4
Battery Health Estimation

4.1 Need for health estimates

Over time, cells in a battery pack will age and their performance will degrade. They will eventually reach a point where they no longer meet the performance requirements of the battery pack, which we consider to be the pack's end of life (although so-called "second-life" applications might be able to make use of the remaining diminished capability). Between a battery pack's beginning of life and end of life it is important to have knowledge regarding the present degradation status of its cells to be able to make accurate calculations of state of charge, available energy and available power.

In Volume I of this series, we focused on understanding *ideal* battery cells: cells that do not age and hence have constant model parameter values. Sadly, such cells do not exist. We now turn our attention to designing algorithms that work with *normal* battery cells: cells that do age, and hence have operational characteristics that degrade. We need to understand which parameters in the model change over time, which of these are most significant, and how to modify our battery-management methods to account for aging.

Note that normal aging is only one cause of cell failure. Failures can also occur because of cell design faults, poorly controlled manufacturing processes or impurities in the materials used during manufacture, abuse, and uncontrolled operations. Cells that have design or manufacturing faults or are abused often appear normal for a period of time and then fail very rapidly. The methods discussed in this chapter cannot predict this sudden-onset failure (it remains an open problem to do so), but there are some remediation methods that can maximize safety even if cells fail in this way.[1] Proper battery management will prevent uncontrolled operation while the BMS is active, but has no influence over external factors or over ambient conditions

[1] See, for example, Kim, G-H, Smith, K, Ireland, J, and Pesaran, A., "Fail-safe design for large capacity lithium-ion battery systems," *Journal of Power Sources*, 210, 2012, pp. 243–253.

while the BMS is inactive (e.g., physical damage during an xEV collision, or an out-of-bounds ambient temperature while the BMS and its thermal controls are turned off).

Referring back to Fig. 3.1, which illustrates the main topics of this book, it turns out that state estimation using an equivalent-circuit cell model works just as well for normal battery cells as it does for ideal battery cells so long as every cell's model parameter values are continuously updated to reflect the cell's present aged characteristics. In this chapter, we begin to examine how to estimate the cell-model parameter values that change relatively slowly. Our focus is on understanding and tracking normal aging processes as well as uncontrolled operations in the sense that overvoltage, overtemperature, and so forth accelerate the normal degradation mechanisms. The battery pack can still be used during the period when it is aging normally until its end of life, but at reduced levels of performance. We do not look at internal faults and abuse, which are usually detected using other means (e.g., those discussed in Chap. 1) and may require shutting down parts or all of the battery pack for immediate servicing in order to prevent propagation of the failure. The BMS designer should work closely with the battery-cell electrochemists and production engineers to understand all of the probable failure mechanisms and their characteristics fully to be able adapt the knowledge from this chapter into their individual designs.

In particular, we are most interested in those quantities that reflect a change in the performance that the battery pack can deliver. These are indicators of battery-pack SOH. There is no universally agreed-upon definition of SOH, but the most commonly estimated quantities used to summarize battery pack health include the present total capacity and present equivalent series resistance of every cell. Accurate estimates of total capacity and equivalent series resistance allow us to compute reliable total energy and available power estimates for the battery pack over its service lifetime.

4.1.1 Total capacity

As a battery cell ages, its total capacity Q decreases. In a lithium-ion cell, this is due primarily to unwanted side reactions that consume lithium that could otherwise be used during charge and discharge of the cell, and to structural deterioration of the electrode active materials that eliminates lithium storage sites.

Fig. 4.1 illustrates a simplified example of ideal-cell behavior. In the figure, both the negative and positive electrodes have sixteen sites that could hold lithium. Presently, four negative-electrode sites and five positive-electrode sites are shown to be occupied. When the

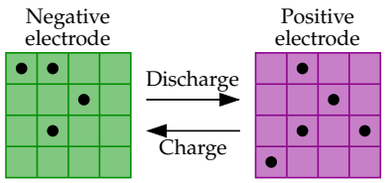

Figure 4.1: Ideal-cell operation.

cell is fully charged, there will be nine occupied sites in the negative electrode, and no occupied sites in the positive electrode. When the cell is fully discharged, there will be no occupied sites in the negative electrode, and nine occupied sites in the positive electrode.[2] The total capacity of the cell is equal to the minimum of the number of storage sites in the negative electrode, the number of storage sites in the positive electrode, and the amount of lithium that can be cycled. In this example, the total capacity is the minimum of 16, 16, and 9, which yields a total capacity of nine lithium atoms.

Continuing the example, a side reaction is an undesired chemical process that consumes lithium while it is in transit from one electrode to another and removes it from cycling. Most side reactions happen while the cell is being charged. This is illustrated in Fig. 4.2, where one lithium atom is shown being consumed by a side reaction as the cell is being charged. The total capacity has now been reduced to the minimum of 16, 16, and 8, which yields a total capacity of eight lithium atoms.

Structural deterioration is something that eliminates lithium storage sites from one of the electrodes, perhaps due to a collapse of part of the crystal structure of the electrode itself. This is illustrated in Fig. 4.3, where damage to the positive electrode is shown as a white scar. Some lithium may be trapped in the structure such that it is no longer free to cycle back and forth as the cell is charged and discharged, so capacity is lost. In the figure, the lithium atom in the lower-right corner of the positive electrode is trapped by the structural collapse. Structural collapse can also eliminate lithium storage sites. In the figure, the positive electrode has only ten good storage sites, but one of those is isolated from cycling, so there are only nine useable storage sites. So, the total capacity is the minimum of 16, 9, and 8, yielding a final total capacity of 8 lithium atoms.

This slow reduction in capacity is often referred to as *capacity fade*. We require algorithms that track capacity fade to provide the other battery-management-system algorithms with up-to-date estimates of every cell's total capacity. This knowledge is critical to be able to calculate battery-pack available energy accurately, where total capacity is a major contributing factor (as we saw in Sect. 1.14). If coulomb counting is being used for SOC estimation, an accurate estimate of total capacity is also needed; however, if Kalman filters are being used instead, the state of charge estimates turn out to be fairly insensitive to a poor total-capacity estimate since the built-in feedback correction mechanism is able to compensate for moderate errors in the total-capacity estimate. The dependence of available-power estimates on the value of total capacity also turns out to be minimal.

[2] This is a simplification. Neither electrode is ever completely full or completely empty during operation. Instead, a cell is considered fully charged or fully discharged when its open-circuit voltage reaches predetermined maximum and minimum levels.

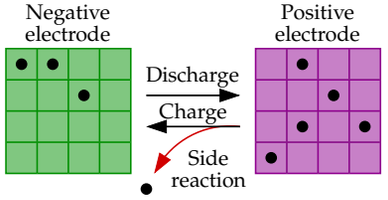

Figure 4.2: Capacity loss due to side reaction.

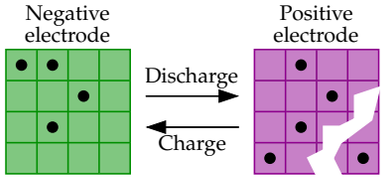

Figure 4.3: Capacity loss due to material loss.

4.1.2 *Equivalent series resistance (ESR)*

As a battery cell ages normally, its equivalent series resistance R_0 increases. This is also due primarily to unwanted side reactions and structural deterioration. The side reactions tend to form resistive films on the surface of the active-material particles that impede the ionic conductivity. Structural deterioration severs electronic pathways between particles and decreases the electronic conductivity.

Having an up-to-date knowledge of equivalent-series resistance is important because it turns out to be a major contributing factor to the available-power calculation.[3] It can be a major contributing factor to state of charge estimation for some voltage-based methods; however, for Kalman-filter-based methods it it is a minor factor, and it does not affect coulomb-counting methods at all. It does not have a significant role in available-energy estimation.

[3] Because resistance and power are so tightly coupled, resistance rise in a cell is commonly referred to as *power fade*.

4.1.3 *Other cell parameters*

As the cell ages, other cell-model parameter values will change as well. For example, while the open-circuit-potential relationships of each electrode remain fixed by the chemistry of the crystal structures, a cell's overall OCV relationship can change due to shifts in the stoichiometric operating windows used by each electrode that occur when capacity is lost due to side reactions and structural deterioration. This effect tends not to be large, but may need to be tracked by future BMS.

Other cell-model parameter values almost certainly change as well; however, few if any present BMS make efforts to estimate the changes. Overall, changes to the equivalent-series resistance and total capacity have the dominant impact on BMS performance.

As we proceed in this chapter, we will first discuss in greater detail the primary specific known electrochemical and structural mechanisms by which a lithium-ion cell can degrade. We will see that this is a lot more complicated than the simple example just presented, but that the dominant outcomes are still well represented by changes in resistance and total capacity in an equivalent-circuit cell model. There are other secondary degradation mechanisms and doubtless some that have not yet been discovered, but so long as they manifest as slow changes in cell resistance and total capacity, the methods of this chapter will continue to work.

We will then explore the concept of *sensitivity*, leading to an understanding of how *observable* the changes in total capacity and resistance are from cell input/output (current/voltage) measurements. This will lead directly to a simple method that can estimate

equivalent-series resistance with a good degree of accuracy. However, it will also show that finding a good estimate of cell total capacity is a very challenging task.

We know that a nonlinear Kalman filter can be used to estimate the quickly time-varying states of a cell model. It turns out that nonlinear Kalman filters can also be used to estimate the slowly time-varying parameter values. We will investigate how to do so, and some remedies to a common pitfall when trying to estimate the states and parameters of a model at the same time. An advantage of this approach is that it can be used to estimate any time-varying parameter in the cell model—not only resistance and capacity.

If we cannot afford the complexity of a Kalman-filter method and yet desire to estimate total capacity, we might consider a regression method. However, it turns out that methods for estimating total capacity that are based in principle on ordinary least-squares regression will yield biased results. We spend a considerable amount of time in this chapter seeing why this is true, and showing how to estimate total capacity correctly using regression. We further explore how we can know that our estimate is correct and how to compute confidence intervals for our total-capacity estimates.

4.2 *Negative-electrode aging*

Lithium-ion battery-cell operational characteristics change slowly over time as the cell ages. This aging can be captured in an equivalent-circuit cell model by adapting the parameter values within the model so that good predictions are made over the entire lifetime of the battery cell. However, since equivalent-circuit-model parameters do not individually describe any of the physical electrochemical processes taking place inside the cell, these changing parameter values do not give any insight into why or how the aging has occurred.

To understand aging, we must consider a cell from a physics-based perspective.[4] In the next sections, we will seek to describe aging *qualitatively*, as even this simplified degree of understanding is valuable to the BMS algorithm designer. For example, it helps to explain why manufacturers put voltage limits and current limits on cells.

For maximum impact, we would need to be able to model aging *quantitatively* as well. Two crude reduced-order quantitative cell-degradation models are introduced in Chap. 7. These models are simple enough to be used alongside an equivalent-circuit model and can be used to track some kinds of aging, leading to some preliminary physics-based control methods. For more advanced controls, which provide more accurate limits on the power available from the bat-

[4] Because aging depends on the physics of the cell, aging mechanisms of a lithium-ion cell will be different from those of other types of cell. We restrict our discussion to lithium-ion cells here, basing this section on the excellent paper by Vetter et al., "Ageing mechanisms in lithium-ion batteries," *Journal of Power Sources*, 147, 2005, 269–281.

tery pack, one would need to combine reduced-order physics-based degradation models with a reduced-order physics-based cell model, such as was developed in Vol. I. This is one topic we will cover in the planned Vol. III of this series.

In a lithium-ion cell, aging processes occur within both the negative and positive electrodes. We consider degradation in the negative electrode first, where aging effects are seen at three scales. First, some aging occurs at the surface of the electrode active-material particles; that is, at the interface between the solid and the electrolyte. Second, other aging mechanisms take place within the interior of the active-material particles. Third, aging processes can take place within the overall composite electrode structure, including changes to the active materials, the conductive additives, the binder materials, the current collectors, the porosity, and so forth. We will consider aging at these three scales separately in the next subsections.

4.2.1 *Negative electrode aging at surface of particles*

Most commercial lithium-ion battery cells have negative-electrode active materials that are comprised of a synthetic or natural graphite. Graphite has good lithium storage capacity, can be charged and discharged repeatedly, and is inexpensive and nontoxic. Most important, perhaps, is that lithiated graphite has very low voltage with respect to a lithium-metal reference. This is key to maximizing overall cell voltage, since cell voltage is equal to the positive-electrode potential minus the negative-electrode potential.

Fig. 4.4 plots the open-circuit-potential relationships for graphite and for another candidate negative-electrode active material: lithium-titanate-oxide (LTO). The horizontal axis is the present operating stoichiometry of the electrode and corresponds to the value of x in Li_xC_6, for graphite or in $Li_{4+3x}Ti_5O_{12}$ for LTO. The black dot at $x = 0.55$ shows that the open-circuit potential of graphite is about 0.1 V at that point. Charging a cell increases x and causes the potential to decrease; discharging does the opposite.

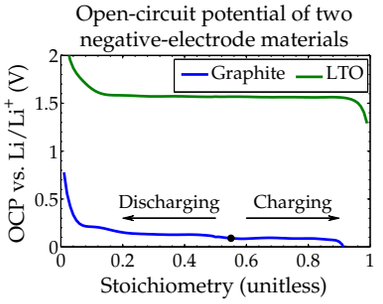

Figure 4.4: Open-circuit-potential relationships for two negative-electrode active materials.

While the low potential of graphite over most of its operational range makes it highly desirable for high-voltage lithium-ion cells, it is also the cause of the dominant degradation mechanism that takes place inside a lithium-ion cell. This low electrical potential is outside the electrochemical voltage stability window of the organic solvents used in lithium-ion cell electrolytes. When the electrolyte solvents come into contact with lithiated graphite, reductive electrolyte decomposition takes place at the electrode/electrolyte interface. The rate

of decomposition is accelerated by lower electrode potentials, which occur when the electrode (and the cell) are in a high SOC.[5]

Lithium-ion cells are constructed in a fully discharged state (all of the lithium is in the positive electrode). The nonlithiated graphite is then at a high-enough potential that the electrolyte-reduction reaction does not occur during cell fabrication. However, when the cell is charged for the first time, the graphite becomes lithiated and its potential drops. Solvent in the electrolyte that comes into contact with the surface of the particle is reduced and forms reaction products that coat the surface of the electrode particle with a *solid–electrolyte interphase (SEI)* surface film. The SEI is a passivating layer that partially insulates the graphite from the remaining solvent in the electrolyte and hence slows down further reaction. Hence, most of the SEI is formed during the initial charge process of the cell, which leads to this first charge being termed the *formation process*. The growth of SEI is illustrated in Fig. 4.5.

[5] LTO has high-enough potential that this side reaction does not occur, greatly extending life. However, it also lowers the overall cell voltage, and hence the cell's energy density.

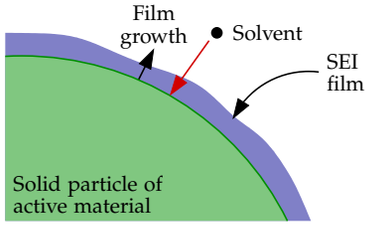

Figure 4.5: Growth of a SEI film.

The side-reaction that produces SEI film consumes lithium while creating the film products. Therefore, cell total capacity decreases due to SEI growth. The SEI film is porous enough to allow intercalation and deintercalation of lithium to and from the graphite, but decreases the conductivity of ion transfer and hence increases cell resistance. Therefore, SEI growth leads to both capacity fade and power fade.

The exact nature of the SEI layer is complicated and not completely understood. It is suspected that numerous reaction products form, then decompose, and then combine into more stable products. However, we do know that once lithium is consumed by SEI growth it is never returned to a form that enables cycling. That is, once capacity is lost to grow SEI it is permanently lost.

While SEI grows fastest during the formation cycle, it continues to build over time. Anything that exposes graphite to solvents in the electrolyte will cause SEI growth. For example, while the SEI film tends to impede the solvent from reaching the graphite surface, the film has enough porosity that some solvent continues to permeate the film and contact the surface of the particle. When this happens, more SEI forms, and the SEI layer grows. High temperatures can lead to reactions that break down the SEI layer, which can lead to new SEI forming on the newly exposed graphite.

Charging at high rates can force solvent to co-intercalate into the graphite along with lithium and so the SEI reaction can take place *inside* a graphite particle. When this happens, gasses generated during SEI formation cause expansive pressures to build up inside the particle, which tend to crack it along internal grain boundaries or to flake off layers (a process termed *exfoliation*). Both of these expose

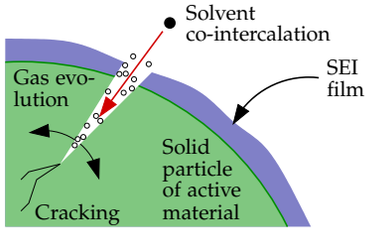

Figure 4.6: Gas generation and cracking due to solvent co-intercalation.

more fresh graphite to solvent in the electrolyte, leading to more SEI formation. This is illustrated in Fig. 4.6.

Trace water in the electrolyte combines with ionized fluorine from the electrolyte salt $LiPF_6$ to form hydrofluoric acid, HF. This acid attacks the SEI, thinning it and allowing more solvent to contact the graphite, forming more SEI. The acid can also accelerate positive-electrode degradation (as we will see in Sect. 4.3.1), leading to dissolved ionized metals such as manganese or cobalt in the electrolyte. These can become part of products comprising the SEI layer when they propagate through the separator to the negative-electrode region of the cell, a process known as *anode poisoning*. The SEI products formed by these metals tend not to have high electronic conductivity and so increase cell resistance. They can also plug pores that would otherwise be used for lithium, preventing lithium cycling and causing capacity fade. These two mechanisms are illustrated in Fig. 4.7.

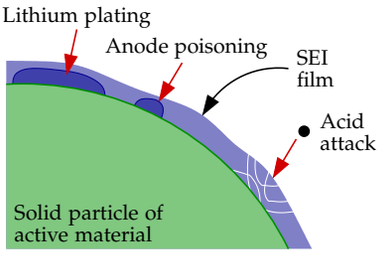

Figure 4.7: Acid attack, anode poisoning, and lithium plating.

A final surface effect that we consider is that of *lithium plating*. This side reaction can cause severe capacity loss and is most acute at cold temperatures where diffusion of lithium in the solid particles is slower.[6] If charging is forced, the local particle surface overpotential can reach a level that causes lithium ions from the electrolyte to join with electrons from the external circuit and plate solid lithium metal on the surface of the particle (this happens when the surface solid–electrolyte potential difference drops below 0 V). Capacity is irreversibly lost. The lithium metal tends to further catalyze SEI growth and forms a metallic annealing site that promotes growth of metal dendrites, which can penetrate the separator and eventually lead to a cell short circuit. This is also illustrated in Fig. 4.7.

[6] For this reason, great care should be taken when charging a lithium-ion cell at cold temperatures. Many manufacturers do not recommend charging a cell at ambient temperatures below about 0 °C.

4.2.2 *Negative electrode aging in bulk*

Charging and discharging lithium-ion cells increases and decreases the amount of lithium present in the negative-electrode active particles. Lithiation causes stresses that tend to lead to an increase in volume; delithiation tends to decrease volume. In graphite, this volume change is modest, usually less than 10 %.[7]

Over time, these volume changes can lead to cracking and splitting of particles along internal grain boundaries. As mentioned in Sect. 4.2.1, solvent co-intercalation and gas formation can also split particles. In graphite electrodes, this will cause more SEI to form on the exposed graphite. Alternately, the volume changes can crack only the SEI layer, which will expose more surface graphite and lead to more SEI formation.

[7] In silicon, a very-high-capacity possible graphite replacement in future lithium-ion cells, the volume changes can be more than 300 %!

4.2.3 *Negative electrode aging in composite electrode*

Within the composite negative electrode of a lithium-ion cell, conductive additives such as carbon black are often added to improve the electrode's electronic conductivity and binders such as PVdF are added to help maintain contact between particles. These inactive components are not often mentioned when describing the parts of a lithium-ion cell as they do not take part in charge and discharge operations. However, they are critical to the proper functioning of the cell, and a great deal of care is taken when designing a cell to arrive at a good ratio of the inactive materials to the active materials in the electrode.[8] Fig. 4.8 illustrates the structure of the electrode, including the binders and conductive additives that coat the particles.

When lithiating and delithiating the active materials in an electrode, stresses leading to deformations can cause the binder to fail, leading to mechanical and electronic contact loss between the graphite particles themselves, between the particles and the current collector, between the binder and the particles, and between the binder and the current collector. This results in higher cell resistance as fewer pathways are available for electrons to flow through the electrode matrix. It can also lead to capacity loss if particles become completely disconnected electronically from the current collectors.

Porosity of the electrode can be reduced by volume changes and by the evolution of the SEI layer, which grows into the space normally occupied by the electrolyte. This impedes movement of lithium ions through the electrolyte and increases cell resistance.

If a cell becomes overdischarged, the open-circuit potential of its graphite material can increase to the point where copper in the negative-electrode current collector corrodes, releasing Cu^{2+} into the electrolyte. This has several consequences. First, there is reduced current-collector/electrode contact, which leads to higher cell resistance. Second, corrosion products that deposit on electrode particles have poor electronic conductivity, which increases SEI film resistance and hence overall cell resistance. Third, the corroded current collector has uneven resistance, which can lead to inhomogeneous current and potential distributions across the cell plate area, resulting in accelerated aging in parts of the cell and a preference toward lithium plating. Finally, copper plating on the negative-electrode particles also make metallic annealing sites that can accelerate lithium plating, dendrite growth, and hence short circuits.

[8] See, for example, Y-H Chen, C-W Wang, G. Liu, X-Y Song, V.S. Battaglia, and A.M. Sastry, "Selection of Conductive Additives in Li-Ion Battery Cathodes, A Numerical Study," *Journal of the Electrochemical Society*, 154(10), 2007, pp. A978–986.

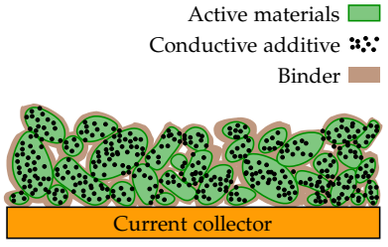

Figure 4.8: Composite electrode structure.

The negative-electrode degradation mechanisms are summarized in Table 4.1. The entries highlighted with a bold font are considered to be the more serious. In particular, we note that some

mechanisms primarily cause power fade while others primarily cause capacity fade. Therefore, changes in cell resistance and total capacity are not necessarily proportional, depending on exactly how the cell has aged.

Table 4.1: Principal aging mechanisms in the negative electrode (adapted from Table 1 in Vetter et al., "Ageing Mechanisms in Lithium-Ion Batteries," *Journal of Power Sources,* 147, 2005, 269–281)

Cause	Effect	Leads to	Enhanced by
Continuous low-rate electrolyte decomposition reaction builds SEI	Lithium loss, impedance rise	**Capacity fade, Power fade**	High temperatures, high cell SOC
Solvent co-intercalation, gas evolution and subsequent graphite exfoliation	Loss of active material, lithium loss	Capacity fade	Overcharge
Decrease of accessible surface area due to continuous SEI growth	Impedance rise	**Power fade**	High temperatures, high cell SOC
Changes in porosity due to volume change and SEI growth	Impedance rise, larger overpotentials	Power face	High cycling rate, high cell SOC
Contact loss of active material particles due to volume changes during cycling	Loss of active material	**Capacity fade**	High cycling rate, low cell SOC
Decomposition of binder	Lithium loss, loss of mechanical stability	Capacity fade	High cell SOC, high temperatures
Current collector corrosion	Larger overpotentials and impedance; Inhomogeneous distribution of current and potential	Power fade, Enhances other aging mechanisms	Overdischarge, low cell SOC
Metallic lithium plating and subsequent electrolyte decomposition by metallic lithium	Lithium loss (electrolyte loss)	Capacity fade (power fade)	Low temperature, high charge rates, geometric misfits

4.3 *Positive-electrode aging*

As with the negative electrode, aging occurs in three locations in the positive electrode: at the surface of the particles, within the active material particles themselves, and in the bulk positive electrode. We discuss these mechanisms in the next three subsections.

4.3.1 *Positive electrode aging at surface of particles*

In the positive electrode, researchers have found that a film can grow on the surface of the active-material particles as well. In part, this is due to a chemical reaction between the solvent in the electrolyte and the positive-electrode active materials; however, this mechanism is not as pronounced as it is in negative electrodes.

A bigger factor is the dissolution of metals from the electrode crystal structures into the electrolyte and products formed from these metals that can reprecipitate onto the particle surface as a high-resistance film. This dissolution is accelerated by hydrofluoric acid in the electrolyte, initiated by trace amounts of water that combine with the $LiPF_6$ salt.

Metal dissolution via acid attack is a primary cause of capacity loss for lithium-manganese-oxide cells as the loss of manganese destroys the crystal structure and eliminates lithium storage sites.[9] Lithium-cobalt-oxide cells also lose capacity due to cobalt loss, but at a slower rate. The actual mechanism depends on which oxide is used in the positive electrode but tends to happen predominantly at low or high cell states of charge and can be accelerated greatly by high temperature and by any HF acid that might be dissolved in the electrolyte.

A side effect of metal dissolution is that the ionized metals can migrate across the separator and poison the negative-electrode, as mentioned in Sect. 4.2.1. This increases cell resistance and lowers total capacity.

[9] See, for example, Dai, Y., Cai, L., and White, R.E., "Capacity Fade Model for Spinel $LiMn_2O_4$ Electrode," *Journal of the Electrochemical Society*, 160(1), 2013, A182–A190.

4.3.2 *Positive electrode aging in bulk*

When lithium intercalates into and deintercalates out of the positive-electrode active particles, stresses cause strains known as *phase transitions* that distort the shape of the crystal structure of the electrode materials without changing the overall structure itself. Transitions in phase are caused by the presence or absence of lithium in the storage sites, leading to different local molecular forces. Some of these phase transitions are normal and reversible, but others lead to collapse of the electrode structure and rapid capacity decrease due to the resulting loss of lithium storage sites. This is most common when

overcharging a cell: too much lithium is removed from the positive electrode, which causes the lithium pathways to collapse.

These cycling stresses can also lead to a phenomena known as *structural disordering* where the crystal structure of the electrode materials breaks down. Chemical bonds between atoms in the crystal are broken and then later reform to different atoms. This collapses the tunnel-like structures that allow lithium movement, which can cause lithium to become trapped within the crystal structure and also the loss of lithium storage sites. Both of these effects decrease the total capacity of the cell.

In some chemistries, phase transitions near the surface have been observed to lead to the formation of permanent subsurface layers that do not allow lithium to move as freely as in the unaltered crystal structure. This increases the resistance of the cell.

Finally, particles in lithium-iron-phosphate positive electrodes have been observed to grow over time as adjacent particles apparently sinter together. This reduces the total surface area of the electrode, and results in a higher cell resistance.

4.3.3 *Positive electrode aging in the composite electrode*

The composite positive electrode experiences degradation in the same ways as the composite negative electrode. Over time, the binder can decompose, the conductive additives can become oxidized, the current collector can become corroded, and contact among particles and between particles and the current collector can become lost due to volume changes.

The positive-electrode degradation mechanisms are summarized in Table 4.2. While the dominant degradation in the negative electrode tends to be due to side reactions that grow the SEI layer, the dominant mechanisms in the positive electrode tend to be those that result in material loss. Both result in decreased cell total capacity and increased cell resistance.

4.4 *Sensitivity of voltage to R_0*

Having discussed the dominant aging mechanisms in a lithium-ion cell from a qualitative point of view, we now turn our attention to the task of estimating the present state of health of the battery cell. Based on our observations to this point, we see that being able to estimate the cell's present total capacity and equivalent-series resistance will go a long way toward a good description of the cell's present health.

The task of estimating total capacity and resistance must some-

Cause	Effect	Leads to	Enhanced by
Phase transitions	Cracking of active particles	Capacity fade	High rates, high/low SOC
Structural disordering	Lithium sites lost and lithium trapped	Capacity fade	High rates, high/low SOC
Metal dissolution and/or electrolyte decomposition	Migration of soluble species,	Capacity fade	High/low SOC, high temperature
	Re-precipitation of new phases,	Power fade	
	Surface layer formation	Power fade	
Electrolyte decomposition	Gas evolution		High temperature
Binder decomposition	Loss of contact	Power fade	
Oxidation of conductive agent	Loss of contact	Power fade	
Corrosion of current collector	Loss of contact	Power fade	High SOC

Table 4.2: Principal aging mechanisms at positive electrode.

how use input/output (current/voltage) data from the cell.[10] As a measure of how readily a good estimate may be made, we can then consider the sensitivity of the cell voltage signal to changes in resistance and capacity. This will give us a feel for how easy or difficult it is to estimate these quantities and will reveal that voltage is very sensitive to a change in resistance but very insensitive to a change in total capacity. This result will motivate a simple way to estimate cell resistance; however, we will need to investigate more complex methods in order to make good total-capacity estimates.

Estimating a cell's equivalent series resistance turns out to be relatively simple because it is highly observable from voltage measurements. To see this, consider the cell's voltage equation,

$$v_k = \mathrm{OCV}(z_k) + Mh_k + M_0 s_k - \sum_i R_i i_{R_i,k} - i_k R_0.$$

We define the unitless *sensitivity* of the voltage measurement to a change in resistance as

$$S_{v_k}^{R_0} = \frac{R_0}{v_k} \frac{\mathrm{d}v_k}{\mathrm{d}R_0}$$

$$= \frac{-R_0}{v_k} i_k.$$

A cell's resistance is usually on the order of milliohms, and its voltage is usually a few volts. When multiplied by cell current i_k, which can be large, the absolute sensitivity of voltage to R_0 is relatively high (a magnitude of 0.01 would not be out of the ordinary). This implies that it should be fairly simple to estimate resistance from the voltage signal.

One approach to estimating R_0 is to subtract voltages at two adjacent time samples

$$\begin{aligned} v_k &= \text{OCV}(z_k) + Mh_k + M_0 s_k - \textstyle\sum_i R_i i_{R_i,k} - i_k R_0 \\ v_{k-1} &= \text{OCV}(z_{k-1}) + Mh_{k-1} + M_0 s_{k-1} - \textstyle\sum_i R_i i_{R_i,k-1} - i_{k-1} R_0 \\ \hline v_k - v_{k-1} &\approx R_0 (i_{k-1} - i_k) + M_0 (s_k - s_{k-1}), \end{aligned}$$

where we use our knowledge that that cell state of charge z_k, diffusion currents $i_{R_i,k}$, and analog hysteresis h_k change relatively slowly compared to how quickly i_k changes.

So, we can estimate

$$\widehat{R}_{0,k} = \frac{(v_k - v_{k-1}) - M_0 (s_k - s_{k-1})}{i_{k-1} - i_k}. \tag{4.1}$$

However, when using this method we run into an immediate divide-by-zero problem when $\Delta i_k = i_{k-1} - i_k = 0$, as would happen during a constant-current event or during cell rest. So, we skip updates to $\widehat{R}_{0,k}$ when $|\Delta i_k|$ is small, which eliminates the divide-by-zero problem and avoids amplification of measurement and approximation noise in Eq. (4.1) as well.

Because of the imperfect fidelity of the ESC cell model with respect to the true cell behavior and because of the inaccuracy introduced via our specific approximations, the estimate of $\widehat{R}_{0,k}$ from Eq. (4.1) is quite noisy. To make a better estimate of R_0, we can filter the $\widehat{R}_{0,k}$ signal. For example, we might implement the one-pole digital filter

$$\widehat{R}^{\text{filt}}_{0,k} = \alpha \widehat{R}^{\text{filt}}_{0,k-1} + (1 - \alpha) \widehat{R}_{0,k}, \tag{4.2}$$

where $0 \ll \alpha < 1$. This method, while very simple, tends to work quite well. Later in this chapter we will see a second method, which is based on nonlinear Kalman filtering, that also works well.

Another factor that merits consideration is that a cell's resistance is both SOC dependent and temperature dependent. If a scalar resistance estimate is updated, it will adapt to model the cell resistance at the present SOC and temperature as both of those change over time. If an adaptive function that describes the entire relationship of resistance versus state of charge is required instead of an adaptive scalar, then a functional form will need to be proposed and the coefficients of the form will need to be adapted based on the present

[10] Here, we focus on passive methods that simply monitor current and voltage and perform computations using these measurements but do not inject signals into the cell. It is also possible to modulate the input current to a cell purposefully using sinusoidal or pulse signals to measure indicators of health more directly. If the battery pack is permanently connected to the load, the passive approach tends to be preferred since the battery state is not altered by the health measurement, but if the battery pack can be disconnected from the load for diagnostics (even for a short period of time), the latter approach is also a possibility. Note, however, that the sinusoidal response of a battery can be computed from the discrete Fourier transform of passively measured data so long as the cell is *persistently excited*, and the pulse responses can be measured opportunistically when balancing circuits are activated and deactivated, so passive approaches are not as limited as it might appear at first.

state of charge, temperature, and operating resistance. For example, temperature dependence can often be well modeled using an Arrhenius-inspired relationship

$$R_0 = R_{0,\text{ref}} \exp\left(\frac{E_{R_0,\text{ref}}}{R}\left(\frac{1}{T_{\text{ref}}} - \frac{1}{T}\right)\right),$$

where only the $R_{0,\text{ref}}$ and $E_{R_0,\text{ref}}$ values need to be adapted online to fix the temperature dependence. However, care must be taken in crafting the adaptation algorithm if the battery pack dwells near one temperature for an extended period of time, because adaptation can overcorrect the predictions at that temperature and cause predictions for resistance at other temperatures to become biased.

Temperature and SOC dependence can also be handled by modeling resistance using local basis functions. Adaptation of the parameters of these basis functions will cause only local updates to the resistance relationship and will not bias predictions of resistance at other temperature and state of charge operating points. However, operational points that are not often visited by the application will not have their resistance updated frequently and the estimates will tend to become out of date over time.

4.5 *Code to estimate R_0*

In this section, we see how to implement the simple equivalent-series-resistance estimator we have just seen in MATLAB. In the code, we first load a datafile comprising voltage and current data measured from a cell in the laboratory. This particular cell has a known $R_0 = 2.53\,\text{m}\Omega$ from the system identification procedure discussed in Chap. 2. It is our goal to estimate this value of R_0 using the simple estimator.

```
%% Initialize "truth" values
R0 = 2.53e-3; % from external system ID of cell
R0_Vect = R0*ones(1,length(voltage));

%% Load data file
load('Resistance_data.mat');
```

For the example we consider here, the profiles of current and voltage versus time are plotted in Fig. 4.9.

Next, we compute and plot the unfiltered resistance estimate. The threshold on $|\Delta i_k|$ was chosen to be 16.5 A in this example, which is approximately equal to the 2C rate for this cell. The code simply loops through all data, evaluating Eq. (4.1) for each time step and updating the resistance estimate if the threshold criterion is met:

```
%% Estimate R0 - unfiltered
threshold = 16.5;   % Define threshold
```

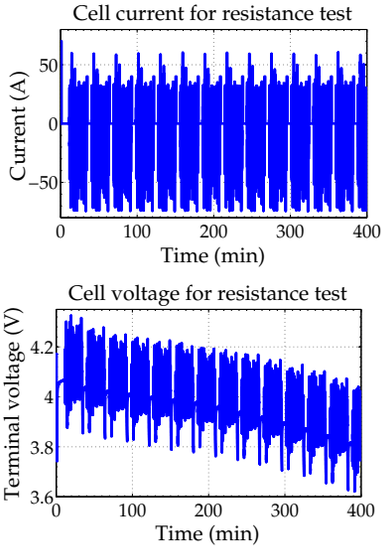

Figure 4.9: Current and voltage data used to estimate cell resistance.

```
R0_hat = 0*R0_Vect; % Reserve memory for estimate

for k = 2:length(v)
  num = voltage(k) - voltage(k-1);
  den = current(k-1) - current(k);
  R0_hat(k) = num / den;
  if abs(den < threshold),
    R0_hat(k) = R0_hat(k-1);
  end
end

% Plot unfiltered R0 estimate
figure(1); plot(time/60,1000*R0_hat,time/60,1000*R0_Vect);
title('R0 estimate (unfiltered)'); grid on;
xlabel('Time (min)'); ylabel('R0 values (m\Omega)'); ylim([0 4]);
legend('Estimate','Actual','location','southeast');
```

Next, we filter the estimate using $\alpha = 0.999$. The code loops through all raw estimates, applies Eq. (4.2) to each point, and plots the final results:

```
%% Estimate R0 - filtered version
alpha = 0.999;        % define filter pole
R0_hat_filt = R0_hat; % initialize filtered estimate

for k = 2:length(voltage)
  R0_hat_filt(k) = alpha*R0_hat_filt(k-1) + (1-alpha)*R0_hat(k);
end

% Plot filtered R0 estimate
figure(2); plot(time/60,1000*R0_hat_filt,time/60,1000*R0_Vect);
title('R0 estimate (filtered)'); grid on;
xlabel('time (min)'); ylabel('R0 values (m\Omega)'); ylim([0 4]);
legend('Estimate','Actual','location','southeast');
```

Fig. 4.10 shows some results from running this program. The top-left frame compares the unfiltered estimate at every time step to the true value. We see that the estimate has reasonable average value, but that it is very noisy. The top-right frame shows a zoom into one portion of the data segment to highlight some detail. In particular, the places where the estimate flattens out are the result of the $|\Delta i_k|$ threshold criterion not being met, which causes the estimates to retain their prior values.

The lower-left frame shows the filtered version of the estimate versus the truth. Because the filtered estimate is initialized to zero, it takes some time to converge to the neighborhood of the true value of resistance. A better initial guess of the true resistance will lead to faster convergence of the filtered estimate. The lower-right frame shows the unfiltered and filtered estimates compared to each other. We see that the filter has greatly reduced the amount of noise in the estimate. After convergence, the estimate stays within about 7.5 % of the true value, which is very nice performance for such a simple algorithm.

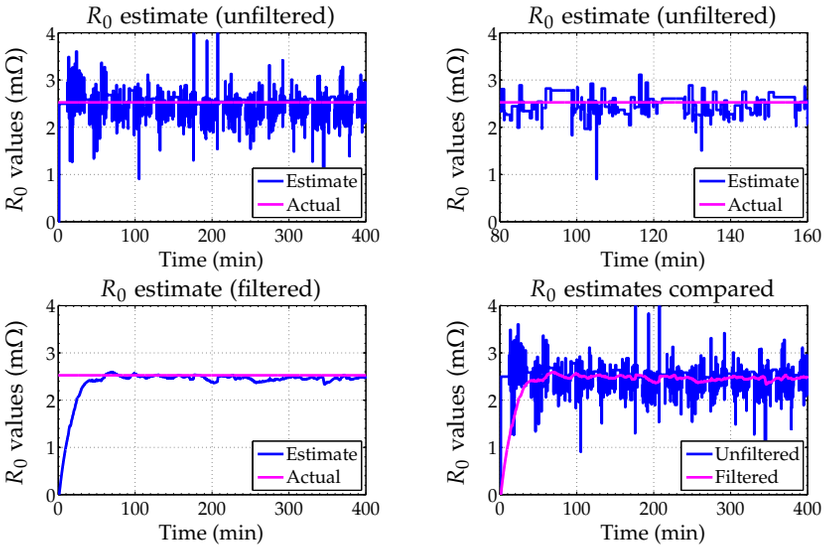

Figure 4.10: Intermediate and final resistance estimates.

4.6 *Sensitivity of voltage to total capacity Q*

Estimating total capacity well turns out to be very difficult. To see why, consider the sensitivity of the voltage measurement to a cell's total capacity

$$\begin{aligned} S_{v_k}^{Q} &= \frac{Q}{v_k}\frac{\mathrm{d}v_k}{\mathrm{d}Q} \\ &= \frac{Q}{v_k}\frac{\mathrm{d}}{\mathrm{d}Q}\left(\mathrm{OCV}(z_k) + Mh_k + M_0 s_k - \sum_i R_i i_{R_i,k} - i_k R_0\right). \end{aligned}$$

While Q/v_k is a reasonably large value, the term inside the parenthesis does not appear to contain capacity at all! To uncover the voltage's sensitivity to total capacity, we need to unfold the total derivatives.

Consider the first term, which we evaluate using the chain rule of total derivatives to be

$$\frac{\mathrm{dOCV}(z_k)}{\mathrm{d}Q} = \frac{\partial \mathrm{OCV}(z_k)}{\partial z_k}\frac{\mathrm{d}z_k}{\mathrm{d}Q}.$$

For most cells, the slope of the open-circuit-voltage curve is very shallow, so $\partial \mathrm{OCV}(z_k)/\partial z_k$ is very small. Further, we expand the total derivative of the SOC equation:

$$\begin{aligned} \frac{\mathrm{d}z_k}{\mathrm{d}Q} &= \frac{\mathrm{d}z_{k-1}}{\mathrm{d}Q} - \eta_{k-1} i_{k-1} \Delta t \frac{\mathrm{d}(1/Q)}{\mathrm{d}Q} \\ &= \frac{\mathrm{d}z_{k-1}}{\mathrm{d}Q} + \frac{\eta_{k-1} i_{k-1} \Delta t}{Q^2}. \end{aligned}$$

The first term on the right-hand side of this result can be calculated recursively. It grows in magnitude when i_k has the same sign

for a considerable number of consecutive iterations and shrinks when i_k changes direction. For random i_k (e.g., for a hybrid-electric-vehicle application) it tends to be about the same order of magnitude as the second term on the right-hand side.

The second term has Q^2 in the denominator. Since Q must be expressed in coulombs for its units to agree with the other terms of the equation, Q^2 is a very large number, making the second term very small. So, $\mathrm{d}z_k/\mathrm{d}Q$ is small and overall sensitivity of voltage to capacity through the OCV term is very small.

Similarly, the sensitivity of the voltage to capacity through the analog hysteresis term is small (it is zero through the other terms). As a consequence, individual voltage measurements have very little information regarding capacity. We must somehow combine many voltage measurements to estimate total capacity well, and it helps if current has been generally in the same direction for most of the period spanning those measurements. That is, a large change in SOC over an interval makes it easier to estimate total capacity using the data collected during that interval than does a small change in SOC.

Generally speaking, simple methods like the one used to estimate $\widehat{R}_0$ will not work well. So, in the remainder of this chapter, we explore two different and more complicated approaches. First, we look at using a nonlinear Kalman filter, which can work well, and which has the additional benefit of being able to estimate all cell-model parameter values simultaneously as they change over time. However, these Kalman-filter-based approaches are relatively complex and there are some algorithm stability issues that must be addressed before the estimates can be trusted. Simpler regression techniques can be used to estimate total capacity, but the most straightforward ordinary least-squares approach gives biased results. So, we will also spend considerable time exploring a total-least-squares approach, which uses the noisy input measurements in an optimal way to produce unbiased total-capacity estimates.

4.7 *Estimating parameters via Kalman filters*

In Chap. 3, we saw that a Kalman filter can be used to estimate the state of a dynamic system using noisy input/output measurements if the model parameter values are known precisely. It turns out that it is also possible to use a nonlinear Kalman filter to estimate the parameters of a dynamic system's model using noisy input/output measurements if its state is known precisely. Because the SOH of a battery cell is summarized by the cell-model parameter values, this approach promises to be very useful.

Unfortunately, we do not know the state precisely. However, if we

are careful, we can also use a nonlinear Kalman filter to estimate *both* the state and the parameters of the system model simultaneously using noisy input/output measurements.

In this section, we proceed by first considering how to estimate the parameter values of a system if its state is known. Then, we combine the state-estimation and parameter-estimation methods to show how to estimate both state and parameter simultaneously.[11]

[11] The EKF presentation is based on Plett, G., "Extended Kalman Filtering for Battery Management Systems of LiPB-Based HEV Battery Packs—Part 3: State and Parameter Estimation," *Journal of Power Sources*, 134(2), 2004, pp. 277–92, and the SPKF presentation is based on Plett, G., "Sigma-Point Kalman Filters for Battery Management Systems of LiPB-Based HEV Battery Packs—Part 2: Simultaneous State and Parameter Estimation," *Journal of Power Sources*, 161(2), 2006, pp. 1369–84.

4.7.1 *A generic approach to parameter estimation*

We begin by collecting the true parameters of a particular model into the vector $\boldsymbol{\theta}$. We will use Kalman-filtering techniques to estimate the parameter values as we did to estimate the state-vector values. Therefore, we require a discrete-time state-space model of the dynamics of the parameters.

By assumption, parameter values change very slowly, so we model the present parameter vector as being equal to the prior parameter vector plus some small perturbation $\boldsymbol{r}_k$

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1}. \tag{4.3}$$

The small zero-mean white-noise input $\boldsymbol{r}_k$ is fictitious, and is used in Eq. (4.3) to model a forcing input for the slow drift in the parameter values of the system. We do not ever synthesize this noise as an input to the Kalman filter; instead, the covariance matrix $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$ is used to indicate how quickly we believe the parameter values can change, which causes the Kalman filter to adjust how quickly it updates parameter value estimates.

The model of Eq. (4.3) is known as a *random walk*. In the absence of feedback, it describes a path for the parameter values that comprises a sequence of purely random steps, where the variance—and hence the uncertainty—of $\boldsymbol{\theta}_k$ increases linearly over time. In the model, the parameter values are just as likely to increase as they are to decrease in value from one time step to another.

Based on the qualitative understanding of cell degradation that we have gained so far in this chapter, we recognize that the random walk is probably not a good description of how cell parameter values actually change. In a random walk, a parameter value is just as likely to increase as it is to decrease. However, if we make a bold generalization, we can state that a real cell's total capacity only decreases with time; it does not increase.[12] Similarly, its resistance only increases with time; it does not decrease.[13]

While the random walk is not an accurate open-loop model for how parameter values change, to model the changes more descriptively would require a detailed quantitative description of how the

[12] Over the first few cycles of a lithium-ion battery cell, the total capacity often *does* increase, as the electrolyte seeps into voids it had not properly wet during cell construction. This creates ionic connection to parts of the cell that were previously isolated, and adds their capacity to the total. Also, if a cell has been stored idle for a long period of time, some researchers have noted that subsequent cycling will recover some lost capacity although the mechanism for doing so is not clear.

[13] Again, resistance has been observed to decrease for the first few cycles of a lithium-ion cell before it begins to increase. This is thought to be caused by the formation of micro-cracks on the surface of the active particles due to intercalation stresses exciting internal defects in the new active materials. These cracks increase the surface area of the graphite that is exposed, decreasing resistance. However, the micro-cracks stop forming after the initial cycles, and the general trend of cell resistance over time is in the increasing direction.

degradation mechanisms actually work. There is very little literature on this subject, although we do look at two such models in Chap. 7. It turns out that even though Eq. (4.3) is not a very accurate description of the details of cell degradation, the feedback mechanism of the Kalman filter will correct for the modeling errors and still make good estimates of the model's parameter values.

To incorporate feedback, we require a model output equation, which must be a measurable function of the system parameters. We use

$$\boldsymbol{d}_k = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k), \tag{4.4}$$

where $\boldsymbol{g}(\cdot)$ is the output equation of the system model being identified, and $\boldsymbol{e}_k$ models the sensor noise and modeling error.[14] Then, the sequence of all measurements made for parameter identification can be written as $\mathbb{D}_k = \{\boldsymbol{d}_0, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_k\}$.

[14] Note that $\boldsymbol{d}_k$ is usually the same measurement as $\boldsymbol{y}_k$, but we maintain a distinction here in case separate outputs are used. Also, note that the noises $\boldsymbol{e}_k$ in Eq. (4.4) and $\boldsymbol{v}_k$ in Eq. (4.6) often play the same role, but are considered distinct here.

We also slightly revise the mathematical model of system state-vector dynamics and measurement relationship to be

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{\theta}, \boldsymbol{w}_{k-1}) \tag{4.5}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{v}_k), \tag{4.6}$$

which includes the parameters $\boldsymbol{\theta}$ in the model explicitly. Time-invariant numeric values required by the model may be embedded within $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ and are not included in $\boldsymbol{\theta}$.

4.8 EKF parameter estimation

With the new parameter-dynamics equation Eq. (4.3), parameter-output equation Eq. (4.4), revised state equation Eq. (4.5), and model-output equation Eq. (4.6), we are now ready to see how to perform parameter estimation using a nonlinear Kalman filter.

We proceed by first deriving the extended Kalman filter for parameter estimation following the same six steps of sequential-probabilistic inference introduced in Chap. 3.

EKF step 1a: Parameter prediction time update.

The parameter prediction step finds the expected value of the parameter update equation Eq. (4.3), which is

$$\begin{aligned}\hat{\boldsymbol{\theta}}_k^- &= \mathbb{E}[\boldsymbol{\theta}_k \mid \mathbb{D}_{k-1}] \\ &= \mathbb{E}[\boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1} \mid \mathbb{D}_{k-1}] \\ &= \hat{\boldsymbol{\theta}}_{k-1}^+,\end{aligned} \tag{4.7}$$

since we have assumed that the fictitious noise $\boldsymbol{r}_k$ is zero mean. That is, the predicted parameter vector for this time step is equal to the

estimated parameter vector computed at the end of the prior time step. This result makes sense because the parameters are assumed to be essentially constant, with very slow variation due to cell aging. We do see, however, that the changes in parameter values due to degradation will be captured only via the measurement feedback because the prediction step in Eq. (4.7) does not forecast any degradation in their values.

EKF step 1b: Error covariance time update.

The covariance matrix of the parameter prediction error is found by first computing $\tilde{\boldsymbol{\theta}}_k^-$, as

$$\begin{aligned}\tilde{\boldsymbol{\theta}}_k^- &= \boldsymbol{\theta}_k - \hat{\boldsymbol{\theta}}_k^- \\ &= \boldsymbol{\theta}_{k-1} + \boldsymbol{r}_k - \hat{\boldsymbol{\theta}}_{k-1}^+ \\ &= \tilde{\boldsymbol{\theta}}_{k-1}^+ + \boldsymbol{r}_k.\end{aligned}$$

We then compute the desired covariance directly as

$$\begin{aligned}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- &= \mathbb{E}[\tilde{\boldsymbol{\theta}}_k^- (\tilde{\boldsymbol{\theta}}_k^-)^T] \\ &= \mathbb{E}[(\tilde{\boldsymbol{\theta}}_{k-1}^+ + \boldsymbol{r}_k)(\tilde{\boldsymbol{\theta}}_{k-1}^+ + \boldsymbol{r}_k)^T] \\ &= \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k-1}^+ + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}},\end{aligned}$$

where we have assumed that the zero-mean fictitious noise is uncorrelated with the parameter prediction error. Therefore, mathematically, the time-updated parameter-error covariance is equal to the covariance prior to the time update, but having additional uncertainty due to the fictitious noise that is assumed to be driving the change in parameter values. Physically, this corresponds to the increasing uncertainty in parameter values over time due to aging unless we can somehow adapt the parameter values via feedback, which is what we will do in step 2b.

EKF step 1c: Output prediction.

The measurable system output based on the parameter model is predicted using Eq. (4.4) and EKF assumption 1 to be

$$\begin{aligned}\hat{\boldsymbol{d}}_k &= \mathbb{E}[\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k) \mid \mathbb{D}_{k-1}] \\ &\approx \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \hat{\boldsymbol{\theta}}_k^-, \bar{\boldsymbol{e}}_k).\end{aligned}$$

That is, we assume that a good way to approximate the mean of the parameter output equation is by propagating the expected parameter vector $\hat{\boldsymbol{\theta}}_k^-$ and the mean estimation error through the parameter output equation.

EKF step 2a: Estimator gain matrix.

The output prediction error may then be written as

$$\begin{aligned}\tilde{\boldsymbol{d}}_k &= \boldsymbol{d}_k - \hat{\boldsymbol{d}}_k \\ &= \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k) - \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \hat{\boldsymbol{\theta}}_k^-, \bar{\boldsymbol{e}}_k).\end{aligned}$$

We employ EKF assumption 2 to linearize this relationship using a Taylor-series expansion of the first term on the left-hand side of the equation:

$$\begin{aligned}\boldsymbol{d}_k \approx \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \hat{\boldsymbol{\theta}}_k^-, \bar{\boldsymbol{e}}_k) &+ \underbrace{\left.\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k^-}}_{\text{Defined as } \hat{\boldsymbol{C}}_k^\theta} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_k^-) \\ &+ \underbrace{\left.\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\mathrm{d}\boldsymbol{e}_k}\right|_{\boldsymbol{e}_k=\bar{\boldsymbol{e}}_k}}_{\text{Defined as } \hat{\boldsymbol{D}}_k^\theta} (\boldsymbol{e}_k - \bar{\boldsymbol{e}}_k).\end{aligned}$$

From this, we can compute such necessary quantities as:

$$\begin{aligned}\boldsymbol{\Sigma}_{\tilde{d},k} &\approx \hat{\boldsymbol{C}}_k^\theta \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- (\hat{\boldsymbol{C}}_k^\theta)^T + \hat{\boldsymbol{D}}_k^\theta \boldsymbol{\Sigma}_{\tilde{e}} (\hat{\boldsymbol{D}}_k^\theta)^T \\ \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}\tilde{d},k}^- &\approx \mathbb{E}[(\tilde{\boldsymbol{\theta}}_k^-)(\hat{\boldsymbol{C}}_k^\theta \tilde{\boldsymbol{\theta}}_k^- + \hat{\boldsymbol{D}}_k^\theta \tilde{\boldsymbol{e}}_k)^T] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- (\hat{\boldsymbol{C}}_k^\theta)^T.\end{aligned}$$

These terms may be combined to compute the Kalman gain

$$\boldsymbol{L}_k^\theta = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- (\hat{\boldsymbol{C}}_k^\theta)^T \left[\hat{\boldsymbol{C}}_k^\theta \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- (\hat{\boldsymbol{C}}_k^\theta)^T + \hat{\boldsymbol{D}}_k^\theta \boldsymbol{\Sigma}_{\tilde{e}} (\hat{\boldsymbol{D}}_k^\theta)^T\right]^{-1}.$$

Note the superscript θ used in the notation for $\hat{\boldsymbol{C}}_k^\theta$, $\hat{\boldsymbol{D}}_k^\theta$, and $\boldsymbol{L}_k^\theta$, which is added to keep these EKF matrices for parameter estimation distinct from their counterparts for state estimation. This will be important later when we estimate both states and parameters simultaneously.

We must be very careful when computing $\hat{\boldsymbol{C}}_k^\theta$. By the chain rule of total differentials, we have

$$\begin{aligned}\mathrm{d}\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k) = &\frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{x}_k} \mathrm{d}\boldsymbol{x}_k + \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{u}_k} \mathrm{d}\boldsymbol{u}_k \\ &+ \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{\theta}} \mathrm{d}\boldsymbol{\theta} + \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{e}_k} \mathrm{d}\boldsymbol{e}_k.\end{aligned}$$

Dividing both sides of this equation by $\mathrm{d}\boldsymbol{\theta}$, we find

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\mathrm{d}\boldsymbol{\theta}} &= \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{x}_k} \frac{\mathrm{d}\boldsymbol{x}_k}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{u}_k} \underbrace{\frac{\mathrm{d}\boldsymbol{u}_k}{\mathrm{d}\boldsymbol{\theta}}}_{0} \\ &\quad + \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{\theta}} \frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}\boldsymbol{\theta}} + \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{e}_k} \underbrace{\frac{\mathrm{d}\boldsymbol{e}_k}{\mathrm{d}\boldsymbol{\theta}}}_{0} \\ &= \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k)}{\partial \boldsymbol{x}_k} \frac{\mathrm{d}\boldsymbol{x}_k}{\mathrm{d}\boldsymbol{\theta}}.\end{aligned}$$

As with state estimation, some terms drop out since the deterministic cell input current is not a function of cell parameter values, nor is the measurement error $\boldsymbol{e}_k$. However, by Eq. (4.5), the model state *is* a function of the parameter values and so we must include the $\mathrm{d}\boldsymbol{x}_k/\mathrm{d}\boldsymbol{\theta}$ term in the expansion. This is different from what we found for state-vector estimation in Chap. 3, and is the reason why we had a discussion of total versus partial derivatives at the corresponding point in the derivation of the EKF there. Here, we absolutely must use the total derivative instead of the partial derivative, or else the parameter estimator will not work!

We evaluate this new term recursively.

$$\frac{\mathrm{d}\boldsymbol{x}_k}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{\theta}, \boldsymbol{w}_{k-1})}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{\theta}, \boldsymbol{w}_{k-1})}{\partial \boldsymbol{x}_{k-1}} \frac{\mathrm{d}\boldsymbol{x}_{k-1}}{\mathrm{d}\boldsymbol{\theta}}.$$

We see that $\mathrm{d}\boldsymbol{x}_k/\mathrm{d}\boldsymbol{\theta}$ is a function of $\mathrm{d}\boldsymbol{x}_{k-1}/\mathrm{d}\boldsymbol{\theta}$, and so this relationship evolves over time as the state evolves. The term $\mathrm{d}\boldsymbol{x}_0/\mathrm{d}\boldsymbol{\theta}$ is initialized to zero unless side information gives a better estimate of its value.

In summary, in order to calculate $\hat{\boldsymbol{C}}_k^{\theta}$ for any specific model structure, we require methods to calculate all of the above partial derivatives for the model to find the needed total derivatives.

EKF step 2b: Parameter estimate measurement update.

The parameter estimate is computed by updating the prediction using the estimator gain and the output prediction error $\boldsymbol{d}_k - \hat{\boldsymbol{d}}_k$:

$$\hat{\boldsymbol{\theta}}_k^+ = \hat{\boldsymbol{\theta}}_k^- + \boldsymbol{L}_k^{\theta}(\boldsymbol{d}_k - \hat{\boldsymbol{d}}_k).$$

This step is unchanged in principle from the standard EKF.

EKF step 2c: Error covariance measurement update.

Finally, the updated parameter estimation-error covariance is computed as

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^+ = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- - \boldsymbol{L}_k^{\theta}\boldsymbol{\Sigma}_{\tilde{d},k}(\boldsymbol{L}_k^{\theta})^T.$$

EKF for parameter estimation is summarized in the Appendix on p. 229. When starting the algorithm for the first time, we initialize the parameter-vector estimate with our best information regarding its value, $\hat{\boldsymbol{\theta}}_0^+ = \mathbb{E}[\boldsymbol{\theta}_0]$, we initialize the parameter estimation error covariance matrix with our best information regarding our uncertainty in the parameter-vector estimate

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},0}^+ = \mathbb{E}\big[(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_0^+)(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}_0^+)^T\big],$$

and we initialize $\mathrm{d}\boldsymbol{x}_0/\mathrm{d}\boldsymbol{\theta} = 0$ unless side information is available.

4.9 SPKF parameter estimation

Given the time we invested in Chap. 3 studying sigma-point methods, it is also relatively straightforward to show how to estimate parameter values using a sigma-point Kalman filter. As always, we proceed by deriving the six essential steps of sequential probabilistic inference.

SPKF step 1a: Parameter prediction time update.

The parameter prediction step is approximated as

$$\begin{aligned}\hat{\boldsymbol{\theta}}_k^- &= \mathbb{E}[\boldsymbol{\theta}_k \mid \mathbb{D}_{k-1}] \\ &= \mathbb{E}[\boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1} \mid \mathbb{D}_{k-1}] \\ &= \hat{\boldsymbol{\theta}}_{k-1}^+.\end{aligned}$$

This outcome is the same as with EKF since the dynamics of the parameters follow a linear model.

SPKF step 1b: Error covariance time update.

Again, because of the linearity of the parameter dynamics, the error-covariance update equation is the same as for EKF.

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k-1}^+ + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}.$$

SPKF step 1c: Predict system output $\boldsymbol{d}_k$.

To predict the system output, we first define the augmented random vector

$$\boldsymbol{\theta}_k^a = \begin{bmatrix} \boldsymbol{\theta}_k \\ \boldsymbol{e}_k \end{bmatrix},$$

which combines the randomness of the parameter estimates and the sensor noise. We then compute a set of $p+1$ sigma points describing this augmented random vector, which we will denote as $\boldsymbol{\mathcal{W}}_k^{a,-}$:

$$\boldsymbol{\mathcal{W}}_k^{a,-} = \left\{ \hat{\boldsymbol{\theta}}_k^{a,-}, \hat{\boldsymbol{\theta}}_k^{a,-} + \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^{a,-}}, \hat{\boldsymbol{\theta}}_k^{a,-} - \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^{a,-}} \right\}.$$

From the augmented sigma points, the $p+1$ vectors comprising the parameters portion $\boldsymbol{\mathcal{W}}_k^{\theta,-}$ and the $p+1$ vectors comprising the modeling error portion $\boldsymbol{\mathcal{W}}_k^{e,-}$ are extracted.

The output equation Eq. (4.4) is evaluated using all pairs of $\boldsymbol{\mathcal{W}}_{k,i}^{\theta,-}$ and $\boldsymbol{\mathcal{W}}_{k,i}^{e,-}$ (where the subscript i denotes that the ith vector is being extracted from the original set), yielding the sigma points $\boldsymbol{\mathcal{D}}_{k,i}$ for time step k. That is,

$$\boldsymbol{\mathcal{D}}_{k,i} = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{W}}_{k,i}^{\theta,-}, \boldsymbol{\mathcal{W}}_{k,i}^{e,-}).$$

Finally, the output prediction is computed as

$$\begin{aligned}\hat{\boldsymbol{d}}_k^- &= \mathbb{E}\big[\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}, \boldsymbol{e}_k) \mid \mathbb{D}_{k-1}\big] \\ &\approx \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{W}}_{k,i}^{\theta,-}, \boldsymbol{\mathcal{W}}_{k,i}^{e,-}) \\ &= \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{D}}_{k,i}.\end{aligned}$$

SPKF step 2a: Estimator gain matrix $\boldsymbol{L}_k^{\theta}$.

To compute the estimator gain matrix, we must first compute the required covariance matrices.

$$\begin{aligned}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k} &= \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \big(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\big)\big(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\big)^T \\ \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}\tilde{\boldsymbol{d}},k}^{-} &= \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \big(\boldsymbol{\mathcal{W}}_{k,i}^{\theta,-} - \hat{\boldsymbol{\theta}}_k^{a,-}\big)\big(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\big)^T.\end{aligned}$$

Then, we simply compute $\boldsymbol{L}_k^{\theta} = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}\tilde{\boldsymbol{d}},k}^{-} \boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k}^{-1}$.

SPKF step 2b: Parameter estimate measurement update.

The parameter estimate is created from the predicted parameters and the measurement innovation using the equation from the optimal formulation

$$\hat{\boldsymbol{\theta}}_k^{+} = \hat{\boldsymbol{\theta}}_k^{-} + \boldsymbol{L}_k^{\theta}(\boldsymbol{d}_k - \hat{\boldsymbol{d}}_k).$$

SPKF step 2c: Error covariance measurement update.

Finally, the covariance of the parameter estimation error is computed as

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^{+} = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^{-} - \boldsymbol{L}_k^{\theta} \boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k} (\boldsymbol{L}_k^{\theta})^T.$$

SPKF for parameter estimation is summarized in the Appendix on p. 230.

4.10 *Joint and dual estimation*

Having seen how to use Kalman filters to perform state estimation and parameter estimation separately, we now turn our attention to using nonlinear Kalman filters to perform the task of estimating both the state and parameter vectors at the same time if the state-filter output $\boldsymbol{y}_k$ is the same as the parameter-filter output $\boldsymbol{d}_k$. There are two approaches to doing so: *joint estimation* and *dual estimation*. These are discussed in the next sections.

4.10.1 Generic joint estimation

In joint estimation, the states and parameters are combined into a single high-dimension vector, and a nonlinear Kalman filter simultaneously estimates the values within this augmented state vector. Doing so has the disadvantages of large matrix operations due to the high dimensionality of the resulting augmented model and potentially poor numeric conditioning due to the vastly different time scales of the states and parameters in the augmented state vector. However, it is quite straightforward to implement.

To see how to do so, we first combine the state and parameter vectors to form augmented dynamics

$$\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{\theta}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{\theta}_{k-1}, \boldsymbol{w}_{k-1}) \\ \boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1} \end{bmatrix}$$
$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_k, \boldsymbol{v}_k).$$

To simplify notation, let $\mathbf{X}_k$ be the augmented state, $\mathbf{W}_k$ be the augmented noise, and $\mathbf{F}(\cdot)$ be the augmented state equations. Then,

$$\mathbf{X}_k = \mathbf{F}(\mathbf{X}_{k-1}, \boldsymbol{u}_{k-1}, \mathbf{W}_{k-1})$$
$$\boldsymbol{y}_k = \boldsymbol{h}(\mathbf{X}_k, \boldsymbol{u}_k, \boldsymbol{v}_k).$$

With this augmented discrete-time state-space model of the system state dynamics and parameter dynamics, we simply apply a nonlinear Kalman-filter method.

4.10.2 Generic dual estimation

In dual estimation, separate nonlinear Kalman filters are used for state estimation and parameter estimation. The computational complexity is smaller than for joint estimation because smaller matrices are involved in the computations and the matrix operations may be better conditioned numerically. However, by decoupling state from parameters any cross-correlations between the two are lost, leading to the possibility of poorer estimation accuracy.

The mathematical model of state dynamics again explicitly includes the parameters as the vector $\boldsymbol{\theta}_k$

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}, \boldsymbol{\theta}_{k-1})$$
$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{\theta}_{k-1}).$$

Time-invariant numeric values required by the model may be embedded within $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ and are not included in $\boldsymbol{\theta}_k$. We also slightly revise the mathematical model of parameter dynamics to include the

effect of the state equation explicitly.

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1}$$
$$\boldsymbol{d}_k = \boldsymbol{h}\left(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \bar{\boldsymbol{w}}_{k-1}, \boldsymbol{\theta}_{k-1}), \boldsymbol{u}_k, \boldsymbol{e}_k, \boldsymbol{\theta}_{k-1} \right).$$

The interactions between the two Kalman filters are illustrated in Fig. 4.11, where the details will become clearer in the following subsections. We see from the diagram that the dual-estimation process essentially comprises two Kalman filters running in parallel—with KF_x adapting the state and KF_θ adapting parameters—with some information exchange between the filters.

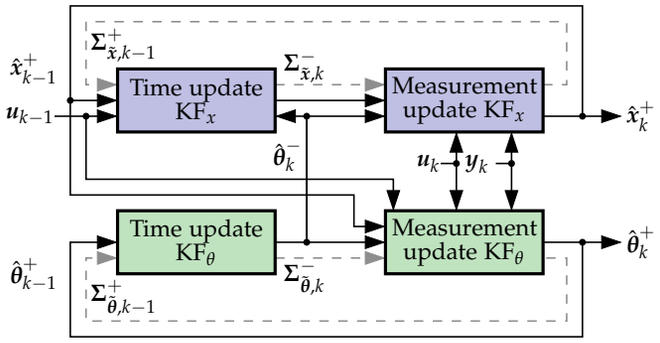

Figure 4.11: Interactions between the two nonlinear Kalman filters used for dual estimation.
(Reproduced from Fig. 8 of Plett, G.L., "Extended Kalman Filtering for Battery Management Systems of LiPB-Based HEV Battery Packs—Part 3: State and Parameter Estimation," *Journal of Power Sources*, 134(2), 2004, pp. 277–92.)

With generic joint and dual estimation defined, we now show how to use these approaches with EKF and SPKF.

4.10.3 *Joint state and parameter estimation via EKF*

Applying EKF to the joint-estimation problem is straightforward. But, we must be careful to evaluate the recursive calculation of $\mathrm{d}\mathbf{F}/\mathrm{d}\mathbf{X}$ correctly when computing the $\hat{\boldsymbol{C}}_k$ matrix. The method is summarized in the Appendix on p. 233.

4.10.4 *Dual state and parameter estimation via EKF*

When implementing dual state and parameter estimation using EKF, two EKFs are implemented with their signals mixed. Again, we need to be careful when computing $\hat{\boldsymbol{C}}^\theta_k$, which requires a total-differential expansion to be correct:

$$\hat{\boldsymbol{C}}^\theta_k = \left. \frac{\mathrm{d}\boldsymbol{g}(\hat{\boldsymbol{x}}^-_k, \boldsymbol{u}_k, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}^-_k}$$
$$\frac{\mathrm{d}\boldsymbol{g}(\hat{\boldsymbol{x}}^-_k, \boldsymbol{u}_k, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \boldsymbol{g}(\hat{\boldsymbol{x}}^-_k, \boldsymbol{u}_k, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{g}(\hat{\boldsymbol{x}}^-_k, \boldsymbol{u}_k, \boldsymbol{\theta})}{\partial \hat{\boldsymbol{x}}^-_k} \frac{\mathrm{d}\hat{\boldsymbol{x}}^-_k}{\mathrm{d}\boldsymbol{\theta}}$$

$$\frac{\mathrm{d}\hat{\boldsymbol{x}}_k^-}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1}, \boldsymbol{\theta})}{\partial \hat{\boldsymbol{x}}_{k-1}^+} \frac{\mathrm{d}\hat{\boldsymbol{x}}_{k-1}^+}{\mathrm{d}\boldsymbol{\theta}}$$

$$\frac{\mathrm{d}\hat{\boldsymbol{x}}_{k-1}^+}{\mathrm{d}\boldsymbol{\theta}} = \frac{\mathrm{d}\hat{\boldsymbol{x}}_{k-1}^-}{\mathrm{d}\boldsymbol{\theta}} - \boldsymbol{L}_{k-1}^x \frac{\mathrm{d}\boldsymbol{g}(\hat{\boldsymbol{x}}_{k-1}^-, \boldsymbol{u}_{k-1}, \boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}}.$$

In the notation, $\boldsymbol{L}_k^x$ is the Kalman gain for the state filter, which is assumed not to be a function of $\boldsymbol{\theta}$. In fact, it *is* a weak function of $\boldsymbol{\theta}$, but the consensus of the literature is that the dependence is weak enough that the simplifications achieved by the assumption outweigh any gains that might be made by a more careful analysis.

To implement this method, the three total derivatives $\mathrm{d}\boldsymbol{g}/\mathrm{d}\boldsymbol{\theta}$, $\mathrm{d}\hat{\boldsymbol{x}}_{k-1}^+/\mathrm{d}\boldsymbol{\theta}$, and $\mathrm{d}\hat{\boldsymbol{x}}_k^-/\mathrm{d}\boldsymbol{\theta}$ are initialized to zero and computed recursively as the filters operate. The method is summarized in the Appendix on p. 233.

4.10.5 Joint state and parameter estimation via SPKF

Joint state and parameter estimation using sigma-point Kalman filters uses the standard SPKF method where the state vector is augmented with the parameters. No other changes are made. The method is summarized in the Appendix on p. 233.

4.10.6 Dual state and parameter estimation via SPKF

Dual state and parameter estimation using SPKF, just like dual estimation using EKF, uses two filters. Both employ the SPKF algorithm and intermix signals. The method is summarized in the Appendix on p. 234.

4.11 Robustness and speed

4.11.1 Ensuring correct convergence

Dual and joint filtering adapt the state estimate $\hat{\boldsymbol{x}}_k^+$ and the parameter estimate $\hat{\boldsymbol{\theta}}_k^+$ so that the model input/output relationship matches the measured input/output data closely. However, there is no built-in guarantee that the state of the model converges to anything with physical meaning. It is possible for the state and parameter estimates to diverge from their true values and, because of cancellations between errors caused by inaccurate state and parameter estimates, still have good agreement with input/output measurements, at least for a time.

When estimating the ESC cell-model state and parameter vectors, we are concerned that the they converge to their true meaning. This will not happen by default with the dual or joint estimation meth-

ods as described so far. Special steps must be taken to ensure that convergence occurs.

Consider the SOC state z_k inside the model state vector. We may use a very crude cell model to derive a synthetic measurement of this state, as we saw in Chap. 3. Specifically, we can model cell terminal voltage as

$$v_k \approx \text{OCV}(z_k) - R_0 i_k.$$

Then, given a measurement of voltage, we can estimate SOC as

$$\hat{z}_k = \text{OCV}^{-1}(v_k + R_0 i_k).$$

While this result is too noisy to use as the primary state of charge estimate, it still has some nice properties. First, if current is zero and hysteresis is negligible, then the estimate converges to the true state of charge. Second, even if current is not zero, the estimate tends to have very little bias: it is noisy, as we saw in Fig. 3.4, but the noise is close to zero mean.

Therefore, by measuring the cell voltage under load v_k, the cell current i_k, and having knowledge of R_0 and the cell's inverse OCV function, we can compute a noisy estimate $\hat{z}_k$ of SOC. We then augment the true measurement of cell voltage with this synthetic measurement of cell SOC in the output equation of the model, which then becomes:

$$\boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}) = \begin{bmatrix} \text{OCV}(z_k) + Mh_k + M_0 s_k - \sum_i R_i i_{R_i,k} - R_0 i_k \\ z_k \end{bmatrix}.$$

The dual or joint nonlinear Kalman filter is run using this modified model, with the measurement used in the measurement update replaced by

$$\boldsymbol{y}_k = \begin{bmatrix} \text{cell voltage at time } k \\ \text{crude SOC estimate } \hat{z}_k \end{bmatrix}.$$

While the errors in $\hat{z}_k$ due to ignoring short-term hysteresis bias and diffusion voltages prohibit it from being used as the primary estimator of SOC, its expected long-term behavior in a dynamic environment is accurate, and it maintains the accuracy of the state of charge state in the dual and joint filters.

4.12 *Unbiased estimate of total capacity using linear regression*

The computational complexity incurred by using a nonlinear Kalman filter to estimate all states and parameters simultaneously is relatively high. If we require estimates of only resistance and total capacity, this complexity is not warranted. The simple estimator of Sect. 4.4

can be used to estimate resistance. But, how can we then estimate total capacity?

Recall from Sect. 4.6 that the sensitivity of voltage to total capacity is low, so separating capacity information from noisy input/output data is difficult, and the noise easily biases the results. The sensitivity improves when SOC has changed appreciably between updates, so we would expect that iteration-by-iteration updates to be unnecessary and computationally inefficient. Fortunately, capacity changes slowly, so updating total-capacity estimates infrequently and only after significant changes to state of charge occur is adequate.[15]

[15] The remainder of this chapter is based in large part on Plett, G., "Recursive approximate weighted total least squares estimation of battery cell total capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2319–2331.

4.12.1 *The problem with least-squares capacity estimates*

For a fresh look at estimating total capacity, consider again the cell state of charge equation, where we compute a cell's state of charge at time index $k_2 > k_1$ starting with an initial known state of charge at time index k_1

$$z_{k_2} = z_{k_1} - \frac{1}{Q}\sum_{k=k_1}^{k_2-1} \eta_k i_k.$$

We can rearrange the terms in this expression to get

$$\underbrace{-\sum_{k=k_1}^{k_2-1} \eta_k i_k}_{y} = Q\underbrace{\left(z_{k_2} - z_{k_1}\right)}_{x},$$

where the obvious linear structure of $y = Qx$ becomes apparent. Using a regression technique, for example, we may compute estimates of Q. We need only to find sets of values for x and y in this equation to do so.[16]

[16] Note that to use this method, coulomb counting cannot be used as the state of charge estimator, since this results in a degenerate equation: we would be comparing coulomb counts on one side of the equation with exactly the same coulomb counts on the other side of the equation. The state of charge estimate must be based—at least in part—on voltage measurements. The Kalman-filter based methods from Chap. 3 for state of charge estimation have enough voltage feedback that they work with this method.

The most common linear regression techniques are based on *ordinary least squares* (*OLS* or simply LS). OLS assumes that the independent variable x is known exactly but that the dependent variable y may have some uncertainty. That is, it finds a solution to the problem $(y - \Delta y) = Qx$ by using known values of x but only partially known values of y to find the best Q to minimize the difference between the line fit and the data in y. We write $y - \Delta y$ in the equation because the true value of the dependent variable is an unknown distance Δy away from the known measurement y.

The issue with using standard ordinary-least-squares regression on the total-capacity-estimation problem is that both the summed current value y and the difference between state of charge values x have sensor noise or estimation noise associated with them. Not only does our coulomb count y have sensor noise, but our estimates of state of charge are also generally imperfect, causing uncertainty on the x

variable as well. That is, the total-capacity-estimation problem is implicitly of the form $(y - \Delta y) = Q(x - \Delta x)$ since both the integrated current and state of charge estimates have errors. This is a different problem from the one solved by ordinary least squares, so using the ordinary-least-squares solution biases the results.

The common approach to counteract this problem is to try to ensure that the state of charge estimates used in the regression are as accurate as possible and then use standard least-squares estimation anyway. For example, we might put constraints on when and how the total capacity is estimated. We could force the cell current to be zero for some period before the test begins and after the test ends so that the cell is in an equilibrium state and so the SOC estimates are as accurate as possible. This procedure eliminates to a large extent—but not completely—the error in the x variable, and makes the regression reasonably accurate. This method still does not handle the residual uncertainty in x correctly: while it minimizes the error, it never totally eliminates it.

A better approach is to recognize that the solution to problems that look like $(y - \Delta y) = Q(x - \Delta x)$ is to use *total-least-squares* regression instead of ordinary-least-squares regression. Total least squares takes into account the uncertainties in both the x and y variables when finding an estimate of Q. However, there are challenges when attempting to make total-least-squares solutions computationally efficient so that they can be implemented on a cost-effective BMS.

In the next sections, we derive the ordinary-least-squares and total-least-squares solutions and some variants thereof to show their similarities and differences. We will develop a computationally efficient approximate total-least-squares solution that works very well to estimate battery-cell capacity from noisy data.

4.13 *Weighted ordinary least squares*

Both ordinary least squares (OLS) and total least squares (TLS), as applied to battery-cell total-capacity estimation, seek to find a constant $\widehat{Q}_n$ such that $\boldsymbol{y} \approx \widehat{Q}_n \boldsymbol{x}$ using n-vectors of measured data $\boldsymbol{x}$ and $\boldsymbol{y}$. The ith data pair, comprising x_i in $\boldsymbol{x}$ and y_i in $\boldsymbol{y}$, correspond to data collected from a cell over the ith interval of time, where x_i is the estimated change in state of charge over that interval and y_i is the accumulated ampere-hours passing through the cell during that period.

Specifically,

$$x_i = \hat{z}_{k_2^{(i)}} - \hat{z}_{k_1^{(i)}} \qquad \text{for time interval } i$$

$$y_i = -\sum_{k=k_1^{(i)}}^{k_2^{(i)}-1} \eta_k i_k,$$

where $k_1^{(i)}$ is the discrete-time index of the start of the ith interval, and $k_2^{(i)}$ is the discrete-time index of the end of the ith interval. The data vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ used to produce a capacity estimate must be at least one sample long ($n \geq 1$), but larger values of n may be used to obtain better estimates as the uncertainties in the data can be averaged out over many measurements.

The OLS approach assumes that there is no error on the x_i and models the data as $\boldsymbol{y} = Q\boldsymbol{x} + \Delta\boldsymbol{y}$, where $\Delta\boldsymbol{y}$ is a vector of unknown measurement errors. This is illustrated in Fig. 4.12, where the error bars on the data points are meant to depict the uncertainties. We assume that $\Delta\boldsymbol{y}$ comprises zero-mean Gaussian random variables with known variances $\sigma_{y_i}^2$ that may be different for every data point.

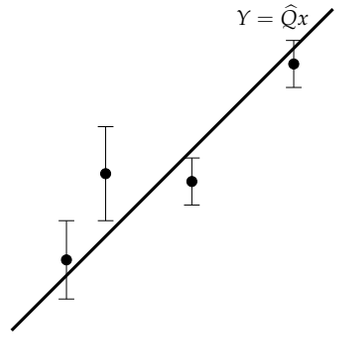

Figure 4.12: The OLS paradigm. (Reproduced from Fig. 1a of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

OLS attempts to find an estimate $\widehat{Q}_n$ of the true cell total capacity Q, based on n data pairs (x_i, y_i), that minimizes the sum of squared errors Δy_i. We generalize the OLS approach slightly here to allow for finding a $\widehat{Q}_n$ that minimizes the sum of *weighted* squared errors, where the weighting takes into account the uncertainty of the measurement. That is, we desire to find a $\widehat{Q}_n$ that minimizes the *weighted least squares (WLS)* cost function[17]

$$\chi^2_{\text{WLS}} = \sum_{i=1}^{n} \frac{(y_i - Y_i)^2}{\sigma_{y_i}^2} = \sum_{i=1}^{n} \frac{(y_i - \widehat{Q}_n x_i)^2}{\sigma_{y_i}^2}.$$

In this equation, Y_i is the final optimized mapping of the data (x_i, y_i) to the line. That is, it is a point on the line $Y_i = \widehat{Q}_n x_i$ corresponding to the measured data pair (x_i, y_i), where y_i is assumed to have noise but x_i has no uncertainty.

[17] The cost function is named with the symbol χ^2 since it turns out to be a chi-squared random variable: it is the sum of the squares of uncorrelated zero-mean unit-variance Gaussian random variables.

There are a number of approaches that may be taken to solve this problem, but one that will serve our purposes well is to differentiate the cost function with respect to $\widehat{Q}_n$ and then to solve for $\widehat{Q}_n$ by setting the partial derivative to zero. The biggest need for care is to recognize that all the $\sigma_{y_i}^2$ inside the summation may be distinct, so we cannot simply multiply both sides of the equation by some constant σ_y^2 to eliminate it from the final result.

First, differentiating gives us

$$\frac{\partial \chi^2_{\text{WLS}}}{\partial \widehat{Q}_n} = -2\sum_{i=1}^{n} \frac{x_i(y_i - \widehat{Q}_n x_i)}{\sigma_{y_i}^2} = 0.$$

Then, dividing both sides of the equation by -2, splitting the summation into two separate summations, and solving for $\widehat{Q}_n$ gives

$$\widehat{Q}_n \sum_{i=1}^{n} \frac{x_i^2}{\sigma_{y_i}^2} = \sum_{i=1}^{n} \frac{x_i y_i}{\sigma_{y_i}^2}$$

$$\widehat{Q}_n = \sum_{i=1}^{n} \frac{x_i y_i}{\sigma_{y_i}^2} \Bigg/ \sum_{i=1}^{n} \frac{x_i^2}{\sigma_{y_i}^2}.$$

If we define two new variables

$$c_{1,n} = \sum_{i=1}^{n} \frac{x_i^2}{\sigma_{y_i}^2} \qquad \text{and} \qquad c_{2,n} = \sum_{i=1}^{n} \frac{x_i y_i}{\sigma_{y_i}^2},$$

then we can write $\widehat{Q}_n = c_{2,n}/c_{1,n}$.

The two quantities $c_{1,n}$ and $c_{2,n}$ may be computed recursively to minimize storage requirements and to even out computational requirements when updating $\widehat{Q}_n$ for large n via

$$\begin{aligned} c_{1,n} &= c_{1,n-1} + x_n^2/\sigma_{y_n}^2 \\ c_{2,n} &= c_{2,n-1} + x_n y_n/\sigma_{y_n}^2. \end{aligned}$$

The recursive approach requires an initial estimate of $c_{1,0}$ and $c_{2,0}$. One possible initialization is simply to set $c_{1,0} = c_{2,0} = 0$. Alternately, we can recognize that a cell with nominal capacity Q_{nom} has that capacity over a SOC range of 1.0. Therefore, we can initialize with a synthetic zeroth "measurement" where $x_0 = 1$ and $y_0 = Q_{\text{nom}}$. The value for $\sigma_{y_0}^2$ can be set to the manufacturing variance of the nominal capacity. That is, $c_{1,0} = 1/\sigma_{y_0}^2$ and $c_{2,0} = Q_{\text{nom}}/\sigma_{y_0}^2$.

This method may be adapted easily to allow fading memory of past measurements. We modify the WLS cost function to place more emphasis on recent measurements by defining the *fading-memory weighted least-squares (FMWLS)* cost function as

$$\chi_{\text{FMWLS}}^2 = \sum_{i=1}^{n} \gamma^{n-i} \frac{(y_i - \widehat{Q}_n x_i)^2}{\sigma_{y_i}^2},$$

where the forgetting factor γ is in the range $0 \ll \gamma \leq 1$. If we expand a few terms, we see that this looks like

$$\begin{aligned} \chi_{\text{FMWLS}}^2 = \frac{(y_n - \widehat{Q}_n x_n)^2}{\sigma_{y_n}^2} + \gamma \frac{(y_{n-1} - \widehat{Q}_n x_{n-1})^2}{\sigma_{y_{n-1}}^2} \\ + \gamma^2 \frac{(y_{n-2} - \widehat{Q}_n x_{n-2})^2}{\sigma_{y_{n-2}}^2} + \cdots \end{aligned}$$

and so more recent data points have higher weighting in the cost function than do data points collected far in the past. With this fading-memory cost function, the solution becomes

$$\widehat{Q}_n = \sum_{i=1}^{n} \gamma^{n-i} \frac{x_i y_i}{\sigma_{y_i}^2} \Bigg/ \sum_{i=1}^{n} \gamma^{n-i} \frac{x_i^2}{\sigma_{y_i}^2},$$

which may also be computed easily in a recursive manner. To do so, we keep track of the two running sums:

$$\tilde{c}_{1,n} = \sum_{i=1}^{n} \gamma^{n-i} x_i^2 / \sigma_{y_i}^2$$
$$\tilde{c}_{2,n} = \sum_{i=1}^{n} \gamma^{n-i} x_i y_i / \sigma_{y_i}^2.$$

Then, the optimal total-capacity estimate is

$$\widehat{Q}_n = \tilde{c}_{2,n} / \tilde{c}_{1,n} \tag{4.8}$$

and the value of the cost function for this estimate can be written as

$$\chi^2_{\text{FMWLS}} = \tilde{c}_{1,n} \widehat{Q}_n^2 - 2\tilde{c}_{2,n} \widehat{Q}_n + \tilde{c}_{3,n}. \tag{4.9}$$

When an additional data point becomes available, we update these quantities via

$$\tilde{c}_{1,n} = \gamma \tilde{c}_{1,n-1} + x_n^2 / \sigma_{y_n}^2$$
$$\tilde{c}_{2,n} = \gamma \tilde{c}_{2,n-1} + x_n y_n / \sigma_{y_n}^2.$$

In summary, the WLS and FMWLS solutions have a number of nice properties:

1. They give a closed-form solution for $\widehat{Q}_n$. Only simple operations—multiplication, addition, and division—are required.
2. The solutions can be computed very easily in a recursive manner.
3. Fading memory can be added easily, allowing adaptation of $\widehat{Q}_n$ to adjust for changes in true cell total capacity.

4.14 Weighted total least squares

The TLS approach assumes that there are errors on both the x_i and y_i measurements and models the data as $(\boldsymbol{y} - \Delta\boldsymbol{y}) = Q(\boldsymbol{x} - \Delta\boldsymbol{x})$. This is illustrated in Fig. 4.13, where the error bars on the data points are meant to show the uncertainties in each dimension. We assume that $\Delta\boldsymbol{x}$ is a zero-mean Gaussian random vector with known element variances $\sigma_{x_i}^2$ and that $\Delta\boldsymbol{y}$ is a zero-mean Gaussian random vector with known element variances $\sigma_{y_i}^2$, where $\sigma_{x_i}^2$ is not necessarily equal to or related to $\sigma_{y_i}^2$, and where all of the $\sigma_{x_i}^2$ and $\sigma_{y_i}^2$ may be distinct.

TLS attempts to find an estimate $\widehat{Q}_n$ of the true cell total capacity Q that minimizes the sum of squared errors Δx_i plus the sum of squared errors Δy_i. We generalize that approach here slightly to allow for finding a $\widehat{Q}_n$ that minimizes the sum of *weighted* squared errors, where the weighting takes into account the uncertainty of the measurement.

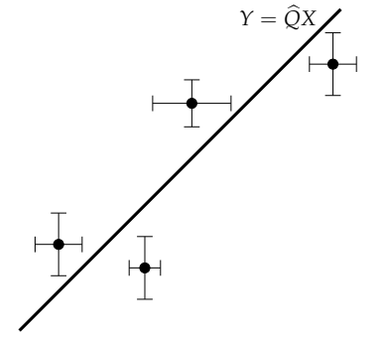

Figure 4.13: The TLS paradigm. (Reproduced from Fig. 1b of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

That is, we desire to find a $\widehat{Q}_n$ that minimizes the *weighted total-least-squares (WTLS)* cost function

$$\chi^2_{\text{WTLS}} = \sum_{i=1}^{n} \frac{(x_i - X_i)^2}{\sigma^2_{x_i}} + \frac{(y_i - Y_i)^2}{\sigma^2_{y_i}}.$$

In this equation, (X_i, Y_i) is the final optimized mapping of the data (x_i, y_i) to the line. That is, X_i and Y_i are the points on the line $Y_i = \widehat{Q}_n X_i$ corresponding to the noisy measured data pair (x_i, y_i) for the optimized value of $\widehat{Q}_n$. In general, $x_i \neq X_i$ and $y_i \neq Y_i$. As part of the WTLS solution, we will need to find this optimal mapping from (x_i, y_i) to (X_i, Y_i).

Because both x_i and y_i have noise, we must handle this optimization problem differently from the way we handled the WLS problem. We augment the cost function with Lagrange multipliers λ_i to enforce the constraint that $Y_i = \widehat{Q}_n X_i$. This yields augmented cost function

$$\chi^2_{\text{WTLS},a} = \sum_{i=1}^{n} \frac{(x_i - X_i)^2}{\sigma^2_{x_i}} + \frac{(y_i - Y_i)^2}{\sigma^2_{y_i}} - \lambda_i (Y_i - \widehat{Q}_n X_i).$$

To solve, we set the partial derivatives of $\chi^2_{\text{WTLS},a}$ with respect to X_i, Y_i, and λ_i to zero. First, taking the partial derivative with respect to the Lagrange multipliers recovers the constraint:

$$\begin{aligned} \frac{\partial \chi^2_{\text{WTLS},a}}{\partial \lambda_i} &= -(Y_i - \widehat{Q}_n X_i) = 0 \\ Y_i &= \widehat{Q}_n X_i. \end{aligned} \tag{4.10}$$

Next, we take the partial derivative with respect to Y_i. This allows us to solve for the Lagrange multipliers:

$$\begin{aligned} \frac{\partial \chi^2_{\text{WTLS},a}}{\partial Y_i} &= \frac{-2(y_i - Y_i)}{\sigma^2_{y_i}} - \lambda_i = 0 \\ \lambda_i &= \frac{-2(y_i - Y_i)}{\sigma^2_{y_i}}. \end{aligned} \tag{4.11}$$

Finally, we take the partial derivative with respect to X_i:

$$\frac{\partial \chi^2_{\text{WTLS},a}}{\partial X_i} = \frac{-2(x_i - X_i)}{\sigma^2_{x_i}} + \lambda_i \widehat{Q}_n = 0. \tag{4.12}$$

We substitute the solution for the Lagrange multiplier from Eq. (4.11) into Eq. (4.12) and multiply both sides of the equation by $\sigma^2_{y_i}\sigma^2_{x_i}$ to simplify the form

$$\begin{aligned} 0 &= -\frac{2(x_i - X_i)}{\sigma^2_{x_i}} - \frac{2(y_i - Y_i)}{\sigma^2_{y_i}} \widehat{Q}_n \\ &= \sigma^2_{y_i}(x_i - X_i) + \sigma^2_{x_i}(y_i - Y_i)\widehat{Q}_n. \end{aligned}$$

Finally, we substitute the constraint Eq. (4.10) and solve for X_i:

$$0 = \sigma_{y_i}^2 x_i - \sigma_{y_i}^2 X_i + \sigma_{x_i}^2 y_i \widehat{Q}_n - \sigma_{x_i}^2 X_i \widehat{Q}_n^2$$
$$X_i = \frac{x_i \sigma_{y_i}^2 + \widehat{Q}_n y_i \sigma_{x_i}^2}{\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2}.$$

With these results, we can rewrite the original WTLS cost function in terms of measured and known quantities as

$$\begin{aligned}
\chi^2_{\text{WTLS}} &= \sum_{i=1}^{n} \frac{(x_i - X_i)^2}{\sigma_{x_i}^2} + \frac{(y_i - Y_i)^2}{\sigma_{y_i}^2} \\
&= \sum_{i=1}^{n} \frac{\left(x_i - \frac{x_i \sigma_{y_i}^2 + \widehat{Q}_n y_i \sigma_{x_i}^2}{\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2}\right)^2}{\sigma_{x_i}^2} + \frac{\left(y_i - \widehat{Q}_n \frac{x_i \sigma_{y_i}^2 + \widehat{Q}_n y_i \sigma_{x_i}^2}{\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2}\right)^2}{\sigma_{y_i}^2} \\
&= \sum_{i=1}^{n} \frac{\left(x_i \left(\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2\right) - \left(x_i \sigma_{y_i}^2 + \widehat{Q}_n y_i \sigma_{x_i}^2\right)\right)^2}{\sigma_{x_i}^2 \left(\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2\right)^2} + \\
&\qquad \frac{\left(y_i \left(\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2\right) - \widehat{Q}_n \left(x_i \sigma_{y_i}^2 + \widehat{Q}_n y_i \sigma_{x_i}^2\right)\right)^2}{\sigma_{y_i}^2 \left(\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2\right)^2} \\
&= \sum_{i=1}^{n} \frac{\widehat{Q}_n^2 \sigma_{x_i}^4 \left(y_i - \widehat{Q}_n x_i\right)^2}{\sigma_{x_i}^2 \left(\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2\right)^2} + \frac{\sigma_{y_i}^4 \left(y_i - \widehat{Q}_n x_i\right)^2}{\sigma_{y_i}^2 \left(\sigma_{y_i}^2 + \widehat{Q}_n^2 \sigma_{x_i}^2\right)^2} \\
&= \sum_{i=1}^{n} \frac{(y_i - \widehat{Q}_n x_i)^2}{\widehat{Q}_n^2 \sigma_{x_i}^2 + \sigma_{y_i}^2}.
\end{aligned} \tag{4.13}$$

To find the value of $\widehat{Q}_n$ that minimizes this cost function, we set the partial derivative $\partial \chi^2_{\text{WTLS}} / \partial \widehat{Q}_n = 0$. After a few steps, we can find that this is equivalent to solving for $\widehat{Q}_n$ in

$$\frac{\partial \chi^2_{\text{WTLS}}}{\partial \widehat{Q}_n} = \sum_{i=1}^{n} \frac{2(\widehat{Q}_n x_i - y_i)(\widehat{Q}_n y_i \sigma_{x_i}^2 + x_i \sigma_{y_i}^2)}{(\widehat{Q}_n^2 \sigma_{x_i}^2 + \sigma_{y_i}^2)^2} = 0. \tag{4.14}$$

Unfortunately, this solution has none of the nice properties of the WLS solution. Namely,

1. There is no closed-form solution in the general case. Instead, a numerical optimization method must be used in order to find $\widehat{Q}_n$. One possibility is to perform a Newton–Raphson search for $\widehat{Q}_n$, where several iterations of the equation

$$\widehat{Q}_{n,k} = \widehat{Q}_{n,k-1} - \frac{\partial \chi^2_{\text{WTLS}} / \partial \widehat{Q}_{n,k-1}}{\partial^2 \chi^2_{\text{WTLS}} / \partial \widehat{Q}^2_{n,k-1}} \tag{4.15}$$

   are performed every time the data vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are updated with a new data pair. That is, after n data pairs are gathered, the

total-least-squares estimate $\widehat{Q}_{n,0}$ is initialized to some value. We could use an estimate $\widehat{Q}_n$ from a weighted-least-squares solution, or we could use the prior estimate $\widehat{Q}_{n-1}$ from a weighted-total-least-squares solution for this initialization. Then, Eq. (4.15) is computed at time index n for $k = 1$ to some number of total iterations K. Then, the final weighted-total-least-squares solution at this time step is set to $\widehat{Q}_n = \widehat{Q}_{n,K}$.

The numerator of Eq. (4.15) is the *Jacobian* of the original cost function and is given by Eq. (4.14), if $\widehat{Q}_n$ is replaced by $\widehat{Q}_{n,k-1}$. The denominator of this update equation is the *Hessian* of the original metric function, which can be found using known quantities to be

$$\frac{\partial^2 \chi^2_{\text{WTLS}}}{\partial \widehat{Q}_n^2} = 2\sum_{i=1}^{n}\left(\frac{\sigma_{y_i}^4 x_i^2 + \sigma_{x_i}^4(3\widehat{Q}_n^2 y_i^2 - 2\widehat{Q}_n^3 x_i y_i)}{(\widehat{Q}_n^2\sigma_{x_i}^2 + \sigma_{y_i}^2)^3}\cdot \right. \\ \left. - \frac{\sigma_{x_i}^2\sigma_{y_i}^2(3\widehat{Q}_n^2 x_i^2 - 6\widehat{Q}_n x_i y_i + y_i^2)}{(\widehat{Q}_n^2\sigma_{x_i}^2 + \sigma_{y_i}^2)^3}\right). \tag{4.16}$$

This result is used in the Newton–Raphson iteration of Eq. (4.15) with $\widehat{Q}_n$ replaced by $\widehat{Q}_{n,k-1}$.

This Newton–Raphson search has the property that the number of significant figures in the solution doubles with each iteration of the update. In practice, we find that around four iterations produce double-precision results. Also note that the metric function χ^2_{WTLS} is convex, which guarantees that this iterative method will converge to the global solution.

2. There is no recursive update in the general case. This has both storage and computational implications. To use WTLS, the entire vector $\boldsymbol{x}$ and $\boldsymbol{y}$ must be retained, which implies increasing storage as the number of measurements increase. Furthermore, the number of computations grows as n grows. That is, WTLS is not well suited for an embedded-system application that must run in real time within limited memory.

3. There is no fading memory recursive update (because there is no recursive update). A nonrecursive *fading memory weighted total least squares (FMWTLS)* cost function may be defined, however, as

$$\chi^2_{\text{FMWTLS}} = \sum_{i=1}^{n}\gamma^{n-i}\frac{(y_i - \widehat{Q}_n x_i)^2}{\widehat{Q}_n^2\sigma_{x_i}^2 + \sigma_{y_i}^2}.$$

The Jacobian of this cost function is

$$\frac{\partial \chi^2_{\text{FMWTLS}}}{\partial \widehat{Q}_n} = 2\sum_{i=1}^{n}\gamma^{n-i}\frac{(\widehat{Q}_n x_i - y_i)(\widehat{Q}_n y_i\sigma_{x_i}^2 + x_i\sigma_{y_i}^2)}{(\widehat{Q}_n^2\sigma_{x_i}^2 + \sigma_{y_i}^2)^2}. \tag{4.17}$$

The Hessian is

$$\frac{\partial^2 \chi^2_{\text{FMWTLS}}}{\partial \widehat{Q}_n^2} = 2\sum_{i=1}^{n} \gamma^{n-i} \left(\frac{\sigma_{y_i}^4 x_i^2 + \sigma_{x_i}^4 (3\widehat{Q}_n^2 y_i^2 - 2\widehat{Q}_n^3 x_i y_i)}{(\widehat{Q}_n^2 \sigma_{x_i}^2 + \sigma_{y_i}^2)^3} - \frac{\sigma_{x_i}^2 \sigma_{y_i}^2 (3\widehat{Q}_n^2 x_i^2 - 6\widehat{Q}_n x_i y_i + y_i^2)}{(\widehat{Q}_n^2 \sigma_{x_i}^2 + \sigma_{y_i}^2)^3} \right). \quad (4.18)$$

Using the Jacobian and Hessian of this cost function, we can use a Newton–Raphson search to find the solution to the fading-memory cost function to find an estimate of Q.

In Sect. 4.17, we will address a special case of WTLS that gives a closed-form solution with recursive update and fading memory. We will then give an approximate solution to the general WTLS problem that also has the nice characteristics of the WLS solution. Before we do so, we first consider two important properties of both the WLS and WTLS solutions.

4.15 *Goodness of model fit*

When the measurement errors $\Delta \boldsymbol{x}$ and $\Delta \boldsymbol{y}$ are uncorrelated and Gaussian, the metric functions χ^2_{WLS} and χ^2_{WTLS} are chi-squared random variables.

- χ^2_{WLS} is a chi-squared random variable with $n-1$ degrees of freedom because n data points y_i were used in its creation and one degree of freedom is lost when fitting $\widehat{Q}_n$.
- χ^2_{WTLS} is a chi-squared random variable with $2n-1$ degrees of freedom because n data points x_i and n additional data points y_i are used in its creation and one degree of freedom is lost when fitting $\widehat{Q}_n$.

Knowledge of the distribution and the number of degrees of freedom can be used to determine, from the optimized values of the metric functions, whether the model fit is reliable; that is, whether the linear fit is a good fit to the data and whether the optimized value of $\widehat{Q}_n$ is a good estimate of the cell's total capacity.

The incomplete gamma function $P(\chi^2 \mid \nu)$ is defined as the probability that the observed chi-square for a correct model should be less than a computed value χ^2 for degree of freedom ν. Its complement, $Q(\chi^2 \mid \nu) = 1 - P(\chi^2 \mid \nu)$, is the probability that the observed chi-square will exceed the value χ^2 by chance *even* for a correct model.[18] Therefore, to test for goodness of fit of a model, we must evaluate

$$Q(\chi^2 \mid \nu) = \frac{1}{\Gamma(\nu/2)} \int_{\chi^2/2}^{\infty} e^{-t} t^{(\nu/2-1)} \, \mathrm{d}t. \quad (4.19)$$

Methods for computing this function are built into many engineering analysis programs, and C-language code is also easy to find.

[18] The nomenclature $Q(\chi^2 \mid \nu)$ is standard for the (complementary) incomplete gamma function, and is not to be confused with the symbol used to denote true cell total capacity Q, or with the symbol used to denote the estimate of cell total capacity $\widehat{Q}_n$.

If the value obtained for $Q(\chi^2 \mid \nu)$ is small for some cost χ^2 and degree of freedom ν, then either the model is wrong and can be statistically rejected, or the variances $\sigma^2_{x_i}$ or $\sigma^2_{y_i}$ are poorly known, or the variances are not actually Gaussian. The third possibility is common, but is also generally benign if we are willing to accept low values of $Q(\chi^2 \mid \nu)$ as representing a valid model. It is not unusual to accept models if $Q(\chi^2 \mid \nu) \gtrsim 0.001$ and to reject them otherwise.

We will see that when the hypothesized model is not a good fit to the data, the value of $Q(\chi^2 \mid \nu)$ becomes extremely small. However, when the hypothesized model is equal to the true model generating the data, even when $\widehat{Q}_n$ is not precisely equal to Q, the value of $Q(\chi^2 \mid \nu)$ tends to be very close to unity. We will use this information later to show that the WLS model is not a good approach to total-capacity estimation, whereas WTLS is much better. It could also be used online in a battery-management system as a check for the validity of the present total-capacity estimate.

4.16 *Confidence intervals*

When computing an estimate of cell total capacity $\widehat{Q}_n$, it is also important to be able to specify the certainty of that estimate. Specifically, we would like to estimate the variance $\sigma^2_{\widehat{Q}_n}$ of the total capacity estimate, with which we can compute confidence intervals such as three-sigma bounds $(\widehat{Q}_n - 3\sigma_{\widehat{Q}_n}, \widehat{Q}_n + 3\sigma_{\widehat{Q}_n})$ within which we have high certainty that the true value of cell total capacity Q lies.

To derive confidence limits, we must recast the least-squares type optimization problem as a maximum-likelihood optimization problem. With the assumption that all errors are Gaussian, this is straightforward. If we form a vector $\boldsymbol{y}$ comprising elements y_i and a vector $\boldsymbol{x}$ comprising corresponding elements x_i and a diagonal matrix $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}$ having corresponding diagonal elements $\sigma^2_{y_i}$, then minimizing χ^2_{WLS} is equivalent to maximizing the value of the multivariable Gaussian pdf:

$$\begin{aligned} ML_{\text{WLS}} &= \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{y} - \widehat{Q}_n\boldsymbol{x})^T\boldsymbol{\Sigma}^{-1}_{\tilde{\boldsymbol{y}}}(\boldsymbol{y} - \widehat{Q}_n\boldsymbol{x})\right) \\ &= \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}|^{1/2}} \exp\left(-\frac{1}{2}\chi^2_{\text{WLS}}\right), \end{aligned} \tag{4.20}$$

which is a maximum-likelihood problem. That is, minimizing χ^2_{WLS} yields the same $\widehat{Q}_n$ that maximizes ML_{WLS}.[19]

Similarly, if we form a vector $\boldsymbol{d}$ by concatenating $\boldsymbol{y}$ and $\boldsymbol{x}$, and a vector $\hat{\boldsymbol{d}}$ by concatenating the corresponding elements Y_i and X_i, and a diagonal matrix $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}}}$ having diagonal elements $\sigma^2_{y_i}$ followed by $\sigma^2_{x_i}$,

[19] The constant to the left of the exponential in Eq. (4.20) causes the function to integrate to 1, yielding a valid probability density function.

then minimizing χ^2_{WTLS} is equivalent to maximizing the value of

$$ML_{\text{WTLS}} = \frac{1}{(2\pi)^n |\mathbf{\Sigma}_{\tilde{\boldsymbol{d}}}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{d} - \hat{\boldsymbol{d}})^T \mathbf{\Sigma}_{\tilde{\boldsymbol{d}}}^{-1} (\boldsymbol{d} - \hat{\boldsymbol{d}})\right)$$
$$= \frac{1}{(2\pi)^n |\mathbf{\Sigma}_{\tilde{\boldsymbol{d}}}|^{1/2}} \exp\left(-\frac{1}{2}\chi^2_{\text{WTLS}}\right).$$

The maximum-likelihood formulation makes it possible to determine confidence intervals on $\widehat{Q}_n$. According to the Cramer–Rao theorem, a tight lower bound on the variance of $\widehat{Q}_n$ is given by the negative inverse of the second derivative of the argument of the exponential function, evaluated at the $\widehat{Q}_n$ that minimizes the least-squares cost function or maximizes the maximum-likelihood cost function. Therefore, we have

$$\sigma^2_{\widehat{Q}_n} \geq 2\left(\frac{\partial^2 \chi^2_{\text{WLS}}}{\partial \widehat{Q}_n^2}\right)^{-1} \quad \text{for WLS}$$
$$\sigma^2_{\widehat{Q}_n} \geq 2\left(\frac{\partial^2 \chi^2_{\text{WTLS}}}{\partial \widehat{Q}_n^2}\right)^{-1} \quad \text{for WTLS.}$$

These lower bounds on the variance are often quite tight, so are reasonable to be used in computing confidence intervals on our total-capacity estimates.

The second partial derivatives (i.e., the Hessians) of the WTLS and FMWTLS metric functions have already been described in the context of a Newton–Raphson iteration by Eqs. (4.14) and (4.16) for WTLS and Eqs. (4.17) and (4.18) for FMWTLS. For WLS and FMWLS, the situation is easier. We have

$$\frac{\partial^2 \chi^2_{\text{WLS}}}{\partial \widehat{Q}_n^2} = 2\sum_{i=1}^{n} \frac{x_i^2}{\sigma^2_{y_i}}$$
$$\frac{\partial^2 \chi^2_{\text{FMWLS}}}{\partial \widehat{Q}_n^2} = 2\sum_{i=1}^{n} \gamma^{n-i} \frac{x_i^2}{\sigma^2_{y_i}},$$

which may be computed using the previously defined recursive parameters as:

$$\frac{\partial^2 \chi^2_{\text{WLS}}}{\partial \widehat{Q}_n^2} = 2c_{1,n}$$
$$\frac{\partial^2 \chi^2_{\text{FMWLS}}}{\partial \widehat{Q}_n^2} = 2\tilde{c}_{1,n}. \tag{4.21}$$

We will use these results to produce confidence intervals on our estimate in the examples in Sect. 4.20 and 4.21. Note that this method produces lower bounds on the width of the confidence interval: it is possible for the real bounds to be wider. Keeping this in mind, the

plots in these future sections are somewhat optimistic. However, we will note that the confidence intervals are still uncomfortably wide in some cases. This is unavoidable. We are computing optimal estimates of total capacity. The wide error bounds are due to the total-capacity estimation problem being very hard and this is the best that can be done.

4.17 Simplified total least squares

4.17.1 TLS with proportional confidence on x_i and y_i

The general WTLS and FMWTLS solutions provide excellent results but are impractical to implement in an embedded system due to ever-growing memory and computational requirements. Therefore, we search for cases that lead to simpler implementations. Here, we look at an exact solution for the case when the uncertainties on the x_i and y_i data points are proportional to each other for all i, which can be implemented easily in an embedded system. With insights from this solution we will next look at an approximate WTLS solution that also has nice implementation properties.

If $\sigma_{x_i} = k\sigma_{y_i}$ for all i, then the WTLS cost function reduces to a generalization of the standard TLS cost function. We substitute $\sigma_{x_i} = k\sigma_{y_i}$ into χ^2_{WTLS} and associated results to get the proportional total-least-squares (PTLS) cost function

$$\chi^2_{\text{PTLS}} = \sum_{i=1}^{n} \frac{(x_i - X_i)^2}{k^2\sigma^2_{y_i}} + \frac{(y_i - Y_i)^2}{\sigma^2_{y_i}} = \sum_{i=1}^{n} \frac{(y_i - \widehat{Q}_n x_i)^2}{(\widehat{Q}_n^2 k^2 + 1)\sigma^2_{y_i}}.$$

Furthermore, the Jacobian of the WTLS cost function reduces (again, via the substitution $\sigma_{x_i} = k\sigma_{y_i}$) to

$$\frac{\partial \chi^2_{\text{PTLS}}}{\partial \widehat{Q}_n} = 2\sum_{i=1}^{n} \frac{(\widehat{Q}_n x_i - y_i)(\widehat{Q}_n k^2 y_i + x_i)}{(\widehat{Q}_n^2 k^2 + 1)^2 \sigma^2_{y_i}}.$$

This equation may be solved for an exact solution to $\widehat{Q}_n$ without requiring iteration to do so.

We set the Jacobian to zero and collect terms

$$\begin{aligned}
\frac{\partial \chi^2_{\text{PTLS}}}{\partial \widehat{Q}_n} &= 2\sum_{i=1}^{n} \frac{(\widehat{Q}_n x_i - y_i)(\widehat{Q}_n k^2 y_i + x_i)}{(\widehat{Q}_n^2 k^2 + 1)^2 \sigma^2_{y_i}} = 0 \\
&= \widehat{Q}_n^2 \underbrace{\sum_{i=1}^{n} k^2 \frac{x_i y_i}{\sigma^2_{y_i}}}_{a = k^2 c_{2,n}} + \widehat{Q}_n \underbrace{\sum_{i=1}^{n} \frac{x_i^2 - k^2 y_i^2}{\sigma^2_{y_i}}}_{b = c_{1,n} - k^2 c_{3,n}} + \underbrace{\sum_{i=1}^{n} \frac{-x_i y_i}{\sigma^2_{y_i}}}_{c = -c_{2,n}} \\
&= a\widehat{Q}_n^2 + b\widehat{Q}_n + c = 0,
\end{aligned}$$

where we have defined $c_{3,n} = \sum_{i=1}^{n} y_i^2/\sigma_{y_i}^2$. Note that the solution for $\widehat{Q}_n$ can be found via the quadratic formula

$$\widehat{Q}_n = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Substituting the recursive quantities, we can write

$$\widehat{Q}_n = \frac{-(c_{1,n} - k^2 c_{3,n}) \pm \sqrt{(c_{1,n} - k^2 c_{3,n})^2 + 4k^2 c_{2,n}^2}}{2k^2 c_{2,n}}. \tag{4.22}$$

Which of these two solutions should we to choose for our final total-capacity estimate? We can show using a Routh test that this quadratic equation always has exactly one positive root and one negative root. Because total capacity must be positive, we will choose the positive solution to Eq. (4.22).

To see this, we compute the Routh array for this quadratic equation, which is:

$$\begin{array}{c|cc} \widehat{Q}_n^2 & k^2 c_{2,n} & -c_{2,n} \\ \widehat{Q}_n^1 & c_{1,n} - k^2 c_{3,n} & 0 \\ \widehat{Q}_n^0 & -c_{2,n} & 0 \end{array}$$

The Routh test examines the left column of the Routh array and counts the number of sign changes as we traverse from the first row to the final row. The number of roots in the open right-half complex plane is equal to the number of sign changes that we count. For this array, we note that the top entry must be positive since $c_{2,n}$ is always positive and similarly the bottom entry must be negative. Therefore, regardless of the sign on the middle entry, there is exactly one sign change from positive to negative as we traverse the first column.

Consequently, there is exactly one root of the polynomial in the right-half complex plane. The other root, therefore, must be in the left-half plane or on the imaginary axis. By the fundamental theorem of algebra, because the coefficients $c_{1,n}$, $c_{2,n}$, and $c_{3,n}$ are real, the polynomial roots must either both be real or be complex conjugates. The fact that they are in different halves of the complex plane shows that they cannot be complex conjugates, and therefore must both be real. So, we choose the larger root from the solution of the quadratic equation, which corresponds to the positive root.

Recursive calculation is done via

$$\widehat{Q}_n = \frac{-c_{1,n} + k^2 c_{3,n} + \sqrt{(c_{1,n} - k^2 c_{3,n})^2 + 4k^2 c_{2,n}^2}}{2k^2 c_{2,n}},$$

where initialization is done by setting $x_0 = 1$, $y_0 = Q_{\text{nom}}$, and $\sigma_{y_i}^2$ to a value representing the uncertainty of the total capacity at

manufacture. Therefore,

$$c_{1,0} = 1/\sigma_{y_i}^2$$
$$c_{2,0} = Q_{\text{nom}}/\sigma_{y_i}^2$$
$$c_{3,0} = Q_{\text{nom}}^2/\sigma_{y_i}^2$$

and

$$c_{1,n} = c_{1,n-1} + x_n^2/\sigma_{y_i}^2$$
$$c_{2,n} = c_{2,n-1} + x_n y_n/\sigma_{y_i}^2$$
$$c_{3,n} = c_{3,n-1} + y_n^2/\sigma_{y_i}^2.$$

The Hessian, which is required to compute the uncertainty of the estimate, may also be found in terms of the recursive parameters:

$$\frac{\partial^2 \chi_{\text{PTLS}}^2}{\partial \widehat{Q}_n^2} = \frac{(-4k^4 c_2)\widehat{Q}_n^3 + 6k^4 c_3 \widehat{Q}_n^2}{(\widehat{Q}_n^2 k^2 + 1)^3} + \frac{(-6c_1 + 12c_2)k^2 \widehat{Q}_n + 2(c_1 - k^2 c_3)}{(\widehat{Q}_n^2 k^2 + 1)^3}.$$

This can be used to predict error bounds on the estimate $\widehat{Q}_n$. One-sigma bounds are computed as

$$\sigma_{\widehat{Q}_n} = \sqrt{2/(\partial^2 \chi_{\text{PTLS}}^2/\partial \widehat{Q}_n^2)}.$$

Fading memory may be incorporated easily. Following the same steps, we find that recursive calculation is done via

$$\widehat{Q}_n = \frac{-\tilde{c}_{1,n} + k^2 \tilde{c}_{3,n} + \sqrt{(\tilde{c}_{1,n} - k^2 \tilde{c}_{3,n})^2 + 4k^2 \tilde{c}_{2,n}^2}}{2k^2 \tilde{c}_{2,n}}, \tag{4.23}$$

where initialization is done by setting $x_0 = 1$, $y_0 = Q_{\text{nom}}$, and $\sigma_{y_i}^2$ to a value representing the uncertainty of the initial total capacity. Therefore,

$$\tilde{c}_{1,0} = 1/\sigma_{y_i}^2$$
$$\tilde{c}_{2,0} = Q_{\text{nom}}/\sigma_{y_i}^2$$
$$\tilde{c}_{3,0} = Q_{\text{nom}}^2/\sigma_{y_i}^2$$

and

$$\tilde{c}_{1,n} = \gamma \tilde{c}_{1,n-1} + x_n^2/\sigma_{y_i}^2$$
$$\tilde{c}_{2,n} = \gamma \tilde{c}_{2,n-1} + x_n y_n/\sigma_{y_i}^2$$
$$\tilde{c}_{3,n} = \gamma \tilde{c}_{3,n-1} + y_n^2/\sigma_{y_i}^2.$$

After some straightforward manipulations, we can obtain the Hessian in terms of the recursive parameters $\tilde{c}_1$ through $\tilde{c}_3$:

$$\frac{\partial^2 \chi^2_{\text{FMPTLS}}}{\partial \widehat{Q}_n^2} = \frac{(-4k^4\tilde{c}_2)\widehat{Q}_n^3 + 6k^4\tilde{c}_3\widehat{Q}_n^2}{(\widehat{Q}_n^2k^2+1)^3} + \frac{(-6\tilde{c}_1+12\tilde{c}_2)k^2\widehat{Q}_n + 2(\tilde{c}_1 - k^2\tilde{c}_3)}{(\widehat{Q}_n^2k^2+1)^3}. \quad (4.24)$$

This can be used to predict error bounds on the estimate $\widehat{Q}_n$. One-sigma bounds are computed as

$$\sigma_{\widehat{Q}_n} = \sqrt{2/(\partial^2\chi^2_{\text{FMPTLS}}/\partial\widehat{Q}_n^2)}.$$

Further, the cost function for the optimizing $\widehat{Q}_n$ can be computed in terms of the recursive parameters as

$$\chi^2_{\text{FMPTLS}} = \frac{\tilde{c}_{1,n}\widehat{Q}_n^2 - 2\tilde{c}_{2,n}\widehat{Q}_n + \tilde{c}_{3,n}}{\widehat{Q}_n^2k^2+1}. \quad (4.25)$$

In summary, the proportional-total-least-squares solution shares the nice properties of the WLS solution:

1. It gives a closed-form solution for $\widehat{Q}_n$. No iteration or advanced algorithms are required—only simple mathematical operations.
2. The solution can be computed very easily in a recursive manner. We keep track of the three running sums $c_{1,n}$, $c_{2,n}$ and $c_{3,n}$. When an additional data point becomes available, we update the sums and compute an updated total-capacity estimate.
3. Fading memory is easily added.

Unfortunately, this solution does not allow $\sigma^2_{x_i}$ and $\sigma^2_{y_i}$ to be arbitrary—they must be proportionally related by the scaling factor $\sigma_{x_i} = k\sigma_{y_i}$ for every data point. The next section describes an approximation to PTLS that allows an arbitrary relationship.

4.18 Approximate full solution

4.18.1 Deriving the approximate weighted total-least-squares cost function

We desire an approximate solution to the WTLS problem that allows $\sigma^2_{x_i}$ and $\sigma^2_{y_i}$ to be nonproportional and that yields a recursive solution for feasible implementation in an embedded system. We will do so by considering Fig. 4.14, which illustrates the geometry of the WTLS and PTLS solutions and motivates the geometry of the approximate solution to be developed in this section.

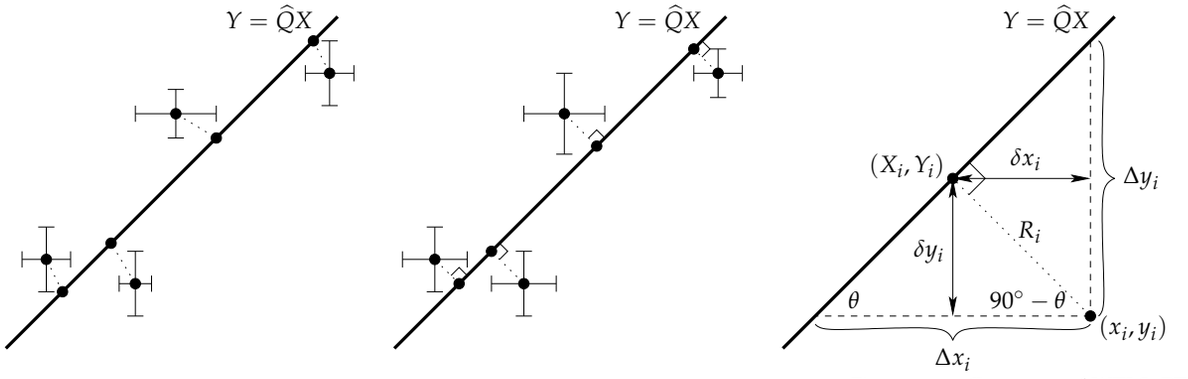

Figure 4.14: Geometry of WTLS, PTLS, and AWTLS.
(Reproduced from Fig. 2 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

The left frame of the figure shows the WTLS relationship between data point (x_i, y_i) and its optimized map (X_i, Y_i) on $Y_i = \widehat{Q}_n X_i$ when $\sigma_{x_i}^2$ and $\sigma_{y_i}^2$ are arbitrary. The data points (x_i, y_i) are drawn with error bars to illustrate the uncertainties in each dimension, which are proportional to σ_{x_i} and σ_{y_i}. A dotted line joins each data point (x_i, y_i) to its map (X_i, Y_i) on the line $Y_i = \widehat{Q}_n X_i$. We see that the distance between x_i and X_i is not necessarily equal to the distance between y_i and Y_i. If the quality of the x_i measurement is better (poorer) than the quality of the y_i measurement, the distance to its map X_i should be shorter (greater) than the distance from y_i to its map Y_i.

The middle frame shows the PTLS relationship between data point (x_i, y_i) and its optimized map (X_i, Y_i) on $Y_i = \widehat{Q}_n X_i$ when $\sigma_{x_i}^2$ and $\sigma_{y_i}^2$ are equal. In this case, the distance between x_i and X_i is equal to the distance between y_i and Y_i, and the line joining data point (x_i, y_i) and its map (X_i, Y_i) is perpendicular to the line $Y_i = \widehat{Q}_n X_i$. If σ_{x_i} and σ_{y_i} are not equal but proportional, either of the x- or y-axes may be scaled to yield transformed data points having equal variances and hence the same idea applies.

The right frame of the figure illustrates definitions that will be used to derive an approximate weighted total-least-squares (AWTLS) solution. As with the PTLS solution, we enforce that the line joining data point (x_i, y_i) and (X_i, Y_i) be perpendicular to the line $Y_i = \widehat{Q}_n X_i$. This will result in a solution that may be solved recursively. However, as with the WTLS solution, we weight the distance between x_i and X_i differently from the distance between y_i and Y_i in the optimization cost function.

We define Δx_i be the x-distance between data point i and the line, and Δy_i be the y-distance between data point i and the line. The slope of the line is $\widehat{Q}_n = \Delta y_i / \Delta x_i$ for all i. The angle of the line is $\theta = \tan^{-1} \widehat{Q}_n$. The shortest distance between the line and a given data

point is

$$R_i = \Delta y_i \cos\theta = \Delta y_i / \sqrt{1 + \tan^2\theta} = \Delta y_i / \sqrt{1 + \widehat{Q}_n^2}.$$

We let $\delta x_i = R_i \sin\theta$ and $\delta y_i = R_i \cos\theta$. These are the x- and y-components of the perpendicular distance between data point i and the fitting line. We then weight our fitting cost function according to these variances. Therefore, we define the *approximate weighted total least squares (AWTLS)* cost function as

$$\chi^2_{\text{AWTLS}} = \sum_{i=1}^{n} \frac{\delta x_i^2}{\sigma_{x_i}^2} + \frac{\delta y_i^2}{\sigma_{y_i}^2}.$$

Note that $\sin^2\theta = 1 - \cos^2\theta = \widehat{Q}_n^2/(1 + \widehat{Q}_n^2)$. Therefore,

$$\delta x_i^2 = \left(\frac{\Delta y_i^2}{1 + \widehat{Q}_n^2}\right)\left(\frac{\widehat{Q}_n^2}{1 + \widehat{Q}_n^2}\right)$$
$$\delta y_i^2 = \left(\frac{\Delta y_i^2}{1 + \widehat{Q}_n^2}\right)\left(\frac{1}{1 + \widehat{Q}_n^2}\right).$$

Because $\Delta y_i = y_i - \widehat{Q}_n x_i$, we can then write

$$\chi^2_{\text{AWTLS}} = \sum_{i=1}^{n} \frac{(y_i - \widehat{Q}_n x_i)^2}{(1 + \widehat{Q}_n^2)^2}\left(\frac{\widehat{Q}_n^2}{\sigma_{x_i}^2} + \frac{1}{\sigma_{y_i}^2}\right).$$

To verify that AWTLS is an approximation to WTLS in at least some cases, we note that the two cost functions are equal when $\sigma_{x_i} = \sigma_{y_i}$. However, they are not equal when $\sigma_{x_i} = k\sigma_{y_i}$, but this will be corrected in Sect. 4.18.4.

4.18.2 Minimizing the AWTLS cost function

The Jacobian of the AWTLS cost function (found with the help of Mathematica) is

$$\frac{\partial \chi^2_{\text{AWTLS}}}{\partial \widehat{Q}_n} = \frac{2}{(\widehat{Q}_n^2 + 1)^3}\sum_{i=1}^{n} \widehat{Q}_n^4\left(\frac{x_i y_i}{\sigma_{x_i}^2}\right) + \widehat{Q}_n^3\left(\frac{2x_i^2}{\sigma_{x_i}^2} - \frac{x_i^2}{\sigma_{y_i}^2} - \frac{y_i^2}{\sigma_{x_i}^2}\right)$$
$$+ \widehat{Q}_n^2\left(\frac{3x_i y_i}{\sigma_{y_i}^2} - \frac{3x_i y_i}{\sigma_{x_i}^2}\right) + \widehat{Q}_n\left(\frac{x_i^2 - 2y_i^2}{\sigma_{y_i}^2} + \frac{y_i^2}{\sigma_{x_i}^2}\right) + \left(\frac{-x_i y_i}{\sigma_{y_i}^2}\right).$$

We define additional recursive quantities

$$c_{4,n} = \sum_{i=1}^{n} \frac{x_i^2}{\sigma_{x_i}^2}, \qquad c_{5,n} = \sum_{i=1}^{n} \frac{x_i y_i}{\sigma_{x_i}^2}, \qquad c_{6,n} = \sum_{i=1}^{n} \frac{y_i^2}{\sigma_{x_i}^2}.$$

This allows us to write the cost function in terms of recursively computed running summations

$$\frac{\partial \chi^2_{\text{AWTLS}}}{\partial \widehat{Q}_n} = \frac{2}{(\widehat{Q}_n^2 + 1)^3}\Big(c_5\widehat{Q}_n^4 + (2c_4 - c_1 - c_6)\widehat{Q}_n^3$$
$$+ (3c_2 - 3c_5)\widehat{Q}_n^2 + (c_1 - 2c_3 + c_6)\widehat{Q}_n - c_2\Big),$$

where initialization is done by setting $x_0 = 1$, $y_0 = Q_{\text{nom}}$, $\sigma^2_{y_0}$ to a representative value of the uncertainty of the total capacity, and $\sigma^2_{x_0}$ to a representative value of the uncertainty of a difference between two state of charge estimates.

Therefore, we initialize

$$\begin{aligned} c_{1,0} &= 1/\sigma^2_{y_0} & c_{4,0} &= 1/\sigma^2_{x_0} \\ c_{2,0} &= Q_{\text{nom}}/\sigma^2_{y_0} & c_{5,0} &= Q_{\text{nom}}/\sigma^2_{x_0} \\ c_{3,0} &= Q^2_{\text{nom}}/\sigma^2_{y_0} & c_{6,0} &= Q^2_{\text{nom}}/\sigma^2_{x_0}, \end{aligned}$$

and recursively compute

$$\begin{aligned} c_{1,n} &= c_{1,n-1} + x_n^2/\sigma^2_{y_n} & c_{4,n} &= c_{4,n-1} + x_n^2/\sigma^2_{x_n} \\ c_{2,n} &= c_{2,n-1} + x_n y_n/\sigma^2_{y_n} & c_{5,n} &= c_{5,n-1} + x_n y_n/\sigma^2_{x_n} \\ c_{3,n} &= c_{3,n-1} + y_n^2/\sigma^2_{y_n} & c_{6,n} &= c_{6,n-1} + y_n^2/\sigma^2_{x_n}. \end{aligned}$$

We minimize the cost function by setting its Jacobian to zero. Therefore, any of the roots of the quartic equation

$$c_5\widehat{Q}_n^4 + (2c_4 - c_1 - c_6)\widehat{Q}_n^3 + (3c_2 - 3c_5)\widehat{Q}_n^2 + (c_1 - 2c_3 + c_6)\widehat{Q}_n - c_2 = 0 \tag{4.26}$$

is a candidate solution for $\widehat{Q}_n$. We will discuss how to solve Eq. (4.26) for these roots and select the optimal value in Sect. 4.18.3.

When the assumptions made in deriving AWTLS are approximately true, the Hessian yields a good value for the error bounds on the total-capacity estimate. After some straightforward but messy mathematics, we can find the Hessian in terms of the recursive parameters to be

$$\begin{aligned} \frac{\partial^2 \chi^2_{\text{AWTLS}}}{\partial \widehat{Q}_n^2} = \frac{2}{(\widehat{Q}_n^2+1)^4}\Big(&-2c_5\widehat{Q}_n^5 + (3c_1 - 6c_4 + 3c_6)\widehat{Q}_n^4 \\ &+ (-12c_2 + 16c_5)\widehat{Q}_n^3 + (-8c_1 + 10c_3 + 6c_4 - 8c_6)\widehat{Q}_n^2 \\ &+ (12c_2 - 6c_5)\widehat{Q}_n + (c_1 - 2c_3 + c_6)\Big). \end{aligned}$$

Fading memory can be incorporated easily. The cost function is

$$\chi^2_{\text{FMAWTLS}} = \sum_{i=1}^{n} \gamma^{n-i} \frac{(y_i - \widehat{Q}_n x_i)^2}{(1+\widehat{Q}_n^2)^2}\left(\frac{\widehat{Q}_n^2}{\sigma^2_{x_i}} + \frac{1}{\sigma^2_{y_i}}\right).$$

The Jacobian is

$$\begin{aligned} \frac{\partial \chi^2_{\text{AWTLS}}}{\partial \widehat{Q}_n} = \frac{2}{(\widehat{Q}_n^2+1)^3}\sum_{i=1}^{n}\gamma^{n-i}\Bigg[&\widehat{Q}_n^4\left(\frac{x_i y_i}{\sigma^2_{x_i}}\right) + \widehat{Q}_n^3\left(\frac{2x_i^2}{\sigma^2_{x_i}} - \frac{x_i^2}{\sigma^2_{y_i}} - \frac{y_i^2}{\sigma^2_{x_i}}\right) \\ &+ \widehat{Q}_n^2\left(\frac{3x_i y_i}{\sigma^2_{y_i}} - \frac{3x_i y_i}{\sigma^2_{x_i}}\right) + \widehat{Q}_n\left(\frac{x_i^2 - 2y_i^2}{\sigma^2_{y_i}} + \frac{y_i^2}{\sigma^2_{x_i}}\right) + \left(\frac{-x_i y_i}{\sigma^2_{y_i}}\right)\Bigg], \end{aligned}$$

which can be rewritten in terms of recursively computed running summations as

$$\frac{\partial \chi^2_{\text{FMAWTLS}}}{\partial \widehat{Q}_n} = \frac{2}{(\widehat{Q}_n^2+1)^3}\Bigg(\tilde{c}_5\widehat{Q}_n^4 + (-\tilde{c}_1+2\tilde{c}_4-\tilde{c}_6)\widehat{Q}_n^3 + (3\tilde{c}_2-3\tilde{c}_5)\widehat{Q}_n^2 + (\tilde{c}_1-2\tilde{c}_3+\tilde{c}_6)\widehat{Q}_n - \tilde{c}_2\Bigg).$$

Initialization is done by setting $x_0 = 1$, $y_0 = Q_{\text{nom}}$, $\sigma^2_{y_0}$ to a representative value of the uncertainty of the total capacity, and $\sigma^2_{x_0}$ to a representative value of the uncertainty of a difference between two state of charge estimates. Therefore, we initialize

$$\begin{aligned} \tilde{c}_{1,0} &= 1/\sigma^2_{y_0} & \tilde{c}_{4,0} &= 1/\sigma^2_{x_0} \\ \tilde{c}_{2,0} &= Q_{\text{nom}}/\sigma^2_{y_0} & \tilde{c}_{5,0} &= Q_{\text{nom}}/\sigma^2_{x_0} \\ \tilde{c}_{3,0} &= Q^2_{\text{nom}}/\sigma^2_{y_0} & \tilde{c}_{6,0} &= Q^2_{\text{nom}}/\sigma^2_{x_0}, \end{aligned}$$

and recursively compute

$$\begin{aligned} \tilde{c}_{1,n} &= \tilde{c}_{1,n-1} + x_n^2/\sigma^2_{y_n} & \tilde{c}_{4,n} &= \tilde{c}_{4,n-1} + x_n^2/\sigma^2_{x_n} \\ \tilde{c}_{2,n} &= \tilde{c}_{2,n-1} + x_n y_n/\sigma^2_{y_n} & \tilde{c}_{5,n} &= \tilde{c}_{5,n-1} + x_n y_n/\sigma^2_{x_n} \\ \tilde{c}_{3,n} &= \tilde{c}_{3,n-1} + y_n^2/\sigma^2_{y_n} & \tilde{c}_{6,n} &= \tilde{c}_{6,n-1} + y_n^2/\sigma^2_{x_n}. \end{aligned}$$

We minimize the cost function by setting its Jacobian to zero. Therefore, any of the roots of the quartic equation

$$\tilde{c}_5\widehat{Q}_n^4 + (2\tilde{c}_4-\tilde{c}_1-\tilde{c}_6)\widehat{Q}_n^3 + (3\tilde{c}_2-3\tilde{c}_5)\widehat{Q}_n^2 + (\tilde{c}_1-2\tilde{c}_3+\tilde{c}_6)\widehat{Q}_n - \tilde{c}_2 = 0 \tag{4.27}$$

is a candidate solution for $\widehat{Q}_n$. We will discuss how to solve for these roots and select the optimal value in Sect. 4.18.3. The Hessian, which may be used to compute error bounds, is

$$\frac{\partial^2 \chi^2_{\text{FMAWTLS}}}{\partial \widehat{Q}_n^2} = \frac{2}{(\widehat{Q}_n^2+1)^4}\Bigg(-2\tilde{c}_5\widehat{Q}_n^5 + (3\tilde{c}_1-6\tilde{c}_4+3\tilde{c}_6)\widehat{Q}_n^4 + (-12\tilde{c}_2+16\tilde{c}_5)\widehat{Q}_n^3 + (-8\tilde{c}_1+10\tilde{c}_3+6\tilde{c}_4-8\tilde{c}_6)\widehat{Q}_n^2 + (12\tilde{c}_2-6\tilde{c}_5)\widehat{Q}_n + (\tilde{c}_1-2\tilde{c}_3+\tilde{c}_6)\Bigg). \tag{4.28}$$

4.18.3 Solving the quartic equation

To be able to find total capacity, we must find the roots of a quartic equation, either Eq. (4.26) or (4.27). There are several methods that could be used to do so. In the following, we consider three approaches to finding the roots of the generic quartic

$$x^4 + ax^3 + bx^2 + cx + d = 0. \tag{4.29}$$

First, the roots of the quartic may be found by iterative techniques. The set of four roots of the quartic found at time step $n-1$ are used as the initial guess for the set of roots at time step n. Then, one or more Newton–Raphson iterations are performed to refine the set of roots for time step n. While this approach is conceptually simple, there are some subtle difficulties that can turn into big problems. For example, the roots may start out distinct, but over time two or more roots may come together to make a repeated root. Or, a repeated root may split into multiple distinct roots.[20] It is possible to compute how many real roots a polynomial has within an interval via Sturm's theorem,[21] and to search for only those roots using a line search, but the amount of computation required to do so is considerable.

[20] For this reason, it is unwise to use an iterative technique to track a single real root as the total-capacity estimate, because you may be tracking the wrong root! Instead, the entire set of roots should be solved every iteration.

[21] See, for example, Bruce E. Meserve, *Fundamental Concepts of Algebra*, Dover, 1982.

Second, analytic solutions exist to compute the roots of a quartic equation directly. These include Ferrari's method, Cardano's method, and others. At first, it would seem that this is the best approach to solving the quartic—all roots can be found in closed form every time. However, it turns out that all the known methods for solving quartic equations analytically have numeric instabilities.[22] The methods can be shown to be stable for only certain combinations of the signs of $\{a,b,c,d\}$ in Eq. (4.29). Further, while we know that $d<0$ for the problem we are trying to solve, we have observed positive, negative, and zero values for b. In the examples later in this chapter, we encounter only negative values for a and c, but there is no guarantee of this in the general case. In short, using a closed-form analytic solution for the roots of the quartic is problematic, at best.

[22] See, for example, D. Herbison-Evans, "Solving quartics and cubics for graphics," *Graphics Gems V (IBM Version)*, 1995, pp. 3–15.

Third, it turns out that the roots of Eq. (4.29) are the eigenvalues of either of the following *companion matrices*:

$$\begin{bmatrix} -a & 1 & 0 & 0 \\ -b & 0 & 1 & 0 \\ -c & 0 & 0 & 1 \\ -d & 0 & 0 & 0 \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} -a & -b & -c & -d \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \text{or}$$

$$\begin{bmatrix} 0 & 0 & 0 & -d \\ 1 & 0 & 0 & -c \\ 0 & 1 & 0 & -b \\ 0 & 0 & 1 & -a \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -d & -c & -b & -a \end{bmatrix}.$$

While finding the eigenvalues of a general matrix is extremely difficult, it is considerably simpler to find the eigenvalues of a companion matrix.[23] This is also the most stable method numerically and is how MATLAB finds the zeros of a polynomial using its built-in `roots` command. This is probably the best approach for finding the roots of the quartic equation.

[23] See, for example, D.A. Bini, P. Boito, Y. Eidelman, L. Gemignani, and I. Gohberg, "A fast implicit QR eigenvalue algorithm for companion matrices," *Linear Algebra and its Applications*, 432, 2010, pp. 2006–2031.

Once the set of roots is found, however, we still need to decide which root is the one to use as the capacity estimate. Negative and

complex roots can be discarded immediately, but we have sometimes encountered solutions with multiple positive real roots. Which is the capacity estimate?

The only foolproof method of which we are aware is to evaluate χ^2_{AWTLS} at each of the positive-real candidate solutions and to retain the one that gives the lowest computed value. Computing the cost function may be done very readily if we rewrite it in terms of the running summations and the total-capacity estimate. For the approximate weighted TLS solution, we have

$$\chi^2_{\text{AWTLS}} = \frac{1}{(\widehat{Q}_n^2+1)^2}\left(c_4\widehat{Q}_n^4 - 2c_5\widehat{Q}_n^3 + (c_1+c_6)\widehat{Q}_n^2 - 2c_2\widehat{Q}_n + c_3\right). \tag{4.30}$$

Similarly, for the fading memory version, we have

$$\chi^2_{\text{FMAWTLS}} = \frac{1}{(\widehat{Q}_n^2+1)^2}\left(\tilde{c}_4\widehat{Q}_n^4 - 2\tilde{c}_5\widehat{Q}_n^3 + (\tilde{c}_1+\tilde{c}_6)\widehat{Q}_n^2 - 2\tilde{c}_2\widehat{Q}_n + \tilde{c}_3\right). \tag{4.31}$$

4.18.4 *Summary of approximate weighted total least squares*

To use the AWTLS or FMAWTLS solution, we first initialize the six recursive variables. Then, every time a data point becomes available, we update the recursive variables and solve the quartic equation Eq. (4.26) or (4.27) for the four possible capacity estimates. We substitute these capacity estimates back into the cost function Eq. (4.30) or (4.31) and choose the value that gives the lowest cost.

Note that the AWTLS cost function does not equal the WTLS cost function when $\sigma_{x_i} = k\sigma_{y_i}$. This can be remedied easily by defining scaled measurements $\tilde{y}_i = ky_i$. Then $\sigma_{\tilde{y}_i} = \sigma_{x_i}$. We invoke the AWTLS or FMAWTLS methods to find total capacity estimate $\widehat{Q}_n$ and Hessian H_n using input sequences composed of the original $\boldsymbol{x}$ vector and the scaled $\tilde{\mathbf{y}}$ vector (i.e., $(x_i, \tilde{y}_i)$ with corresponding variances $(\sigma^2_{x_i}, k^2\sigma^2_{y_i})$). The true slope estimate can be found as $\widehat{Q}_{n,\text{corrected}} = \widehat{Q}_n/k$ and the corrected Hessian can be found as $H_{n,\text{corrected}} = k^2 H_n$. This is the method used in the simulation results in Sect. 4.20 and 4.21, where the proportionality constant is estimated as $k = \sigma_{x_1}/\sigma_{y_1}$. This scaling improves results even when σ_{y_i} and σ_{x_i} are not proportionally related if k is chosen to give an order of magnitude proportionality or average proportionality between the uncertainties of x_i and y_i.

In summary, these AWTLS solutions share the nice properties of the WLS solution:

1. They give a closed-form solution for $\widehat{Q}_n$. No iteration is required. The only complication is the need for finding the roots of a quartic

polynomial, but a manageable approach to doing so has been proposed.

2. The solution can be computed very easily in a recursive manner. We keep track of the six running sums $c_{1,n}$ through $c_{6,n}$. When an additional data point becomes available, we update the sums and compute an updated total-capacity estimate.
3. Fading memory can be added easily to allow the estimate $\widehat{Q}_n$ to place greater emphasis on more recent measurements than on earlier measurements, allowing adaptation of $\widehat{Q}_n$ to adjust for true cell total capacity changes.
4. Further, this method is superior to the PTLS solution since it allows individual weighting on the x_i and y_i data points.

4.19 *Code to simulate the methods*

The regressive total-capacity estimation algorithms are implemented in the MATLAB code `xLSalgos.m`. The function begins with some comments that describe its input and output arguments:

```
% Tests the recursive performance of the xLS algorithms on a particular
% dataset
% [Qhat,SigmaQ,Fit] = xLSalgos(measX,measY,SigmaX,SigmaY,gamma,Qnom)
%   - measX = noisy z(2)-z(1)
%   - measY = noisy integral(i(t)/3600 dt)
%   - SigmaX = variance of X
%   - SigmaY = variance of Y
%   - gamma = geometric forgetting factor (gamma = 1 for perfect memory)
%   - Qnom = nominal value of Q: if nonzero, used to initialize recursions
%
%   - Qhat = estimate of capacity at every time step
%     - column 1 = WLS - weighted, recursive
%     - column 2 = WTLS - weighted, but not recursive
%     - column 3 = SCTLS - scaled confidence PTLS; recursive and weighted,
%                  but using SigmaX(1) and SigmaY(1) only to determine
%                  factor by which all SigmaX and SigmaY are assumed to be
%                  related
%     - column 4 = AWTLS - recursive and weighted
%   - SigmaQ = variance of Q, computed via Hessian method (columns
%                  correspond to methods in the same way as for Qhat)
%   - Fit = goodness of fit metric for each method (columns
%                  correspond to methods in the same way as for Qhat)
function [Qhat,SigmaQ,Fit]=xLSalgos(measX,measY,SigmaX,SigmaY,gamma,Qnom)
```

Next, memory is reserved for the function output matrices and the proportionality constant k between σ_x and σ_y is estimated as $k = \sigma_{x_1}/\sigma_{y_1}$. The recursive algorithm parameters are initialized to zero. If, however, we have known nonzero Q_{nom}, then we can initialize to better values. Note that uppercase "`C`" variables are scaled fading-memory recursive parameters for use with AWTLS and lowercase "`c`" variables are fading-memory recursive parameters for all other

methods (except for nonrecursive WTLS). The recursive parameters are then updated based on the current measurement.

```
% Reserve some memory
Qhat = zeros(length(measX),4); SigmaQ = Qhat; Fit = Qhat;
K = sqrt(SigmaX(1)/SigmaY(1));

% Initialize some variables used for the recursive methods
c1 = 0; c2 =0; c3 = 0; c4 = 0; c5 = 0; c6 = 0;
C1 = 0; C2 =0; C3 = 0; C4 = 0; C5 = 0; C6 = 0;
if Qnom ~= 0,
  c1 = 1/SigmaY(1); c2 = Qnom/SigmaY(1); c3 = Qnom^2/SigmaY(1);
  c4 = 1/SigmaX(1); c5 = Qnom/SigmaX(1); c6 = Qnom^2/SigmaX(1);
  C1 = 1/(K^2*SigmaY(1)); C2 = K*Qnom/(K^2*SigmaY(1));
  C3 = K^2*Qnom^2/(K^2*SigmaY(1));
  C4 = 1/SigmaX(1); C5 = K*Qnom/SigmaX(1); C6 = K^2*Qnom^2/SigmaX(1);
end

for iter = 1:length(measX),
  % Compute some variables used for the recursive methods
  c1 = gamma*c1 + measX(iter)^2/SigmaY(iter);
  c2 = gamma*c2 + measX(iter)*measY(iter)/SigmaY(iter);
  c3 = gamma*c3 + measY(iter)^2/SigmaY(iter);
  c4 = gamma*c4 + measX(iter)^2/SigmaX(iter);
  c5 = gamma*c5 + measX(iter)*measY(iter)/SigmaX(iter);
  c6 = gamma*c6 + measY(iter)^2/SigmaX(iter);

  C1 = gamma*C1 + measX(iter)^2/(K^2*SigmaY(iter));
  C2 = gamma*C2 + K*measX(iter)*measY(iter)/(K^2*SigmaY(iter));
  C3 = gamma*C3 + K^2*measY(iter)^2/(K^2*SigmaY(iter));
  C4 = gamma*C4 + measX(iter)^2/SigmaX(iter);
  C5 = gamma*C5 + K*measX(iter)*measY(iter)/SigmaX(iter);
  C6 = gamma*C6 + K^2*measY(iter)^2/SigmaX(iter);
```

Next, the fading-memory recursive capacity estimate using WLS is evaluated via Eq. (4.8). Its Hessian is computed via Eq. (4.21) to be able to evaluate the estimate's error-bounds variable `SigmaQ`. The value of the cost function χ^2_{FMWLS} is computed using Eq. (4.9) and used to compute goodness of fit, stored in `Fit`, using MATLAB's implementation of the complementary incomplete gamma function and Eq. (4.19).

```
  % Method 1: WLS
  Q = c2./c1; Qhat(iter,1) = Q;
  H = 2*c1;   SigmaQ(iter,1) = 2/H;
  J = Q.^2.*c1 -2*Q.*c2 + c3;
  Fit(iter,1) = gammainc(J/2,(iter-1)/2,'upper');
```

The next code segment computes the present total-capacity estimate using the WTLS method. This method is not recursive, and requires executing multiple Newton–Raphson search iterations to converge toward the solution every time a new data point is measured. Each WTLS solution is initialized with the WLS estimate just computed, and then the cost-function Jacobian and Hessian matrices are repeatedly computed using Eq. (4.17) and (4.18). These are used with Eq. (4.15) to converge toward the WTLS solution. The optimized

cost-function value is computed via Eq. (4.13) and used to find the goodness of fit of the model.

```
    % Method 2: WTLS -- not recursive
    g = flipud((gamma.^(0:(iter-1)))');   % vector of forgetting factors
    x = measX(1:iter); y = measY(1:iter); % all measurements until now
    sx = sqrt(SigmaX(1:iter));            % all sigma-x until now
    sy = sqrt(SigmaY(1:iter));            % all sigma-y until now
    Q = Qhat(iter,1); % initialize WTLS with WLS total capacity estimate
    for kk = 1:10,    % ten Newton--Raphson iterations
      jacobian = sum(g.*(2*(Ctls2*x-y).*(Ctls2*y.*sx.^2+x.*sy.^2))./...
                     ((Ctls2^2*sx.^2+sy.^2).^2));
      hessian = sum(g.*(2*sy.^4.*x.^2+sx.^4.*...
                    (6*Ctls2^2*y.^2-4*Ctls2^3*x.*y) - ...
                     sx.^2.*sy.^2.*...
                    (6*Ctls2^2*x.^2-12*Ctls2*x.*y+2*y.^2))./...
                    ((Ctls2^2*sx.^2+sy.^2).^3));
      Q = Q - jacobian/hessian;
    end
    Qhat(iter,2) = Q;                     % save capacity estimate
    SigmaQ(iter,2) = 2/hessian;           % save bounds
    J = sum(g.*(y-Q*x).^2./(sx.^2*Q^2+sy.^2)); % cost-function value
    Fit(iter,2) = gammainc(J/2,(2*iter-1)/2,'upper'); % goodness of fit
```

Next, we implement the PTLS method. The total-capacity estimate is computed using Eq. (4.23), its Hessian via Eq. (4.24), and the resulting minimum cost with Eq. (4.25).

```
    % Method 3: PTLS
    Q = (-c1+K^2*c3+sqrt((c1-K^2*c3)^2+4*K^2*c2^2))/(2*K^2*c2);
    Qhat(iter,3) = Q;
    H = ((-4*K^4*c2)*Q^3+6*K^4*c3*Q^2+...
         (-6*c1+12*c2)*K^2*Q+2*(c1-K^2*c3))/(Q^2*K^2+1)^3;
    SigmaQ(iter,3) = 2/H;
    J = (Q^2*c1 -2*Q*c2 + c3)/(Q^2*K^2+1);
    Fit(iter,3) = gammainc(J/2,(2*iter-1)/2,'upper');
```

Finally, we implement the AWTLS method. We first search for roots of the quartic equation of Eq. (4.27). We discard any that are not positive and real. Then, we evaluate the cost function of Eq. (4.31) using the remaining candidate roots and keep the root that minimizes the cost function. The Hessian is computed via Eq. (4.28).

```
    % Method 4: AWTLS with pre-scaling
    r = roots([C5 (-C1+2*C4-C6) (3*C2-3*C5) (C1-2*C3+C6) -C2]);
    r = r(r==conj(r)); % discard complex-conjugate roots
    r = r(r>0); % discard negative roots
    Jr = ((1./(r.^2+1).^2).*(r.^4*C4-2*C5*r.^3+(C1+C6)*r.^2-2*C2*r+C3))';
    J = min(Jr);
    Q = r(Jr==J); % keep Q that minimizes cost function
    H = (2/(Q^2+1)^4)*...
          (-2*C5*Q^5+(3*C1-6*C4+3*C6)*Q^4+(-12*C2+16*C5)*Q^3 ...
          +(-8*C1+10*C3+6*C4-8*C6)*Q^2+(12*C2-6*C5)*Q+(C1-2*C3+C6));
    Qhat(iter,4) = Q/K;
    SigmaQ(iter,4) = 2/H/K^2;
    Fit(iter,4) = gammainc(J/2,(2*iter-1)'/2,'upper');
  end
  Fit = real(Fit);
return
```

4.20 Example HEV simulations

In closing, we examine a number of usage scenarios to exercise the WLS, WTLS, PTLS, and AWTLS total-capacity estimation methods and to compare their performance. All scenarios use the fading-memory version of the four methods, but we omit the prefix "FM" for brevity. Unless otherwise stated, the fading-memory forgetting factor is chosen as $\gamma = 1.0$.

We assume that the individual SOC estimates that are input to these methods can be determined to an accuracy of $\sigma_z = 0.01$. This is being very generous, since the best method of which we are aware, SPKF, achieves only around $\sigma_z = 0.01$ for LMO or NMC cells and $\sigma_z = 0.03$ for LFP cells in practice when Q_{nom} is used instead of Q in the estimator. Other methods that we have used, such as EKF, achieve around $\sigma_z = 0.02$ or higher for LMO cells in practice. A nice advantage of both EKF and SPKF is that they give dynamic estimates of the variance of the state of charge estimate that ensure that the values of σ_{x_i} used in total-capacity estimation are accurate.

We use computer simulation rather than cell testing to validate the algorithms because it allows us to constrain a variety of factors that would be difficult to control in a real-time embedded system. These include:

- The efficacy and accuracy of the SOC estimation algorithms used to provide input to the total-capacity estimation algorithms;
- The accuracy and precision of the raw sensor measurements used as input (including the challenges of bias errors, nonlinear errors, and random errors, for example);
- The repeatability of the experiment; and
- The fact that total capacity of a physical cell fades over time and the associated difficulty or even impossibility of knowing the true value of total capacity against which to compare our results.

Therefore, we choose to use synthetic data to isolate the performance of the total-capacity estimation algorithms themselves, when all other factors are in some sense idealized. The x_i and y_i values are mathematically generated, as described in the individual subsections below.

4.20.1 HEV application, scenario 1

The first sets of simulations that we present are for HEV scenarios. From the perspective of total-capacity estimation, these applications are characterized by the narrow window of battery-cell SOC used.

We assume that the vehicle uses a SOC range of 40 % to 60 %. Therefore, each time the total capacity estimate is updated, the true

change in SOC can range from -0.2 to $+0.2$. We simulate this by choosing the true value of x_i to be a uniform random number selected between these limits.

The HEV application is also characterized by the fact that the battery pack is never fully charged to a precisely known SOC; therefore, each time the total capacity estimate is updated, two estimates of SOC are required to compute $x_i = \hat{z}_{k_2}^{(i)} - \hat{z}_{k_1}^{(i)}$. This gives an overall $\sigma_x^2 = 2\sigma_z^2 = 2(0.01)^2$. We simulate this by computing the measured value of x_i to be equal to the true value of x_i added to a zero-mean Gaussian random number having variance σ_x^2.

We compute the true value of y_i to be equal to the nominal capacity of the cell Q_{nom} multiplied by the true value of x_i. Noise on the y_i measurement is assumed to comprise accumulated quantization noises. For y_i computed by summing m_i measurements, taken at a 1 Hz rate, from a sensor having quantizer resolution q, the total noise is $\sigma_{y_i}^2 = q^2 m_i/(12 \times 3{,}600^2)$.[24] For HEV scenario 1, we assumed that the maximum range of the current sensor is $\pm 30 Q_{\text{nom}}$ and that a 10-bit analog-to-digital converter is used to measure current. This leads to $q = 60 Q_{\text{nom}}/1024$. We chose $m_i = 300\,\text{s}$ for every measurement and a nominal capacity of $Q_{\text{nom}} = 10\,\text{Ah}$.

[24] This is the variance of a uniform random variable distributed between $-q/2$ and $q/2$, scaled by $m_i/3{,}600^2$

To implement this scenario, we use the following code:

```
Q0 = 10;           % initial true cell total capacity
slope = 0;         % total capacity does not change with time
maxI = 30*Q0;      % must be able to measure current up to +/- maxI
precisionI = 1024; % 10-bit precision on current sensor
Qnom = 0;          % initialization for recursive variables
xmax = 0.2;        % maximum change in state of charge per measurement
xmin = -xmax;      % minimum change in state of charge per measurement
m = 300;           % interval length between measurements
socnoise = sqrt(2)*0.01; % sigma_x
gamma = 1;         % fading-memory forgetting factor (no forgetting)
plotTitle = 'HEV Scenario 1'; % for the output plot
runScenario
```

This code calls `runScenario.m`, which we now describe. The first part of this script synthesizes random truth values for x_i and y_i and the corresponding noisy values thereof. Most scenarios have a constant-length interval, corresponding to a scalar value of the `m` variable; however, if `m` is not a scalar, a variable-length interval is required. The details of data generation for a variable-length interval are discussed in Sect. 4.21.2. The second part of the script calls the `xLSalgos.m` function, which has already been described, and plots results:

```
% runScenario.m: run a particular scenario and plot results

% Make up some data
n = 1000;              % number of data points
Q = (Q0+slope*(1:n))'; % true capacity as a function of time
x = ((xmax-xmin)*rand(n,1)+xmin); % true values of "x" measurements
y = Q.*x;              % true values of "y" measurements
```

```
% Add in some noise to both variables.
binsize = 2*maxI/precisionI;
rn1 = ones(n,1);       % ones the size of the "x" measurements
if isscalar(m),        % constant number of measurements per iteration
  rn2 = rn1;           % ones the size of the "y" measurements
  sy = binsize*sqrt(m/12)/3600*rn2; % sigma-y for each measurement
else                   % variable number of measurements per iteration
  mu = log(m(1))+m(2)^2; % input to lognrnd command
  m = 3600*lognrnd(mu,sigma,n,1); % create log-normal rand variables
  sy = binsize*sqrt(m/12)/3600; % sigma-y for each measurement
end
sx = socnoise*rn1;     % sigma-x for each measurement

x = x + sx.*randn(n,1); % noisy measurement of "x"
y = y + sy.*randn(n,1); % noisy measurement of "y"

% Call the xLSalgos.m function. Returned variables have format:
%  - column 1 = WLS - weighted, recursive
%  - column 2 = WTLS - weighted, but not recursive
%  - column 3 = PTLS - recursive weighted, but using SigmaX(1) and
%               SigmaY(1) only (i.e., simplified method using c0 and c1)
%  - column 4 = AWTLS - recursive and weighted
[Qhat,SigmaQ,Fit] = xLSalgos(x,y,sx.^2,sy.^2,gamma,Qnom);
Qrep = repmat(Q,1,size(Qhat,2));

figure(1); clf; plot(Qhat); hold on; box on;
xlabel('Algorithm update index'); ylabel('Capacity estimate (Ah)');
title(sprintf('%s: Capacity Estimates with Error Bounds',...
              plotTitle));
legend('WLS','WTLS','TLS','AWTLS','location','northeast');
plot(Qhat+3*sqrt(SigmaQ),'linewidth',0.5); % error bounds
plot(Qhat-3*sqrt(SigmaQ),'linewidth',0.5);
plot(Qhat);       % overlay true capacity one more time so plots on top
plot(1:n,Q,'k:'); % plot true capacity

figure(2); clf; plot(Fit); hold on; box on;
xlabel('Algorithm update index'); ylabel('Goodness of fit');
title(sprintf('%s: Goodness of fit for each method',plotTitle))
legend('WLS','WTLS','TLS','AWTLS','location','east'); ylim([-0.02 1.02]);
```

The recursive parameters were initialized to zero prior to the first data point being received. Results are presented in Fig. 4.15. The top frame of the figure shows estimates made using the four methods evolving over time as thick lines, and their three-sigma error bounds computed using the Hessian method as thin lines. We see that WTLS, PTLS, and AWTLS give identical estimates and error bounds for this scenario and converge to the neighborhood of the true total capacity. The WLS estimate is biased and its error bounds are (incorrectly) so tight that they are indistinguishable from the estimate itself.

The bottom frame of the figure shows the goodness-of-fit metric as applied to the four methods. Again, WTLS, PTLS, and AWTLS give identical results, quickly converging to a value of 1.0 (that is, these methods are confident that their estimate is reliable). The WLS method returns a vanishingly small value for goodness of fit, re-

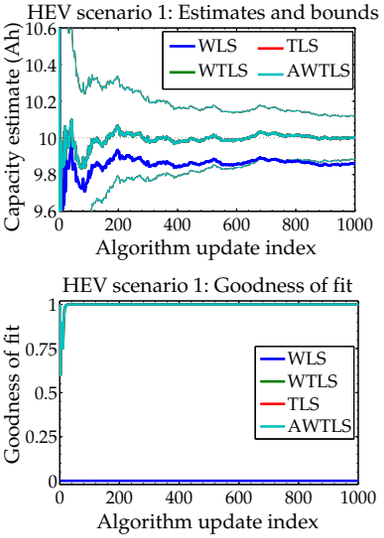

Figure 4.15: Simulation results for HEV scenario 1.
(Reproduced from Fig. 3 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

flecting the fact that the method does not give a good value for its total-capacity estimates and/or the bounds thereon.

4.20.2 *HEV application, scenario 2*

The second HEV scenario is identical to the first except that the recursive methods are initialized with a total capacity estimate before any measurements are received. In this case, the methods were initialized with a nominal capacity estimate of 9.9 Ah (the true total capacity was still 10.0 Ah).

To implement this scenario, we use the following code:

```
Q0 = 10;           % initialize true cell total capacity
slope = 0;         % total capacity does not change with time
maxI = 30*Q0;      % must be able to measure current up to +/- maxI
precisionI = 1024; % 10-bit precision on current sensor
Qnom = 0.99*Q0;    % initialization for recursive variables
xmax = 0.2;        % maximum change in state of charge per measurement
xmin = -xmax;      % minimum change in state of charge per measurement
m = 300;           % interval length between measurements
socnoise = sqrt(2)*0.01; % sigma_x
gamma = 1;         % fading-memory forgetting factor (no forgetting)
plotTitle = 'HEV Scenario 2'; % for the output plot
runScenario
```

Results from running this code are presented in Fig. 4.16. In this scenario, PTLS and AWTLS give identical results for their estimates, error bounds, and goodness of fit. WTLS cannot be calculated recursively, so its estimate cannot be initialized; subsequently, the WTLS results are the same as for scenario 1. Once again, WLS is inferior to the other methods. PTLS and AWTLS give the most accurate and most confident estimates because the ability to initialize the estimate with a reasonable value has resulted in of tighter error bounds while maintaining a high value for goodness of fit.

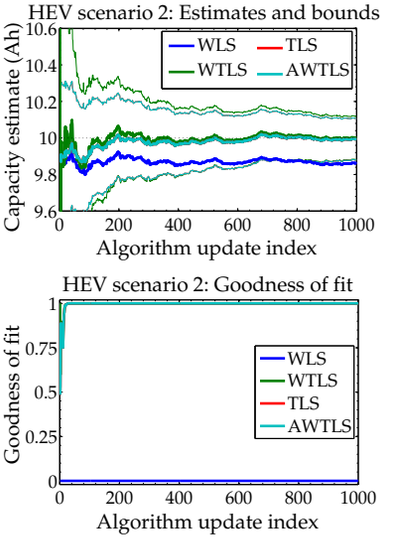

Figure 4.16: Simulation results for HEV scenario 2.
(Reproduced from Fig. 4 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

4.20.3 *HEV application, scenario 3*

In the third HEV scenario, we explore the ability of the algorithms to track a moving value of total capacity. This scenario is identical to HEV scenario 2, except that the true total capacity is changing with a slope of -0.001 Ah per measurement update, and a fading memory forgetting factor of $\gamma = 0.99$ is used for all methods.

To implement this scenario, we use the following code:

```
Q0 = 10;           % initialize true cell total capacity
slope = -0.001;    % true capacity changes this much every iteration
maxI = 30*Q0;      % must be able to measure current up to +/- maxI
precisionI = 1024; % 10-bit precision on current sensor
Qnom = 0.99*Q0;    % initialization of recursive variables
xmax = 0.2;        % maximum change in state of charge per measurement
xmin = -xmax;      % minimum change in state of charge per measurement
m = 300;           % interval length between measurements
```

```
socnoise = sqrt(2)*0.01; % sigma_x
gamma = 0.99;      % fading-memory forgetting factor (slow forgetting)
plotTitle = 'HEV Scenario 3'; % for the output plot
runScenario
```

Results are presented in Fig. 4.17, where the true total capacity is drawn as a dotted black line. In this example, the WLS method *appears* to give good results, but its goodness of fit value is still vanishingly small because the error bounds are unreasonably tight and almost never surround the true value of total capacity. WTLS, PTLS, and AWTLS are also able to track the moving value of total capacity—PTLS and AWTLS give the best results due to the ability to initialize their recursive parameters with reasonable values, yielding better estimates having narrower error bounds.

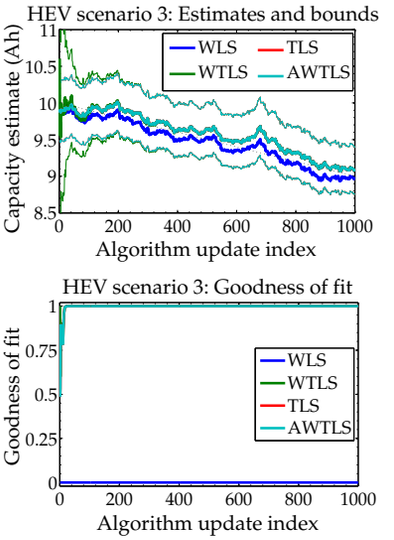

Figure 4.17: Simulation results for HEV scenario 3.
(Reproduced from Fig. 5 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

4.21 Example EV simulations

4.21.1 EV application, scenario 1

The next scenarios that we consider are typical of EV and PHEV (or E-REV) operation. These are different from the HEV application in several respects: the battery total capacity is larger, the relative rate of energy usage is lower, the range of SOC used by the vehicle is larger, and the EV battery pack is sometimes fully charged to a known setpoint.

In all cases, we consider a battery pack having total capacity of $Q = 100\,\text{Ah}$, and a maximum rate of $\pm 5Q$. We again assume a 10-bit current sensor, which gives $q = 10Q_{\text{nom}}/1024$ and $\sigma_{y_i}^2 = q^2 m_i/(12 \times 3600^2)$. For the first EV scenario we assume that the total-capacity estimate is updated on a regular basis as the vehicle operates, with $m_i = 7200\,\text{s}$ (i.e., every 2 h or about 120 mi of highway-driven distance). We assume that the battery SOC can change by $\pm 40\,\%$ in that interval, so the true value of x_i is chosen to be a uniform random variable between -0.4 and $+0.4$. Again, noise on x_i is Gaussian with variance $\sigma_{x_i}^2 = 2(0.01)^2$. The recursive methods are initialized with an initial total-capacity estimate of 99 Ah.

To implement this scenario, we use the following code:

```
Q0 = 100;          % initialize true cell total capacity
slope = 0;         % true capacity changes this much every iteration
maxI = 5*Q0;       % must be able to measure current up to +/- maxI
precisionI = 1024; % 10-bit precision on current sensor
Qnom = 0.99*Q0;    % initialization of recursive variables
xmax = 0.4;        % maximum change in state of charge per measurement
xmin = -xmax;      % minimum change in state of charge per measurement
m = 7200;          % interval length between measurements
socnoise = sqrt(2)*0.01; % sigma_x
gamma = 1;         % fading-memory forgetting factor (no forgetting)
plotTitle = 'EV Scenario 1'; % for the output plot
runScenario
```

Representative results of this scenario are presented in Fig. 4.18. These are very similar in most respects to the HEV scenario 2 results. Again, WLS fails because its error bounds are far too tight, leading to a vanishingly small goodness of fit. WTLS, PTLS, and AWTLS all give good results, with PTLS and AWTLS giving the best results due to lower error bounds because of the possibility of initialization.

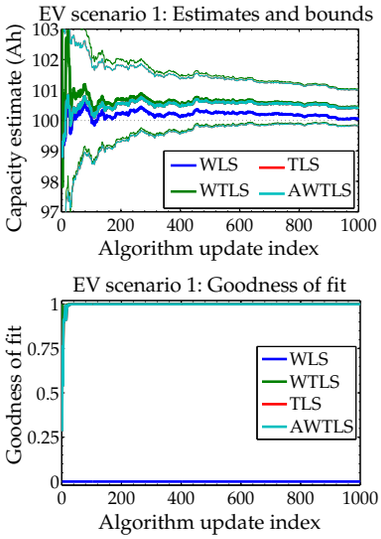

Figure 4.18: Simulation results for EV scenario 1.
(Reproduced from Fig. 6 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

4.21.2 *EV application, scenario 2*

The asymptotic quality of the total-capacity estimates is limited by the noise on the state of charge estimation error. If this noise can be reduced, the total-capacity estimates can become much more accurate. The EV application allows a means to do this: whenever the battery pack is fully charged, we have a precisely known endpoint SOC. Therefore, either $\hat{z}_{k_2}$ or $\hat{z}_{k_1}$ can be known exactly for every total capacity estimate update. This then allows us to use $\sigma_{x_i}^2 = \sigma_z^2 = (0.01)^2$, which is half that used when neither SOC end point is known exactly.

The tradeoff is that we no longer have regular updates. Instead, updates happen randomly, whenever the vehicle is charged. Therefore, m_i becomes a random variable. For the results presented in this section, we treat m_i as a lognormal random variable with mode 0.5 h and standard deviation 0.6 h. This pdf is plotted in Fig. 4.19. Other pdfs could easily be used: this one was chosen to give reasonable-duration drive cycles that encompass a variety of driving behaviors and distances. Also, since a greater fraction of the battery pack would be used for an entire drive cycle than for a regular periodic update, we use an 80 % range of SOC, so the true value of x_i is computed to be a uniform random number from -0.8 to $+0.8$.

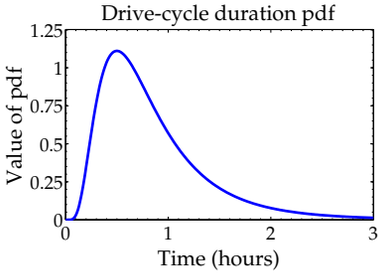

Figure 4.19: Example drive-duration pdf.
(Reproduced from Fig. 7 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

To implement this scenario, we use the following code:

```
Q0 = 100;          % initialize true cell total capacity
slope = 0;         % true capacity changes this much every iteration
maxI = 5*Q0;       % must be able to measure current up to +/- maxI
precisionI = 1024; % 10-bit precision on current sensor
Qnom = 0.99*Q0;    % initialization of recursive variables
xmax = 0.8;        % maximum change in state of charge per measurement
xmin = -xmax;      % minimum change in state of charge per measurement
m = [0.5, 0.6];    % mode = 0.5; sigma = 0.6 in pdf
socnoise = 0.01;   % sigma_x
gamma = 1;         % fading-memory forgetting factor (no forgetting)
plotTitle = 'EV Scenario 2'; % for the output plot
runScenario
```

Results from this scenario are presented in Fig. 4.20. WLS fails once again. However, this time PTLS also fails because $\sigma_{x_i} \neq k\sigma_{y_i}$ due to the variable-length drive cycles. The estimate given by PTLS is actually quite reasonable, but the goodness of fit is very small. WTLS gives good results, and AWTLS gives the best results (based on its

lower error bounds) because of the ability to initialize the estimate. Note that the asymptotic three-sigma error bounds drop from about $\pm 1\,\%$ for HEV scenario 1 to about $\pm 0.15\,\%$ of the true total capacity in EV scenario 2 due to having a lower value of σ_{x_i} and also due to the wider range in x_i.

4.21.3 *EV application, scenario 3*

The final scenario that we consider is identical to EV scenario 2, except that we simulate a changing total capacity. The slope of the total capacity curve is chosen to be -0.01 Ah per measurement update, and $\gamma = 0.98$ was used.

To implement this scenario, we use the following code:

```
Q0 = 100;          % initialize true cell total capacity
slope = -0.01;     % true capacity changes this much every iteration
maxI = 5*Q0;       % must be able to measure current up to +/- maxI
precisionI = 1024; % 10-bit precision on current sensor
Qnom = 0.99*Q0;    % initialization of recursive variables
xmax = 0.8;        % maximum change in state of charge per measurement
xmin = -xmax;      % minimum change in state of charge per measurement
m = [0.5, 0.6];    % mode = 0.5; sigma = 0.6 in pdf
socnoise = 0.01;   % sigma_x
gamma = 0.98;      % fading-memory forgetting factor (slow forgetting)
plotTitle = 'EV Scenario 3'; % for the output plot
runScenario
```

Representative results of this scenario are presented in Fig. 4.21. Once again, WLS fails and PTLS is uncertain of its estimate for nearly 100 updates. However, PTLS does recover and do quite well. The AWTLS method gives the best results.

4.22 *Discussion of simulations*

The simulation results have illustrated a few key properties of the four methods we have introduced to estimate total capacity. They confirm that noise on the state of charge estimates used as input to the total-capacity estimator must be considered in order to estimate battery-cell total capacity properly. Least squares, weighted least squares, and other similar methods that do not consider this noise simply fail. They give biased estimates of total capacity and have unreliable error bounds. Methods related to total least squares, where noises on the SOC estimates are explicitly recognized and incorporated in the calculations, are required for reliable total-capacity estimation.

In principle, WTLS always gives the best results. However, we have seen that the PTLS and AWTLS methods can give better results in practice because they can be initialized with a nominal-capacity

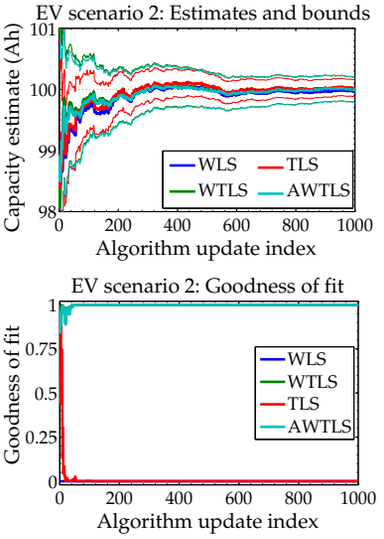

Figure 4.20: Simulation results for EV scenario 2.
(Reproduced from Fig. 8 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

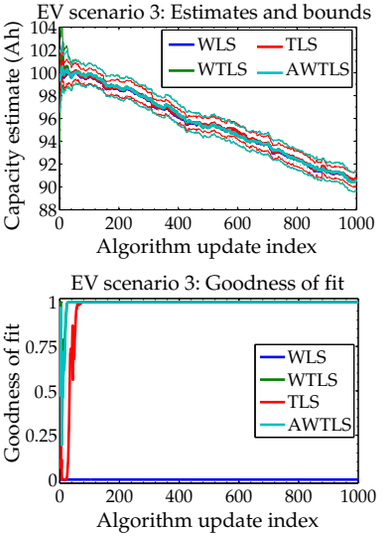

Figure 4.21: Simulation results for EV scenario 3.
(Reproduced from Fig. 9 of Plett, G.L., "Recursive Approximate Weighted Total Least Squares Estimation of Battery Cell Total Capacity," *Journal of Power Sources*, 196(4), 2011, pp. 2,319–31.)

estimate. Furthermore, since PTLS and AWTLS give nice recursive solutions, one of them should always be used instead of WTLS.

If the measurement update interval m_i is constant and therefore $\sigma_{x_i} = k\sigma_{y_i}$ for all measurements, PTLS and AWTLS give identical results. Therefore, the simpler PTLS method is preferred. However, if $\sigma_{x_i} \neq k\sigma_{y_i}$, the AWTLS method gives better results than PTLS and sometimes PTLS fails. This is particularly important for the EV application where updates to the estimate of total capacity are done when charging the battery: these yield a greatly improved total-capacity estimate because of the reduction in σ_{x_i} due to knowing one SOC value exactly. AWTLS always gives results at least as good as the other methods.

The noises that contribute to σ_{y_i} are assumed to be due to current-sensor errors. In practice, these can include gain errors, bias errors, noise errors, and nonlinear errors. We have considered only noise errors here. Gain errors and nonlinear errors will bias all of the methods; however, we believe that the biased value of the total-capacity estimate will be consistent with the perceived capacity of the battery cell if the same current sensor is used to compute the battery-cell total-capacity estimate and to monitor pack operations. Bias error can be subtracted in a EV setting if we can assume that the coulombic efficiency of the cells is $\eta \approx 1$ by matching the discharged ampere-hours from usage with the charged ampere-hours when charging.

The error bounds on the total-capacity estimate, even with the optimum WTLS estimator, are larger than some might expect. This underscores the need for a method that predicts not only the estimate but also dynamic error bounds on the estimate, as do the ones proposed in this chapter. Without dynamic error bounds, the user of the total-capacity estimate has no idea how good or bad that estimate is. If the estimate is used to compute battery-pack available energy, for example, the energy estimate may be overly optimistic or overly pessimistic, neither of which is acceptable.

4.23 Where to from here?

We have now looked at the fundamental state- and parameter-estimation problems that must be solved by a BMS. These estimates can be fed into energy- and power-estimation algorithms. We have already seen some simple examples of this in Chap. 1. We will look at more advanced methods in Chaps. 6 and 7. Next, however, we look at the somewhat simpler but still very important issue of balancing or equalizing the cells in a battery pack.

4.24 Appendices: Nonlinear Kalman-filter algorithms

The following pages have summary tables for the nonlinear Kalman-filter based algorithms developed in this chapter.

Appendix: Nonlinear EKF for parameter estimation

Nonlinear state-space model:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{r}_k,$$
$$\boldsymbol{d}_k = \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_k, \boldsymbol{e}_k).$$

where $\boldsymbol{r}_k$ and $\boldsymbol{e}_k$ are independent Gaussian noise processes with means zero and $\bar{\boldsymbol{e}}$, respectively, and having covariance matrices $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$ and $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{e}}}$, respectively.

Definitions:

$$\hat{\boldsymbol{C}}_k^{\theta} = \left.\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x}_k,\boldsymbol{u}_k,\boldsymbol{\theta},\boldsymbol{e}_k)}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k^-} \qquad \hat{\boldsymbol{D}}_k^{\theta} = \left.\frac{\mathrm{d}\boldsymbol{g}(\boldsymbol{x}_k,\boldsymbol{u}_k,\boldsymbol{\theta},\boldsymbol{e}_k)}{\mathrm{d}\boldsymbol{e}_k}\right|_{\boldsymbol{e}_k=\bar{\boldsymbol{e}}_k}$$

Caution: Be careful to compute $\hat{\boldsymbol{C}}_k^{\theta}$ using the recursive chain rule described in the chapter!

Initialization: For $k = 0$, set

$$\hat{\boldsymbol{\theta}}_0^+ = \mathbb{E}[\boldsymbol{\theta}_0]$$
$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},0}^+ = \mathbb{E}\big[(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)^T\big].$$
$$\frac{\mathrm{d}\boldsymbol{x}_0}{\mathrm{d}\boldsymbol{\theta}} = 0, \text{ unless side information is available.}$$

Computation: For $k = 1, 2, \ldots$ compute:

Param.-prediction time update: $\hat{\boldsymbol{\theta}}_k^- = \hat{\boldsymbol{\theta}}_{k-1}^+$

Error-covariance time update: $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k-1}^+ + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$

Kalman gain matrix:
$$\boldsymbol{L}_k^{\theta} = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^-(\hat{\boldsymbol{C}}_k^{\theta})^T[\hat{\boldsymbol{C}}_k^{\theta}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^-(\hat{\boldsymbol{C}}_k^{\theta})^T + \hat{\boldsymbol{D}}_k^{\theta}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{e}}}(\hat{\boldsymbol{D}}_k^{\theta})^T]^{-1}$$

Param.-estimate meas. update: $\hat{\boldsymbol{\theta}}_k^+ = \hat{\boldsymbol{\theta}}_k^- + \boldsymbol{L}_k^{\theta}[\boldsymbol{d}_k - \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \hat{\boldsymbol{\theta}}_k^-, \bar{\boldsymbol{e}}_k)]$

Error-covariance meas. update: $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^+ = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- - \boldsymbol{L}_k^{\theta}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k}(\boldsymbol{L}_k^{\theta})^T$

Appendix: Nonlinear SPKF for parameter estimation

Nonlinear state-space model:

$$\begin{aligned}\boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \boldsymbol{r}_k,\\ \boldsymbol{d}_k &= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_k, \boldsymbol{e}_k).\end{aligned}$$

where $\boldsymbol{r}_k$ and $\boldsymbol{e}_k$ are independent Gaussian noise processes with means zero and $\bar{\boldsymbol{e}}$, respectively, and having covariance matrices $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$ and $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{e}}}$, respectively.

Definitions: Let

$$\boldsymbol{\theta}_k^a = \left[\boldsymbol{\theta}_k^T,\ \boldsymbol{e}_k^T\right]^T, \quad \boldsymbol{\mathcal{W}}_k^a = \left[(\boldsymbol{\mathcal{W}}_k^\theta)^T,\ (\boldsymbol{\mathcal{W}}_k^e)^T\right]^T, \quad p = 2 \times \dim(\boldsymbol{\theta}_k^a).$$

Initialization: For $k = 0$, set

$$\begin{aligned}\hat{\boldsymbol{\theta}}_0^+ &= \mathbb{E}\left[\boldsymbol{\theta}_0\right] & \hat{\boldsymbol{\theta}}_0^{a,+} &= \mathbb{E}\left[\boldsymbol{\theta}_0^a\right] = \left[(\hat{\boldsymbol{\theta}}_0^+)^T,\ \bar{\boldsymbol{e}}\right]^T\\ \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},0}^+ &= \mathbb{E}\left[(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)^T\right] & \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},0}^{a,+} &= \mathbb{E}\left[(\boldsymbol{\theta}_0^a - \hat{\boldsymbol{\theta}}_0^{a,+})(\boldsymbol{\theta}_0^a - \hat{\boldsymbol{\theta}}_0^{a,+})^T\right]\\ & & &= \mathrm{diag}\left(\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},0}^+, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{e}}}\right).\end{aligned}$$

Computation: For $k = 1, 2, \ldots$ compute:

Parameter-prediction time update:

$$\hat{\boldsymbol{\theta}}_k^- = \hat{\boldsymbol{\theta}}_{k-1}^+$$

Error-covariance time update:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k-1}^+ + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$$

Output estimate:

$$\begin{aligned}\boldsymbol{\mathcal{W}}_k^{a,-} &= \left\{\hat{\boldsymbol{\theta}}_k^{a,-},\ \hat{\boldsymbol{\theta}}_k^{a,-} + \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^{a,-}},\ \hat{\boldsymbol{\theta}}_k^{a,-} - \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^{a,-}}\right\}\\ \boldsymbol{\mathcal{D}}_{k,i} &= \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\mathcal{W}}_{k,i}^{\theta,-}, \boldsymbol{\mathcal{W}}_{k,i}^{e,-})\\ \hat{\boldsymbol{d}}_k &= \textstyle\sum_{i=0}^p \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{D}}_{k,i}\end{aligned}$$

Estimator gain matrix:

$$\begin{aligned}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k} &= \textstyle\sum_{i=0}^p \alpha_i^{(\mathrm{c})} \left(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\right) \times \left(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\right)^T\\ \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}\tilde{\boldsymbol{d}},k}^- &= \textstyle\sum_{i=0}^p \alpha_i^{(\mathrm{c})} \left(\boldsymbol{\mathcal{W}}_{k,i}^{\theta,-} - \hat{\boldsymbol{\theta}}_k^-\right) \times \left(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\right)^T\\ \boldsymbol{L}_k^\theta &= \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}}\tilde{\boldsymbol{d}},k}^- \boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k}^{-1}\end{aligned}$$

Parameter-estimate meas. update:

$$\hat{\boldsymbol{\theta}}_k^+ = \hat{\boldsymbol{\theta}}_k^- + \boldsymbol{L}_k^\theta\left(\boldsymbol{d}_k - \hat{\boldsymbol{d}}_k\right)$$

Error-covariance meas. update:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^+ = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- - \boldsymbol{L}_k^\theta \boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k} (\boldsymbol{L}_k^\theta)^T$$

Appendix: Joint EKF for state and parameter estimation

State-space model:

$$\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{\theta}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}, \boldsymbol{\theta}_{k-1}) \\ \boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1} \end{bmatrix}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{\theta}_k)$$

or

$$\mathbf{X}_k = \mathbf{F}(\mathbf{X}_{k-1}, \boldsymbol{u}_{k-1}, \mathbf{W}_{k-1})$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\mathbf{X}_k, \boldsymbol{u}_k, \boldsymbol{v}_k),$$

where $\boldsymbol{w}_k$, $\boldsymbol{r}_k$, and $\boldsymbol{v}_k$ are independent, Gaussian noise processes having means $\bar{\boldsymbol{w}}$, zero, and $\bar{\boldsymbol{v}}$, and covariance matrices $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}$, $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$, and $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}}$, respectively. For brevity, we let $\mathbf{X}_k = \left[\boldsymbol{x}_k^T, \ \boldsymbol{\theta}_k^T\right]^T$, $\mathbf{W}_k = \left[\boldsymbol{w}_k^T, \ \boldsymbol{r}_k^T\right]^T$ and $\boldsymbol{\Sigma}_{\widetilde{\mathbf{W}}} = \mathrm{diag}(\boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}})$.

Definitions:

$$\hat{\boldsymbol{A}}_k = \left.\frac{\mathrm{d}\mathbf{F}(\mathbf{X}_k, \boldsymbol{u}_k, \mathbf{W}_k)}{\mathrm{d}\mathbf{X}_k}\right|_{\mathbf{X}_k = \hat{\mathbf{X}}_k^+} \qquad \hat{\boldsymbol{B}}_k = \left.\frac{\mathrm{d}\mathbf{F}(\mathbf{X}_k, \boldsymbol{u}_k, \mathbf{W}_k)}{\mathrm{d}\mathbf{W}_k}\right|_{\mathbf{W}_k = \bar{\mathbf{W}}_k}$$

$$\hat{\boldsymbol{C}}_k = \left.\frac{\mathrm{d}\boldsymbol{h}(\mathbf{X}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)}{\mathrm{d}\mathbf{X}_k}\right|_{\mathbf{X}_k = \hat{\mathbf{X}}_k^-} \qquad \hat{\boldsymbol{D}}_k = \left.\frac{\mathrm{d}\boldsymbol{h}(\mathbf{X}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)}{\mathrm{d}\boldsymbol{v}_k}\right|_{\boldsymbol{v}_k = \bar{\boldsymbol{v}}_k}.$$

Initialization: For $k = 0$, set

$$\hat{\mathbf{X}}_0^+ = \mathbb{E}\left[\mathbf{X}_0\right]$$

$$\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},0}^+ = \mathbb{E}\left[(\mathbf{X}_0 - \hat{\mathbf{X}}_0^+)(\mathbf{X}_0 - \hat{\mathbf{X}}_0^+)^T\right]$$

Computation: For $k = 1, 2, \ldots$ compute:

State estimate time update: $\hat{\mathbf{X}}_k^- = \mathbf{F}(\hat{\mathbf{X}}_{k-1}^+, \boldsymbol{u}_{k-1}, \bar{\mathbf{W}}_{k-1})$

Error covariance time update: $\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^- = \hat{\boldsymbol{A}}_{k-1} \boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k-1}^+ \hat{\boldsymbol{A}}_{k-1}^T + \hat{\boldsymbol{B}}_{k-1} \boldsymbol{\Sigma}_{\widetilde{\mathbf{W}}} \hat{\boldsymbol{B}}_{k-1}^T$

Output estimate: $\hat{\boldsymbol{y}}_k = \boldsymbol{h}(\hat{\mathbf{X}}_k^-, \boldsymbol{u}_k, \bar{\boldsymbol{v}}_k)$

Estimator gain matrix: $\boldsymbol{L}_k = \boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^- \hat{\boldsymbol{C}}_k^T [\hat{\boldsymbol{C}}_k \boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^- \hat{\boldsymbol{C}}_k^T + \hat{\boldsymbol{D}}_k \boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}} \hat{\boldsymbol{D}}_k^T]^{-1}$

State estimate meas. update: $\hat{\mathbf{X}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{L}_k (\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)$

Error covariance meas. update: $\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^+ = \boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^- - \boldsymbol{L}_k \boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}},k} \boldsymbol{L}_k^T$

Appendix: Dual EKF for state and parameter estimation

Nonlinear state-space models:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_k, \boldsymbol{w}_k) \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_k, \boldsymbol{v}_k) \end{aligned} \quad \text{and} \quad \begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \boldsymbol{r}_k, \\ \boldsymbol{d}_k &= \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\theta}_k, \boldsymbol{e}_k). \end{aligned}$$

where $\boldsymbol{w}_k$, $\boldsymbol{v}_k$, $\boldsymbol{r}_k$ and $\boldsymbol{e}_k$ are independent Gaussian noise processes with means $\bar{\boldsymbol{w}}$, $\bar{\boldsymbol{v}}$, zero, and $\bar{\boldsymbol{e}}$ and covariance matrices $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}$, $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}}$, $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$ and $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{e}}}$, respectively.

Definitions:

$$\hat{\boldsymbol{A}}_k = \left.\frac{\mathrm{d}\boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{u}_k,\hat{\boldsymbol{\theta}}_k^-,\boldsymbol{w}_k)}{\mathrm{d}\boldsymbol{x}_k}\right|_{\boldsymbol{x}_k=\hat{\boldsymbol{x}}_k^+} \qquad \hat{\boldsymbol{B}}_k = \left.\frac{\mathrm{d}\boldsymbol{f}(\boldsymbol{x}_k,\boldsymbol{u}_k,\hat{\boldsymbol{\theta}}_k^-,\boldsymbol{w}_k)}{\mathrm{d}\boldsymbol{w}_k}\right|_{\boldsymbol{w}_k=\bar{\boldsymbol{w}}}$$

$$\hat{\boldsymbol{C}}_k^x = \left.\frac{\mathrm{d}\boldsymbol{h}(\boldsymbol{x}_k,\boldsymbol{u}_k,\hat{\boldsymbol{\theta}}_k^-,\boldsymbol{v}_k)}{\mathrm{d}\boldsymbol{x}_k}\right|_{\boldsymbol{x}_k=\hat{\boldsymbol{x}}_k^-} \qquad \hat{\boldsymbol{D}}_k^x = \left.\frac{\mathrm{d}\boldsymbol{h}(\boldsymbol{x}_k,\boldsymbol{u}_k,\hat{\boldsymbol{\theta}}_k^-,\boldsymbol{v}_k)}{\mathrm{d}\boldsymbol{v}_k}\right|_{\boldsymbol{v}_k=\bar{\boldsymbol{v}}}$$

$$\hat{\boldsymbol{C}}_k^\theta = \left.\frac{\mathrm{d}\boldsymbol{g}(\hat{\boldsymbol{x}}_k^-,\boldsymbol{u}_k,\boldsymbol{\theta},\boldsymbol{e}_k)}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k^-} \qquad \hat{\boldsymbol{D}}_k^\theta = \left.\frac{\mathrm{d}\boldsymbol{g}(\hat{\boldsymbol{x}}_k^-,\boldsymbol{u}_k,\boldsymbol{\theta},\boldsymbol{e}_k)}{\mathrm{d}\boldsymbol{e}_k}\right|_{\boldsymbol{e}_k=\bar{\boldsymbol{e}}}.$$

Initialization: For $k = 0$, set

$$\hat{\boldsymbol{\theta}}_0^+ = \mathbb{E}[\boldsymbol{\theta}_0], \qquad \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},0}^+ = \mathbb{E}\big[(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)^T\big],$$

$$\hat{\boldsymbol{x}}_0^+ = \mathbb{E}[\boldsymbol{x}_0], \qquad \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},0}^+ = \mathbb{E}\big[(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+)(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+)^T\big].$$

Computation: For $k = 1, 2, \ldots$ compute:

Param.-pred. time update:

$$\hat{\boldsymbol{\theta}}_k^- = \hat{\boldsymbol{\theta}}_{k-1}^+$$

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k-1}^+ + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$$

State-pred. time update:

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1}, \hat{\boldsymbol{\theta}}_k^-, \bar{\boldsymbol{w}})$$

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k}^- = \hat{\boldsymbol{A}}_{k-1}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k-1}^+\hat{\boldsymbol{A}}_{k-1}^T + \hat{\boldsymbol{B}}_{k-1}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}\hat{\boldsymbol{B}}_{k-1}^T$$

State filter meas. update:

$$\boldsymbol{L}_k^x = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k}^-(\hat{\boldsymbol{C}}_k^x)^T\big[\hat{\boldsymbol{C}}_k^x\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k}^-(\hat{\boldsymbol{C}}_k^x)^T + \hat{\boldsymbol{D}}_k^x\boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}}(\hat{\boldsymbol{D}}_k^x)^T\big]^{-1}$$

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{L}_k^x\big[\boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-, \boldsymbol{u}_k, \hat{\boldsymbol{\theta}}_k^-, \bar{\boldsymbol{v}})\big]$$

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k}^+ = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k}^- - \boldsymbol{L}_k^x\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}},k}(\boldsymbol{L}_k^x)^T$$

Param.-est. meas. update:

$$\boldsymbol{L}_k^\theta = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^-(\hat{\boldsymbol{C}}_k^\theta)^T\big[\hat{\boldsymbol{C}}_k^\theta\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^-(\hat{\boldsymbol{C}}_k^\theta)^T + \hat{\boldsymbol{D}}_k^\theta\boldsymbol{\Sigma}_{\tilde{\boldsymbol{e}}}(\hat{\boldsymbol{D}}_k^\theta)^T\big]^{-1}$$

$$\hat{\boldsymbol{\theta}}_k^+ = \hat{\boldsymbol{\theta}}_k^- + \boldsymbol{L}_k^\theta\big[\boldsymbol{y}_k - \boldsymbol{g}(\hat{\boldsymbol{x}}_k^-, \boldsymbol{u}_k, \hat{\boldsymbol{\theta}}_k^-, \bar{\boldsymbol{e}})\big]$$

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^+ = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- - \boldsymbol{L}_k^\theta\boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k}(\boldsymbol{L}_k^\theta)^T$$

Appendix: Joint SPKF for state and parameter estimation

State-space model:

$$\begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{\theta}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}, \boldsymbol{\theta}_{k-1}) \\ \boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1} \end{bmatrix}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, \boldsymbol{\theta}_k)$$

or

$$\mathbf{X}_k = \mathbf{F}(\mathbf{X}_{k-1}, \boldsymbol{u}_{k-1}, \mathbf{W}_{k-1})$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\mathbf{X}_k, \boldsymbol{u}_k, \boldsymbol{v}_k),$$

where $\boldsymbol{w}_k$, $\boldsymbol{r}_k$, and $\boldsymbol{v}_k$ are independent, Gaussian noise processes with means $\bar{\boldsymbol{w}}$, zero, and $\bar{\boldsymbol{v}}$, and covariance matrices $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}$, $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$, and $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}}$, respectively. For brevity, we let $\mathbf{X}_k = \left[\boldsymbol{x}_k^T,\ \boldsymbol{\theta}_k^T\right]^T$, $\mathbf{W}_k = \left[\boldsymbol{w}_k^T,\ \boldsymbol{r}_k^T\right]^T$, and $\boldsymbol{\Sigma}_{\widetilde{\mathbf{W}}} = \mathrm{diag}(\boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}})$.

Definitions: Let

$$\mathbf{X}_k^a = \left[\mathbf{X}_k^T,\ \mathbf{W}_k^T,\ \boldsymbol{v}_k^T\right]^T, \quad \boldsymbol{\mathcal{X}}_k^a = \left[(\boldsymbol{\mathcal{X}}_k^{\mathbf{X}})^T,\ (\boldsymbol{\mathcal{X}}_k^{\mathbf{W}})^T,\ (\boldsymbol{\mathcal{X}}_k^{v})^T\right]^T,$$

$$p = 2 \times \dim(\mathbf{X}_k^a).$$

Initialization: For $k = 0$, set

$$\hat{\mathbf{X}}_0^+ = \mathbb{E}\left[\mathbf{X}_0\right] \qquad \hat{\mathbf{X}}_0^{a,+} = \mathbb{E}\left[\mathbf{X}_0^a\right] = \left[(\hat{\mathbf{X}}_0^+)^T,\ \bar{\mathbf{W}},\ \bar{\boldsymbol{v}}\right]^T$$

$$\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},0}^+ = \mathbb{E}\left[(\mathbf{X}_0 - \hat{\mathbf{X}}_0^+)(\mathbf{X}_0 - \hat{\mathbf{X}}_0^+)^T\right] \qquad \boldsymbol{\Sigma}_{\tilde{\mathbf{X}},0}^{a,+} = \mathbb{E}\left[(\mathbf{X}_0^a - \hat{\mathbf{X}}_0^{a,+})(\mathbf{X}_0^a - \hat{\mathbf{X}}_0^{a,+})^T\right]$$

$$= \mathrm{diag}\left(\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},0}^+, \boldsymbol{\Sigma}_{\widetilde{\mathbf{W}}}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}}\right).$$

Computation: For $k = 1, 2, \ldots$ compute:

State pred. time update:

$$\boldsymbol{\mathcal{X}}_{k-1}^{a,+} = \left\{\hat{\mathbf{X}}_{k-1}^{a,+},\ \hat{\mathbf{X}}_{k-1}^{a,+} + \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k-1}^{a,+}},\ \hat{\mathbf{X}}_{k-1}^{a,+} - \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k-1}^{a,+}}\right\}$$

$$\boldsymbol{\mathcal{X}}_{k,i}^{\mathbf{X},-} = \mathbf{F}(\boldsymbol{\mathcal{X}}_{k-1,i}^{\mathbf{X},+}, \boldsymbol{u}_{k-1}, \boldsymbol{\mathcal{X}}_{k-1,i}^{\mathbf{W},+})$$

$$\hat{\mathbf{X}}_k^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{X}}_{k,i}^{\mathbf{X},-}$$

Error covar. time update:

$$\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \left(\boldsymbol{\mathcal{X}}_{k,i}^{\mathbf{X},-} - \hat{\mathbf{X}}_k^-\right)\left(\boldsymbol{\mathcal{X}}_{k,i}^{\mathbf{X},-} - \hat{\mathbf{X}}_k^-\right)^T$$

Output estimate:

$$\boldsymbol{\mathcal{Y}}_{k,i} = \boldsymbol{h}(\boldsymbol{\mathcal{X}}_{k,i}^{\mathbf{X},-}, \boldsymbol{u}_k, \boldsymbol{\mathcal{X}}_{k-1,i}^{v,+})$$

$$\hat{\boldsymbol{y}}_k = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{Y}}_{k,i}$$

Estimator gain matrix:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}},k} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \left(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k\right)\left(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k\right)^T$$

$$\boldsymbol{\Sigma}_{\tilde{\mathbf{X}}\tilde{\boldsymbol{y}},k}^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \left(\boldsymbol{\mathcal{X}}_{k,i}^{\mathbf{X},-} - \hat{\mathbf{X}}_k^-\right)\left(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k\right)^T$$

$$\boldsymbol{L}_k = \boldsymbol{\Sigma}_{\tilde{\mathbf{X}}\tilde{\boldsymbol{y}},k}^- \boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}},k}^{-1}$$

State est. meas. update:

$$\hat{\mathbf{X}}_k^+ = \hat{\mathbf{X}}_k^- + \boldsymbol{L}_k\left(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right)$$

Error covar. meas. update:

$$\boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^+ = \boldsymbol{\Sigma}_{\tilde{\mathbf{X}},k}^- - \boldsymbol{L}_k \boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}},k} \boldsymbol{L}_k^T$$

Appendix: Dual SPKF for state and parameter estimation

Nonlinear state-space models:

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, w_{k-1}, \boldsymbol{\theta}_{k-1})$$
$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k, v_k, \boldsymbol{\theta}_k)$$

and

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{r}_{k-1},$$
$$\boldsymbol{d}_k = \boldsymbol{h}(\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \bar{\boldsymbol{w}}_{k-1}, \boldsymbol{\theta}_{k-1}),\ \boldsymbol{u}_k, \bar{\boldsymbol{v}}_k, \boldsymbol{\theta}_{k-1}, \boldsymbol{e}_k).$$

where $\boldsymbol{w}_k$, $\boldsymbol{v}_k$, $\boldsymbol{r}_k$ and $\boldsymbol{e}_k$ are independent, Gaussian noise processes with means $\bar{\boldsymbol{w}}$, $\bar{\boldsymbol{v}}$, zero, and $\bar{\boldsymbol{e}}$, and covariance matrices $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}$, $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}}$, $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$ and $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{e}}}$, respectively.

Definitions:

$$\boldsymbol{x}_k^a = \left[\boldsymbol{x}_k^T,\ \boldsymbol{w}_k^T,\ \boldsymbol{v}_k^T\right]^T, \quad \boldsymbol{\mathcal{X}}_k^a = \left[(\boldsymbol{\mathcal{X}}_k^x)^T,\ (\boldsymbol{\mathcal{X}}_k^w)^T,\ (\boldsymbol{\mathcal{X}}_k^v)^T\right]^T,$$
$$p = 2 \times \dim(\boldsymbol{x}_k^a).$$

Initialization: For $k = 0$, set

$$\hat{\boldsymbol{\theta}}_0^+ = \mathbb{E}[\boldsymbol{\theta}_0], \qquad \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},0}^+ = \mathbb{E}\left[(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)(\boldsymbol{\theta}_0 - \hat{\boldsymbol{\theta}}_0^+)^T\right]$$
$$\hat{\boldsymbol{x}}_0^+ = \mathbb{E}[\boldsymbol{x}_0], \qquad \hat{\boldsymbol{x}}_0^{a,+} = \mathbb{E}\left[\boldsymbol{x}_0^a\right] = \left[(\hat{\boldsymbol{x}}_0^+)^T,\ \bar{\boldsymbol{w}},\ \bar{\boldsymbol{v}}\right]^T$$
$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},0}^+ = \mathbb{E}\left[(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+)(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+)^T\right] \qquad \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},0}^{a,+} = \mathbb{E}\left[(\boldsymbol{x}_0^a - \hat{\boldsymbol{x}}_0^{a,+})(\boldsymbol{x}_0^a - \hat{\boldsymbol{x}}_0^{a,+})^T\right]$$
$$= \operatorname{diag}\left(\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},0}^+, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{w}}}, \boldsymbol{\Sigma}_{\tilde{\boldsymbol{v}}}\right).$$

Computation: For $k = 1, 2, \ldots$ compute:

Param.-pred. time update:
$$\hat{\boldsymbol{\theta}}_k^- = \hat{\boldsymbol{\theta}}_{k-1}^+$$

Param.-covar. time update:
$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^- = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k-1}^+ + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{r}}}$$

State-estimate time update:
$$\boldsymbol{\mathcal{X}}_{k-1}^{a,+} = \left\{\hat{\boldsymbol{x}}_{k-1}^{a,+}, \hat{\boldsymbol{x}}_{k-1}^{a,+} + \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k-1}^{a,+}}, \hat{\boldsymbol{x}}_{k-1}^{a,+} - \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k-1}^{a,+}}\right\}$$
$$\boldsymbol{\mathcal{X}}_{k,i}^{x,-} = \boldsymbol{f}(\boldsymbol{\mathcal{X}}_{k-1,i}^{x,+}, \boldsymbol{u}_{k-1}, \boldsymbol{\mathcal{X}}_{k-1,i}^{w,+}, \hat{\boldsymbol{\theta}}_k^-)$$
$$\hat{\boldsymbol{x}}_k^- = \textstyle\sum_{i=0}^p \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{X}}_{k,i}^{x,-}$$

State-covar. time update:
$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}},k}^- = \textstyle\sum_{i=0}^p \alpha_i^{(\mathrm{c})} \left(\boldsymbol{\mathcal{X}}_{k,i}^{x,-} - \hat{\boldsymbol{x}}_k^-\right)\left(\boldsymbol{\mathcal{X}}_{k,i}^{x,-} - \hat{\boldsymbol{x}}_k^-\right)^T$$

Output est., param. filter:
$$\boldsymbol{\mathcal{W}}_k = \left\{\hat{\boldsymbol{\theta}}_k^-, \hat{\boldsymbol{\theta}}_k^- + \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^-}, \hat{\boldsymbol{\theta}}_k^- - \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{\theta}},k}^-}\right\}$$
$$\boldsymbol{\mathcal{D}}_{k,i} = \boldsymbol{h}(\boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1}, \bar{\boldsymbol{w}}_{k-1}, \boldsymbol{\mathcal{W}}_{k,i}), \boldsymbol{u}_k, \bar{\boldsymbol{v}}_k, \boldsymbol{\mathcal{W}}_{k,i})$$
$$\hat{\boldsymbol{d}}_k = \textstyle\sum_{i=0}^p \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{D}}_{k,i}$$

Output est., state filter:
$$\boldsymbol{\mathcal{Y}}_{k,i} = \boldsymbol{h}(\boldsymbol{\mathcal{X}}_{k,i}^{x,-}, \boldsymbol{u}_k, \boldsymbol{\mathcal{X}}_{k-1,i}^{v,+}, \hat{\boldsymbol{\theta}}_k^-)$$
$$\hat{\boldsymbol{y}}_k = \textstyle\sum_{i=0}^p \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{Y}}_{k,i}$$

(continued on next page...)

Computation (continued from prior page): For $k = 1, 2, \ldots$ compute:

State filter gain matrix:
$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}},k} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k\right)\left(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k\right)^T$$
$$\boldsymbol{\Sigma}^{-}_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{z}},k} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\boldsymbol{\mathcal{X}}^{x,-}_{k,i} - \hat{\boldsymbol{x}}_k^{-}\right)\left(\boldsymbol{\mathcal{Y}}_{k,i} - \hat{\boldsymbol{y}}_k\right)^T$$
$$\boldsymbol{L}_k^x = \boldsymbol{\Sigma}^{-}_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{y}},k} \boldsymbol{\Sigma}^{-1}_{\tilde{\boldsymbol{y}},k}$$

Param. filter gain matrix:
$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\right)\left(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\right)^T$$
$$\boldsymbol{\Sigma}^{-}_{\tilde{\boldsymbol{\theta}}\tilde{\boldsymbol{d}},k} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\boldsymbol{\mathcal{W}}_{k,i} - \hat{\boldsymbol{\theta}}_k^{-}\right)\left(\boldsymbol{\mathcal{D}}_{k,i} - \hat{\boldsymbol{d}}_k\right)^T$$
$$\boldsymbol{L}_k^{\theta} = \boldsymbol{\Sigma}^{-}_{\tilde{\boldsymbol{\theta}}\tilde{\boldsymbol{d}},k} \boldsymbol{\Sigma}^{-1}_{\tilde{\boldsymbol{d}},k}$$

State-est. meas. update:
$$\hat{\boldsymbol{x}}_k^{+} = \hat{\boldsymbol{x}}_k^{-} + \boldsymbol{L}_k^x \left(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right)$$

State-covar. meas. update:
$$\boldsymbol{\Sigma}^{+}_{\tilde{\boldsymbol{x}},k} = \boldsymbol{\Sigma}^{-}_{\tilde{\boldsymbol{x}},k} - \boldsymbol{L}_k^x \boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}},k} (\boldsymbol{L}_k^x)^T$$

Param.-est. meas. update:
$$\hat{\boldsymbol{\theta}}_k^{+} = \hat{\boldsymbol{\theta}}_k^{-} + \boldsymbol{L}_k^{\theta} \left(\boldsymbol{y}_k - \hat{\boldsymbol{d}}_k\right)$$

Param.-covar. meas. update:
$$\boldsymbol{\Sigma}^{+}_{\tilde{\boldsymbol{\theta}},k} = \boldsymbol{\Sigma}^{-}_{\tilde{\boldsymbol{\theta}},k} - \boldsymbol{L}_k^{\theta} \boldsymbol{\Sigma}_{\tilde{\boldsymbol{d}},k} (\boldsymbol{L}_k^{\theta})^T$$