**System Identification of Nonlinear State-Space Battery Model**

**December 2012 Progress Report**

Author:

Wei He

[weihe@calce.umd.edu](mailto:weihe@calce.umd.edu)

Center for Advanced Life Cycle Engineering

Department of Mechanical Engineering

University of Maryland, College Park

Advisor:

Dr. Chaochao Chen

chaochao@umd.edu

Center for Advanced Life Cycle Engineering

Department of Mechanical Engineering

University of Maryland, College Park

**Abstract**

The goal of this project is to solve the parameter estimation problem of the nonlinear state-space model for the battery state of charge estimation. An Expectation Maximization (EM) algorithm is employed to solve this problem. The Expectation (E) step involves solving a nonlinear state problem, which will be solved using the particle filter and smoother algorithm in this project. In this semester, the particle filter and smoother were implemented and validated in Matlab.

**1. Introduction**

Electric vehicles (EVs) powered by lithium-ion batteries is going to penetrate the automobile market within the next few years, due to the increasing concerns on global warming and fossil fuel depletion. However, challenges still exist for EVs that remain to be solved. The most notable one is the state of charge (SOC) estimation, which can be used for remaining range prediction of EVs and optimal battery control. SOC by definition is the remaining charge in the battery expressed as the percentage of its maximum capacity. When the battery is full, the SOC is 100%; when it is empty, the SOC is 0 %. To estimate the SOC, a lot of equivalent circuit models (ECMs) have been developed to model the dynamics of the battery system [1-3]. Fig.1 shows a commonly used ECM [1]. In this figure Vt is the terminal voltage of the battery that can be measured by a voltage sensor. OCV is the open circuit voltage of a battery, which is a monotonic nonlinear function of SOC. The nonlinear relationship between OCV and SOC can be established by battery tests. The *R*p and *C*p are the double layer resistance and capacitance respectively, and *R*0 is the series resistance of the battery. It should be noted that the ECM in Fig.1 is just an approximation of the real system. If more capacitors and resistors are added into the model, the modeling accuracy may be increased. Eq. (1) shows the equations of the ECM in Fig. 1 in a continuous form.

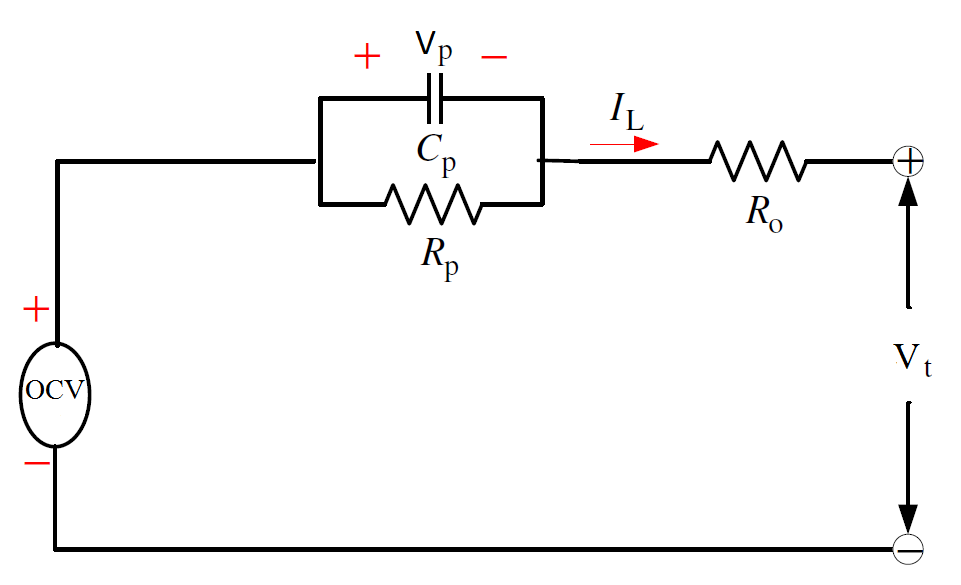


Figure 1 An equivalent circuit model of batteries

 (1)

We can reformulate the Eq.(1) to a discrete state-space representation as follows:

 (2)

 (3)

where *k* is the sampling time and  is the sampling interval.

To model the dynamic evolution of the hidden state *SOC*k, the Coulomb counting principle can be used, which is defined by:

 (4)

where *Q*max is the maximum capacity of a battery, and IL is the current flowing through the battery.

The combination of Eq.(2), Eq.(3) and Eq.(4) forms a state-space model for the battery SOC estimation. Since SOC and Vp are not direct observable, they are set as the state variables. As a result, Eq.(2) and Eq.(4) are the process functions in the model. Vt can be directly measured by a sensor, so Eq.(3) is the measurement function. The input of the model is *IL,*k and output is *VL,*k. We assume that the states: SOC and Vp, and the output Vt are corrupted with independent zero mean Gaussian noise. The state-space representation of the battery system is summarized as follows:

 (5)

Therefore, the model parameters of this model are . The model parameters will change with the loading conditions of batteries. For example, *R*0 may decrease with the increase of temperature. Thus, we need to update these parameters to reflect the true system response, in order to get accurate SOC estimations.

**2. Approach**

**2.1 Expectation Maximization**

This project considers estimating the unknown parametersin state-space models

 (6)

based on the information in the measured input-output responses

 (7)

using a maximum likelihood (ML) framework

 (8)

Eq. (8) is equivalent to maximize a log-likelihood function of Y1*:N*

 (9)

where  is:

 (10)

Given a set of observed data, a set of unobserved latent data or missing values, and a vector of unknown parameters, the EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying the following two steps [4]:

1. Expectation step (E step): calculate the expected value of the log likelihood function, with respect to the conditional distribution of given under the current estimate of the parameters :

 (11)

 (12)

1. Maximization step (M step): find the parameter that maximizes this quantity:

 (13)

If not converged, update *ii+*1 and return to step 2

It has been proved in Ref.[4] that , which implies that the increase of can insure the increase of the log likelihood of 

When the model (6) is linear and the process noise and measure noise and are Gaussian, then Eq. (11) can be simply computed by a standard Kalman filter. However, in nonlinear and/or non-Gaussian case, other approaches should be employed. In this study the particle filter and smoother will be used to compute Eq. (11). Apply the conditional expectation operator to both side of Eq. (12), we have [4]:

**** (14)

where

 (15)

 (16)

 (17)

 in Eq. (15) and in Eq. (17) are smoothing problems and can be solved using a particle smoother [4-6]. In Eq. (16), can be rewritten as

 (18)

Therefore, the particle filter and smoother representations can be used deliver an importance sampling approximation to *I2* .

If we substitute the particle smoother representation: and particle filter representation: into Eq. (15) , Eq. (16) and Eq. (17), then we have:

 (19)

Eq. (19) provides a solution to calculate for any nonlinear state space model. The EM method with particle approximation is called particle EM in the literature. Below provides a summary of the particle EM algorithm [4].

1. Set *i* = 0 and initialize 
2. Expectation (E) Step:
   1. Run particle filter and particle smoother
   2. Calculate 
3. Maximization (M) Step:

Compute: 

1. Check the non-termination condition . If satisfied update and return to step 2, otherwise terminate.

**2.2 The algorithm of particle filter**

This section will introduce the principles of particle filters, which follows a Bayesian filtering framework. It includes two steps: prediction and update. Prediction is to propagate the distribution to the next time point based on the process model to get a prior for the updating step:

 (20)

In the updating step, the posterior distribution is updated with the measurement using Bayes’ rule:

 (21)

where 

Eq. (20) and (21) forms a recursive Baysian solution for the filtering problem. But it is just conceptual in general, because the propagation of the distributions is hard to solve analytically in most case. However it is possible to find an approximated numerical solution by using Monte Carlo sampling. The idea is to represent the distribution by a set of random samples with the associated weights:

 (22)

where , *i =* 1,2,3,…,*M* is a set of independent random samples draw from a proposal distribution , and is the Bayesian importance weights associated with each sample . The weight can be obtained by:

 (23)

Why sampling from instead of is because the target distribution is usually unknown. If we choose our proposal distribution to be, then the weights become to

 (24)

Since the denominator is independent of , we have:

 (25)

The main steps of the particle filter are summarized as follows [4]:

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. Initialize particles, and set k = 1.
2. Predict the particles by drawing *M* i.i.d samples according to  
    
3. Compute the importance weights 



1. For each *j = 1,…,M* draw a new particle  with replacement (resample) according to 
2. If *k* < *N* increment *k k+*1 and return to step 2, otherwise terminate.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

In step 4 of the algorithm, there is a resample step. This resample step is to solve the degeneracy problem of particle filters. Degeneracy problem means that after several iterations, the weights of the most particles are close to zero, which implies that a large computational effort is devoted to updating particles whose contribution to  is almost zero. The basic idea of resample is to eliminate the particles with small weights and concentrate on the particles with high weights. In this study, we adopted the system resampling algorithm presented in Ref. [5], and the steps of the algorithm are shown below:

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Initialize the CDF: *c*1 = 0

For *i* = 2:*M*

Construct CDF: 

End For

Start at the bottom of the CDF: *i* = 1

Draw a starting point: 

For j=1:M

Move along the CDF: 

While 

*i* = *i*+1

End while

Record indices: ind*j* = *i*

End For

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

In this algorithm, the indices of the particles after resampling are recorded, and these particles kept after resampling are used as the parent particles for the state prediction in the next time points.

**2.3 Particle smoother**

Similar to particle filter, the particle smoother is to approximate the distribution of the posterior distribution of *xk* given the entire measurements Y1:*N* by a set of particles , *i = 1,2,3,…,M* with associated weights :

 (26)

To compute the , we note the following fact based on the law of total probability and Bayes’ rule:

 (27)

It is easy to find that when *k = N*, the smoothing density and the filtering density are the same, and hence the weights in Eq. (26) and the particles are identical. As a result, we can work backwards to compute Eq. (27) by assuming the particle smoothing approximation is available at time *k*+1 and use it to compute Eq.(27) as:

 (28)

Therefore, based on Eq. (27) and Eq. (28) , we have



The steps of the particle smoother are listed as follows [4]:

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. Run the particle filter and store the predicted particles  and their weights , for *k = 1*,…,*N*.
2. Initialize the smoothed weights to be the terminal filtered weights at time *k* = *N*: and set *k* = *N-1*.
3. Compute the smoothed weights using the filtered weights  and particles  via:  
   
4. Update *k k-*1. If *k* > 0 return to step 3, otherwise terminate.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**3. Validation**

**3.1 Simulated signal generation**

In this project, the particle filter and smoother will be validated using simulated data. Here, we assume that the model parameters in Eq. (5) are known, and simulated data were generated based on Eq. (5). The parameter settings are shown as follows:



Fig. 2 shows the voltage and current signal and Fig. 3 shows the evolution of the states.



Fig.2 Simulated input current and output voltage



Fig. 3 Simulated SOC and Vp

**3.2 Particle Filtering and Smoothing Results**

The particle filter and particle smoother were implemented to estimate the states and compared with the real states to evaluate the algorithm performance and correctness of the implementation. Fig.4 and Fig. 5 shows the particle filtering and smoothing results respectively. The red line is the real state and the blue line is the estimation. It can be seen from the figures that the particle filter and smoother can provide accurate state estimations. The root mean square (RMS) error of particle filter is 0.0063 and that of particle smoother is 0.0050. For comparison, the Kalman filter was also implemented. Fig. 6 shows the Kalman filtering result. Since we assume the OCV(SOC) here is a linear function, the Kalman filter provides a optimal solution here, and RMS error is only 0.0045, which outperforms particle filter. Based on the principle of particle filter, the accuracy of the particle filter will be increased if we use more particles to approximate the distribution. Fig. 7 shows the RMS error as a function of the particle number. It can be seen that as the particle number increases, the RMS error is approaching 0.0045. In the next step, the OCV(SOC) will be changed to a nonlinear function, then it can be expected that the particle filter will outperform the Kalman filter.



Fig. 4 Particle filter-based SOC estimation



Fig.5 Particle smoothing-based SOC estimation



Fig. 6 Kalman filter-based SOC estimation

Fig. 7 The RMS estimation error of the SOC as a function of particle filter size

**4 Future works and Deliverables**

The future works are to implement the full expectation maximization algorithm based on the particle smoother and validate the full algorithm. The timeline is listed as follows:

* for the full algorithm: February 1
* Validation: March 15
* Testing: April 15
* Final Report: May 1

The algorithm is being written in Matlab 2010 using a Dell laptop. Once the project finished, Deliverables include the codes of the particle filter, particle smoother and particle EM. The datasets of the simulated battery discharge process, and the end-of-the year progress reports.

**5 References**

1. H. He, R. Xiong, and H. Guo, Online estimation of model parameters and state-of-charge of LiFePO4 batteries in electric vehicles. Applied Energy, 2012. 89(1): p. 413-420.
2. C. Hu, B.D. Youn, and J. Chung, A Multiscale Framework with Extended Kalman Filter for Lithium-Ion Battery SOC and Capacity Estimation. Applied Energy, 2012. 92: p. 694-704.
3. H.W. He, R. Xiong, and J.X. Fan, Evaluation of Lithium-Ion Battery Equivalent Circuit Models for State of Charge Estimation by an Experimental Approach. Energies, 2011. 4(4): p. 582-598.
4. T.B. Schön, A. Wills, and B. Ninness, System identification of nonlinear state-space models. Automatica, 2011. 47(1): p. 39-49.
5. M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. Signal Processing, IEEE Transactions on, 2002. 50(2): p. 174-188.
6. A. Doucet and A.M. Johansen, A tutorial on particle filtering and smoothing: fifteen years later. Handbook of Nonlinear Filtering, 2009: p. 656-704.