```
clear; clc;

% Folder base path
baseFolder = "C:\Users\owner's\Desktop\matlab hands on\IDP_MATLAB";

% Battery nominal capacity (Ah)
C_rated_Ah = 2; % change accordingly
C_rated_As = C_rated_Ah * 3600; % convert Ah to As

% Initialize variables for concatenated plot
time_all = [];
SOC_all = [];

% Initial SOC at start of first cycle (assumed fully charged)
SOC_initial = 1;

% Running time offset for continuous time axis
time_offset = 0;

for cycleNum = 1:100
    fprintf('Processing cycle %d...\n', cycleNum);

    % Construct folder and filenames
    cycleFolder = fullfile(baseFolder, ['cycle' num2str(cycleNum)]);
    chargeFile = fullfile(cycleFolder, ['charge' num2str(cycleNum) '.csv']);
    dischargeFile = fullfile(cycleFolder, ['discharge' num2str(cycleNum) '.csv']);

    % Read charge and discharge data tables
    chargeData = readtable(chargeFile);
    dischargeData = readtable(dischargeFile);

    % Extract time and currents
    charge_time = chargeData.Time;
    charge_current = chargeData.Current_charge; % positive charging current

    discharge_time = dischargeData.Time;
    discharge_current = dischargeData.Current_load; % positive load current

    % Calculate SOC during charging (cumulative integration)
    SOC_charge = SOC_initial + cumtrapz(charge_time, charge_current) / C_rated_As;

    % Adjust discharge current sign to negative (discharging)
    discharge_current = -abs(discharge_current);

    % Calculate SOC during discharging starting from end of charging SOC
    SOC_discharge = SOC_charge(end) + cumtrapz(discharge_time, discharge_current) / ↙
C_rated_As;

    % Combine time vectors, adjusting discharge time to be continuous after charge
    time_cycle = [charge_time; discharge_time + charge_time(end)];
    SOC_cycle = [SOC_charge; SOC_discharge];
    % Offset time for continuous plotting across cycles
    time_cycle = time_cycle + time_offset;

    % Append cycle data to full arrays
    time_all = [time_all; time_cycle];
    SOC_all = [SOC_all; SOC_cycle];

    % Update initial SOC for next cycle as last SOC of this cycle
    SOC_initial = SOC_cycle(end);

    % Update time offset for next cycle
    time_offset = time_all(end);
end

% Plot continuous SOC profile over 100 cycles
figure;
plot(time_all, SOC_all, 'LineWidth', 2);
grid on;
xlabel('Time (seconds)');
ylabel('State of Charge (SOC)');
title('Battery SOC over 100 Charge-Discharge Cycles');
ylim([0 1.1]);
% Create a table with time and SOC
resultTable = table(time_all, SOC_all, 'VariableNames', {'Time_seconds', 'SOC'});
for cycleNum = 1:100
    fprintf('Processing cycle %d...\n', cycleNum);

    % File paths for current cycle
    cycleFolder = fullfile(baseFolder, ['cycle' num2str(cycleNum)]);
    chargeFile = fullfile(cycleFolder, ['charge' num2str(cycleNum) '.csv']);
    dischargeFile = fullfile(cycleFolder, ['discharge' num2str(cycleNum) '.csv']);

    % Read charge and discharge data
    chargeData = readtable(chargeFile);
    dischargeData = readtable(dischargeFile);

    % Extract relevant data
    charge_time = chargeData.Time;
    charge_current = chargeData.Current_charge;  % Positive charging current

    discharge_time = dischargeData.Time;
    discharge_current = dischargeData.Current_load; % Positive load current

    % Calculate SOC during charging
    SOC_charge = SOC_initial + cumtrapz(charge_time, charge_current) / C_rated_As;

    % Discharge current is negative (current out of battery)
    discharge_current = -abs(discharge_current);

    % Calculate SOC during discharging starting from end of charging SOC
    SOC_discharge = SOC_charge(end) + cumtrapz(discharge_time, discharge_current) /↙
C_rated_As;

    % Combine time and SOC vectors for this cycle
    % Note: time is relative within cycle (no global offset here)
    time_cycle = [charge_time; discharge_time + charge_time(end)];
    SOC_cycle = [SOC_charge; SOC_discharge];

    % Save SOC and time for this cycle in separate CSV file
    cycleTable = table(time_cycle, SOC_cycle, 'VariableNames', {'Time_seconds', 'SOC'});
    filename = fullfile(baseFolder, ['SOC_cycle' num2str(cycleNum) '.csv']);
    writetable(cycleTable, filename);

    % Update SOC_initial for next cycle to carry over SOC
    SOC_initial = SOC_cycle(end);
end
```