Compute cross variance between the state and measurements:

$$P_{xz} = \Sigma w_i^c (y_i - \bar{x})(z_i - \mu_z)^T \tag{38}$$

$$K = P_{xz} P_z^{-1} \tag{39}$$

   e) Compute new state estimate using residual and Kalman gain:

$$x = \bar{x} + Ky \tag{40}$$

$$P = \bar{P} - K P_z K^T \tag{41}$$

The σ-points are generated based on Van Der Merwe's Scaled Sigma point algorithm which is as follows:

1) Let X_i be the i^{th} σ-point. In all, we have $2n + 1$ σ-points, where n is the dimension of the state variable. (In our case, $n = 3$)
2) $X_0 = \mu$
3) $X_i = \mu + \left[\sqrt{(n+\lambda)\Sigma}\right]_i$ for $i = 1, \ldots, n$
4) $X_i = \mu - \left[\sqrt{(n+\lambda)\Sigma}\right]_i$ for $i = (n+1), \ldots, 2n$
5) $\lambda = \alpha^2(n+\kappa) - n$, where α and κ are hyper-parameters.
6) Weights computation (for X_0:

$$w_0^m = \frac{\lambda}{n+\lambda} \tag{42}$$

$$w_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta \tag{43}$$

7) Weights computation (for X_i, $i \neq 0$):

$$w_m^i, w_c^i = \frac{1}{2(n+\lambda)} \text{ for } i = 1, 2, \ldots, 2n \tag{44}$$

8) Usually the values of the hyper-parameters are chosen as follows: $\alpha \in [0, 1]$, $\beta \approx 2$, $\kappa \approx 0$

D. Gauss-Hermite Kalman Filter

The Gauss-Hermite Kalman Filter algorithm only differs from the UKF algorithm in the σ-points generation - and is based on the Gauss-Hermite quadrature integration rule, which is in turn based on the Golub-Welsch Algorithm [3] instead of the unscented transform which UKF uses.

1) The Gauss Hermite quadrature approximates the value of integrals of the following kind

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \tag{45}$$

   and which can be approximated as:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx = \Sigma_{i=1}^{n} w_i f(x_i) \tag{46}$$

2) Hermite Polynomial $H_n(x)$:

$$w_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 \left[H_{n-1}(x_i)\right]^2} \tag{47}$$

   n denotes the number of sample points used, x_i are the roots of the physicist's version of the Hermite polynomial and w_i's are the associated weights. Note that the weights are not different for means and covariances as in the Unscented transform.
3) How does the quadrature work?

$$E[h(y)] = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-u)^2}{2\sigma^2}} h(y) dy \tag{48}$$

   where $x = \frac{y-\mu}{\sqrt{2}\sigma} \longleftrightarrow y = \sqrt{2}\sigma x + \mu$
4) Coupled with integration by substitution:

$$E[h(y)] = \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi}} e^{-x^2} h(\sqrt{2}\sigma x + u) dx \tag{49}$$

   leading to:

$$E[h(y)] \approx \frac{1}{\sqrt{\pi}} \Sigma_{i=1}^{n} w_i h\left(\sqrt{2}\sigma x_i + \mu\right) \tag{50}$$

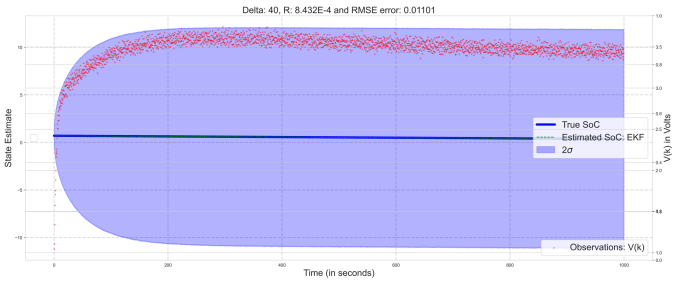

Fig. 4. EKF: SoC Estimation with $\pm 2\sigma$ bounds

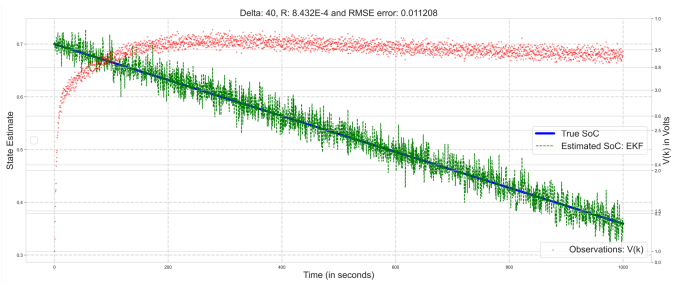

Fig. 5. EKF: SOC Estimation

IV. Results

We first define various parameters we have used:

1) Simulation Parameters:
   a) dt: time step: 0.01 seconds
   b) N_{sims}: simulation steps: 10^5
   c) n_{states}: number of states: 3
   d) t_{final}: final time: $N_{sims}dt$
   e) x_0: initial state: [0.7, 2, 1]. This means that the initial SoC = 0.7, i.e. the battery is initially charged at 70%.
   f) R: observation model covariance: $8.432 * 10^{-4}$ [2]
   g) Q: $\begin{bmatrix} 1000R & 0 & 0 \\ 0 & 1000R & 0 \\ 0 & 0 & 1000R \end{bmatrix}$: process model covariance matrix
   h) δ: observation frequencies: 10, 20, 40, 80 (which means 1 observation after every z iterations). Hence we can obtain an observation after $\delta * dt$ seconds.
   i) Adaptive Window = 10 iterations (only applicable for Adaptive Extended Kalman filter)
2) Battery Parameters:
   a) R_{CT}: equivalent resistance charge transfer: 1.6E-3 Ω
   b) τ_{CT}: equivalent time constant charge transfer: 3.68 seconds
   c) R_{Dif}: equivalent resistance diffusion: 7.7E-3 Ω
   d) τ_{Dif}: equivalent time constant diffusion: 84.34 seconds
   e) Q_{nom}: nominal battery capacity: 10.4 Ah
   f) R_0: internal battery resistance: 6E-3 Ω

Note that we only present plots for a particular case of $\delta = 40$ fior conciseness of the report and plots for other cases are submitted in separate files.

A. Extended Kalman Filter

Here are the results for SoC estimation with an EKF: note that we only plot and compute the tracking RMSE error for the SoC, and not the other states for clearer analysis and plots. We provide the plots with V_{CT} and V_{Dif} tracking errors and standard deviations in separate files for completeness. We can see that the standard deviation of the results is quite large to see the tracking as well as true SOC dynamics. The scattered points are the observations made on the output voltage $V(k)$ which is computed according to the observation model Eq. [12]. Hence we plot the EKF results without the standard deviation here:

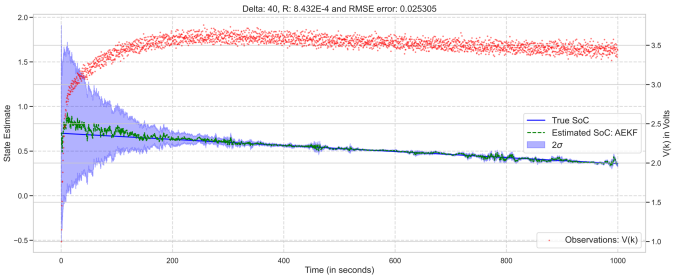

Fig. 6. Adaptive EKF: SOC Estimation with $\pm 2\sigma$ bounds

B. Adaptive Extended Kalman Filter

The AEKF method improves on the EKF by adaptively updating the process model noise covariance matrix based on the innovation history, and we see significant improvements in tracking errors and convergence over EKF. We use an adaptation algorithm with the hyper-parameter 'innovation_window' set to 10. Later, we also do a sensitivity analysis to examine the performance with different adaptation window values. The results are as follows: It can be clearly see that AEKF improves upon the EKF for tracking error as well as uncertainty convergence, as it can be seen that the $\pm 2\sigma$ band tapers off narrowly as the time passes.

C. Unscented Kalman Filter Results

The Unscented Kalman filter is a Gaussian filtering technique which is based on the numerical approximation of the Gaussian integral using the unscented transform and usually improves on the EKF in terms of convergence as well as tracking error. Here too the standard deviation is very large hence, we show both the plots with and without the $\pm 2\sigma$ bands around the estimated states.

D. Gauss-Hermite Kalman Filter Results

The Gauss-Hermite filter too is a Gaussian filtering technique, and we implement filters of 2 orders: 3 and 5. However we only present plots for order 5, reserving the rest for separate files.

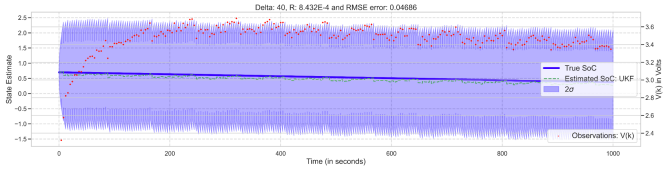

Fig. 7. UKF: SOC Estimation with $\pm 2\sigma$ bounds

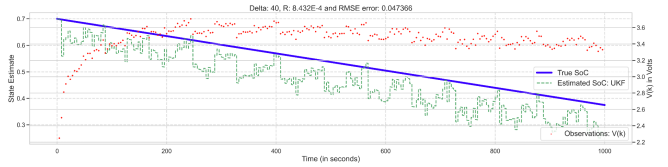

Fig. 8. UKF: SOC Estimation

E. Comparisons

We present a concise summary of the results we have obtained so far - the table below enlists RMSE tracking errors for true SOC and SOC estimates for the various different filters we have implemented including EKF, AEKF, UKF and GHKF. The above table data is also visualized in the figure below

From the RMSE values in the table, we can conclude that the EKF and AEKF outperform the general Gaussian filters UKF and GHKF by a significant margin. The EKF performance is not sensitive towards the measurement frequency, whereas all other filters are, with the Adaptive EKF being the most sensitive.

The main aim of the project was to explore and compare the EKF and the adaptive versions of the EKF in detail. In this section we explore and compare some more detailed aspects of EKF and the AEKF filters. For a range of values of δ and measurement model covariances R, we compute the errors :

$$SOC_{\text{true}} - SOC_{\text{estimated}}(R, \delta) = \text{Error}(R, \delta) \qquad (51)$$

And then we compute time-averaged errors across all the R and δ values, which can be visualized as follows: When observed closely, we can conclude that the Adaptive EKF performs better (in terms of average error over the whole

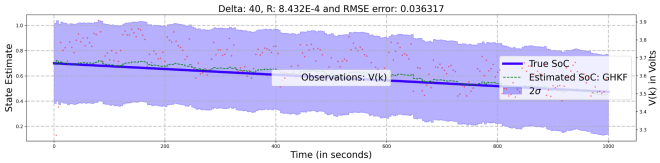

Fig. 9. GHKF Order 5: SOC Estimation with $\pm 2\sigma$

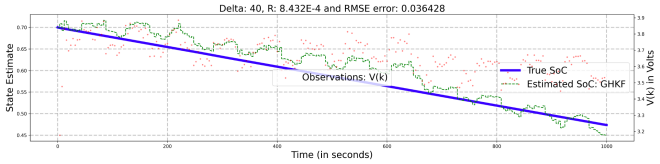

Fig. 10. GHKF Order 5: SOC Estimation

	RMSE (R = 8.432E-4)				
δ	EKF	AEKF	UKF	GHKF3	GHKF5
10	0.011076	0.010003	0.030431	0.029078	0.03627
20	0.011085	0.015705	0.045942	0.028856	0.03369
40	0.011002	0.025305	0.047366	0.029565	0.036428
80	0.011834	0.023846	0.047593	0.028664	0.030856

TABLE I
PERFORMANCE COMPARISON

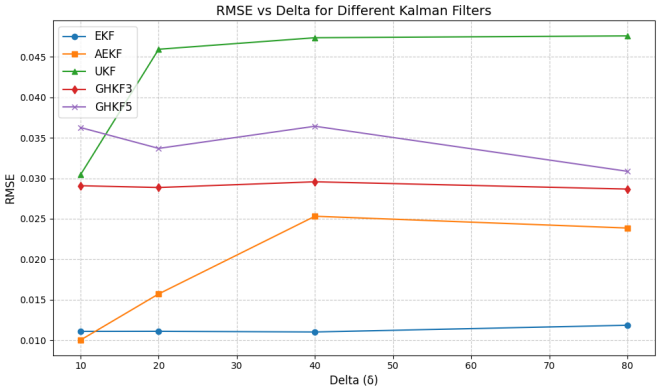

Fig. 11. RMSE errors for various filters

estimation time horizon) for lower values of δ (higher observation frequencies) as well as R, whereas the EKF outperforms the adaptive algorithm in terms of higher values of δ and R. However average error doesn't completely quantify the performance of the algorithms. Another interesting observation is that the EKF performance in terms of average error does not depend much on the observation frequency whereas only depends on the measurement model noise covariance. In real-world phenomena, measurement noise is generally fixed, due to external disturbances, however the measurement frequency is somewhat accessible and controllable to the user - and that they can have a higher observation frequency for the sensor incurring a higher cost. The Adaptive EKF filter on the other hand, shows sensitivity towards the measurement frequency which can be exploited by tuning parameters to obtain optimal performance. One such parameter is the innovation window, referred in Def. [III-B], which we tune to examine effect on AEKF performance: As it can be seen in the figure the AEKF average error (across all values of R and δ) is highly sensitive to the value of the adaptation window. However it doesn't follow a specific trend, as we can see several local maximas in average error at values 10, 70 and 80. In general we can

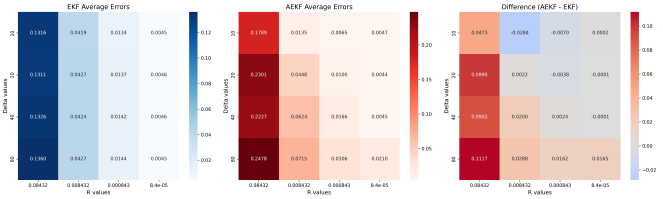

Fig. 12. Errors $SOC_{\text{true}} - SOC_{\text{estimated}}(R, \delta)$

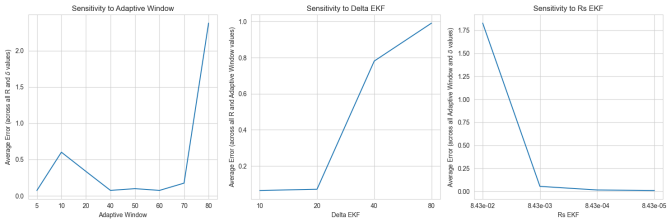

Fig. 13. AEKF Sensitivity Analysis

say that that the adaptation window can be around 0.05% of the total simulation steps, for our case: the simulation steps were 100,000: hence an adaptation window $\in (10, 50)$ should work optimally. However one needs to really try out different values of adaptation and figure out which suits the particular system of interest the best.

V. Unsuccessful Attempts

We also tried implementing particle filters as well as Cubature Quadrature filters. Particle filters are population-based methods, where a set of particles (samples) are used to represent the posterior distribution of a stochastic process given the noisy and/or partial observations. The state-space model can be nonlinear and the initial state and noise distributions can take any form required . Cubature quadrature filters on the other hand are another form of approximating Gaussian integrals, and are structured as follows:

1) Prediction Steps:
   a) The following equations are used to generate the sigma points:

$$\chi_{k-1}^{(i)} = m_{k-1} + \sqrt{P_{k-1}}\zeta^{(i)}$$
$$\zeta^{(i)} = \sqrt{n}e_i \quad i = 1, \ldots, n$$
$$\zeta^{(i)} = -\sqrt{n}e_{i-n} \quad i = n+1, \ldots, 2n$$

   b) Propagate sigma points through dynamic model
   c) Compute predicted mean and predicted covariance
2) Update Step:
   a) Form sigma points:

$$\chi_k^{-(i)} = m_k^- + \sqrt{P_k^-}\zeta^{(i)}, \quad i = 1, \ldots, 2n$$

   b) Pass sigma points through the measurement model:

$$y_k^{(i)} = h(\chi_k^{-(i)}), \quad i = 1, 2, \ldots, 2n$$

   c) Compute predicted mean and variance
   d) Compute filter gain K_k and the filtered state mean m_k and covariance P_k, conditional on the measurement y_k

Note that the codes and the implementation are provided in the .zip files, but the results were as follows, which were not satifactory with respect to tracking accuracy. The results we obtained were as follows: (for $n_{\text{particles}} = 500000$, and 100 time steps, with a time interval of $dt = 0.1$ We suspect that the results aren't good in tracking accuracy due to lesser

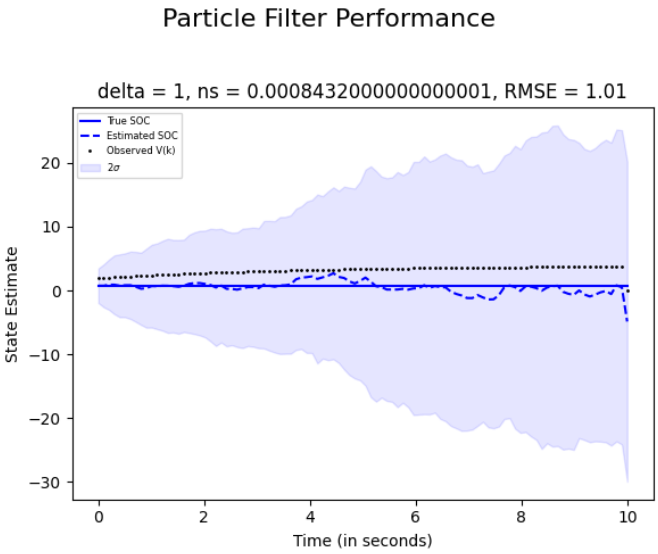

Fig. 14. Particle Filter Results

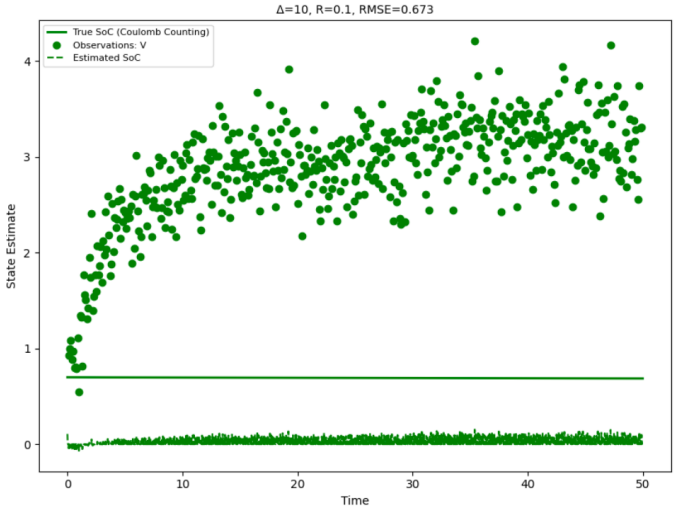

Fig. 15. CQF: SOC Estimation

number of particles, the tracking accuracy is highly affected by the population size. For the above mentioned simulation parameters, it took us 4 minutes to run the code and produce the output. For other filters, we are running 100,000 time steps with a time interval resolution of 0.01 seconds. For such computations, at $n_{\text{particles}} = 500000$, the computation times are intractable. Hence we stopped development of the particle filter results. The preliminary code as well as the results are provided in the zipped file.

Here are the results for the implementation of the Cubature Quadrature filter: On the other hand, we think that the following reasons are why the Cubature Quadrature Filter (CQF) performs poorly in SOC tracking accuracy.

1) Choice of Quadrature Rule and Weights: The CQF uses cubature rules, which weigh all sigma points equally,

and the scheme is designed to integrate symmetric functions in higher dimensions. GHKF and UKF on the other hand, assign tailored weights to sigma points (or quadrature points). Because of this, in scenarios involving highly nonlinear dynamics or observation models, the uniform weighting may fail to capture the influence of more critical regions in the state space accurately. GHKF and UKF adapt weights to better approximate the posterior distribution, which can significantly enhance tracking performance.

2) Non-linear state and measurement equations are central to the battery dynamics system, and CQF may not perform well due to the non-linearities due to its uniform weighing scheme, unlike GHKF or UKF which using higher-order Hermite polynomials or the unscented transform can handle such non-linearities robustly.
3) Again, CQF distributes quadrature points uniformly in a spherical manner for integrating symmetric Gaussian distributions. However the symmetry breaks down in our use-case, as the posterior might be skewed or shaped by constraints such as battery dynamics. Also, by assigning equal weights, CQF cannot differentiate between the contributions of points near high-probability regions versus outliers.

VI. Conclusion

In conclusion, this project focused on the estimation of the State of Charge (SOC) utilizing advanced Kalman filtering techniques. We implemented several filtering methods, including the Extended Kalman Filter (EKF), Adaptive Extended Kalman Filter (AEKF), Unscented Kalman Filter (UKF), and Gauss-Hermite Kalman Filter (GHKF), to enhance the accuracy of SOC predictions based on a second-order equivalent circuit battery model. SOC estimation is a very critical function for the battery management systems, and accurate estimates about SOC help improve electric vehicle range and efficiency. The results demonstrated that both EKF and AEKF significantly outperformed the UKF and GHKF in terms of tracking accuracy and computational efficiency. Notably, the AEKF showed improved adaptability by dynamically updating noise covariance parameters, which contributed to better performance under varying conditions. However, attempts to implement Particle Filters and Cubature Quadrature Filters were less successful, highlighting the challenges associated with tracking accuracy in non-linear systems. Overall, this study underscores the importance of model-based approaches in SOC estimation and provides valuable insights for future developments in BMS technology.

References

[1] A.H. Mohamed and K.P. Schwarz. Adaptive Kalman filtering for INS/GPS. Journal of geodesy, 73(4):193–203, 1999

[2] C. Taborelli and S. Onori, "State of charge estimation using extended Kalman filters for battery management system," 2014 IEEE International Electric Vehicle Conference (IEVC), Florence, Italy, 2014, pp. 1-8, doi: 10.1109/IEVC.2014.7056126.

[3] https://gubner.ece.wisc.edu/gaussquad.pdf

[4] Simo Sarkka (2013). Bayesian Filtering and Smoothing. Cambridge University Press.