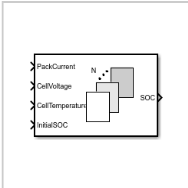
MathWorks®

SOC Estimator (Kalman Filter)

State of charge estimator with Kalman filter

Since R2022b



Libraries:
Simscape / Battery / BMS / Estimators

Description

This block implements an estimator that calculates the state of charge (SOC) of a battery by using the Kalman filter algorithms.

The SOC is the ratio of the released capacity $C_{\text{releasable}}$ to the rated capacity C_{rated}. Manufacturers provide the value of the rated capacity of each battery, which represents the maximum amount of charge in the battery:

$$SOC = \frac{C_{\text{releasable}}}{C_{\text{rated}}}.$$

This block supports single-precision and double-precision floating-point simulation.

Note

To enable single-precision floating-point simulation, the data type of all inputs and parameters, except for the **Sample time (-1 for inherited)** parameter, must be `single`.

For continuous-time simulation, set the **Filter type** parameter to `Extended Kalman-Bucy filter` or `Unscented Kalman-Bucy filter`.

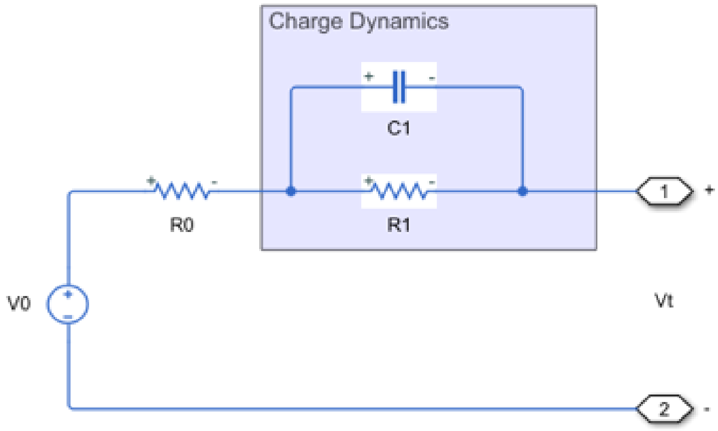
Note

Continuous-time implementation of this block works only in a double-precision floating-point simulation. If you provide single-precision floating-point parameters and inputs, this block casts them to double-precision floating-point values to prevent errors.

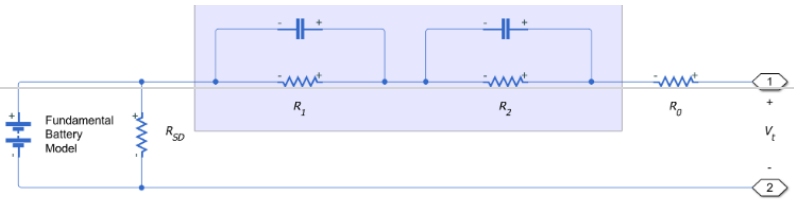
For discrete-time simulation, set the **Filter type** parameter to `Extended Kalman filter` or `Unscented Kalman filter` and the **Sample time (-1 for inherited)** parameter to a positive value or `-1`.

Equations

These figures show the equivalent circuit for a battery with one-time-constant dynamics and two time-constant dynamics, respectively:



Help Center



The equations for the equivalent circuit with two time-constant dynamics are:

$$\frac{dSOC}{dt} = -\frac{i}{3600AH}$$

$$\frac{dV_1}{dt} = \frac{i}{C_1(SOC,T)} - \frac{V_1}{R_1(SOC,T)C_1(SOC,T)}$$

$$\frac{dV_2}{dt} = \frac{i}{C_2(SOC,T)} - \frac{V_2}{R_2(SOC,T)C_2(SOC,T)}$$

$$V_t = V_0(SOC,T) - iR_0 - V_1 - V_2$$

where

- SOC is the state of charge.
- i is the current.
- V_0 is the no-load voltage.
- V_t is the terminal voltage.
- AH is the ampere-hour rating.
- R_1 is the first polarization resistance.
- C_1 is the first parallel RC capacitance.
- R_2 is the second polarization resistance.
- C_2 is the second parallel RC capacitance.
- T is the temperature.
- V_1 is the polarization voltage over the first RC network.
- V_2 is the polarization voltage over the second RC network.

A time constant τ_1 for the first parallel section relates the first polarization resistance R_1 and the first parallel RC capacitance C_1 using the relationship $C_1 = \tau_1/R_1$.

A time constant τ_2 for the second parallel section relates the second polarization resistance R_2 and the second parallel RC capacitance C_2 using the relationship $C_2 = \tau_2/R_2$.

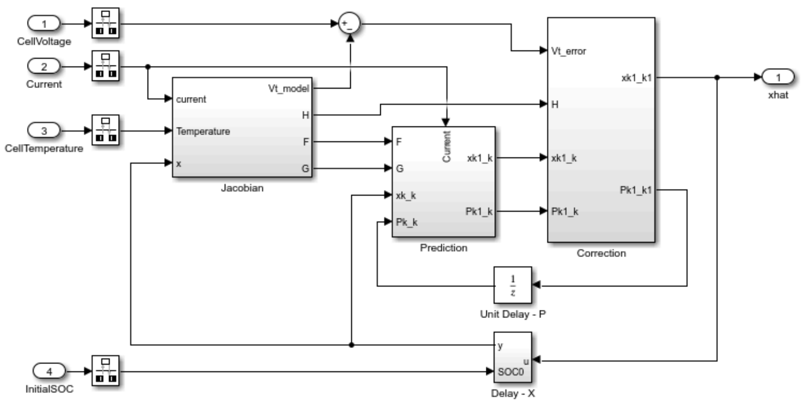
For the Kalman filter algorithms, the block uses this state and these process and observation functions:

$$x = \begin{bmatrix} SOC & V_1 \end{bmatrix}^T$$
$$f(x,i) = \begin{bmatrix} -\dfrac{i}{3600AH} \\ \dfrac{i}{C_1(SOC,T)} - \dfrac{V_1}{R_1(SOC,T)C_1(SOC,T)} \end{bmatrix}$$
$$h(x,i) = V_0(SOC,T) - iR_0 - V_1$$

If you set the **Charge dynamics** parameter to `Two time-constant dynamics`, for the Kalman filter algorithms, the block uses this state and these process and observation functions:

$$x = \begin{bmatrix} SOC & V_1 & V_2 \end{bmatrix}^T$$
$$f(x,i) = \begin{bmatrix} -\dfrac{i}{3600AH} \\ \dfrac{i}{C_1(SOC,T)} - \dfrac{V_1}{R_1(SOC,T)C_1(SOC,T)} \\ \dfrac{i}{C_2(SOC,T)} - \dfrac{V_2}{R_2(SOC,T)C_2(SOC,T)} \end{bmatrix}$$
$$h(x,i) = V_0(SOC,T) - iR_0 - V_1 - V_2$$

Extended Kalman Filter
This diagram shows the structure of the extended Kalman filter (EKF):



The EKF technique relies on a linearization at every time step to approximate the nonlinear system. To linearize the system at every time step, the algorithm computes these Jacobians online:

$$F = \frac{\partial f}{\partial x}$$
$$H = \frac{\partial h}{\partial x}$$

The EKF is a discrete-time algorithm. After the discretization, the Jacobians for the SOC estimation of the battery are:

$$\mathbf{F}_d = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{-T_S}{R_1 C_1}} \end{bmatrix}$$

$$\mathbf{H}_d = \begin{bmatrix} \frac{\partial V_{OC}}{\partial SOC} & -1 \end{bmatrix}$$

where T_S is the sample time and V_{OC} is the open-circuit voltage.

The EKF algorithm comprises these phases:

- **Initialization**
  - $\widehat{x}(0|0)$ — State estimate at time step 0 using measurements at time step 0.
  - $\widehat{P}(0|0)$ — State estimation error covariance matrix at time step 0 using measurements at time step 0.
- **Prediction**
  - Project the states ahead (*a priori*):

    $$\hat{\mathbf{x}}(k+1|k) = f(\hat{\mathbf{x}}(k|k), i).$$

  - Project the error covariance ahead:

    $$\hat{\mathbf{P}}(k+1|k) = \mathbf{F}_d(k)\hat{\mathbf{P}}(k|k)\mathbf{F}_d^T(k) + \mathbf{Q},$$

    where **Q** is the covariance of the process noise.
- **Correction**
  - Compute the Kalman gain:

    $$\mathbf{K}(k+1) = \hat{\mathbf{P}}(k+1|k)\mathbf{H}_d^T(k)(\mathbf{H}_d(k)\hat{\mathbf{P}}(k+1|k)\mathbf{H}_d^T(k) + \mathbf{R})^{-1},$$

    where R is the covariance of the measurement noise.
  - Update the estimate with the measurement $y(k)$ (*a posteriori*):

    $$\hat{\mathbf{x}}(k+1|k+1) = \hat{\mathbf{x}}(k+1|k) + \mathbf{K}(k+1)(V_t(k) - h(\hat{\mathbf{x}}(k|k), i)).$$

  - Update the error covariance:

    $$\hat{\mathbf{P}}(k+1|k+1) = (\mathbf{I} - \mathbf{K}(k+1)\mathbf{H}_d)\hat{\mathbf{P}}(k+1|k).$$

Extended Kalman-Bucy Filter

This diagram shows the structure of the extended Kalman-Bucy filter (EKBF):