EVS24
Stavanger, Norway, May 13 - 16, 2009

Efficient Battery Pack State Estimation using Bar-Delta Filtering

Gregory L. Plett
University of Colorado at Colorado Springs and Consultant to Compact Power Inc.,
1420 Austin Bluffs Parkway, Colorado Springs, CO 80918, USA, glp@eas.uccs.edu

Abstract

Accurate state-of-charge (SOC) estimation is critical for xEV applications but involves considerable computational complexity. We argue that battery-pack SOC is undefined, and that individual estimates of all cell SOCs are required to compute available power and energy. Furthermore, it is not feasible to simply replicate a cell-based method for all cells in a pack. Instead, we propose a new method for pack state estimation that we call "bar-delta filtering" that takes advantage of similar states among pack cells. It estimates all battery-pack cell state-of-charge and state-of-health values using only slightly more computation than for a single cell.

Keywords: state of charge, battery SoH (State of Health), battery management, BMS, control system.

1 Introduction

In this paper, we consider state estimation for battery packs for vehicular application, including hybrid electric vehicles (charge conserving and plug-in), battery electric vehicles, and so forth. We refer to this whole family herein as "xEV". For hybrid EVs, it is critical to be able to determine available power at any point in time; for pure EVs, it is critical to be able to determine total available energy. These, in turn, require knowing (at minimum) cell state-of-charge (SOC), cell resistances, and cell total capacities. There are a variety of methods in the literature for making estimates of these more basic quantities [1–4]. We prefer those that are based on Kalman filtering techniques for their strong theoretical basis in optimal estimation theory and for their track record in a wide variety of applications. In particular, we have found sigma-point Kalman filters (SPKF) to be the most accurate method of estimating cell SOC, cell resistance, and cell total capacity that we have encountered [5, 6]. Unfortunately, they are also among the most computationally complex.

Furthermore, we contend that it is necessary to estimate all individual cell SOC, resistance, and capacity values when determining battery pack power and energy. We have heard some propose estimating a "pack SOC" value instead, but we argue that "pack SOC" does not make sense. In an illustrative example, consider a pack comprising two cells wired in series, one of which has SOC equal to 0% and the other having SOC equal to 100%. What then is the pack SOC? Is it 0%? Is it 50%? Is it 100%? It cannot be 0% because that would indicate that the pack can be charged, but the pack cannot be charged without overcharging the cell that is at 100%. It cannot be 100% because that would indicate that the pack can be discharged, but the pack cannot be discharged without over-discharging the cell that is at 0%. Similarly, it cannot be 50%. While this is an extreme example, it serves to illustrate that "pack SOC" is not a concept that makes sense in any useful way. In order to compute available energy and power of a battery pack, we again emphasize that knowing individual cell SOCs, capacities, and resistances is necessary.

But, how to compute those values? One possibility is to select a method that works well for computing cell SOC, resistance, and capacity, and to replicate that method N times to estimate the desired values for a pack comprising N cells wired in series. Clearly, this will work, but it also requires a prohibitive level of computation. In this paper, we explore a method that takes advantage of the fundamental similarity between all series-connected cells in the pack in order to furnish all

cell SOC, resistance and total capacity estimates in a manner requiring only somewhat more computational effort than for a single cell.

2 Bar-delta filtering

While we have argued that "pack SOC" does not make sense, the concept of "pack-average SOC" can be useful as a step toward finding individual cell SOCs. Since all cells in a series string experience the same current, we expect their SOC values to (1) move in the same direction for any given applied current, by (2) a similar amount (with the differences determined by unequal cell capacities). We can take advantage of this similarity by creating one algorithm to determine the composite average behavior of all cells in the battery pack, and another algorithm to determine the individual differences between specific cells and that composite average behavior.

We define pack-average SOC as the average of all cell SOC values. It is approximately equal to (but not identical to) the value of SOC that would be found via an OCV method if the pack voltage were divided by N to make a scaled pack voltage. We find it adequate, however, to estimate pack-average SOC using any available cell SOC estimation method with inputs: pack current (same for all cells), scaled pack voltage, and average pack temperature. The pack-average SOC is never less than the least SOC value, so is greater than 0%. It is never greater than the greatest SOC value, so is less than 100%. Therefore, its range is within the standard SOC range.

Pack-average and individual cell dynamics depend on more than SOC. Generally, we are interested in estimating other quantities as well, as discussed below. Therefore, we consider estimating pack-average and individual cell "state vectors" at every sampling instant within the battery management system (BMS). For example, we may sample all cell voltages, pack current, and module temperatures once per second. Time within the BMS is then indexed by k, in seconds, and we can denote the state vector of interest for any given cell i at time k as $x_k^{(i)}$. Furthermore, using the idea of pack-average state vector, which extends the idea of "pack-average SOC" to all states of interest, we can write an individual cell's state vector as $x_k^{(i)} = \bar{x}_k + \Delta x_k^{(i)}$ where $\bar{x}_k$ is the pack-average state vector and $\Delta x_k^{(i)}$ is the difference between the state vector of cell i and the pack average state vector. We call $\bar{x}_k$ "x-bar" and we call $\Delta x_k^{(i)}$ "delta-x." So, if we can estimate x-bar for the pack and delta-x for every cell, then we can compute an estimate of the state vector for every cell. We call the method that we have developed "bar-delta filtering," as inspired by the "x-bar" and "delta-x" naming convention.

The first step in bar-delta filtering is to replace the N individual state vector estimators—which each jointly estimate the state and parameters of a single cell—with a single state vector estimator-which estimates the pack-average state and parameters. This is the x-bar filter, or simply, the bar-filter. The second step is to add N estimators that estimate the delta-x portion of every individual cell's state. These are called the x-delta filters, or simply, the delta-filters.

At first, it appears that we have taken a problem of complexity N (because we started with N individual cell state vector estimators) and replaced it with a problem of complexity $N + 1$ (because we ended with one bar filter and N delta filters). However, on closer examination this is not the case—the three different types of estimator involved are not of identical computational complexity. The bar filter *is* of the same computational complexity as the individual state estimators that it uses as a basis. However, the delta filters can be made very simple. Furthermore, the delta filters do not need to be executed at the same rate as the bar filter. While SOC (for example) in individual cells can change very quickly, the difference in SOC between cells changes at a much slower rate. Therefore, the delta filters do not need to be updated as quickly as either the N individual state vector estimators that we started with, or the bar filter. The delta filters may operate at much slower rates, down to $1/N$ times the rate of the bar filter. Therefore, overall, the bar-delta filter method has a computational complexity that can be as low as somewhat more than a single cell state estimator for a multi-cell pack.

3 The ESC cell model

While the bar-delta method may be used with a variety of cell state estimation means, we will proceed in this paper by illustrating how to use it with a sigma-point Kalman filter (SPKF) approach for the bar filter. This method requires a mathematical model of cell dynamics, and we will use the "enhanced self-correcting" (ESC) cell model as a basis for doing so. This model has been well defined elsewhere [5,7], so we will only summarize the necessary features here.

The ESC cell model includes states that describe the dynamics of SOC movement, polarization voltages, and a hysteresis level. The model output blends these characteristics together by computing an open-circuit-voltage (OCV) from the SOC level, mixes in the polarization voltages and hysteresis level, and includes ohmic $i \times R$ losses. The equation that models the dynamics of SOC movement for a given cell i can be written as:

$$z_k^{(i)} = z_{k-1}^{(i)} - i_{k-1}\frac{\Delta t}{C^{(i)}} = z_{k-1}^{(i)} - i_{k-1}\Delta t C_{\text{inv}}^{(i)} \quad (1)$$

where $z_k^{(i)}$ denotes the SOC of cell i at time index k, i_k denotes the pack current at time index k, Δt denotes the sampling period (e.g., one second), and $C_{\text{inv}}^{(i)}$ denotes the inverse of cell capacity for cell i: $C_{\text{inv}}^{(i)} = 1/C^{(i)}$. Note that the capacity may also be modeled as time-varying, denoted as $C_{\text{inv},k}^{(i)}$.

The equation that models the polarization voltage dynamics for a given cell i can be written as:

$$f_k^{(i)} = A_f f_{k-1}^{(i)} + B_f i_{k-1},$$

where $f_k^{(i)}$ denotes a vector of polarization voltages for cell i at time index k, A_f is a matrix comprising the dynamic time constants of the polarization voltages of the cell, and B_f is a constant input matrix.
The equation that models hysteresis voltage dynamics for a given cell i can be written as:

$$h_k^{(i)} = \exp\left(-\left|i_{k-1}\gamma\Delta t C_{\text{inv},k-1}^{(i)}\right|\right) h_{k-1}^{(i)} + \left(1 - \exp\left(-\left|i_{k-1}\gamma\Delta t C_{\text{inv},k-1}^{(i)}\right|\right)\right) \text{sgn}(i_{k-1}),$$

where $h_k^{(i)}$ denotes the hysteresis level for cell i at time index k, γ is a hysteresis rate time constant, and the "sgn" function returns -1 for a negative argument, 0 for a zero argument, and $+1$ for a positive argument.
The cell-model state includes these four dynamic quantities, which are combined in a single vector as:

$$x_k^{(i)} = \begin{bmatrix} z_k^{(i)} \\ f_k^{(i)} \\ h_k^{(i)} \end{bmatrix}.$$

Using the quantities from this cell-model state vector, the cell terminal voltage is expressed as

$$y_k^{(i)} = \text{OCV}(z_k^{(i)}) - i_k R^{(i)} + C f_k^{(i)} + M h_k^{(i)},$$

where the "OCV" function returns the open-circuit-voltage for a given SOC, $R^{(i)}$ is a cell resistance, C is a vector of polarization voltage blending constants, and M is a hysteresis maximum voltage limit.
Temperature dependence may be included in any of the parameters of the model. Specific temperature-dependent notation has been omitted from this paper for purposes of clarity.

4 ESC-SPKF bar-delta filtering

A Kalman filter is an algorithm for estimating the internal state of some dynamic system given only measurements of the system's input and output and given a model of system dynamics. For the purpose of estimating the state of an electrochemical cell, the ESC model may be used with a Kalman filter, where the input to the model is cell current, and the output of the model is cell terminal voltage under load. Every time step, the Kalman filter predicts what it expects to see as the cell terminal voltage given its present state estimate and measured cell input, then compares its estimate of cell terminal voltage to the measured cell terminal voltage, and updates its state estimate accordingly, in the direction of reducing the estimation error of the cell terminal voltage. The original Kalman filter was developed to estimate states of linear systems, but many systems (like electrochemical cells) are in fact nonlinear, so extensions to the original Kalman filter were developed to estimate states of nonlinear systems. We have used extended Kalman filters (EKFs) and sigma-point Kalman filters (SPKFs) for this purpose [5, 6, 8–10]. In our experience, SPKF gives the best cell state estimation results we have seen, although they are also fairly complex mathematically. The implementation that we describe here uses one SPKF per pack to compute pack-average estimates, and N simpler filters for the cells to compute differences between pack-average and cell estimates.

4.1 The pack bar filter

In the implementation that we describe here, the pack-average SPKF estimated the following quantities:

- The pack-average state-of-charge;
- Two pack-average polarization voltages;
- The pack-average hysteresis voltage;
- The pack-average cell resistance;
- The pack-average cell (inverse) capacity; and
- The current sensor bias.

The last item, the current sensor bias, is a critical value to know for any SOC estimation algorithm. This is not something that could be estimated using N standard SPKFs, but is enabled by using the bar-delta filtering method. The bias dynamics are modeled as $i_k^b = i_{k-1}^b + n_{k-1}^b$, where n_k^b is a fictitious noise source that is included in the model to allow SPKF to adapt the bias estimate.
In order to design an SPKF to estimate these quantities, a state-space model of their dynamics is required. The model for all of these quantities is the same as the individual-cell model, except that average voltages and average currents are used as input rather than individual voltages and currents. (Because the cells are connected in series, the bar filter considers all cell currents to be equal, which is true unless some cells are being equalized. Delta filters must consider equalization currents as well.)
For example, (ignoring bias current for now) to find the pack-average SOC dynamics, we take Eq. (1) and add N copies together—one for each cell—and then divide by N.

$$\frac{1}{N}\sum_{i=1}^{N} z_k^{(i)} = \frac{1}{N}\sum_{i=1}^{N} z_{k-1}^{(i)} - \frac{i_{k-1}\Delta t}{N}\sum_{i=1}^{N} C_{\text{inv},k-1}^{(i)}.$$

We can express this as

$$\bar{z}_k = \bar{z}_{k-1} - i_{k-1}\Delta t \bar{C}_{\text{inv},k-1},$$

where $\bar{z}_k$ is the pack-average SOC at time index k, and $\bar{C}_{\text{inv},k}$ denotes the pack-average cell inverse capacity. If we also consider the current-bias state,

$$\bar{z}_k = \bar{z}_{k-1} - (i_{k-1} - i_{k-1}^b)\Delta t \bar{C}_{\text{inv},k-1}.$$

Similarly, the dynamics of all pack-average states and parameters of interest may be summarized

as:

$$\begin{aligned}
\bar{z}_k &= \bar{z}_{k-1} - (i_{k-1} - i^b_{k-1})\Delta t \bar{C}_{\text{inv},k-1} \\
\bar{f}_k &= A_f \bar{f}_{k-1} + B_f (i_{k-1} - i^b_{k-1}) \\
\theta_k &= \exp\left(-\left|(i_{k-1} - i^b_{k-1})\gamma \Delta t \bar{C}_{\text{inv},k-1}\right|\right) \\
\bar{h}_k &= \theta_k \bar{h}_{k-1} + (1-\theta_k)\,\text{sgn}(i_{k-1} - i^b_{k-1}) \\
\bar{R}_k &= \bar{R}_{k-1} + n^{\bar{R}}_{k-1} \\
\bar{C}_{\text{inv},k} &= \bar{C}_{\text{inv},k-1} + n^{\bar{C}_{\text{inv}}}_{k-1} \\
i^b_k &= i^b_{k-1} + n^b_{k-1},
\end{aligned}$$

where $n^{\bar{R}}_k$ and $n^{\bar{C}_{\text{inv}}}_k$ are fictitious noise sources that allow the SPKF to adapt the corresponding pack-average parameters.
The bar-filter for the pack employs an SPKF that uses this model of pack-average states and the measurement equation

$$\bar{y}_k = \text{OCV}(\bar{z}_k) + G_k \bar{f}_k - \bar{R}_k(i_k - i^b_k) + M\bar{h}_k + v_k,$$

where v_k models sensor noise.

4.2 The cell delta filters

The quantities that we are most interested in estimating at the individual cell level are: SOC, resistance, and capacity. These all factor into determining pack available power and lifetime (state-of-health) estimates.
We will first consider the delta filter approach to determining cell SOC. Note, from before, $\Delta z^{(i)}_k = z^{(i)}_k - \bar{z}_k$. Then, using prior equations for the dynamics of $z^{(i)}_k$ and $\bar{z}_k$, we find:

$$\begin{aligned}
\Delta z^{(i)}_k &= z^{(i)}_k - \bar{z}_k \\
&= \left(z^{(i)}_{k-1} - (i_{k-1} - i^b_{k-1})\Delta t C^{(i)}_{\text{inv},k-1}\right) - \\
&\quad \left(\bar{z}_{k-1} - (i_{k-1} - i^b_{k-1})\Delta t \bar{C}_{\text{inv},k-1}\right) \\
&= \Delta z^{(i)}_{k-1} - (i_{k-1} - i^b_{k-1})\Delta t \Delta C^{(i)}_{\text{inv},k-1}
\end{aligned}$$

where $\Delta C^{(i)}_{\text{inv},k} = C^{(i)}_{\text{inv},k} - \bar{C}_{\text{inv},k}$. Because $\Delta C^{(i)}_{\text{inv},k}$ tends to be small, the state $\Delta z^{(i)}_k$ does not change very quickly, and can be updated at a slower rate than the pack-average SOC by accumulating $(i_{k-1} - i^b_{k-1})\Delta t$ in between updates. An output equation suitable for combining with this state equation is

$$\begin{aligned}
y^{(i)}_k &= \text{OCV}(\bar{z}_k + \Delta z^{(i)}_k) + G_k \bar{f}_k \\
&\quad -(\bar{R}_k + \Delta R^{(i)}_k)(i_k - i^b_k) + M\bar{h}_k + v_k.
\end{aligned}$$

To estimate $\Delta z^{(i)}_k$, an SPKF is used with these two equations. Since it is a single-state SPKF, it is very fast.
We can similarly make state-space models of the delta-resistance and delta capacity states. A simple state-space model of the delta-resistance state is:

$$\begin{aligned}
\Delta R^{(i)}_k &= \Delta R^{(i)}_{k-1} + n^{\Delta R}_{k-1} \\
y_k &= \text{OCV}(\bar{z}_k + \Delta z^{(i)}_k) \\
&\quad -(\bar{R}_k + \Delta R^{(i)}_k)(i_k - i^b_k) + v^{\Delta R}_k
\end{aligned}$$

where $\Delta R^{(i)}_k$ is the difference between pack-average resistance and cell resistance and is modeled as a constant value with a fictitious noise process $n^{\Delta R}_k$ allowing adaptation, y_k is a crude estimate of the cell's voltage, and $v^{\Delta R}_k$ models estimation error. The dynamics of the delta-resistance state are simple and linear enough to use a single-state EKF rather than an SPKF. Again, it is very fast.
To estimate cell capacity using an EKF, we again formulate a simple cell model

$$\begin{aligned}
\Delta C^{(i)}_{\text{inv},k} &= \Delta C^{(i)}_{\text{inv},k-1} + n^{\Delta C_{\text{inv}}}_{k-1} \\
d_k &= (z^{(i)}_k - z^{(i)}_{k-1}) + (i_{k-1} - i^b_{k-1})\Delta t \times \\
&\quad \left(\bar{C}_{\text{inv},k-1} + \Delta C^{(i)}_{\text{inv},k-1}\right) + e_k
\end{aligned}$$

The second equation is a reformulation of the SOC state equation such that the expected value of d_k is equal to zero by construction. Again, an EKF is constructed using the model defined by these two equations to produce a capacity estimate. As the EKF runs, the computation for d_k in the second equation is compared to the known value (zero, by construction), and the difference is used to update the inverse capacity estimate. Note that good estimates of the present and previous states-of-charge are required. Here, they come from the pack SPKF combined with the cell SPKF.
The output of the delta filters is computed by combining the average battery pack state with the battery cell module delta states produced by the individual Kalman filters:

$$\begin{aligned}
z^{(i)}_k &= \bar{z}_k + \Delta z^{(i)}_k \\
R^{(i)}_k &= \bar{R}_k + \Delta R^{(i)}_k \\
C^{(i)}_k &= \frac{1}{\bar{C}_{\text{inv},k} + \Delta C^{(i)}_{\text{inv},k}}
\end{aligned}$$

We note again that one desirable feature of the delta filters has been defined by the foregoing equations. Although the battery pack state may change rapidly, the difference between any battery cell module state and the battery pack average state changes very slowly. With this understanding, it is apparent that the bar filter needs to be executed frequently, but the delta filters need be executed much less frequently. Therefore, the computational tasks implemented by bar-delta filtering can approach $1/N$ times the computational tasks utilized by other methods and still produce accurate estimates of the desired battery cell module states.

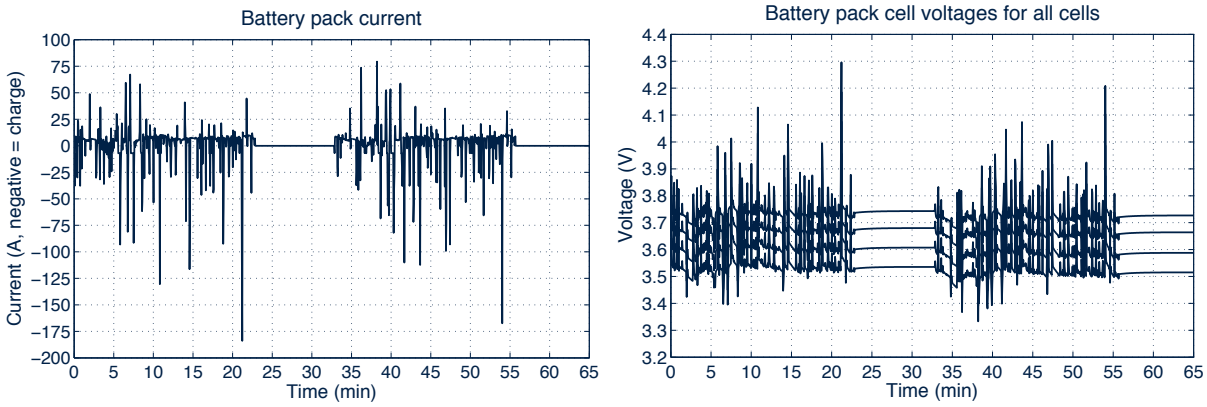

Figure 1: Pack current versus time and individual cell voltages versus time for algorithm testing.

5 Results: Algorithm accuracy

In this section, we present some results obtained using the proposed algorithms. In order to do so, we would ideally present estimates of SOC (etc.) compared with true values of SOC for some example tests. We immediately run into a problem, however: regardless of which BMS estimation algorithm is used, the primary difficulty when validating any such algorithm on physical cells is that the "truth" values are not known. There are no sensors that can directly measure SOC or SOH. At different points in time, laboratory tests can be performed that can be used to determine *a posteriori* what the SOC or SOH was at that time, but cannot determine SOC or SOH in real time, as the battery pack operates.

We have developed a simulation-based validation methodology as one component of a strategy that overcomes this obstacle [11]. A software simulator of cell dynamics first synthesizes various driving, temperature, parameter, and sensor-fault profiles for the battery pack being modeled. All internal variables that are dependent on the cycling history of the cell (e.g., SOC and SOH) are known to the software simulator, so "truth" values are established. The input-output behavior of this model has been tested against physical cells, and works well. We call this the "Data Generator System" (DGS). The BMS algorithms are then executed using this synthetic cell data as input, and the algorithm results are compared to the "truth" values.

We acknowledge that this method is no substitute for actual testing of real cells. However, it can help to minimize the amount of this testing that is required if a careful design-of-experiments approach is taken. A balanced overall validation strategy therefore comprises a large suite of desktop validation tests and a smaller suite of real-time tests on physical battery packs. We present this "desktop validation" means here since it gives an unequivocal truth value for the purpose of evaluating the algorithms themselves.

The synthesized data was generated using a model of a fifth-generation prototype LiPB cell that is very similar to the one presented in [5, 6], but with a somewhat higher capacity. Root-mean-square (RMS) model error versus cell tests conducted using an Arbin BT2000 are on the order of 5–10 mV over all operating conditions.

The tests presented in this section are the result of cycling a four-cell LiPB pack with a UDDS cycle, a rest period, the same UDDS cycle, and a rest period. The pack cells had true capacities of 6.5, 7.0, 7.5, and 8.0 Ah, and resistances of 2.0, 2.25, 2.5, and 2.75 mΩ. The cells had initial SOC values of 40, 45, 50, and 55%. The current-sensor bias was 0.5 A. The algorithms were initialized with all cells having estimated capacity of 6.2 Ah, estimated resistances of 2.25 mΩ, estimated current-sensor bias of 0 A, and initial SOC estimates based on the resting initial voltage of the cells. SPKF was used for the bar filter and the SOC delta filters, and EKF was used for the resistance and capacity-inverse delta filters.

The left frame of Fig. 1 shows the pack current plotted versus time for this test, using the PNGV convention (negative current is charge current). The right frame of Fig. 1 shows the individual cell voltages versus time for this test. Due to different initial SOC levels, the cell voltages are clearly separated. Less apparent, the differing capacities and resistances also affect the cell voltages.

The left frame of Fig. 2 shows SOC plotted versus time for the four cells. Movement is quite similar in all cells, as would be expected. The right frame if Fig. 2 shows the estimation error for the pack-average SOC estimate produced by the bar filter. Error bounds on the SOC state are also plotted based on the output of the SPKF, and we see that (1) the estimation error is always within 1%, and (2) the error bounds always encompass the true error. (One key benefit of using EKF or SPKF is that the estimator algorithm automatically produces both a state estimate *and* dynamic error bounds on that estimate. Here, we see that those bounds are accurate.)

Fig. 3 plots the output of the entire bar-delta algorithm (pack average results added to delta-filter results) to display the estimates of all four cell SOC values versus time. In the left frame, the actual SOC values are plotted versus time, and compared with the estimates. In the right frame, the estimation errors are plotted versus time, and error bounds are also drawn. Root-mean-squared estimation error is on the order of 0.4%.

Fig. 4 displays a similar result for the resistance estimates of the bar-delta filter. The left frame

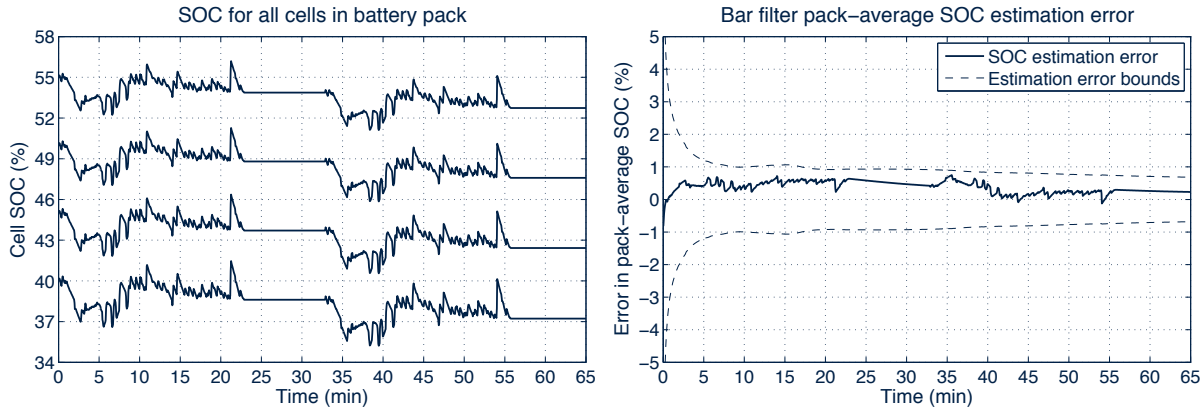

Figure 2: On the left, individual cell SOCs are plotted; on the right, pack-average SOC estimation error is plotted.

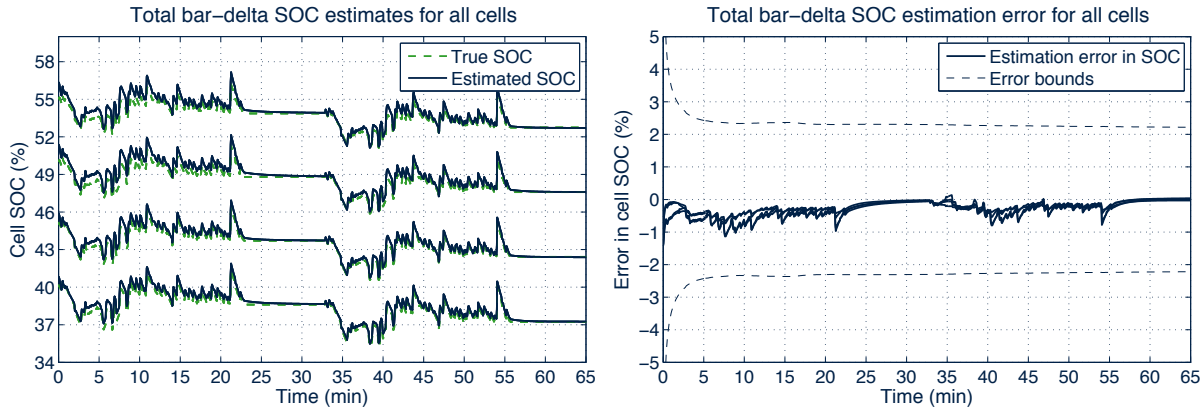

Figure 3: On the left, individual SOC estimates compared to truth for all cells; on the right, SOC estimation error is plotted.

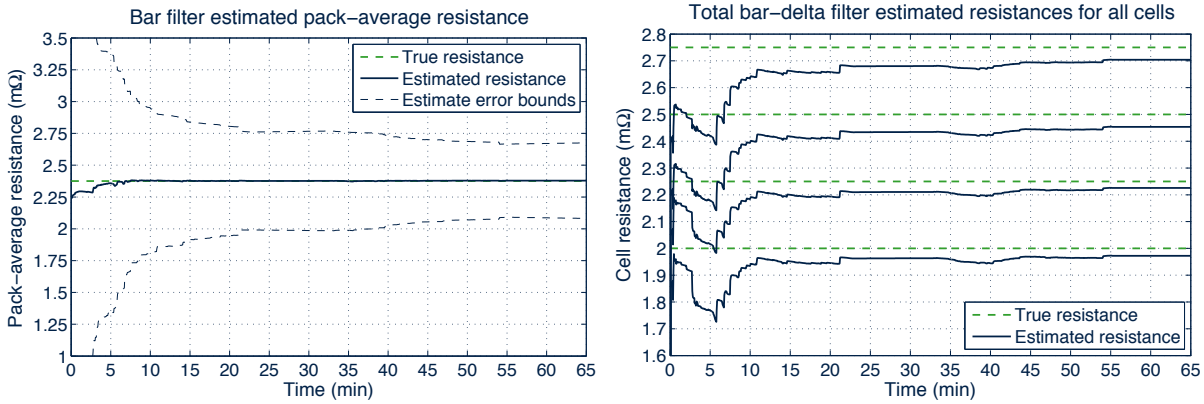

Figure 4: Resistance estimates from the bar-delta filtering method.

shows the pack-average resistance estimate versus time, compared with the true pack average resistance, with error bounds from the SPKF also drawn. The right frame draws the individual resistance estimates versus truth when the bar-filter results are combined with the delta-filter results. Over time, we see that the estimates are converging to the true resistance values.

Capacity estimates evolve in a similar way to resistance estimates. However, the time scale of adaptation is much longer, since capacity is very weakly linked to the output measurement. Abrupt changes in capacity will not be tracked very quickly; but, capacity fade due to normal aging will be tracked very well.

Finally, Fig. 5 shows the current sensor bias estimate versus the true current-sensor bias as a function of time. The bar-delta method is able to quickly respond to the current-sensor bias and converge to the neighborhood of its value.

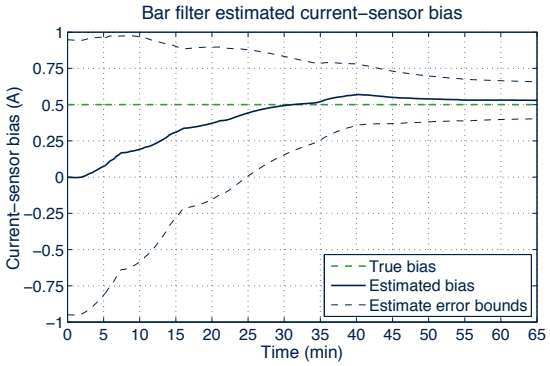

Figure 5: Current sensor bias compared to estimate of current sensor bias as a function of time. Error bars on the estimate, from the SPKF, are also drawn.

6 Results: Algorithm speedup

The goal of the bar-delta filtering is to be both accurate and fast. We have just demonstrated the accuracy of the method; now, it remains to see if the algorithm delivers on its promise of speed. Table 1 lists some results of benchmarking various configurations of the algorithm. In all cases, the code being run was compiled hand-optimized C code, running on a PowerPC platform. The pack test scenario comprised more than one hour of real-time data, so the CPU time per iteration was computed as the run time for the code divided by the number of iterations of the algorithms performed.

The baseline simulation implemented one SPKF per cell in a 100-cell pack. Each iteration required 5.272 ms of processor time to execute the 100 SPKFs. (The absolute CPU time is not as important as the relative CPU time, since the algorithms can be implemented on a variety of processor platforms. However, it is clear that the test platform is overkill in this example, since only 0.5% of the available CPU time is being used to update the algorithms.)

The next simulation implemented the bar filter only (using SPKF). The cell model used in this simulation differed from the baseline simulation in that it is estimating the current-sensor bias state (the baseline simulation did not—in fact, it is not obvious how to do so when multiple SPKFs are being used). Due to the additional state being estimated, the speedup (calculated as baseline simulation time divided by test case simulation time) was not quite 100, as might be predicted, but is very close to it.

The final two simulations include both the bar filter and the delta filters. The first of these two simulations updated all 100 delta filters every iteration, resulting in a speedup of 27.7 over the baseline case. The second updated half of the delta filters every iteration: the first 50 delta filters on even iterations, and the second 50 delta filters on odd iterations. Estimation accuracy was not degraded, and a speedup of 42.9 was achieved. By changing the number of delta filters updated every iteration, we see that speedup of between about 27 and 78 can be achieved overall.

7 Conclusion

It is necessary to monitor all SOC and SOH values in a battery pack in order to be able to accurately estimate power (e.g., using the method of [12]), and to monitor cell health. One method to accomplish this is to independently estimate SOC and SOH of all cells. That is, to execute N SOC estimators and N SOH estimators, if there are N cells in a pack, wired in series. If very accurate methods are used (e.g., SPKF), this is very slow.

We have introduced a method that can be vastly faster and still gives accurate estimates.

- A single "bar filter" jointly estimates the pack-average state and parameters;
- N individual "delta filters" estimate the difference between cell SOC and pack-average SOC;
- N individual "delta filters" estimate the difference between cell and pack-average resistance;
- N individual "delta filters" estimate the difference between cell and pack-average inverse capacity;
- The individual cell filters can operate at a slower rate than the pack filter.

This overall "bar-delta filter" achieves considerable speedup over the benchmark case. Moreover, it is able to estimate a pack current-sensor bias very naturally, to keep the other estimates unbiased by a drifting current sensor. In our implementation we used SPKF for the bar filter and the SOC delta filter, and EKF for the resistance and inverse-capacity delta filter. However, the estimation schemes used for the different filters is very flexible.

References

[1] S. Piller, M. Perrin, and A. Jossen. Methods for state-of-charge determination and their applications. *Journal of Power Sources*, 96:113–120, 2001.

Table 1: Performance measurements of various algorithm configurations.

Description of test (for pack comprising 100 cells)	CPU time per iteration	Speedup
One SPKF per cell	5.272 ms	1.0
One pack bar filter only, no delta filters	0.067 ms	78.7
One pack bar filter, 100 delta filters updated per iteration	0.190 ms	27.7
One pack bar filter, 50 delta filters updated per iteration	0.123 ms	42.9

[2] F. Huet. A review of impedance measurements for determination of the state-of-charge or state-of-health of secondary batteries. *Journal of Power Sources*, 70(1):59 – 69, 1998.

[3] Shalini Rodrigues, N. Munichandraiah, and A. K. Shukla. A review of state-of-charge indication of batteries by means of a.c. impedance measurements. *Journal of Power Sources*, 87(1-2):12 – 20, 2000.

[4] Alvin J. Salkind, Craig Fennie, Pritpal Singh, Terrill Atwater, and David E. Reisner. Determination of state-of-charge and state-of-health of batteries by fuzzy logic methodology. *Journal of Power Sources*, 80(1-2):293 – 300, 1999.

[5] Gregory L. Plett. Sigma-point Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 1. Introduction and state estimation. *Journal of Power Sources*, 152(2):1356–1368, 2006.

[6] Gregory L. Plett. Sigma-point Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 2. Simultaneous state and parameter estimation. *Journal of Power Sources*, 152(2):1369–1384, 2006.

[7] Gregory L. Plett. Results of temperature-dependent LiPB cell modeling. In *CD-ROM Proceedings of the 21st Electric Vehicle Symposium (EVS21)*, Monaco (April, 2005), 9 pages.

[8] Gregory L. Plett. Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs—Part 1: Background. *Journal of Power Sources*, 134(2):252–61, August 2004.

[9] Gregory L. Plett. Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs—Part 2: Modeling and identification. *Journal of Power Sources*, 134(2):262–76, August 2004.

[10] Gregory L. Plett. Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs—Part 3: Parameter estimation. *Journal of Power Sources*, 134(2):277–92, August 2004.

[11] Gregory L. Plett, Robert Billings, and Martin J. Klein. Desktop and HIL validation of hybrid-electric vehicle battery-management-system algorithms. In *Proceedings of SAE Congress 2007*, April 2007. 7 pages.

[12] Gregory L. Plett. High-performance battery-pack power estimation using a dynamic cell model. *IEEE Transactions on Vehicular Technology*, 53(5):1586–93, September 2004.

Authors

Gregory Plett received his Ph.D. in Electrical Engineering from Stanford University. He is Associate Professor of Electrical and Computer Engineering at the University of Colorado at Colorado Springs, and consultant to Compact Power Inc. He specializes in state estimation and control design for battery management systems.