Mondragon Unibertsitatea

ModEst

Battery degradation mode estimation tool

Authors: Iker Lopetegi, Laura Oca, Sergio Fernandez

Date: 23/04/2024

HUMANITY AT WORK MONDRAGON

ÍNDICE

1 INTRODUCCIÓN

The software itself is a tool capable of estimating Li-ion battery degradation modes. Battery degradation modes group different battery degradation mechanism into three main degradation modes: Loss of Active Material (LAM) of both electrodes and Loss of Lithium Inventory (LLI). The core of the tool is a multi-objective optimization algorithm programmed inside MATLAB (any operative system which enables MATLAB's utilization is valid), thus, using MATLAB's programming language. Being more specific, MATLAB's *gamultiobj* (multi-objective genetic algorithm) is used.

The tool can be used in a wide range of applications including lithium-ion batteries. First and foremost, as a degradation diagnosing tool during their operating lifetime. In addition, the tool can be used for electrode balancing purposes, which is applicable in several modelling and characterisation applications.

First disclosure of this software was in the 11th Mar 2024 for the ECCE EUROPE 2024 first submission in the article named "*Intuitive Degradation Mode Estimation Tool: ModEst*".

1.1 GLOSARIO DE TÉRMINOS (opcional)

Term	Acronym	Definition
Loss of Active Material	LAM	The degradation or detachment of the material within the cell's electrodes, reducing its capacity and performance over time.
Loss of Lithium Inventory	LLI	Loss of lithium inventory in a battery occurs when lithium ions become trapped or inactive, diminishing the battery's capacity and overall performance.
Open Circuit Voltage / Potential	OCV / OCP	It's the voltage across the terminals of a battery/electrode when no current is flowing
Negative electrode	NE	The negative electrode of a battery is typically made of a material that can accept and store ions during discharge, such as graphite in lithium-ion batteries.
Positive electrode	PE	The positive electrode of a battery is typically made of a material that can release and absorb ions during charge and discharge, such as lithium cobalt oxide in lithium-ion batteries.
State of Charge	SOC	It indicates the remaining capacity of a battery as a percentage of its full charge.

State of Lithiation	SOL	State of lithiation refers to the extent to which lithium ions have been inserted into the electrode material of a battery during charging.
Beginning of Life	BOL	The beginning of life of a battery refers to its initial performance and capacity when it is first put into use.

1.2 CONTENIDO DE LA CARPETA

- CÓDIGO FUENTE: Que contenga el código fuente del programa.
    - "*ModEst_ini.m*": This is the initialization script, which loads the battery cell data from "*dataNMC*\" folder and calls "*Modest_v3.m*" function to run the tool.
    - "*Modest_v3.m*": The function where the degradation mode estimation tool is performed, called by "*ModEst_ini.m*" main script.
    - "*dataNMC*\": This is the folder that contains the battery cell data, including the OCP data of both electrodes and OCV data of the cell at the beginning of life and at an aged life instant.
- CÓDIGO EJECUTABLE: Que contenga el código ejecutable.

For the software tool in question, there is no traditional executable (.exe) because it runs through a MATLAB script. There is a main script that loads the necessary data from a folder and calls a function to run the tool.

2 DESCRIPCIÓN DEL SOFTWARE

The developed tool performs a multi-objective optimisation using a genetic algorithm (GA). Genetic algorithm is a heuristic approach that seeks the minimisation of the objective function by evolving a population of randomly generated individuals through an iterative process. Hence, the algorithm can be iterated continuously until a sufficiently accurate value is obtained or error threshold is reached. First, the tool obtains the stoichiometry values at 100 \% and 0 \% SOC for the beginning-of-life (BOL) OCV. Later, the constraints for the optimisation of the aged cell are calculated based on the BOL optimisation results. Then, the optimisation for the aged cell is computed, so, two optimisation processes are concatenated. Finally, from the results obtained at both optimisation processes the degradation modes are calculated.

The inputs for the tool are the following:

- BOL OCV containing voltage, capacity and SOC data.
- BOL OCP containing potential, capacity and SOL data.
- Scaling information: Electrode size data or reconstructed cell capacity.
- Aged OCV containing voltage, capacity and SOC data.

The outputs of the tool are the following:

- 100 \% and 0 \% SOC stoichiometries of both electrodes at BOL and for the aged cell.
- Loss-of-active-material (LAM) of each electrode.
- Loss-of-lithium-inventory (LLI).

2.1 Entorno operativo

MATLAB's operating environment is a comprehensive platform for numerical computation, visualization, and programming. It comprises:

1. **Graphical User Interface (GUI)**: MATLAB offers an intuitive GUI with windows for editing scripts, exploring variables, managing files, and interacting via a command window.
2. **Programming Language**: MATLAB employs a high-level, interpreted language optimized for mathematical operations and array manipulations. Its syntax is user-friendly and akin to traditional mathematical notation.
3. **Built-in Functions and Toolboxes**: MATLAB provides an extensive library of built-in functions and specialized toolboxes for tasks ranging from basic arithmetic operations to advanced data analysis, image processing, and control system design.
4. **Data Visualization**: MATLAB enables users to create visually appealing plots and graphs for data analysis and presentation. It supports various plot types, including 2D and 3D plots, histograms, and surface plots.
5. **Integration Capabilities**: MATLAB seamlessly integrates with other programming languages like C/C++, Java, and Python, allowing users to incorporate external libraries and extend its functionality as needed.
6. **Cross-Platform Compatibility**: MATLAB is available for multiple operating systems, including Windows, macOS, and Linux, ensuring users can work across different environments without compatibility issues.

In essence, MATLAB's operating environment provides a powerful and versatile platform for scientific and engineering computing, offering a rich set of tools and resources for tackling diverse computational tasks efficiently.

2.2 Composición. Diagramas de flujo.

A. Half-cell scaling

The cell OCV is obtained by subtracting the negative electrode OCP to the positive electrode OCP within their stoichiometry values between 100 % and 0% SOC. The OCV curves can be obtained by applying techniques as quasi-static voltage measurement or voltage relaxation in the full commercial or prototyped cell. However, the obtention of the OCP of each electrode is not as immediate as obtaining the OCV. To obtain the OCPs of the electrodes, reconstructed cells are usually built using a small portion of the electrode. The tool admits two ways of scaling techniques chosen by the user: by capacity or by electrochemically active area.

The first method uses the reconstructed cell level electrode area and the full-size cell electrode area to obtain the scaling factor. The second method requires an extra full reconstructed cell (in addition to half-cells) to perform the scaling.

B. Optimisation Algorithm

The next step of the optimisation process is the core of the developed tool. To estimate the degradation modes, 100 % and 0 % SOC stoichiometry values at BOL and for the aged cell are needed. The optimisation procedure for both conditions is the same. The parameters to optimise are:

- Upper stoichiometry capacity value [Q^{n}(θ^{n}_{100}), [Q^{p}θ^{p}_{100}]
- Electrode capacity variation coefficients $[\alpha^n, \alpha^p]$

The optimisation process is limited by physical constraints. Upper stoichiometry capacity value cannot be lower than 0 nor a capacity value that makes the remaining capacity in the electrode between the 100 \% and 0 \% SOC stoichiometries to be lower than the cell capacity. Electrode capacity coefficient cannot make the electrode capacity to be lower than the actual cell capacity and can go as high as the total electrode capacity.

Therefore, the optimisation global constraints at BOL are calculated as:

$$0 \leq Q^r_{BOL}(\theta^r_{100}) \leq Q^r - Q$$
$$\frac{Q}{Q^r} \leq \alpha^r_{BOL} \leq 1$$

As stated above, aged optimisation maximum constraints may be different to BOL constraints, as they depend on the result of BOL optimisation. Upper stoichiometry capacity value and maximum capacity coefficient constraints for the aged cell optimisation are redefined as follows:

$$0 \leq Q^r_a(\theta^r_{100}) \leq Q^r_{BOL} - Q$$
$$\frac{Q}{Q^r} \leq \alpha^r_a \leq \frac{Q}{Q^r_{BOL}}$$

To keep the capacity of the electrodes above the capacity of the cell, a nonlinear inequality constraint is added in both optimisations as

$$\alpha^r Q^r - Q^r(\theta^r_{100}) \geq Q$$

The lower stoichiometry capacity value is obtained as follows either for the BOL or for the aged cell:

$$Q^r(\theta^r_0) = Q^r(\theta^r_{100}) + Q$$

The OCP of each electrode is calculated as:

$$OCP^r(\theta^r) = OCP^r\left(Q^r(\theta^r_{100}), Q^r(\theta^r_0)\right)$$

From the calculated OCP curves, OCV is calculated, and the differential voltage is calculated as:

$$\frac{dV}{dQ} = \frac{dOCV}{dQ}$$

The cost functions to minimise are the OCV and the dV/dQ root mean square errors, which emphasises big errors and is consistent with average values.

$$J_1 = \sqrt{\overline{(OCV - OCV^*)}^2}$$

$$J_2 = \sqrt{\overline{\left(\frac{dOCV}{dQ} - \frac{dOCV^*}{dQ}\right)}^2}$$

C. Degradation modes estimation

The LAM of an electrode is given by:

$$LAM^r = \left(1 - \frac{Q_a^r}{Q_{BOL}^r}\right) \times 100$$

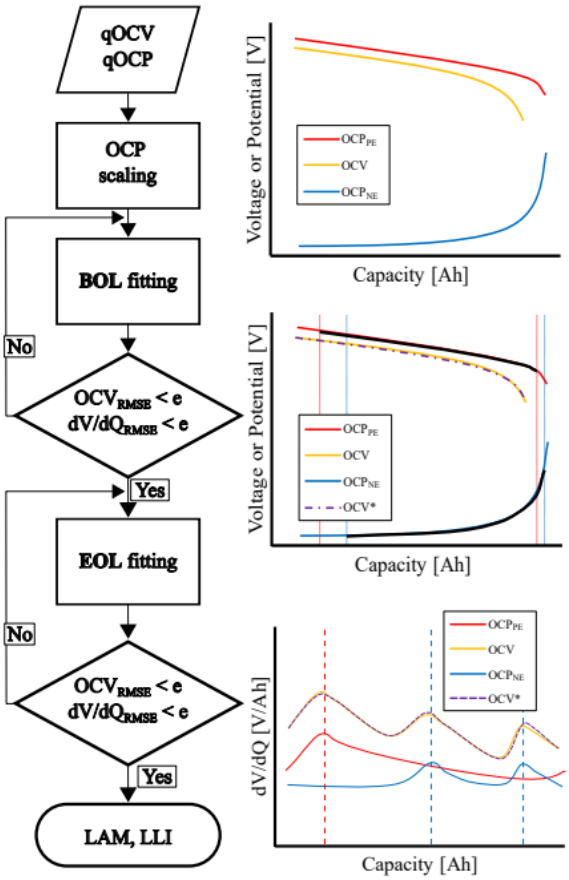
To calculate the LLI, first, the total amount of intercalated lithium in the cell at any SOC was as

$$n_{Li} = \frac{3600}{F}(z^p Q^p + z^n Q^n)$$

Then, the LLI can be calculated as

$$LLI = \left(1 - \frac{n_{Li}^a}{n_{Li}^{BOL}}\right), \times 100$$

Flow charts:



3 INSTALACIÓN Y USO

3.1 Instrucciones o proceso de instalación.

Matlab instalation: https://www.mathworks.com/help/install/install-products.html

3.2 Interface de usuario

In the folder where the software is delivered there is an initialization script that calls to the software. This initialization script is the user interface with the software. This initialization script is tuneable to insert the data you want to be analysed by the tool. The script is already ready to be run and work with an example.

4 DESCRIPCIÓN DETALLADA y SEGMENTOS SIGNIFICATIVOS DE CÓDIGO

```
function [STCH_LIMITS_BOL,STCH_LIMITS_Aged,Error,BOLopt,AGEDopt] =
ModEst_v3(NEdata,PEdata,CELLdata,SCALINGdata,scaling_type,Fresh_OCV,size,analyze_AGED,AGEDdata,optio
ns)

%%%%%%%%%%%%%%%%%
% ELECTRODE data %
%%%%%%%%%%%%%%%%%

% negative electrode %
qQ = NEdata.C_curve(~isnan(NEdata.C_curve));
coeficients = polyfit(1:length(qQ),qQ,1);
if coeficients(1) < 0
    qQ = flip(qQ);
else
end
qSOL = NEdata.SOL_curve(~isnan(NEdata.SOL_curve));
coeficients = polyfit(1:length(qSOL),qSOL,1);
if coeficients(1) > 0
    qSOL = flip(qSOL);
end
if mean(qSOL) > 10
    qSOL = qSOL./100;
else
end
qOCP = NEdata.V_curve(~isnan(NEdata.V_curve));
coeficients = polyfit(1:length(qOCP),qOCP,1);
if coeficients(1) < 0
    qOCP = flip(qOCP);
else
end
[qQ, qSOL, qOCP] = MonotonicArray(qQ,qSOL,qOCP,size);
qQneg = qQ; qSOLneg = qSOL; qOCPneg = qOCP;

figure
    subplot(1,2,1)
        plot(qQ,qOCP)
        hold on
        xlabel('Capacity (mAh)')
        ylabel('Potential (V)')
    subplot(1,2,2)
        plot(qSOL,qOCP)
        hold on
        xlabel('SOL ()')
        ylabel('Potential (V)')

% positive electrode %
qQ = PEdata.C_curve(~isnan(PEdata.C_curve));
coeficients = polyfit(1:length(qQ),qQ,1);
if coeficients(1) < 0
    qQ = flip(qQ);
```

```
else
end
qSOL = PEdata.SOL_curve(~isnan(PEdata.SOL_curve));
coeficients = polyfit(1:length(qSOL),qSOL,1);
if coeficients(1) < 0
    qSOL = flip(qSOL);
end
if mean(qSOL) > 10
    qSOL = qSOL./100;
else
end
qOCP = PEdata.V_curve(~isnan(PEdata.V_curve));
coeficients = polyfit(1:length(qOCP),qOCP,1);
if coeficients(1) > 0
    qOCP = flip(qOCP);
else
end
[qQ, qSOL, qOCP] = MonotonicArray(qQ,qSOL,qOCP,size);
qQpos = qQ; qSOLpos = qSOL; qOCPpos = qOCP;

    subplot(1,2,1)
        plot(qQ,qOCP)
        legend('neg','pos')
    subplot(1,2,2)
        plot(qSOL,qOCP)
        legend('neg','pos')

%%%%%%%%%%%%
% CELL DATA %
%%%%%%%%%%%%
qQ = CELLdata.C_curve(~isnan(CELLdata.C_curve));
coeficients = polyfit(1:length(qQ),qQ,1);
if coeficients(1) < 0
    qQ = flip(qQ);
else
end
qSOL = CELLdata.SOL_curve(~isnan(CELLdata.SOL_curve));
coeficients = polyfit(1:length(qSOL),qSOL,1);
if coeficients(1) > 0
    qSOL = flip(qSOL);
else
end
if mean(qSOL) > 5
    qSOL = qSOL./100;
else
end
qOCP = CELLdata.V_curve(~isnan(CELLdata.V_curve));
coeficients = polyfit(1:length(qOCP),qOCP,1);
if coeficients(1) > 0
    qOCP = flip(qOCP);
else
end
[qQ, qSOL, qOCP] = MonotonicArray(qQ,qSOL,qOCP,size);
qQcell = qQ; qSOLcell = qSOL; qOCPcell = qOCP;

figure
    subplot(1,2,1)
        plot(qQ,qOCP)
        hold on
        xlabel('Capacity (Ah)')
        ylabel('Voltage (V)')
    subplot(1,2,2)
        plot(qSOL,qOCP)
        hold on
        xlabel('SOC ()')
        ylabel('Voltage (V)')

%%%%%%%%%%
% SCALING %
%%%%%%%%%%
switch scaling_type
    case 'OCV'
        qQ = SCALINGdata.C_curve(~isnan(SCALINGdata.C_curve));
        coeficients = polyfit(1:length(qQ),qQ,1);
```

```
        if coeficients(1) < 0
            qQ = flip(qQ);
        else
        end
        qSOL = SCALINGdata.SOL_curve(~isnan(SCALINGdata.SOL_curve));
        coeficients = polyfit(1:length(qSOL),qSOL,1);
        if coeficients(1) > 0
            qSOL = flip(qSOL);
        end
        if mean(qSOL) > 10
            qSOL = qSOL./100;
        else
        end
        qOCP = SCALINGdata.V_curve(~isnan(SCALINGdata.V_curve));
        coeficients = polyfit(1:length(qOCP),qOCP,1);
        if coeficients(1) > 0
            qOCP = flip(qOCP);
        else
        end
        [qQ, qSOL, qOCP] = MonotonicArray(qQ,qSOL,qOCP,size);
        qQscaling = qQ; qSOLscaling = qSOL; qOCPscaling = qOCP;
        scaling_factor = max(qQcell)/max(qQscaling);
        qQnegFull = qQneg.*scaling_factor;
        qQposFull = qQpos.*scaling_factor;
    case 'NOT'
    case 'size'
        neg_size_factor = SCALINGdata.neg_size/SCALINGdata.cell_size;
        pos_size_factor = SCALINGdata.pos_size/SCALINGdata.cell_size;
        qQnegFull = qQneg.*neg_size_factor;
        qQposFull = qQpos.*pos_size_factor;
end

switch scaling_type
    case 'NOT'
        qQnegFull = qQneg;
        qQposFull = qQpos;
end

figure
    yyaxis left
        plot(qQposFull,qOCPpos,'LineWidth',1.2,'Color','r','LineStyle','-')
        hold on
        plot(qQcell,qOCPcell,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        xlabel('Capacity (Ah)')
        ylabel('OCP^{p} or OCV (V)')
    yyaxis right
        plot(qQnegFull,qOCPneg,'LineWidth',1.2,'Color','b','LineStyle','-')
        xlabel('Capacity (Ah)')
        ylabel('OCP^{n} (V)')
        legend('PE','OCV','NE')

switch scaling_type
    case {'OCV','size'}
                axes('Position',[.2 .3 .2 .2])
                box on
                    yyaxis left
                        plot(qQpos,qOCPpos,'LineWidth',1.2,'Color','r','LineStyle','-')
                        xlabel('Capacity (mAh)')
                        ylabel('OCP^{p} (V)')
                    yyaxis right
                        plot(qQneg,qOCPneg,'LineWidth',1.2,'Color','b','LineStyle','-')
                        xlabel('Capacity (mAh)')
                        ylabel('OCP^{n} (V)')
    case 'NOT'
end

figure
    yyaxis left
        plot(qSOLpos,qOCPpos,'LineWidth',1.2,'Color','r','LineStyle','-')
        hold on
        plot(qSOLcell,qOCPcell,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        xlabel('SOL or SOC ()')
        ylabel('OCP^{p} or OCV (V)')
    yyaxis right
```

```
        plot(qSOLneg,qOCPneg,'LineWidth',1.2,'Color','b','LineStyle','-')
        xlabel('SOL or SOC  (Ah)')
        ylabel('OCP^{n} (V)')
        legend('PE','OCV','NE')
%%
%%%%%%%%%%%%%%%%%%%
% BOL OPTIMIZATION %
%%%%%%%%%%%%%%%%%%%

% Maximum bounds
Qneg0_max = max(qQnegFull)-max(qQcell);
Qpos0_max = max(qQposFull)-max(qQcell);
ub = [Qneg0_max Qpos0_max 1 1];
lb = [min(qQnegFull) min(qQposFull) max(qQcell)/max(qQnegFull) max(qQcell)/max(qQposFull)];

% Linear inequality constraints
N = max(qQnegFull);
P = max(qQposFull);
C = max(qQcell);
A = [1 0 -N 0;
     0 1 0 -P];
b = [-C;
     -C];

% Computational cost:
tStart = tic;
fprintf('genetic algorithm Optimization is running...    ');

% Cost function handle
CostFunction = @(x) [OCVopt(qQcell,qOCPcell,qQnegFull,qOCPneg,qQposFull,qOCPpos,x),
dVdQopt(qQcell,qOCPcell,qQnegFull,qOCPneg,qQposFull,qOCPpos,x),
dQdVopt(qQcell,qOCPcell,qQnegFull,qOCPneg,qQposFull,qOCPpos,x)]

switch Fresh_OCV
    case 'NOT'
        nvars = 4;
        [x,fval,exitflag,output] = gamultiobj(CostFunction,nvars,A,b,[],[],lb,ub,[],[],options)
    case 'FRESH'
        nvars = 2;
        [x,fval,exitflag,output] =
gamultiobj(CostFunction,nvars,[],[],[],[],lb(1:2),ub(1:2),[],[],options)
end

% Optimization output
fprintf('BOL OCV fitting computational cost:');
tEnd = toc(tStart)
[~,pos] = min(fval(:,1).*fval(:,2));
x = x(pos,:);

%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZATION RESULTS %
%%%%%%%%%%%%%%%%%%%%%%
switch Fresh_OCV
    case 'NOT'
        coef = x(3);
        Qneg_opt = qQnegFull.*coef;
        coef = x(4);
        Qpos_opt = qQposFull.*coef;
    case 'FRESH'
        Qneg_opt = qQnegFull;
        Qpos_opt = qQposFull;
end


% negative electrode %
    Qneg100 = x(1);                                         % Lower limit of effective electrode
capacity
    Qneg0 = Qneg100+max(qQcell);                            % Upper limit of effective electrode
capacity
    a = find(Qneg_opt>=Qneg100);
    b = find(Qneg_opt<=Qneg0);
    if isempty(a) || a(1)<=1
        Qneg_NEW = Qneg_opt(1:b(end));                       % Efective electrode capacity array
        Pneg_NEW = qOCPneg(1:b(end));                        % Effective electrode potential array
```

```
        SOLneg_NEW = qSOLneg(1:b(end));                         % Effective electrode SOL array
    else
        Qneg_NEW = Qneg_opt(a(1):b(end));                       % Efective electrode capacity array
        Pneg_NEW = qOCPneg(a(1):b(end));                        % Effective electrode potential
array
        SOLneg_NEW = qSOLneg(a(1):b(end));                      % Effective electrode SOL array
    end
    OCPneg = interp1(Qneg_NEW-Qneg_NEW(1),Pneg_NEW,qQcell,"linear","extrap"); % Electrode OCP at BOL
(referenced to 0)

    STCH_LIMITS_BOL.max_Qneg = max(Qneg_opt);
    STCH_LIMITS_BOL.Qneg100 = Qneg_NEW(1);
    STCH_LIMITS_BOL.Qneg0 = Qneg_NEW(end);
    STCH_LIMITS_BOL.SOLneg100 = SOLneg_NEW(1);
    STCH_LIMITS_BOL.SOLneg0 = SOLneg_NEW(end);

% positive electrode %
    Qpos100 = x(2);                                             % Lower limit of effective electrode
capacity
    Qpos0 = Qpos100+max(qQcell);                                % Upper limit of effective electrode
capacity
    a = find(Qpos_opt>=Qpos100);
    b = find(Qpos_opt<=Qpos0);
    if isempty(a) || a(1)<=1
        Qpos_NEW = Qpos_opt(1:b(end));                          % Efective electrode capacity array
        Ppos_NEW = qOCPpos(1:b(end));                           % Effective electrode potential array
        SOLpos_NEW = qSOLpos(1:b(end));                         % Effective electrode SOL array
    else
        Qpos_NEW = Qpos_opt(a(1):b(end));                       % Efective electrode capacity array
        Ppos_NEW = qOCPpos(a(1):b(end));                        % Effective electrode potential
array
        SOLpos_NEW = qSOLpos(a(1):b(end));                      % Effective electrode SOL array
    end
    OCPpos = interp1(Qpos_NEW-Qpos_NEW(1),Ppos_NEW,qQcell,"linear","extrap"); % Electrode OCP at BOL
(referenced to 0)

    STCH_LIMITS_BOL.max_Qpos = max(Qpos_opt);
    STCH_LIMITS_BOL.Qpos100 = Qpos_NEW(1);
    STCH_LIMITS_BOL.Qpos0 = Qpos_NEW(end);
    STCH_LIMITS_BOL.SOLpos100 = SOLpos_NEW(1);
    STCH_LIMITS_BOL.SOLpos0 = SOLpos_NEW(end);

% ocv %
    OCV_calculated = OCPpos - OCPneg;
    e = qOCPcell-OCV_calculated;
    a = ~isnan(e);
    e = e(a);
    Error.OCV_RMSe = sqrt(mean((e).^2));
    Error.OCV_Max = max(e);

% dV/dQ & dQ/dV %

   % cell %
        [dQdVcell,dVdQcell,Qcell,Vcell] = IC_DV(qQcell,qOCPcell);
        dQdVcell = flip(dQdVcell);
        dVdQcell = flip(dVdQcell);

    % negative electrode %
        [dQdPneg,dPdQneg,Qneg,Pneg] = IC_DV(Qneg_opt,qOCPneg);
            dPdQneg_opt = interp1(Qneg,dPdQneg,Qneg_opt,"linear","extrap");
            dPdQneg_NEW = interp1(Qneg,dPdQneg,Qneg_NEW,"linear","extrap");
            dPdQneg_NEW = interp1(Qneg_NEW-Qneg_NEW(1),dPdQneg_NEW,Qcell,"linear","extrap");

            dQdPneg_opt = interp1(Qneg,dQdPneg,Qneg_opt,"linear","extrap");
            dQdPneg_NEW = interp1(Qneg,dQdPneg,Qneg_NEW,"linear","extrap");
            dQdPneg_NEW = interp1(Qneg_NEW-Qneg_NEW(1),dQdPneg_NEW,Qcell,"linear","extrap");

    % positive electrode %
        [dQdPpos,dPdQpos,Qpos,Ppos] = IC_DV(Qpos_opt,qOCPpos);
        dQdPpos = flip(dQdPpos);
        dPdQpos = flip(dPdQpos);
            dPdQpos_opt = interp1(Qpos,dPdQpos,Qpos_opt,"linear","extrap");
            dPdQpos_NEW = interp1(Qpos,dPdQpos,Qpos_NEW,"linear","extrap");
            dPdQpos_NEW = interp1(Qpos_NEW-Qpos_NEW(1),dPdQpos_NEW,Qcell,"linear","extrap");
```

```
            dQdPpos_opt = interp1(Qpos,dQdPpos,Qpos_opt,"linear","extrap");
            dQdPpos_NEW = interp1(Qpos,dQdPpos,Qpos_NEW,"linear","extrap");
            dQdPpos_NEW = interp1(Qpos_NEW-Qpos_NEW(1),dQdPpos_NEW,Qcell,"linear","extrap");

    % calculated %
        [dQdVcalc,dVdQcalc,Qcalc,Vcalc] = IC_DV(qQcell,OCV_calculated);
        dQdVcalc = flip(dQdVcalc);
        dVdQcalc = flip(dVdQcalc);

        dVdQcalc = interp1(Qcalc,dVdQcalc,Qcell,"linear","extrap");
        e = dVdQcell-dVdQcalc;
        a = ~isnan(e);
        e = e(a);
        Error.dVdQ_RMSe = rms(e);
        Error.dVdQ_Max = max(e);

        dQdVcalc = interp1(Qcalc,dQdVcalc,Qcell,"linear","extrap");
        e = dQdVcell-dQdVcalc;
        a = ~isnan(e);
        e = e(a);
        Error.dQdV_RMSe = rms(e);
        Error.dQdV_Max = max(e);

% Optimization output %

    % negative electrode % .neg
    BOLopt.neg.Q = Qneg_opt;
    BOLopt.neg.Q_stch = Qneg_NEW;
    BOLopt.neg.SOL = qSOLneg;
    BOLopt.neg.SOL_stch = SOLneg_NEW;
    BOLopt.neg.OCP = qOCPneg;
    BOLopt.neg.OCP_stch = OCPneg;
    BOLopt.neg.Q100 = Qneg_NEW(1);
    BOLopt.neg.Q0 = Qneg_NEW(end);
    BOLopt.neg.SOL100 = SOLneg_NEW(1);
    BOLopt.neg.SOL0 = SOLneg_NEW(end);
    BOLopt.neg.dPdQ = dPdQneg_opt;
    BOLopt.neg.dPdQ_stch = dPdQneg_NEW;
    BOLopt.neg.dQdP = dQdPneg_opt;
    BOLopt.neg.dQdP_stch = dQdPneg_NEW;

    % positive electrode % .pos
    BOLopt.pos.Q = Qpos_opt;
    BOLopt.pos.Q_stch = Qpos_NEW;
    BOLopt.pos.SOL = qSOLpos;
    BOLopt.pos.SOL_stch = SOLpos_NEW;
    BOLopt.pos.OCP = qOCPpos;
    BOLopt.pos.OCP_stch = OCPpos;
    BOLopt.pos.Q100 = Qpos_NEW(1);
    BOLopt.pos.Q0 = Qpos_NEW(end);
    BOLopt.pos.SOL100 = SOLpos_NEW(1);
    BOLopt.pos.SOL0 = SOLpos_NEW(end);
    BOLopt.pos.dPdQ = dPdQpos_opt;
    BOLopt.pos.dPdQ_stch = dPdQpos_NEW;
    BOLopt.pos.dQdP = dQdPpos_opt;
    BOLopt.pos.dQdP_stch = dQdPpos_NEW;

    % commercial cell % .cell
    BOLopt.cell.Q = qQcell;
    BOLopt.cell.Q_for_dVdQ = Qcell;
    BOLopt.cell.SOL = qSOLcell;
    BOLopt.cell.OCP = qOCPcell;
    BOLopt.cell.dPdQ = dVdQcell;
    BOLopt.cell.dQdP = dQdVcell;

    % calculated % .calc
    BOLopt.calc.Q = qQcell;
    BOLopt.calc.Q_for_dVdQ = Qcell;
    BOLopt.calc.SOL = qSOLcell;
    BOLopt.calc.OCP = OCV_calculated;
    BOLopt.calc.dPdQ = dVdQcalc;
    BOLopt.calc.dQdP = dQdVcalc;
```

```
    % error %
    BOLopt.error.OCV_RMSE = Error.OCV_RMSe;
    BOLopt.error.OCV_Max = Error.OCV_Max;
    BOLopt.error.dVdQ_RMSE = Error.dVdQ_RMSe;
    BOLopt.error.dVdQ_Max = Error.dVdQ_Max;
    BOLopt.error.dQdV_RMSE = Error.dQdV_RMSe;
    BOLopt.error.dQdV_Max = Error.dQdV_Max;

% Visualizaton %
figure
subplot(3,2,1)
    yyaxis left
        plot(Qpos_opt,qOCPpos,'LineWidth',1.2)
        xlabel('Capacity (Ah)')
        ylabel('OCP (V)')
    yyaxis right
        plot(Qpos_opt,dPdQpos_opt,'LineWidth',1.2,'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('-dV/dQ (V/Ah)')
        xline([STCH_LIMITS_BOL.Qpos100 STCH_LIMITS_BOL.Qpos0],'-.k','LineWidth',1)
subplot(3,2,3)
    yyaxis left
        plot(qQcell,qOCPcell,'LineWidth',1.2)
        hold on
        plot(qQcell,OCV_calculated,'LineWidth',1.2,'Color',"k",'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('OCV (V)')
    yyaxis right
        plot(Qcell,dVdQcell,'LineWidth',1.2)
        hold on
        plot(Qcell,dVdQcalc,'LineWidth',1.2,'Color',"k",'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('-dV/dQ (V/Ah)')
        legend('OCV','OCV*','','')
subplot(3,2,5)
    yyaxis left
        plot(Qneg_opt,qOCPneg,'LineWidth',1.2)
        xlabel('Capacity (Ah)')
        ylabel('OCP (V)')
    yyaxis right
        plot(Qneg_opt,dPdQneg_opt,'LineWidth',1.2,'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('-dV/dQ (V/Ah)')
        xline([STCH_LIMITS_BOL.Qneg100 STCH_LIMITS_BOL.Qneg0],'-.k','LineWidth',1)

subplot(3,2,2)
    yyaxis left
        plot(Qpos_opt,qOCPpos,'LineWidth',1.2)
        xlabel('Capacity (Ah)')
        ylabel('OCP (V)')
    yyaxis right
        plot(Qpos_opt,dQdPpos_opt,'LineWidth',1.2,'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('-dQ/dV (Ah/V)')
        xline([STCH_LIMITS_BOL.Qpos100 STCH_LIMITS_BOL.Qpos0],'-.k','LineWidth',1)
subplot(3,2,4)
    yyaxis left
        plot(qQcell,qOCPcell,'LineWidth',1.2)
        hold on
        plot(qQcell,OCV_calculated,'LineWidth',1.2,'Color',"k",'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('OCV (V)')
    yyaxis right
        plot(Qcell,dQdVcell,'LineWidth',1.2)
        hold on
        plot(Qcell,dQdVcalc,'LineWidth',1.2,'Color',"k",'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('-dQ/dV (Ah/V)')
        legend('OCV','OCV*','','')
subplot(3,2,6)
    yyaxis left
        plot(Qneg_opt,qOCPneg,'LineWidth',1.2)
        xlabel('Capacity (Ah)')
        ylabel('OCP (V)')
```

```
    yyaxis right
        plot(Qneg_opt,dQdPneg_opt,'LineWidth',1.2,'LineStyle','--')
        xlabel('Capacity (Ah)')
        ylabel('-dQ/dV (Ah/V)')
        xline([STCH_LIMITS_BOL.Qneg100 STCH_LIMITS_BOL.Qneg0],'-.k','LineWidth',1)

figure
subplot(3,1,1)
    plot(qSOLpos,qOCPpos,'LineWidth',1.2)
    xlabel('SOL ()')
    ylabel('OCP (V)')
    xline([STCH_LIMITS_BOL.SOLpos0 STCH_LIMITS_BOL.SOLpos100],'-.k','LineWidth',1)
    xlim([0 1])

subplot(3,1,2)
    plot(qSOLcell,qOCPcell,'LineWidth',1.2)
    xlabel('SOC ()')
    ylabel('OCV (V)')
    xlim([0 1])

subplot(3,1,3)
    plot(qSOLneg,qOCPneg,'LineWidth',1.2)
    xlabel('SOL ()')
    ylabel('OCP (V)')
    xline([STCH_LIMITS_BOL.SOLneg0 STCH_LIMITS_BOL.SOLneg100],'-.k','LineWidth',1)
    xlim([0 1])


figure(10)
subplot(3,2,1)
    yyaxis left
        plot(Qpos_opt,qOCPpos,'LineWidth',1.2,'Color','r')
        hold on
        plot(qQcell,qOCPcell,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(qQcell,OCV_calculated,'LineWidth',1.2,'Color',"#7E2F8E",'LineStyle','--')
        plot(Qpos_NEW,Ppos_NEW,'k','LineWidth',1.2,'LineStyle','-')
        xline([STCH_LIMITS_BOL.Qpos100 STCH_LIMITS_BOL.Qpos0],'-.r','LineWidth',1)
        xlabel('Capacity (Ah)')
        ylabel('OCP^{p} or OCV (V)')

    yyaxis right
        plot(Qneg_opt,qOCPneg,'LineWidth',1.2,'Color','b')
        hold on
        plot(Qneg_NEW,Pneg_NEW,'k','LineWidth',1.2,'LineStyle','-')
        xline([STCH_LIMITS_BOL.Qneg100 STCH_LIMITS_BOL.Qneg0],'-.b','LineWidth',1)
        ylabel('OCP^{n} (V)')
        legend('PE','OCV','OCV*','','','','NE','','','','Location','best')

subplot(3,2,3)
    yyaxis right
        plot(Qcell,dPdQneg_NEW,'LineWidth',1.2,'Color','b')
        ylabel('dV/dQ^n (V/Ah)')
    yyaxis left
        plot(Qcell,dPdQpos_NEW,'LineWidth',1.2,'Color','r')
        hold on
        plot(Qcell,dVdQcell,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(Qcell,dVdQcalc,'LineWidth',1.2,'Color',"#7E2F8E",'LineStyle','--')
        ylabel('dV/dQ^p (V/Ah)')
        xlabel('Capacity (Ah)')

subplot(3,2,5)
    yyaxis right
        plot(Qcell,dQdPneg_NEW,'LineWidth',1.2,'Color','b')
        ylabel('dQ/dV^n (Ah/V)')
    yyaxis left
        plot(Qcell,dQdPpos_NEW,'LineWidth',1.2,'Color','r')
        hold on
        plot(Qcell,dQdVcell,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(Qcell,dQdVcalc,'LineWidth',1.2,'Color',"#7E2F8E",'LineStyle','--')
        ylabel('dQ/dV^p (Ah/V)')
        xlabel('Capacity (Ah)')

Part = {'neg';'pos';'cell'};
SOL100 = [STCH_LIMITS_BOL.SOLneg100; STCH_LIMITS_BOL.SOLpos100; max(qSOLcell)];
```

```
SOL0 = [STCH_LIMITS_BOL.SOLneg0; STCH_LIMITS_BOL.SOLpos0; min(qSOLcell)];
Q100 = [STCH_LIMITS_BOL.Qneg100; STCH_LIMITS_BOL.Qpos100; max(qQcell)];
Q0 = [STCH_LIMITS_BOL.Qneg0; STCH_LIMITS_BOL.Qpos0; min(qQcell)];
Q = [STCH_LIMITS_BOL.max_Qneg; STCH_LIMITS_BOL.max_Qpos; max(qQcell)];

fprintf('BOL optimization results:')
BOL_Optimization = table(Part,SOL100,SOL0,Q100,Q0,Q);

fprintf('BOL optimization error:')
Curve = {'OCV';'DV';'IC'};
RMS = [Error.OCV_RMSe; Error.dVdQ_RMSe; Error.dQdV_RMSe];
Max = [Error.OCV_Max; Error.dVdQ_Max; Error.dQdV_Max];
BOL_Optimization_error = table(Curve,RMS,Max);

%%
%%%%%%%
% AGED %
%%%%%%%
switch analyze_AGED
    case 'AGED'
        qQ = AGEDdata.C_curve(~isnan(AGEDdata.C_curve));
        coeficients = polyfit(1:length(qQ),qQ,1);
        if coeficients(1) < 0
            qQ = flip(qQ);
        else
        end
        qSOL = AGEDdata.SOL_curve(~isnan(AGEDdata.SOL_curve));
        coeficients = polyfit(1:length(qSOL),qSOL,1);
        if coeficients(1) > 0
            qSOL = flip(qSOL);
        end
        if mean(qSOL) > 10
            qSOL = qSOL./100;
        else
        end
        qOCP = AGEDdata.V_curve(~isnan(AGEDdata.V_curve));
        coeficients = polyfit(1:length(qOCP),qOCP,1);
        if coeficients(1) > 0
            qOCP = flip(qOCP);
        else
        end
        [qQ, qSOL, qOCP] = MonotonicArray(qQ,qSOL,qOCP,size);
        qQaged= qQ; qSOLaged = qSOL; qOCPaged = qOCP;

figure
    yyaxis left
        plot(qQposFull,qOCPpos,'LineWidth',1.2,'Color','r','LineStyle','-')
        hold on
        plot(qQcell,qOCPcell,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(qQaged,qOCPaged,'LineWidth',1.2,'Color',"#77AC30",'LineStyle','-')
        xlabel('Capacity (Ah)')
        ylabel('OCP^{p} or OCV (V)')
    yyaxis right
        plot(qQnegFull,qOCPneg,'LineWidth',1.2,'Color','b','LineStyle','-')
        xlabel('Capacity (Ah)')
        ylabel('OCP^{n} (V)')
        legend('PE','OCV','aged','NE')

switch scaling_type
    case {'OCV','size'}
                axes('Position',[.2 .3 .2 .2])
                box on
                    yyaxis left
                        plot(qQpos,qOCPpos,'LineWidth',1.2,'Color','r','LineStyle','-')
                        xlabel('Capacity (mAh)')
                        ylabel('OCP^{p} (V)')
                    yyaxis right
                        plot(qQneg,qOCPneg,'LineWidth',1.2,'Color','b','LineStyle','-')
                        xlabel('Capacity (mAh)')
                        ylabel('OCP^{n} (V)')
end

figure
    yyaxis left
```

```
        plot(qSOLpos,qOCPpos,'LineWidth',1.2,'Color','r','LineStyle','-')
        hold on
        plot(qSOLcell,qOCPcell,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(qSOLaged,qOCPaged,'LineWidth',1.2,'Color',"#77AC30",'LineStyle','-')
        xlabel('SOL or SOC ()')
        ylabel('OCP^{p} or OCV (V)')
    yyaxis right
        plot(qSOLneg,qOCPneg,'LineWidth',1.2,'Color','b','LineStyle','-')
        xlabel('SOL or SOC  (Ah)')
        ylabel('OCP^{n} (V)')
        legend('PE','OCV','aged','NE')

%%%%%%%%%%%%%%%
% OPTIMIZATION %
%%%%%%%%%%%%%%%
clear x
clear fval

% New bounds dependant on BOL results
nvars = 4;
Qneg0_max = max(qQnegFull)-max(qQaged);
Qpos0_max = max(qQposFull)-max(qQaged);
ub = [Qneg0_max Qpos0_max max(Qneg_opt)/max(qQnegFull) max(Qpos_opt)/max(qQposFull)];
lb = [min(qQnegFull) min(qQposFull) max(qQaged)/max(qQnegFull) max(qQaged)/max(qQposFull)];

% Linear inequality constraints
N = max(qQnegFull);
P = max(qQposFull);
C = max(qQaged);

A = [1 0 -N 0;
     0 1 0 -P];
b = [-C;
     -C];

% Computational cost
tStart = tic;
fprintf('genetic algorithm Optimization is running...    ');

% Cost function handle
CostFunction = @(x) [OCVopt(qQaged,qOCPaged,qQnegFull,qOCPneg,qQposFull,qOCPpos,x),
dVdQopt(qQaged,qOCPaged,qQnegFull,qOCPneg,qQposFull,qOCPpos,x),
dQdVopt(qQaged,qOCPaged,qQnegFull,qOCPneg,qQposFull,qOCPpos,x)]

% Algorithm
[x,fval,exitflag,output] = gamultiobj(CostFunction,nvars,A,b,[],[],lb,ub,[],[],options)

% Optimization output
fprintf('AGED OCV fitting computational cost:')
tEnd = toc(tStart)
[~,pos] = min(fval(:,1).*fval(:,2));
x = x(pos,:);

%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZATION RESULTS %
%%%%%%%%%%%%%%%%%%%%%%
% negative electrode %
    coef = x(3);
    Qneg_opt = qQnegFull.*coef;

    Qneg100 = x(1);
    Qneg0 = Qneg100+max(qQaged);                              % Upper limit of effective electrode
capacity
    a = find(Qneg_opt>=Qneg100);
    b = find(Qneg_opt<=Qneg0);
    if isempty(a) || a(1)<=1
        Qneg_NEW = Qneg_opt(1:b(end));                         % Efective electrode capacity array
        Pneg_NEW = qOCPneg(1:b(end));                          % Effective electrode potential array
        SOLneg_NEW = qSOLneg(1:b(end));                        % Effective electrode SOL array
    else
        Qneg_NEW = Qneg_opt(a(1):b(end));                         % Efective electrode capacity array
        Pneg_NEW = qOCPneg(a(1):b(end));                          % Effective electrode potential
array
        SOLneg_NEW = qSOLneg(a(1):b(end));                        % Effective electrode SOL array
```

```
    end
    OCPneg = interp1(Qneg_NEW-Qneg_NEW(1),Pneg_NEW,qQaged,"linear","extrap"); % Electrode OCP at BOL
(referenced to 0)

    STCH_LIMITS_Aged.max_Qneg = max(Qneg_opt);
    STCH_LIMITS_Aged.Qneg100 = Qneg_NEW(1);
    STCH_LIMITS_Aged.Qneg0 = Qneg_NEW(end);
    STCH_LIMITS_Aged.SOLneg100 = SOLneg_NEW(1);
    STCH_LIMITS_Aged.SOLneg0 = SOLneg_NEW(end);

% positive electrode %
    coef = x(4);
    Qpos_opt = qQposFull.*coef;

    Qpos100 = x(2);
    Qpos0 = Qpos100+max(qQaged);                              % Upper limit of effective electrode
capacity
    a = find(Qpos_opt>=Qpos100);
    b = find(Qpos_opt<=Qpos0);
    if isempty(a) || a(1)<=1
        Qpos_NEW = Qpos_opt(1:b(end));                         % Efective electrode capacity array
        Ppos_NEW = qOCPpos(1:b(end));                          % Effective electrode potential array
        SOLpos_NEW = qSOLpos(1:b(end));                        % Effective electrode SOL array
    else
        Qpos_NEW = Qpos_opt(a(1):b(end));                         % Efective electrode capacity array
        Ppos_NEW = qOCPpos(a(1):b(end));                          % Effective electrode potential
array
        SOLpos_NEW = qSOLpos(a(1):b(end));                        % Effective electrode SOL array
    end
    OCPpos = interp1(Qpos_NEW-Qpos_NEW(1),Ppos_NEW,qQaged,"linear","extrap"); % Electrode OCP at BOL
(referenced to 0)

    STCH_LIMITS_Aged.max_Qpos = max(Qpos_opt);
    STCH_LIMITS_Aged.Qpos100 = Qpos_NEW(1);
    STCH_LIMITS_Aged.Qpos0 = Qpos_NEW(end);
    STCH_LIMITS_Aged.SOLpos100 = SOLpos_NEW(1);
    STCH_LIMITS_Aged.SOLpos0 = SOLpos_NEW(end);

% ocv %
    OCV_calculated = OCPpos - OCPneg;
    e = qOCPaged-OCV_calculated;
    a = ~isnan(e);
    e = e(a);
    Error.RMSe_AGED = rms(e);
    Error.Max_AGED = max(e);

% dV/dQ & dQ/dV %

   % aged %
        [dQdVaged,dVdQaged,Qaged,Vaged] = IC_DV(qQaged,qOCPaged);
        dQdVaged = flip(dQdVaged);
        dVdQaged = flip(dVdQaged);

    % negative electrode %
        [dQdPneg,dPdQneg,Qneg,Pneg] = IC_DV(Qneg_opt,qOCPneg);
            dPdQneg_opt = interp1(Qneg,dPdQneg,Qneg_opt,"linear","extrap");
            dPdQneg_NEW = interp1(Qneg,dPdQneg,Qneg_NEW,"linear","extrap");
            dPdQneg_NEW = interp1(Qneg_NEW-Qneg_NEW(1),dPdQneg_NEW,Qaged,"linear","extrap");

            dQdPneg_opt = interp1(Qneg,dQdPneg,Qneg_opt,"linear","extrap");
            dQdPneg_NEW = interp1(Qneg,dQdPneg,Qneg_NEW,"linear","extrap");
            dQdPneg_NEW = interp1(Qneg_NEW-Qneg_NEW(1),dQdPneg_NEW,Qaged,"linear","extrap");

    % positive electrode %
        [dQdPpos,dPdQpos,Qpos,Ppos] = IC_DV(Qpos_opt,qOCPpos);
        dQdPpos = flip(dQdPpos);
        dPdQpos = flip(dPdQpos);
            dPdQpos_opt = interp1(Qpos,dPdQpos,Qpos_opt,"linear","extrap");
            dPdQpos_NEW = interp1(Qpos,dPdQpos,Qpos_NEW,"linear","extrap");
            dPdQpos_NEW = interp1(Qpos_NEW-Qpos_NEW(1),dPdQpos_NEW,Qaged,"linear","extrap");

            dQdPpos_opt = interp1(Qpos,dQdPpos,Qpos_opt,"linear","extrap");
            dQdPpos_NEW = interp1(Qpos,dQdPpos,Qpos_NEW,"linear","extrap");
            dQdPpos_NEW = interp1(Qpos_NEW-Qpos_NEW(1),dQdPpos_NEW,Qaged,"linear","extrap");
```

```
    % calculated %
        [dQdVcalc,dVdQcalc,Qcalc,Vcalc] = IC_DV(qQaged,OCV_calculated);
        dQdVcalc = flip(dQdVcalc);
        dVdQcalc = flip(dVdQcalc);

        dVdQcalc = interp1(Qcalc,dVdQcalc,Qaged,"linear","extrap");
        e = dVdQaged-dVdQcalc;
        a = ~isnan(e);
        e = e(a);
        Error.dVdQ_RMSe_AGED = rms(e);
        Error.dVdQ_Max_AGED = max(e);

        dQdVcalc = interp1(Qcalc,dQdVcalc,Qaged,"linear","extrap");
        e = dQdVaged-dQdVcalc;
        a = ~isnan(e);
        e = e(a);
        Error.dQdV_RMSe_AGED = rms(e);
        Error.dQdV_Max_AGED = max(e);

% Optimization output %
    % negative electrode % .neg
    AGEDopt.neg.Q = Qneg_opt;
    AGEDopt.neg.Q_stch = Qneg_NEW;
    AGEDopt.neg.SOL = qSOLneg;
    AGEDopt.neg.SOL_stch = SOLneg_NEW;
    AGEDopt.neg.OCP = qOCPneg;
    AGEDopt.neg.OCP_stch = OCPneg;
    AGEDopt.neg.Q100 = Qneg_NEW(1);
    AGEDopt.neg.Q0 = Qneg_NEW(end);
    AGEDopt.neg.SOL100 = SOLneg_NEW(1);
    AGEDopt.neg.SOL0 = SOLneg_NEW(end);
    AGEDopt.neg.dPdQ = dPdQneg_opt;
    AGEDopt.neg.dPdQ_stch = dPdQneg_NEW;
    AGEDopt.neg.dQdP = dQdPneg_opt;
    AGEDopt.neg.dQdP_stch = dQdPneg_NEW;

    % positive electrode % .pos
    AGEDopt.pos.Q = Qpos_opt;
    AGEDopt.pos.Q_stch = Qpos_NEW;
    AGEDopt.pos.SOL = qSOLpos;
    AGEDopt.pos.SOL_stch = SOLpos_NEW;
    AGEDopt.pos.OCP = qOCPpos;
    AGEDopt.pos.OCP_stch = OCPpos;
    AGEDopt.pos.Q100 = Qpos_NEW(1);
    AGEDopt.pos.Q0 = Qpos_NEW(end);
    AGEDopt.pos.SOL100 = SOLpos_NEW(1);
    AGEDopt.pos.SOL0 = SOLpos_NEW(end);
    AGEDopt.pos.dPdQ = dPdQpos_opt;
    AGEDopt.pos.dPdQ_stch = dPdQpos_NEW;
    AGEDopt.pos.dQdP = dQdPpos_opt;
    AGEDopt.pos.dQdP_stch = dQdPpos_NEW;

    % commercial aged % .aged
    AGEDopt.cell.Q = qQaged;
    AGEDopt.cell.Q_for_dVdQ = Qaged;
    AGEDopt.cell.SOL = qSOLaged;
    AGEDopt.cell.OCP = qOCPaged;
    AGEDopt.cell.dPdQ = dVdQaged;
    AGEDopt.cell.dQdP = dQdVaged;

    % calculated % .calc
    AGEDopt.calc.Q = qQaged;
    AGEDopt.calc.Q_for_dVdQ = Qaged;
    AGEDopt.calc.SOL = qSOLaged;
    AGEDopt.calc.OCP = OCV_calculated;
    AGEDopt.calc.dPdQ = dVdQcalc;
    AGEDopt.calc.dQdP = dQdVcalc;

    % error %
    AGEDopt.error.OCV_RMSE = Error.RMSe_AGED;
    AGEDopt.error.OCV_Max = Error.Max_AGED;
    AGEDopt.error.dVdQ_RMSE = Error.dVdQ_RMSe_AGED;
    AGEDopt.error.dVdQ_Max = Error.dVdQ_Max_AGED;
```

```
    AGEDopt.error.dQdV_RMSE = Error.dQdV_RMSe_AGED;
    AGEDopt.error.dQdV_Max = Error.dQdV_Max_AGED;

% Visualization %
figure(10)
subplot(3,2,2)
    yyaxis left
        plot(Qpos_opt,qOCPpos,'LineWidth',1.2,'Color','r')
        hold on
        plot(qQaged,qOCPaged,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(qQaged,OCV_calculated,'LineWidth',1.2,'Color',"#7E2F8E",'LineStyle','--')
        plot(Qpos_NEW,Ppos_NEW,'k','LineWidth',1.2,'LineStyle','-')
        xline([STCH_LIMITS_Aged.Qpos100 STCH_LIMITS_Aged.Qpos0],'-.r','LineWidth',1)
        xlabel('Capacity (Ah)')
        ylabel('OCP^{p} or OCV (V)')

    yyaxis right
        plot(Qneg_opt,qOCPneg,'LineWidth',1.2,'Color','b')
        hold on
        plot(Qneg_NEW,Pneg_NEW,'k','LineWidth',1.2,'LineStyle','-')
        xline([STCH_LIMITS_Aged.Qneg100 STCH_LIMITS_Aged.Qneg0],'-.b','LineWidth',1)
        ylabel('OCP^{n} (V)')
        legend('PE','OCV','OCV*','','','','NE','','','','Location','best')

subplot(3,2,4)
    yyaxis right
        plot(Qaged,dPdQneg_NEW,'LineWidth',1.2,'Color','b')
        ylabel('dV/dQ^n (V/Ah)')
    yyaxis left
        plot(Qaged,dPdQpos_NEW,'LineWidth',1.2,'Color','r')
        hold on
        plot(Qaged,dVdQaged,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(Qaged,dVdQcalc,'LineWidth',1.2,'Color',"#7E2F8E",'LineStyle','--')
        ylabel('dV/dQ^p (V/Ah)')
        xlabel('Capacity (Ah)')

subplot(3,2,6)
    yyaxis right
        plot(Qaged,dQdPneg_NEW,'LineWidth',1.2,'Color','b')
        ylabel('dQ/dV^n (Ah/V)')
    yyaxis left
        plot(Qaged,dQdPpos_NEW,'LineWidth',1.2,'Color','r')
        hold on
        plot(Qaged,dQdVaged,'LineWidth',1.2,'Color',"#EDB120",'LineStyle','-')
        plot(Qaged,dQdVcalc,'LineWidth',1.2,'Color',"#7E2F8E",'LineStyle','--')
        ylabel('dQ/dV^p (Ah/V)')
        xlabel('Capacity (Ah)')

Part = {'neg';'pos';'aged'};
SOL100 = [STCH_LIMITS_Aged.SOLneg100; STCH_LIMITS_Aged.SOLpos100; max(qSOLaged)];
SOL0 = [STCH_LIMITS_Aged.SOLneg0; STCH_LIMITS_Aged.SOLpos0; min(qSOLaged)];
Q100 = [STCH_LIMITS_Aged.Qneg100; STCH_LIMITS_Aged.Qpos100; max(qQaged)];
Q0 = [STCH_LIMITS_Aged.Qneg0; STCH_LIMITS_Aged.Qpos0; min(qQaged)];
Q = [STCH_LIMITS_Aged.max_Qneg; STCH_LIMITS_Aged.max_Qpos; max(qQaged)];

fprintf('Aged optimization results:')
Aged_Optimization = table(Part,SOL100,SOL0,Q100,Q0,Q);

fprintf('Aged optimization error:')
Curve = {'OCV';'DV';'IC'};
RMS = [Error.RMSe_AGED; Error.dVdQ_RMSe_AGED; Error.dQdV_RMSe_AGED];
Max = [Error.Max_AGED; Error.dVdQ_Max_AGED; Error.dQdV_Max_AGED];
Aged_Optimization_error = table(Curve,RMS,Max);

%%
%%%%%%%%%%%%%%%%%%%%%%%%%
% DEGRADATION ESTIMATION %
%%%%%%%%%%%%%%%%%%%%%%%%%
% Limits %
STCH_LIMITS_BOL;
STCH_LIMITS_Aged;

% Loss of Active Material
    STCH_LIMITS_Aged.LAMneg = (1-STCH_LIMITS_Aged.max_Qneg/STCH_LIMITS_BOL.max_Qneg);
```

```
    STCH_LIMITS_Aged.LAMpos = (1-STCH_LIMITS_Aged.max_Qpos/STCH_LIMITS_BOL.max_Qpos);

% Loss of Lithium Inventory
    n_Li_BOL =
3600/96.485e3*(STCH_LIMITS_BOL.SOLneg100*STCH_LIMITS_BOL.max_Qneg+STCH_LIMITS_BOL.SOLpos100*STCH_LIM
ITS_BOL.max_Qpos);
    n_Li_AGED =
3600/96.485e3*(STCH_LIMITS_Aged.SOLneg100*STCH_LIMITS_Aged.max_Qneg+STCH_LIMITS_Aged.SOLpos100*STCH_
LIMITS_Aged.max_Qpos);
    STCH_LIMITS_Aged.LLI = (1-n_Li_AGED/n_Li_BOL);

% Capacity fade
Qfade = max(qQaged)/max(qQcell);

DegMode = {'Est'};
LAMneg = STCH_LIMITS_Aged.LAMneg;
LAMpos = STCH_LIMITS_Aged.LAMpos;
LLI = STCH_LIMITS_Aged.LLI;
Fade = Qfade;
DegModeEstimation = table(DegMode,LAMneg,LAMpos,LLI,Fade);

    case 'NOT'
        STCH_LIMITS_Aged = 0;
        AGEDopt = [];
end

BOL_Optimization
BOL_Optimization_error

switch analyze_AGED
    case 'AGED'
        Aged_Optimization
        Aged_Optimization_error
        DegModeEstimation
    case 'NOT'
end
end

%% Monotonic array for ModEst
function [Qnew,SOCnew,OCVnew] = MonotonicArray(Qold,SOCold,OCVold,size)

if length(Qold) ~= length(OCVold)
    Qold = min(Qold):max(Qold)/(length(OCVold)-1):max(Qold);
end

[Qs, vec] = sort(Qold(:).');
uvec1(vec) = [true, diff(Qs) ~= 0];

Qold = Qold(uvec1);
OCVold = OCVold(uvec1);
SOCold = SOCold(uvec1);

Qnew = min(Qold):max(Qold)/size:max(Qold);
OCVnew = interp1(Qold,OCVold,Qnew);
SOCnew = interp1(Qold,SOCold,Qnew);

end

%% OCV cost function
function y = OCVopt(Q,OCV,Qneg,Pneg,Qpos,Ppos,x)

OCV_calculated = OCVfromOCPs(Q,Qneg,Pneg,Qpos,Ppos,x);

e = OCV-OCV_calculated;
a = ~isnan(OCV-OCV_calculated);
e = e(a);
y = rms(e);

end

%% OCV function
function OCV_calc = OCVfromOCPs(Q,Qneg,Pneg,Qpos,Ppos,x)

x_param = x;
```

```
if length(x) > 2    % check x length to see if capacity coeficients need to be calculated
    coef = x_param(3);
    Qneg = Qneg.*coef;
    coef = x_param(4);
    Qpos = Qpos.*coef;
else
    % FRESH OCV curves consideration %
end

Qneg_ini = x_param(1);
a = find(Qneg>=Qneg_ini);
if isempty(a) || a(1)<=1
    Qneg_new = Qneg(1:end);
    Pneg_new = Pneg(1:end);
else
    Qneg_new = Qneg(a(1):end);
    Pneg_new = Pneg(a(1):end);
end
OCPneg = interp1(Qneg_new-Qneg_new(1),Pneg_new,Q,"linear","extrap");  % Effective OCP

Qpos_ini = x_param(2);
a = find(Qpos>=Qpos_ini);
if isempty(a) || a(1)<=1
    Qpos_new = Qpos(1:end);
    Ppos_new = Ppos(1:end);
else
    Qpos_new = Qpos(a(1):end);
    Ppos_new = Ppos(a(1):end);
end
OCPpos = interp1(Qpos_new-Qpos_new(1),Ppos_new,Q,"linear","extrap");

OCV_calc = OCPpos - OCPneg;

end

%% dVdQ cost function
function y = dVdQopt(Q,OCV,Qneg,Pneg,Qpos,Ppos,x)

OCV_calculated = OCVfromOCPs(Q,Qneg,Pneg,Qpos,Ppos,x);

% cell %
[dQdVcell,dVdQcell,Qcell,Vcell] = IC_DV(Q,OCV);

% calculated %
[dQdVcalc,dVdQcalc,Qcalc,Vcalc] = IC_DV(Q,OCV_calculated);
dVdQcalc = interp1(Qcalc,dVdQcalc,Qcell);

e = dVdQcell-dVdQcalc;
a = ~isnan(e);
e = e(a);
y = rms(e);

end

%% dQdV cost function
function y = dQdVopt(Q,OCV,Qneg,Pneg,Qpos,Ppos,x)

OCV_calculated = OCVfromOCPs(Q,Qneg,Pneg,Qpos,Ppos,x);

% cell %
[dQdVcell,dVdQcell,Qcell,Vcell] = IC_DV(Q,OCV);

% calculated %
[dQdVcalc,dVdQcalc,Qcalc,Vcalc] = IC_DV(Q,OCV_calculated);
dQdVcalc = interp1(Qcalc,dQdVcalc,Qcell);

e = dQdVcell-dQdVcalc;
a = ~isnan(e);
e = e(a);
y = rms(e);

end
```

```
%% IC & DV function
function [IC,DV,Q,V] = IC_DV(Q,V)

coeficients = polyfit(1:length(V),V,1);
if coeficients(1) < 0                                % dV will always be positive, so it will need to
be flipped for positive electrode and cell curves as Voltage array goes ascending
    V = flip(V);
else
end

Qold = Q;
Vold = V;

% DV Curve: Monotonic dQ.
% [Qs, vec] = sort(Qold(:).');
% uvec1(vec) = [true, diff(Qs) ~= 0];
[~,uvec1] = unique(Qold,'stable');
Q = Qold(uvec1);
V = Vold(uvec1);

dQ = [0, diff(Q)];
dV = [0, diff(V)];

DV = dV./dQ;
DV = interp1(Q,DV,Qold,'linear','extrap');

% IC Curve: Monotonic dV.
[~,uvec2] = unique(Vold,'stable');
V = Vold(uvec2);
Q = Qold(uvec2);

dV = [0, diff(V)];
    a = find(dV > 3e-6);
    dV = dV(a);
dQ = [0, diff(Q)];
    dQ = dQ(a);

IC = dQ./dV;
IC = interp1(Q(a),IC,Qold,'linear','extrap');

% [Vs, vec] = sort(Vold(:).');
% uvec2(vec) = [true, diff(Vs) ~= 0];
% V = Vold(uvec2);
% Q = Qold(uvec2);
%
% dV = [0, diff(V)];
% dQ = [0, diff(Q)];
% % a = find(dV>1e-6);
% %     dV = dV(a);
% %     dQ = dQ(a);
%
% IC = dQ./dV;
% IC = interp1(Q,IC,Qold,'linear','extrap');


% Result
% figure
% subplot(2,1,1)
% plot(Qold,DV)
% subplot(2,1,2)
% plot(Qold,IC)

% Smoothing
x_5 = [-5 -4 -3 -2 -1 0 1 2 3 4 5];
alpha_5 = normpdf(x_5,0,3);
filt_5 = alpha_5./sum(alpha_5);
windowWidth = 400; % Increase for more smoothing.

trunc = 500;                                                    % some of the beginning and end
data can cause the capacity trend to be non-increasing, which cause error in findpeaks
IC = conv(IC, filt_5);
    IC = IC(trunc:end-trunc);
    IC = conv(IC, ones(windowWidth,1)/windowWidth, 'same');
```

```
DV = conv(DV, filt_5);
    DV = DV(trunc:end-trunc);
    DV = conv(DV, ones(windowWidth,1)/windowWidth, 'same');

Q = Qold;
Q = conv(Q, filt_5);                          % add in the capacity x-axis
    Q = Q(trunc:end-trunc);

% figure
% subplot(2,1,1)
% plot(Q,DV)
% subplot(2,1,2)
% plot(Q,IC)

end
```

5 AUTORES

Laura Oca Perez

Iker Lopetegi Tapia

Sergio Fernandez Gonzalez