While standard Subspace System Identification (SSI) finds abstract matrices from data, the code uses Structured System Identification (Grey-Box Identification). This means one enforces a known physical structure (the 3-RC circuit) and identify the physical parameters (R and C) that populate the state matrices.

1. The Physical Model (Continuous Time)

We model the battery dynamics using an Equivalent Circuit with:

- An ideal voltage source (OCV).
- A series resistance (R_0).
- Three parallel Resistor-Capacitor (R-C) branches to capture diffusion (fast, medium, slow).

Variables:

- $i(t)$: Input current (positive for discharge).
- $v(t)$: Terminal voltage.
- $i_{R_j}(t)$: Current flowing through the resistor $R_j$of the j-th branch (j=1, 2, 3).

Differential Equation for One R-C Branch

The current flowing through the resistor R_j in an R-C pair is governed by the following Ordinary Differential Equation (ODE):

$$\frac{di_{R_j}(t)}{dt} = -\frac{1}{R_j C_j} i_{R_j}(t) + \frac{1}{R_j C_j} i(t)$$

This is a standard first-order linear equation of the form $\dot{x} = ax + bu$, where:

- $a = -1/\tau_j$ (Where $\tau\_j \ = \ R\_j \ C\_j$)
- $b = 1/\tau_j$

2. Discretization (The "Filter" Logic)

Digital computers and data logs work in discrete time steps (Δt).We must convert the continuous ODE into a discrete difference equation.

Using the exact discretization method (assuming zero-order hold on input current), the solution for $x[k+1]$ based on $x[k]$ is:

$$i_{R_j}[k+1] = \underbrace{\exp\left(-\frac{\Delta t}{\tau_j}\right)}_{A_{rc}} i_{R_j}[k] + \underbrace{\left(1 - \exp\left(-\frac{\Delta t}{\tau_j}\right)\right)}_{B_{rc}} i[k]$$

This equation is exactly what the filter function in the code solves.

- Code Mapping:
  - a = exp(-deltaT / tau(j)) corresponds to the system eigenvalue A_{rc}.
  - b = 1 - a corresponds to the input gain B_{rc}.

3. The State-Space Matrix Formulation

We can now stack the equations for all 3 branches into the standard State-Space form required for system identification:

$$x_{k+1} = Ax_k + Bu_k$$

$$y_k = Cx_k + Du_k$$

State Vector (x_k)

The "state" of the system describes the internal energy stored in the capacitors. We represent this using the diffusion currents:

$$\mathbf{x}_k = \begin{bmatrix} i_{R_1}[k] \\ i_{R_2}[k] \\ i_{R_3}[k] \end{bmatrix}$$

System Matrix (A)

The A matrix contains the time constants. Since the branches are parallel and independent, A is diagonal:

$$\mathbf{A} = \begin{bmatrix} \exp\left(-\frac{\Delta t}{\tau_1}\right) & 0 & 0 \\ 0 & \exp\left(-\frac{\Delta t}{\tau_2}\right) & 0 \\ 0 & 0 & \exp\left(-\frac{\Delta t}{\tau_3}\right) \end{bmatrix}$$

Input Matrix (B)

The B matrix distributes the input current i[k] to the states:

$$\mathbf{B} = \begin{bmatrix} 1 - \exp\left(-\frac{\Delta t}{\tau_1}\right) \\ 1 - \exp\left(-\frac{\Delta t}{\tau_2}\right) \\ 1 - \exp\left(-\frac{\Delta t}{\tau_3}\right) \end{bmatrix}$$

Output Equation (y_k)

The measurable output is the "Polarization Voltage" (the voltage drop caused by the impedances).

$$v_{polarization}[k] = v[k] - OCV(z[k])$$

This drop is the sum of the ohmic drop and the voltage across all RC pairs ($v_{RC} = R \cdot i_R$).

$$v_{pol}[k] = -R_0 i[k] - R_1 i_{R_1}[k] - R_2 i_{R_2}[k] - R_3 i_{R_3}[k]$$

In matrix form $y_k = Cx_k + Du_k$:

(Note: The negative signs usually appear because discharge current is typically negative in datasets or defined as voltage "drops" relative to OCV.)

4. Summary of Relevant Formulae

Running the matlab script, present in the Github repo mathematically solves this optimization problem

1. The Objective Function (Cost):

Find the parameter set $\theta = \{R_0, R_1, R_2, R_3, \tau_1, \tau_2, \tau_3\}$ that minimizes the Root Mean Square Error (RMSE):

$$J(\theta) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(v_{measured}[k] - v_{model}[k])^2}$$

2. The Model Calculation (Inside the loop):

For every time step k:

$$v_{model}[k] = OCV(z[k]) - R_0 i[k] - \sum_{j=1}^{3} R_j \cdot i_{R_j}[k]$$

Where $i_{R_j}[k]$is updated recursively:

$$i_{R_j}[k] = e^{\frac{-\Delta t}{\tau_j}} i_{R_j}[k-1] + \left(1 - e^{\frac{-\Delta t}{\tau_j}}\right) i[k-1]$$

This theoretical framework perfectly matches the simCell3RC and model_error_3RC functions in the code.