1. Aim

The aim of this project is to simulate a full cycle Single particle model of a Li-ion battery, model its capacity fade over charge and discharge cycles and observe its thermal behavior.

2. Publications Reviewed

1. **Meng Guo et al.; Single-Particle Model for a Lithium-Ion Cell: Thermal Behavior 2011**

   Through this paper, I learned about the basic implementation of the Single Particle model of a Li-ion battery. The key assumption in the single-particle model is that the current distribution is taken to be uniform along the thickness of the porous electrode which can thus be represented by a single spherical intercalation particle. The concentration profile in the cell with time and space was plotted using the following equation for mass balance of lithium ions:

$$\frac{\partial c}{\partial t} = D\left(\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial c}{\partial r}\right)\right)$$

   with the boundary conditions:

$$c(r=0) = c_0$$

$$D\frac{\partial c}{\partial r}\bigg|_{r=R} = -j$$

   This equation was converted to code using Finite Difference Methods.
   Next, the Butler-Volmer equation was used to obtain the cell potential based on the calculated value of the overpotential and interpolated value of the open circuit potential. The equations involved are:

$$\eta = \frac{RT}{\alpha_c F}\sinh^{-1}\left(\frac{i}{2i_0}\right)$$

$$V = E_{\text{oc}} - \eta$$

   The values for various parameters for the code were also derived from this paper.

2. **Gang Ning and Branko N. Popov; Cycle Life Modeling of Lithium-Ion Batteries 2004**

   Through this paper, I learned about a continuous small-scale reduction that can take place on the negative electrode when the solvent percolates through the cracks of its surface film. A part of lithium is irreversibly lost due to this parasitic reaction. No parasitic reaction has been considered on the surface of the positive electrode. This results in a capacity fade across multiple cycles and can be modelled using the Tafel equation:

$$i = -i_0 \cdot \exp\left(-\frac{a_s nF}{RT}\eta_s\right)$$

   The total current density at the negative electrode is the sum of the intercalation/deintercalation current density (Butler-Volmer) and the parasitic reaction current density (Tafel). The loss of the active lithium is estimated using the following equation:

$$Q = \int_0^T i(t)\,dt$$

   These equations will now be used to model the capacity fade across multiple cycles using the

Newton Raphson method which will be used to estimate the overpotential and subsequently the cell potential.

3. Code

3.A. Full Cycle SP Model

Listing 1: Full Cycle SP model

```
clc
clearvars
close all

% Constants
F = 96485;                 % Faraday's constant (C/mol)
C_Li = 1000;               % Lithium concentration (mol/m^3)
T = 298;                   % Temperature (K)
Rgc = 8.314;               % Gas constant (J/(mol*K))

% Time Parameters
dt = 0.05;                 % Time step (s)
t = 0:dt:3600;             % Time vector (s)
Nt = numel(t);             % Number of time steps

% Voltage Initialization
voltage = zeros(Nt,1);     % Voltage array (V)

% Cathode Parameters
R_p = 8.5E-6;              % Cathode radius (m)
dr_p = R_p/20;             % Cathode radial step size (m)
D_p = 1E-14;               % Diffusion coefficient for cathode (m^2/s)
Nr_p = numel(r_p);         % Number of radial steps for cathode
C_p = zeros(Nr_p,Nt);      % Concentration matrix for cathode
Cmax_p = 51410;            % Maximum concentration for cathode (mol/m^3)
curr_den_p = 1.5;          % Current density for cathode (A/m^2)
a_p = 0.5;                 % Butler Volmer slope for cathode
k_p = 6.67E-11;            % Reaction rate constant for cathode
    (m^2/(s*mol^(1-a)))
over_pot_p = zeros(Nt,1);    % Overpotential array for cathode (V)
U_ocp_p = zeros(Nt,1);     % Open circuit potential array for cathode (V)
phi_p = zeros(Nt,1);       % Electric potential array for cathode (V)

% Anode Parameters
R_n = 12.5E-6;             % Anode radius (m)
dr_n = R_n/20;             % Anode radial step size (m)
D_n = 3.9E-14;             % Diffusion coefficient for anode (m^2/s)
Nr_n = numel(r_n);         % Number of radial steps for anode
C_n = zeros(Nr_n,Nt);      % Concentration matrix for anode
Cmax_n = 31833;            % Maximum concentration for anode (mol/m^3)
curr_den_n = curr_den_p;     % Current density for anode (A/m^2)
a_n = 0.5;                 % Butler Volmer slope for anode
k_n = 1.764E-11;           % Reaction rate constant for anode
    (m^2/(s*mol^(1-a)))
over_pot_n = zeros(Nt,1);    % Overpotential array for anode (V)
U_ocp_n = zeros(Nt,1);     % Open circuit potential array for anode (V)
phi_n = zeros(Nt,1);       % Electric potential array for anode (V)

```

```
% Initial Concentrations
Ci_p = 0.95*Cmax_p;      % Initial concentration for cathode (mol/m^3)
Ci_n = 0.5*Cmax_n;       % Initial concentration for anode (mol/m^3)

% Initial Charge
Q(1,1) = Ci_n;           % Initial charge in the cell (mol)

% Counter for Cycles
counter = 0;             % Counter for switching current direction

% Main Loop for Cycles
for cycle = 1:10

    % Reset Concentrations
    C_p = zeros(Nr_p, Nt);
    C_n = zeros(Nr_n, Nt);
    C_p(:,1) = Ci_p;
    C_n(:,1) = Ci_n;

    % Control Current Direction
    if counter == 0
        counter = 1;
        if curr_den_p < 0
            curr_den_p = -curr_den_p;
            curr_den_n = -curr_den_n;
        end

    elseif counter == 1
        counter = 0;
        if curr_den_p > 0
            curr_den_p = -curr_den_p;
            curr_den_n = -curr_den_n;
        end
    end

    % Boundary Elements
    J_0_p = k_p*((Cmax_p-C_p(Nr_p,1))*C_p(Nr_p,1)*C_Li)^a_p;
    J_0_n = k_n*((Cmax_n-C_n(Nr_n,1))*C_n(Nr_n,1)*C_Li)^a_n;

    over_pot_p(1,1) = (Rgc*T/(a_p*F))*asinh(curr_den_p/(2*J_0_p));
    over_pot_n(1,1) = (Rgc*T/(a_n*F))*asinh(curr_den_n/(2*J_0_n));

    SOC_p = C_p(Nr_p,1)/Cmax_p;
    SOC_n = C_n(Nr_n,1)/Cmax_n;

    U_ocp_p(1,1) = Uocp_interp(SOC_p);
    U_ocp_n(1,1) = Uocp_interp2(SOC_n);

    phi_p(1,1) = over_pot_p(1,1) + U_ocp_p(1,1);
    phi_n(1,1) = over_pot_n(1,1) + U_ocp_n(1,1);

    voltage(1,1) = phi_n(1,1) - phi_p(1,1);

    % Time Stepping
    for j = 2:Nt

        for i = 2:Nr_p-1
            % Cathode Concentration
            C_p(i,j) = C_p(i,j-1) +
               D_p*dt*(((C_p(i+1,j-1)-2*C_p(i,j-1)+ ...
            C_p(i-1,j-1))/(dr_p)^2)+((C_p(i+1,j-1)-C_p(i-1,j-1))/ ...
            (r_p(i)*dr_p)));
        end

        for i = 2:Nr_n-1
            % Anode Concentration
            C_n(i,j) = C_n(i,j-1) +
               D_n*dt*(((C_n(i+1,j-1)-2*C_n(i,j-1)+ ...
            C_n(i-1,j-1))/(dr_n)^2)+((C_n(i+1,j-1)-C_n(i-1,j-1))/ ...
            (r_n(i)*dr_n)));
        end

        % Cathode Boundary Conditions
        C_p(1,j) = C_p(2,j);
        C_p(Nr_p,j) = C_p(Nr_p-1,j) - curr_den_p*dr_p/(D_p*F);

        % Anode Boundary Conditions
        C_n(1,j) = C_n(2,j);
        C_n(Nr_n,j) = C_n(Nr_n-1,j) + curr_den_n*dr_n/(D_n*F);

        % Update Current Density
        J_0_p = k_p*((Cmax_p-C_p(Nr_p,j))*C_p(Nr_p,j)*C_Li)^a_p;
        J_0_n = k_n*((Cmax_n-C_n(Nr_n,j))*C_n(Nr_n,j)*C_Li)^a_n;

        over_pot_p(j,1) = (Rgc*T/(a_p*F))*asinh(curr_den_p/(2*J_0_p));
        over_pot_n(j,1) = (Rgc*T/(a_n*F))*asinh(curr_den_n/(2*J_0_n));

        SOC_p = C_p(Nr_p,j)/Cmax_p;
        SOC_n = C_n(Nr_n,j)/Cmax_n;

        U_ocp_p(j,1) = Uocp_interp(SOC_p);
        U_ocp_n(j,1) = Uocp_interp2(SOC_n);

        phi_p(j,1) = over_pot_p(j,1) + U_ocp_p(j,1);
        phi_n(j,1) = over_pot_n(j,1) + U_ocp_n(j,1);

        voltage(j,1) = phi_n(j,1) - phi_p(j,1);

        % Break Condition
        if (voltage(j,1) < 3 || SOC_p < 0.05) && counter == 1
            break
        elseif  (voltage(j,1) > 4.2 || SOC_n < 0.5) && counter == 0
            break
        end

        % Plotting
        figure(1)
        timevec = 0+(cycle-1)*3600:dt:3600+(cycle-1)*3600;
        axis([0 36000 3  5])
        plot(timevec,voltage,'-');
        xlabel("Time")
        ylabel("Cell Voltage")
        pause(1)
        hold on

    end
    % Values for Next Cycle
    Ci_p = C_p(Nr_p, Nt);
    Ci_n = C_n(Nr_n, Nt);

end
```

3.B. OCP interpolation function

Listing 2: OCP interpolation function

```
% Function to interpolate open circuit potential (U_ocp) based on state
    of charge (SOC)
function [U_ocp] = Uocp_interp(SOC)
% Data points for interpolation
x = [0.1551155966462897 7, 0.05082515796370215 6, 0.0600660408381556 4,
    0.3399340598799139, 0.07194724790393106, ...
     0.1155116402390845, 0.1907591171255465 5, 0.2237624141315508 7,
        0.2686469182033306 4, 0.3095709863471621, ...
     0.3742574686225445, 0.4059406538919224 6, 0.4415841743711793 6,
        0.4772276948504361 5, 0.5102309918564405, ...
     0.5379538419159398, 0.5735973623951964, 0.6052805476645746,
        0.6369637329339526, 0.6726072534132095, ...
     0.7148514332936673, 0.7504950544909936, 0.7993398937726521,
        0.8442243978444317, 0.8838284549697064, ...
     0.9141913270663193, 0.9353135177246177, 0.9511550600002721,
        0.0455446102991276 5, 0.0415841743711792 8, ...
     0.0363037274246741 9, 0.0336634032335212 4, 0.0283828555687776 2,
        0.0257426320955249 2, 0.0217821961675770 6, ...
     0.0191419726943241 6, 0.0178218609576979 16];
% Corresponding U_ocp values
U = [0.1636735056122742 3, 0.2869387920336689 5, 0.2542857409774652 6,
    0.1171429016291087, 0.21755105853923606, ...
     0.1938775217865856 8, 0.1440816936627777, 0.1277551681346758 6,
        0.1179592465897395 2, 0.1179592465897395 2, ...
     0.1146939290279687 1, 0.1138776152077140 6, 0.1106122976459432 4,
        0.1032653798425230 7, 0.0959183686179745 4, ...
     0.0902041407358159 8, 0.0828572229323958 1, 0.082040877971765,
        0.082040877971765, 0.0812245641515103 5, ...
     0.0804082191908795 9, 0.0795918742302488 4, 0.0787755604099941 3,
        0.0755102428482233 2, 0.0714286114661978 6, ...
     0.0624490348818923 3, 0.0493878269155612 8, 0.0346939290279686 8,
        0.3065306039831655, 0.3285714196741782 4, ...
     0.3506122353651909, 0.3726530354860156, 0.3979591531686112,
        0.424897914061904, 0.4461224003624739 7, ...
     0.4689795454439294, 0.4869387063976346];
% Interpolation using Piecewise Cubic Hermite Interpolating Polynomial
    (pchip)
U_ocp = interp1(x, U, SOC, 'pchip');
end
```

3.C. Understanding Newton Raphson Method

Listing 3: Small code for understanding Newton Raphson method

```
clc
clearvars
```

```

% Initial Guess
x = 0.5;

% Euler's number (e)
e = exp(1); % Value of Euler's number

% Iterative Process (Newton-Raphson Method)
while true
    x_new = x - (e^x - x - 1) / (e^x - 1);

    % Check if the absolute difference between the current and new
       approximation is less than 1e-6
    if (norm(x_new - x) < 1e-6)
        break
    end

    x = x_new;
end
```

4. Plots

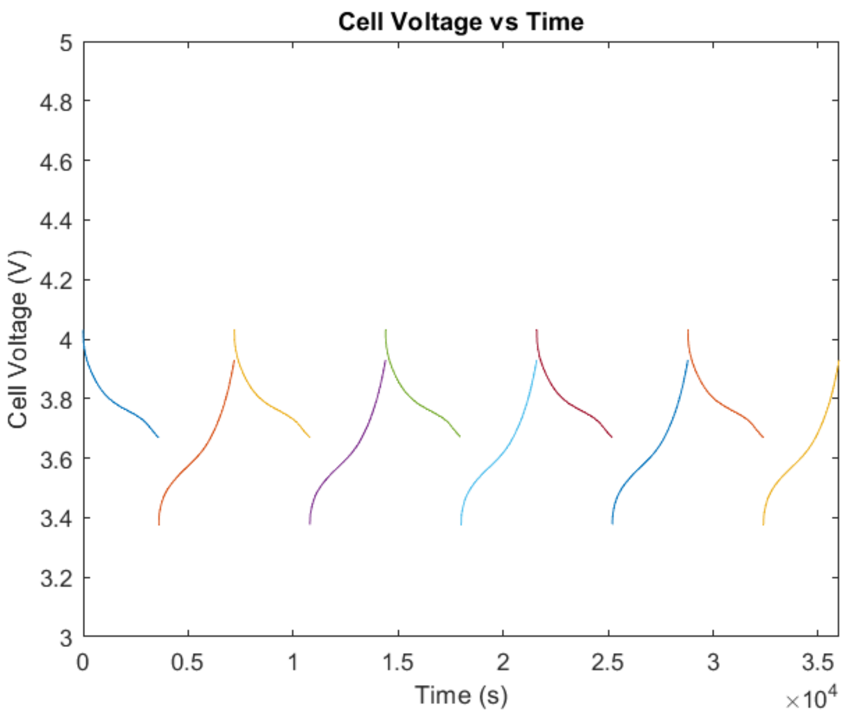

Figure 1: Charge and Discharge cycles showing variation of cell voltage with time

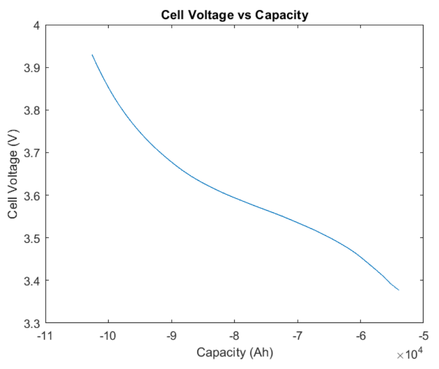

Figure 2: Cell Voltage variation with Discharge Capactiy for one half cycle

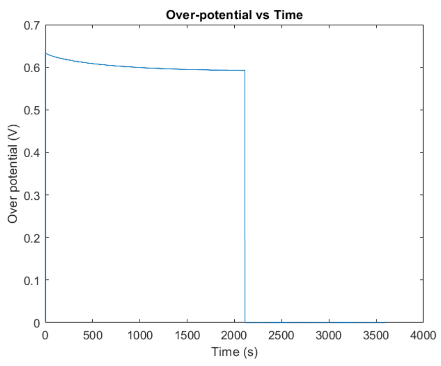

Figure 3: Overpotential vs Time for one half cycle

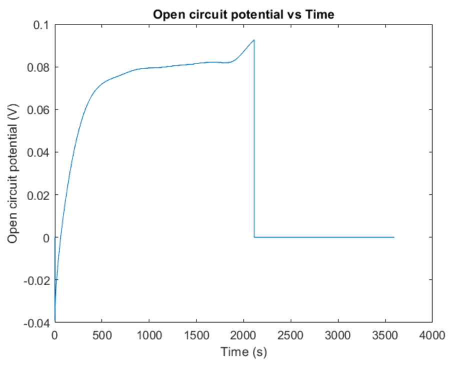

Figure 4: Open Circuit Potential vs Time for one half cycle

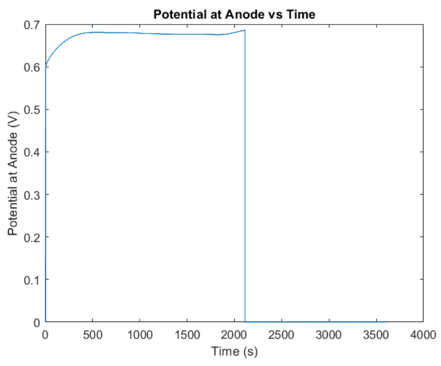

Figure 5: Anode Potential vs Time for one half cycle