Technische Universität Berlin
Faculty for Electrical Engineering and Computer Science
Professor Energieversorgungsnetze und Integration Erneuerbarer Energien
Renewable Energy Technology for WS 20/21

Laboratory Assignment 1

Renewable Energy Technology WS 20/21
Professor: Kai Strunz
Submission date: 08 January 2021

Name	Matriculation number
Heesun Joo	16905214327
Hutomo Rachmat Saleh	461980

Structure

Chapter 1. Exercise (1) Sun Elevation and Azimuth

Description

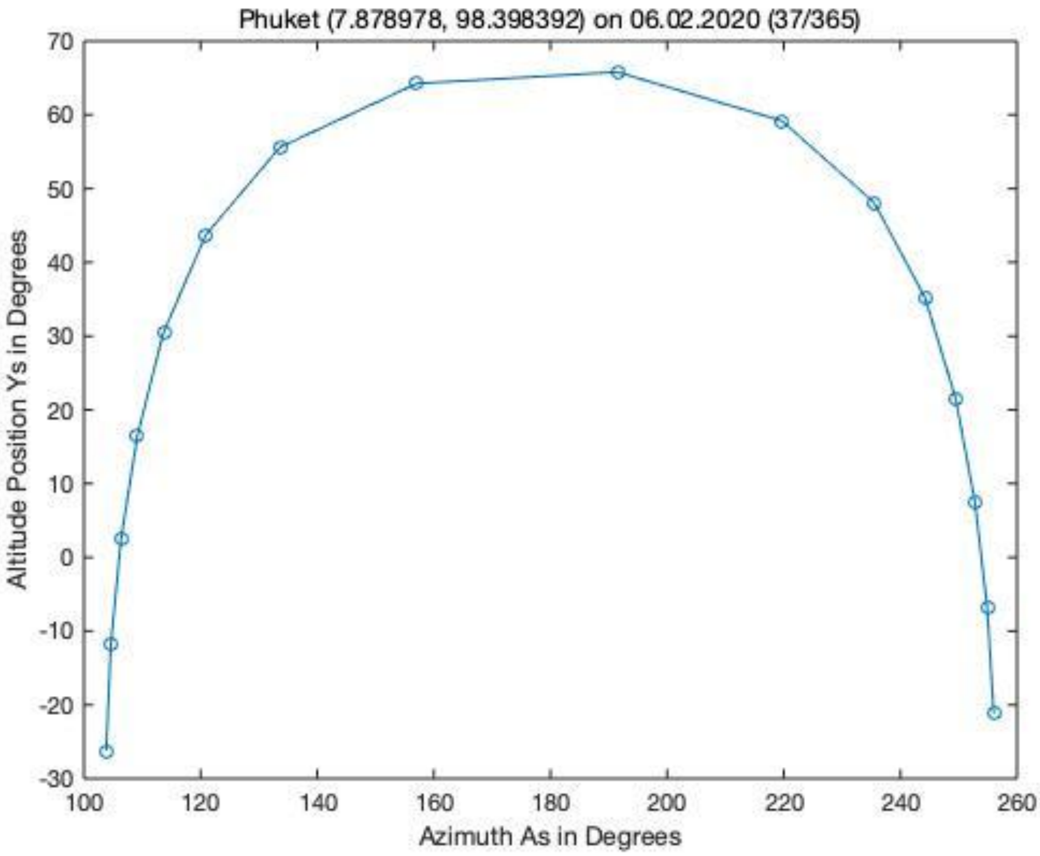
- Calculate the position of the sun and Air Mass factor from 5 am to 8 pm on the 06. February 2020 and plot the results.

Code:

```
%% 1. Assignment
function Assignment_1()
    %% Variables
    DOY = 37; % Day of the year, 6th of February 2020
    LT = (5:20);  % Local time, 5am to 20pm
    TZ = 7;  % Phuket time zone (UTF + 7)
    latitude = 7.878978;  % in degrees
    longitude = 98.398392;

    %% Plot
    [air_mass, sun_elevation, sun_azimuth] = SunDate(latitude, longitude,
DOY, LT, TZ);
    figure(1);
    plot(sun_azimuth, sun_elevation, '-o');
    title('Phuket (7.878978, 98.398392) on 06.02.2020 (37/365)')
    ylabel('Altitude Position Ys in Degrees')
    xlabel('Azimuth As in Degrees')
end

function [AM, sun_elevation, sun_azimuth] = SunDate(latitude, longitude,
DOY, LT, TZ)
    J = 360 * DOY / 365;  % Day angle [?]
    declination = 0.3948 ...  % [?]
                - 23.2559 * cosd(J + 9.1) ...
                - 0.3915 * cosd(2*J + 5.4) ...
                - 0.176 * cosd(3*J + 26);

    % Time Equation (In Minutes)
    TEQ = (0.0066 + 7.3525 * cosd(J + 85.9) ...
         + 9.9359 * cosd(2*J + 108.9) ...
         + 0.3387 * cosd(3*J + 105.2));

    % Preallocate matrix size
    AM = zeros(1, length(LT));
    sun_azimuth = zeros(1, length(LT));
    sun_elevation = zeros(1, length(LT));

    for i = 1:length(LT)
        % True local time (in Hour)
        TLT = LT(i) - TZ + (4 * longitude + TEQ) / 60;
        % Hour angle
        w = (12 - TLT) * 15;
        % Sun elevation 0? - 90?
        sun_elevation(i) = asind(cosd(w) * cosd(latitude) *
cosd(declination) ...
                          + sind(latitude) * sind(declination));
```

```
        % Air mass
        AM(i) = 1 / sind(sun_elevation(i));
        % Sun azimuth
        azimuth = acosd((sind(sun_elevation(i)) * sind(latitude) -
sind(declination)) ...
                    / (cosd(sun_elevation(i)) * cosd(latitude)));
        if (TLT > 12)
            sun_azimuth(i) = 180 + azimuth;
        else
            sun_azimuth(i) = 180 - azimuth;
        end
    end
end
```

Results (Plots and Calculation) including explanation:



Interpretation:

The result of the plot is similar to the example plot shown in the slides, the difference being only the location of the simulation, which is in Berlin. Due to the location of Phuket, Thailand being located near the equator line, the result shows an average of higher altitude position throughout the day.

Chapter 2. Exercise (2) Model of Photovoltaic Module

Description:

- Models the described PV module.
- Plot the voltage-current and voltage-power characteristics under STC.

  Rs= 0.007 Ω, Rsh = 530 Ω and γ = 1.3 (Diode Quality factor).
- Describe the impact of the three parameters (Rs, Rsh, γ).
- Discuss the effects of increasing and decreasing the temperature and irradiance.
- Plot the results.

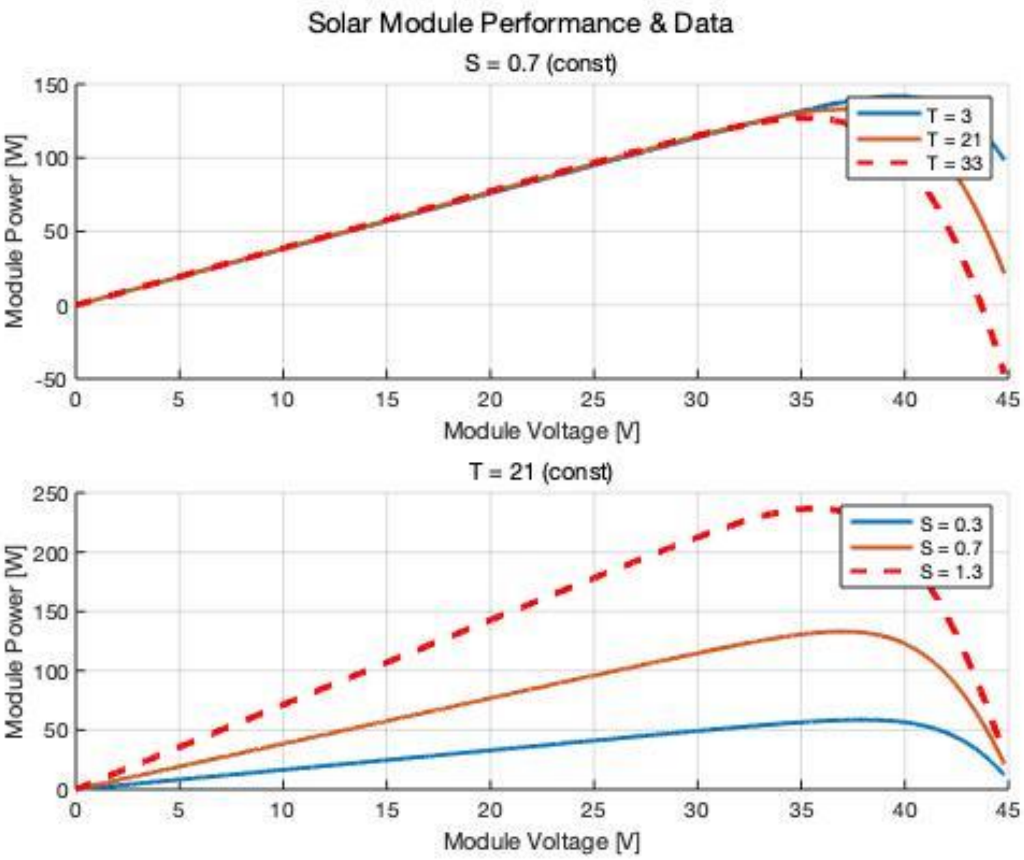
Code:

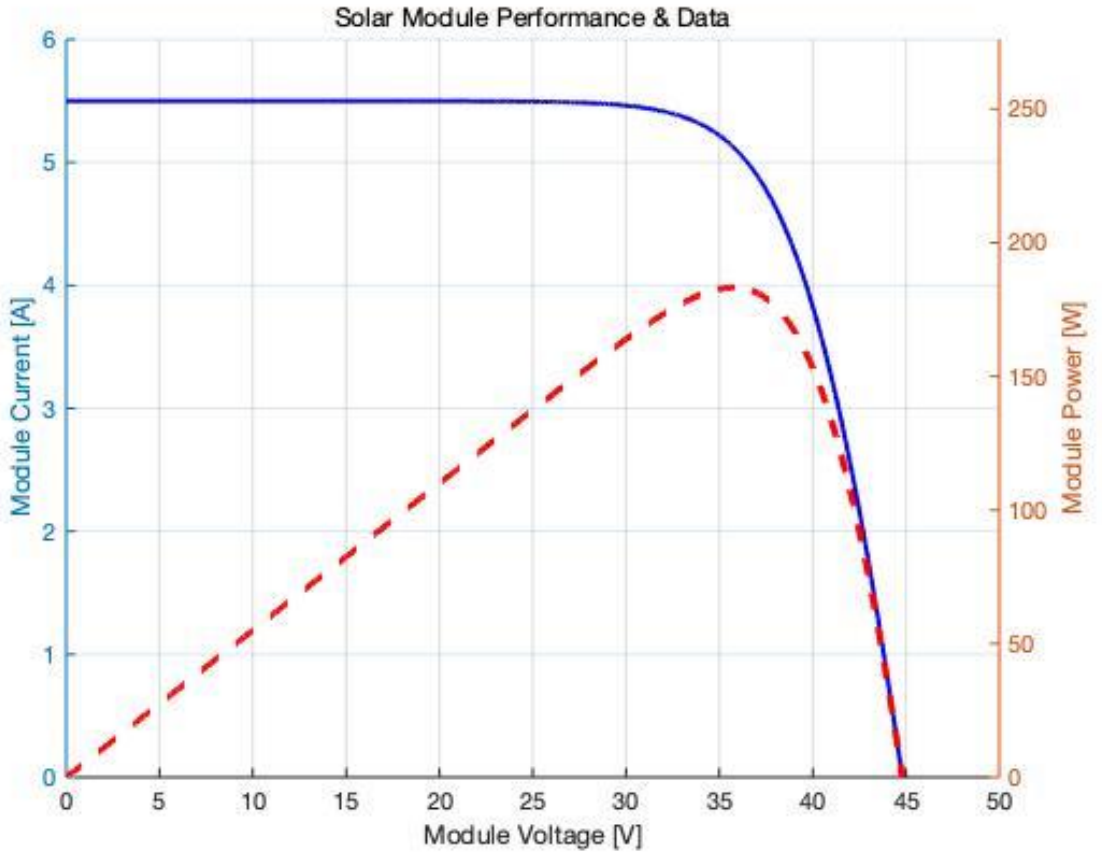
```
%% 2. Assignment
function Assignment_2()
    %% Variables
    V = 0:0.01:44.8;
    I = PVmod(V, 1, 25);
    P = V .* I;

    %% Plot
    figure(2);
    title('Solar Module Performance & Data')
    xlim([0 50])
    grid on

    yyaxis left
    hold on
    plot(V, I, 'b', 'linewidth', 2)
    ylim([0 6])
    xlabel('Module Voltage [V]')
    ylabel('Module Current [A]')

    yyaxis right
    hold on
    plot(V, P, '--r', 'linewidth', 2)
    ylim([0 276])
    ylabel('Module Power [W]')

    %% Effects of temperature and irradiance
    S_var = [0.3, 0.7, 1.3];
    T_var = [3, 21, 33];

    % Matrix preallocation
    N = length(T_var);
    I_1 = zeros(N, length(I));
    I_2 = zeros(N, length(I));
    P_1 = zeros(N, length(I));
    P_2 = zeros(N, length(I));
    for i=1:N
        I_1(i, :) = PVmod(V, 0.7, T_var(i)); % Fixed S, varying T
```

```matlab
        P_1(i, :) = V .* I_1(i, :);
        I_2(i, :) = PVmod(V, S_var(i), 21); % Fixed T, varying S
        P_2(i, :) = V .* I_2(i, :);
    end

    figure(3);

    subplot(2,1,1)
    title('S = 0.7 (const)')
    grid on
    hold on
    plot(V, P_1(1, :), V, P_1(2, :), V, P_1(3, :), '--r', 'linewidth', 2)
    legend('T = 3', 'T = 21', 'T = 33');
    ylabel('Module Power [W]')
    xlabel('Module Voltage [V]')

    subplot(2,1,2)
    title('T = 21 (const)')
    grid on
    hold on
    plot(V, P_2(1, :), V, P_2(2, :), V, P_2(3, :), '--r', 'linewidth', 2)
    legend('S = 0.3', 'S = 0.7', 'S = 1.3');
    xlabel('Module Voltage [V]')
    ylabel('Module Power [W]')

    sgtitle('Solar Module Performance & Data')

end

function I = PVmod(V, S, T)
    %% Variables
    n = 72;  % Number of PV
    V = V / n;
    I_SC = 5.5;  % Short circuit current [A]
    k_o = 65e-3 / 100;  % Temp coeff of I_SC [%]
    S1 = 1;  % STC irradiance [kW / m^2]
    T = T + 273;  % Current Temperature [K]
    T1 = 25 + 273;  % STC temperature [K]
    q = 1.6e-19;  % Elementary charge [C]
    K = 1.381e-23;  % Boltzmann Constant
    gamma = 1.3;  % Diode Quality Factor

    V_G = 1.12;  % Band gap voltage
    V_OC = 44.8 / n;  % Open curcuit voltage

    R_S = 0.007;  % Resistance in series
    R_SH = 530;  % Resistance of shunt

    %% Calculate I_L
    if (S == S1)
        I_L = I_SC;
    else
        I_L = I_SC * (S / S1) * (1 + k_o * (T - T1));
    end

    %% Calculate I_S
    I_S1 = I_L / (exp((q * V_OC) / (gamma * K * T1)) - 1);
    I_S = I_S1 * (T / T1) ^ (3 / gamma) * exp((-q * V_G * (inv(T) -
```

```
inv(T1))) / (gamma * K));

    %% Newton Iteration
    I = 0;  % Start value
    for i = 1:1:10  % 10 Iteration
        y = I_L - I ...
            - I_S * (exp(q * (V + I * R_S) / (gamma * K * T)) - 1) ...
            - (V + I * R_S) / R_SH;
        dy = -1 - (I_S * q * R_S / (gamma * K * T)) ...
             * exp(q * (V + I * R_S) / (gamma * K * T)) ...
             - R_S / R_SH;
        I = I - y ./ dy;
    end
end
```

Results (Plots and Calculation) including explanation:





Interpretation:

By lowering the value of Rsh we will see a decrease in performance. In contrast, a decrease in performance will occur with higher value of Rs value.

One of the problems of solar cell is to keep the temperature as low as possible. The equation also shows the dependency of T with the output I. It shows that with increasing T, the V and therefore P decreases. This is supported with the result shown in Figure 3.

By definition irradiance is the power per unit area received from the sun. It is therefore expected that with increasing irradiance, the power also increases. The result shown in the graph supports this hypothesis.

Chapter 3. Exercise (3) Module Evaluation Parameters

Description:

- Calculate the fill factor and the efficiency of the given module 1740*1030*32 mm^3 under STC.

Code:

```
%% 3. Assignment
function Assignment_3()
    n = 72;
    R_S = 0.007;  % Resistance in series
    R_SH = 530;  % Resistance of shunt
    gamma = 1.3;  % Diode Quality Factor
    I_MPP = 5.1;
    V_MPP = 36.5 / n;
    V_OC = 44.8 / n;
    I_SC = 5.5;
    A = 1740e-3*1030e-3*32e-3;  % Area [m^3]
    S = 1;  % STC irradiance [kW / m^2]

    FF = V_MPP * I_MPP / (V_OC * I_SC);
    efficiency = V_MPP * I_MPP / (A * S);
    disp("Fill Factor: " + FF*100 + " %")
    disp("Efficiency: " + efficiency + " %")
end
```

Results (Plots and Calculation) including explanation:

Fill Factor: 75.5479 %

Efficiency: 45.0811 %

Chapter 4. Exercise (4) Loading PV Modules

Description:

- Develop the MATLAB function that can find the possible values of the pure resistive load to be interfaced to the module keeping it working at the maximum power point MPP for any given irradiance and temperature conditions.
- Fill in the following table.

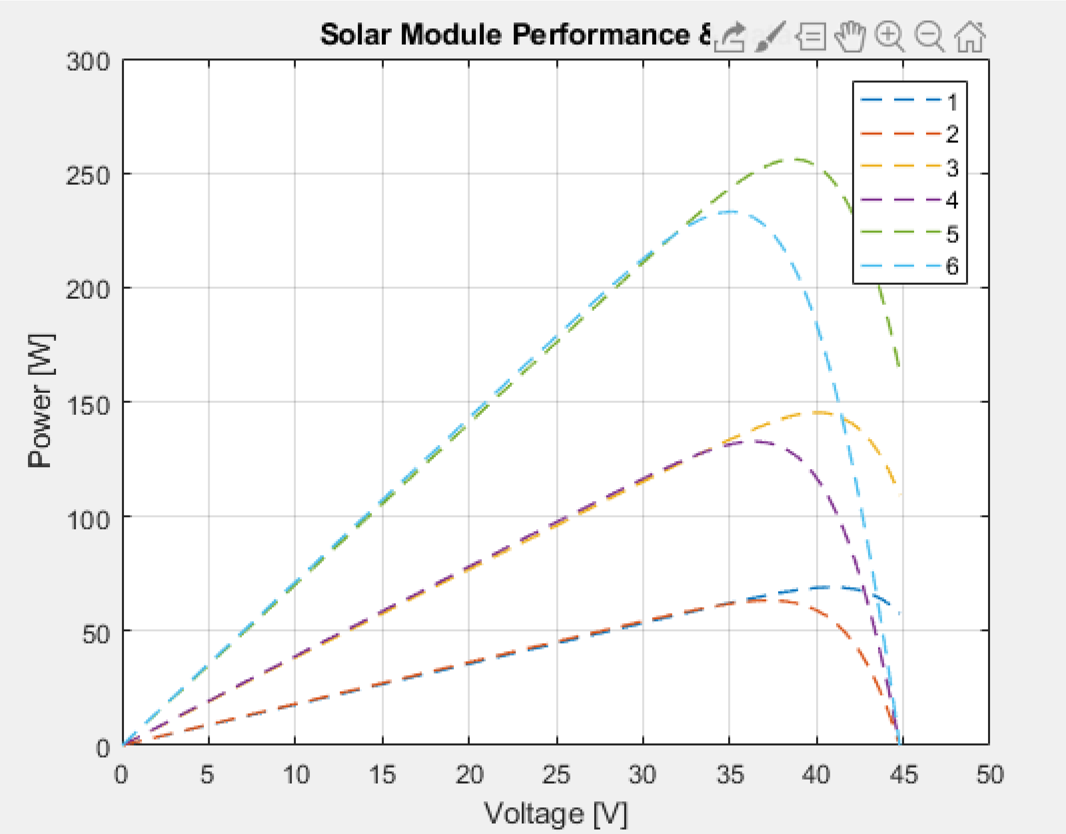
	(1)	(2)	(3)	(4)	(5)	(6)
Irradiance (W/m^2)	330	330	710	710	1300	1300
Temperature (Celcius)	0	25	0	25	0	25
Resistive Load (Ω)	24.1909	21.905	11.025	9.972	5.8426	5.2813

Code:

```
%% 4. Assignment
function Assignment_4()
    %% Variables
    S = [330 330 710 710 1300 1300] * 1e-3;
    T = [0 25 0 25 0 25];
    V = 0:0.01:44.8;
    N = length(S);
    N2 = length(V);
    I = zeros(N, N2);
    P = zeros(N, N2);
    for i = 1:N
        I(i, :) = PVmod(V, S(i), T(i));
        P(i, :) = I(i, :) .* V;
    end

    % Get max P
    R_max = zeros(N, 1);
    for i = 1:N
        P_iter = 0;
        for j = 1:N2
            % If power decrease, Pmax is found
            if (P_iter > P(i, j))
                R_max(i) = V(1, j) / I(i, j);
                break
            end
            P_iter = P(i, j);
        end
    end

    disp(R_max);


    %% Plot
```

```matlab
    figure(2);
    for i = 1:N
        plot(V, P(i, :), '--', 'linewidth', 1)
        hold on
    end
    title('Solar Module Performance & Data')
    xlabel('Power [W]')
    ylabel('Voltage [V]')
    ylim([0 300])
    xlim([0 50])
    grid on

end

function I = PVmod(V, S, T)
    %% Variables
    n = 72;  % Number of PV
    V = V / n;
    I_SC = 5.5;  % Short circuit current [A]
    k_o = 65e-3 / 100;  % Temp coeff of I_SC [%]
    S1 = 1;  % STC irradiance [kW / m^2]
    T = T + 273;  % Current Temperature [K]
    T1 = 25 + 273;  % STC temperature [K]
    q = 1.6e-19;  % Elementary charge [C]
    K = 1.381e-23;  % Boltzmann Constant
    gamma = 1.3;  % Diode Quality Factor

    V_G = 1.12;  % Band gap voltage
    V_OC = 44.8 / n;  % Open curcuit voltage

    R_S = 0.007;  % Resistance in series
    R_SH = 530;  % Resistance of shunt

    %% Calculate I_L
    if (S == S1)
        I_L = I_SC;
    else
        I_L = I_SC * (S / S1) * (1 + k_o * (T - T1));
    end

    %% Calculate I_S
    I_S1 = I_L / (exp((q * V_OC) / (gamma * K * T1)) - 1);
    I_S = I_S1 * (T / T1) ^ (3 / gamma) * exp((-q * V_G * (inv(T) -
inv(T1))) / (gamma * K));

    %% Newton Iteration
    I = 0;  % Start value
    for i = 1:1:10  % 10 Iteration
        y = I_L - I ...
            - I_S * (exp(q * (V + I * R_S) / (gamma * K * T)) - 1) ...
            - (V + I * R_S) / R_SH;
        dy = -1 - (I_S * q * R_S / (gamma * K * T)) ...
             * exp(q * (V + I * R_S) / (gamma * K * T)) ...
             - R_S / R_SH;
        I = I - y ./ dy;
    end
end
```

Results (Plots and Calculation) including explanation:



Resistive load 1: 24.1909

Resistive load 2: 21.9057

Resistive load 3: 11.0255

Resistive load 4: 9.9722

Resistive load 5: 5.8426

Resistive load 6: 5.2813

Interpretation:

We have discussed the effects of Irradiation and Temperature to the power output in Chapter 2. In this chapter we further explore the relationship. We calculated the pure resistive load for the solar module to keep working at its maximum power. It is noticeable that with increasing Irradiance, the pure resistive load decreases. This is due to the power also increasing. Although with increasing temperature, the power output decreases, the pure resistive load marginally decreases.

Chapter 5. Exercise (5) Maximum Power Point Tracking MPPT

Description:

- Calculate the percentage power loss under STC compared to the case of using ideal MPPT for any values of the resistive loads.
- Fill in the following table.

Load (Ω)	0	9	29	57	110
Power loss (%)	100	8.2	64.5	81.6	90.1

Code:

```
%% 5. Assignment
function Assignment_5()
    %% Variables
    R = [0 9 29 57 110];
    S = 1;
    T = 25;
    V = 0:0.01:44.8;
    nV = length(V);
    nR = length(R);
    I = PVmod(V, S, T);
    P = I .* V;
    load_index = zeros(nR, 1);

    % Get max P
    P_iter = 0;
    for j = 1:nV
        % If power decrease, Pmax is found
        if (P_iter > P(1, j))
            P_max = P(1, j); % Get maximum power
            R_ideal = V(1, j) / I(1, j);
            disp("Ideal load is: " + R_ideal + " Ohm")
            break
        end
        P_iter = P(1, j);
    end

    % Get index of equivalent V & I
    for i = 1:nR
        [~, load_index(i)] = min(abs(V./I - R(i)));
    end

    P_R = I(load_index).^2 .* R; % Calculate load power

    % Compare maximum power and calculate power loss
    P_loss = (P_max - P_R) ./ P_max .* 100;
    P_loss;
end

function I = PVmod(V, S, T)
    %% Variables
```

```
    n = 72;  % Number of PV
    V = V / n;
    I_SC = 5.5;  % Short circuit current [A]
    k_o = 65e-3 / 100;  % Temp coeff of I_SC [%]
    S1 = 1;  % STC irradiance [kW / m^2]
    T = T + 273;  % Current Temperature [K]
    T1 = 25 + 273;  % STC temperature [K]
    q = 1.6e-19;  % Elementary charge [C]
    K = 1.381e-23;  % Boltzmann Constant
    gamma = 1.3;  % Diode Quality Factor

    V_G = 1.12;  % Band gap voltage
    V_OC = 44.8 / n;  % Open curcuit voltage

    R_S = 0.007;  % Resistance in series
    R_SH = 530;  % Resistance of shunt

    %% Calculate I_L
    if (S == S1)
        I_L = I_SC;
    else
        I_L = I_SC * (S / S1) * (1 + k_o * (T - T1));
    end

    %% Calculate I_S
    I_S1 = I_L / (exp((q * V_OC) / (gamma * K * T1)) - 1);
    I_S = I_S1 * (T / T1) ^ (3 / gamma) * exp((-q * V_G * (inv(T) -
inv(T1))) / (gamma * K));

    %% Newton Iteration
    I = 0;  % Start value
    for i = 1:1:10  % 10 Iteration
        y = I_L - I ...
            - I_S * (exp(q * (V + I * R_S) / (gamma * K * T)) - 1) ...
            - (V + I * R_S) / R_SH;
        dy = -1 - (I_S * q * R_S / (gamma * K * T)) ...
             * exp(q * (V + I * R_S) / (gamma * K * T)) ...
             - R_S / R_SH;
        I = I - y ./ dy;
    end
end
```

Results (Plots and Calculation) including explanation:

Ideal load is: 6.9727 Ohm

Interpretation:

When no load is connected, no amount of power can be utilized. This corresponds to the result shown with zero load. This is also the same case when too much load is connected. Since the ideal load is 6.97 Ohm, the minimal loss power occurs at 9 Ohm. Therefore, there is a need to use a maximum power point tracker to maximize the power output. This is further discussed in the following chapters.

Chapter 6. Exercise (6) MPPT Techniques

Description:

- Develop a MATLAB function that perform the maximum power point tracking MPPT through pulse width modulation of a buck-boost converter based on the Hill Climbing Algorithm assuming a fixed step for voltage change of 1 volt. Converter output voltage is kept fixed at 32 V.
- Calculate the duty cycle for the buck-boost converter.
- Plot the duty cycle D and Vnew.
- What should be the starting value for Vi?

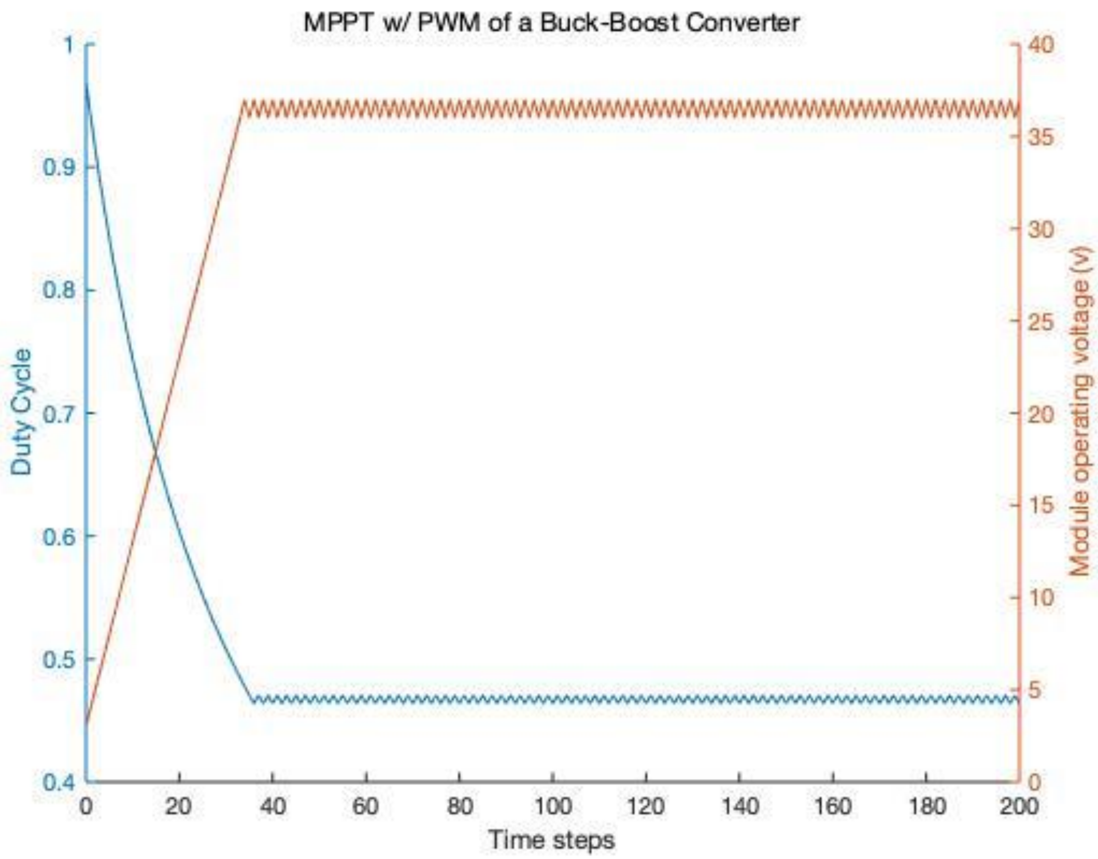
Code:

```
%% 6. Assignment
function Assignment_6()
    % S and T in standard condition
    S = 1; % [kW/m2]
    T = 25; % [C]

    V_step = 1;
    t_step = 1; % time step
    t_span = 0:t_step:2e2;
    [D, V_new] = PVHC(S, T, t_span, V_step);

    %% Plot
    figure(6);
    title('MPPT w/ PWM of a Buck-Boost Converter')
    xlabel('Time steps')

    yyaxis left
    hold on
    plot(t_span, D);
    ylabel('Duty Cycle')

    yyaxis right
    hold on
    plot(t_span, V_new);
    ylabel('Module operating voltage (v)')

end

function [D, V_new] = PVHC(S, T, t_span, V_step)
    % Variables
    V_in = 1; % Start value
    V_out = 32;

    % Calculate Duty Cycle
    D = zeros(1, length(t_span));
    V_new = ones(1, length(t_span));
```

```matlab
    for i = 1:length(t_span)
        D(i) = V_out ./ (V_in + V_out);
        if i == 1
            % Start values
            V_old = V_in;
            V_in = V_old + 1;
            V_new(i) = HClimb(V_old, V_in, V_step, S, T);
            V_old = V_in;
        else
            V_in = V_new(i-1);
            V_new(i) = HClimb(V_old, V_in, V_step, S, T);
            V_old = V_in;
        end
    end
end

function V_new = HClimb(V_old, V_cur, V_step, S, T)
    I_old = PVmod(V_old, S, T);
    I_cur = PVmod(V_cur, S, T);

    P_old = I_old * V_old;
    P_cur = I_cur * V_cur;

    % Compare
    if P_cur > P_old
        V_new = V_cur + V_step;
    elseif P_cur < P_old
        V_new = V_cur - V_step;
    else
        V_new = V_cur;
    end

end

function I = PVmod(V, S, T)
    % Returns current of PV module under certain parameters.
    % Useful for calculating power.

    %% Variables
    n = 72;  % Number of PV
    V = V / n;
    I_SC = 5.5;  % Short circuit current [A]
    k_o = 65e-3 / 100;  % Temp coeff of I_SC [%]
    S1 = 1;  % STC irradiance [kW / m^2]
    T = T + 273;  % Current Temperature [K]
    T1 = 25 + 273;  % STC temperature [K]
    q = 1.6e-19;  % Elementary charge [C]
    K = 1.381e-23;  % Boltzmann Constant
    gamma = 1.3;  % Diode Quality Factor

    V_G = 1.12;  % Band gap voltage
    V_OC = 44.8 / n;  % Open curcuit voltage

    R_S = 0.007;  % Resistance in series
    R_SH = 530;  % Resistance of shunt

    %% Calculate I_L
```

```
    if (S == S1)
        I_L = I_SC;
    else
        I_L = I_SC * (S / S1) * (1 + k_o * (T - T1));
    end

    %% Calculate I_S
    I_S1 = I_L / (exp((q * V_OC) / (gamma * K * T1)) - 1);
    I_S = I_S1 * (T / T1) ^ (3 / gamma) * exp((-q * V_G * (inv(T) -
inv(T1))) / (gamma * K));

    %% Newton Iteration
    I = 0;  % Start value
    for i = 1:1:10  % 10 Iteration
        y = I_L - I ...
            - I_S * (exp(q * (V + I * R_S) / (gamma * K * T)) - 1) ...
            - (V + I * R_S) / R_SH;
        dy = -1 - (I_S * q * R_S / (gamma * K * T)) ...
             * exp(q * (V + I * R_S) / (gamma * K * T)) ...
             - R_S / R_SH;
        I = I - y ./ dy;
    end
end
```

Results (Plots and Calculation) including explanation:



Interpretation:

The starting value of Vi could be any number, since the hill climbing algorithm will near the value to the maximum power point. In our case, since a suitable starting point would be around but below the output voltage of 32V.

Chapter 7. Exercise (7) MPPT Techniques

Description:

- Compare the performance in exercise (6) with the case of a step of 0.2 voltage change regarding the speed of convergence and accuracy of tracking.
- Support your explanation with plots.

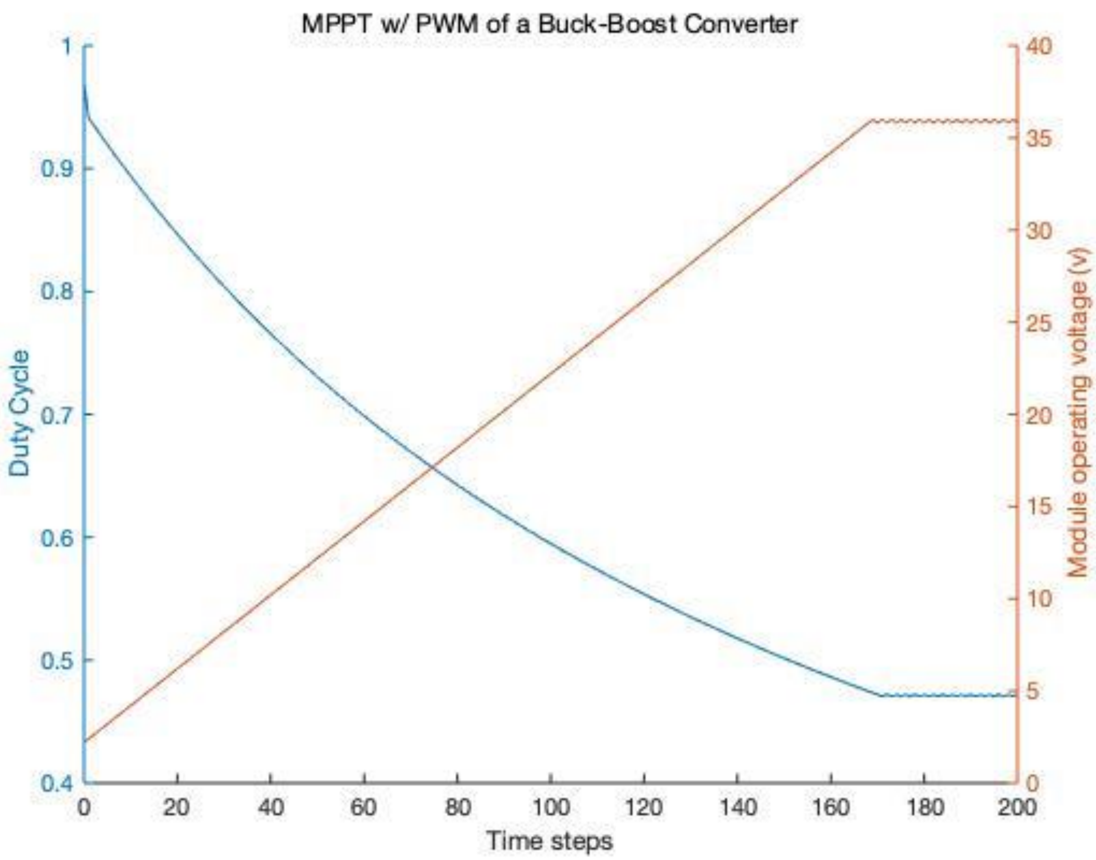
Code:

```
%% 7. Assignment
function Assignment_7()
    % S and T in standard condition
    S = 1; % [kW/m2]
    T = 25; % [C]

    t_step = 1; % time step
    t_span = 0:t_step:2e2;
    [D, V_new] = PVHC(S, T, t_span);

    %% Plot
    figure(6);
    title('MPPT w/ PWM of a Buck-Boost Converter')
    xlabel('Time steps')

    yyaxis left
    hold on
    plot(t_span, D);
    ylabel('Duty Cycle')

    yyaxis right
    hold on
    plot(t_span, V_new);
    ylabel('Module operating voltage (v)')

end

function [D, V_new] = PVHC(S, T, t_span)
    % Variables
    V_in = 1; % Start value
    V_out = 32;

    % Calculate Duty Cycle
    D = zeros(1, length(t_span));
    V_new = ones(1, length(t_span));

    for i = 1:length(t_span)
        D(i) = V_out ./ (V_in + V_out);
        if i == 1
            % Start values
            V_old = V_in;
            V_in = V_old + 1;
            V_new(i) = HClimb(V_old, V_in, S, T);
```

```
            V_old = V_in;
        else
            V_in = V_new(i-1);
            V_new(i) = HClimb(V_old, V_in, S, T);
            V_old = V_in;
        end
    end
end

function V_new = HClimb(V_old, V_cur, S, T)
    V_step = 0.2;

    I_old = PVmod(V_old, S, T);
    I_cur = PVmod(V_cur, S, T);

    P_old = I_old * V_old;
    P_cur = I_cur * V_cur;

    % Compare
    if P_cur > P_old
        V_new = V_cur + V_step;
    elseif P_cur < P_old
        V_new = V_cur - V_step;
    else
        V_new = V_cur;
    end

end

function I = PVmod(V, S, T)
    % Returns current of PV module under certain parameters.
    % Useful for calculating power.

    %% Variables
    n = 72;  % Number of PV
    V = V / n;
    I_SC = 5.5;  % Short circuit current [A]
    k_o = 65e-3 / 100;  % Temp coeff of I_SC [%]
    S1 = 1;  % STC irradiance [kW / m^2]
    T = T + 273;  % Current Temperature [K]
    T1 = 25 + 273;  % STC temperature [K]
    q = 1.6e-19;  % Elementary charge [C]
    K = 1.381e-23;  % Boltzmann Constant
    gamma = 1.3;  % Diode Quality Factor

    V_G = 1.12;  % Band gap voltage
    V_OC = 44.8 / n;  % Open curcuit voltage

    R_S = 0.007;  % Resistance in series
    R_SH = 530;  % Resistance of shunt

    %% Calculate I_L
    if (S == S1)
        I_L = I_SC;
    else
        I_L = I_SC * (S / S1) * (1 + k_o * (T - T1));
    end
```

```
    %% Calculate I_S
    I_S1 = I_L / (exp((q * V_OC) / (gamma * K * T1)) - 1);
    I_S = I_S1 * (T / T1) ^ (3 / gamma) * exp((-q * V_G * (inv(T) -
inv(T1))) / (gamma * K));

    %% Newton Iteration
    I = 0;  % Start value
    for i = 1:1:10  % 10 Iteration
        y = I_L - I ...
            - I_S * (exp(q * (V + I * R_S) / (gamma * K * T)) - 1) ...
            - (V + I * R_S) / R_SH;
        dy = -1 - (I_S * q * R_S / (gamma * K * T)) ...
             * exp(q * (V + I * R_S) / (gamma * K * T)) ...
             - R_S / R_SH;
        I = I - y ./ dy;
    end
end
```

Results (Plots and Calculation) including explanation:



Interpretation:

Compared to the plot from the previous chapter, which uses a larger step size, the plot produced from 0.2 voltage change converges slower but the oscillations after reaching the stable region is smaller. This shows higher accuracy but slower convergence speed on smaller step sizes.

Chapter 8. Exercise (8) MPPT Techniques

Description:

- Develop a MATLAB function that can perform the Adaptive Hill Climbing Algorithm and compare tracking speed and accuracy with the previous two exercises.
- What is the effect of C?
- Support your explanation with plots.

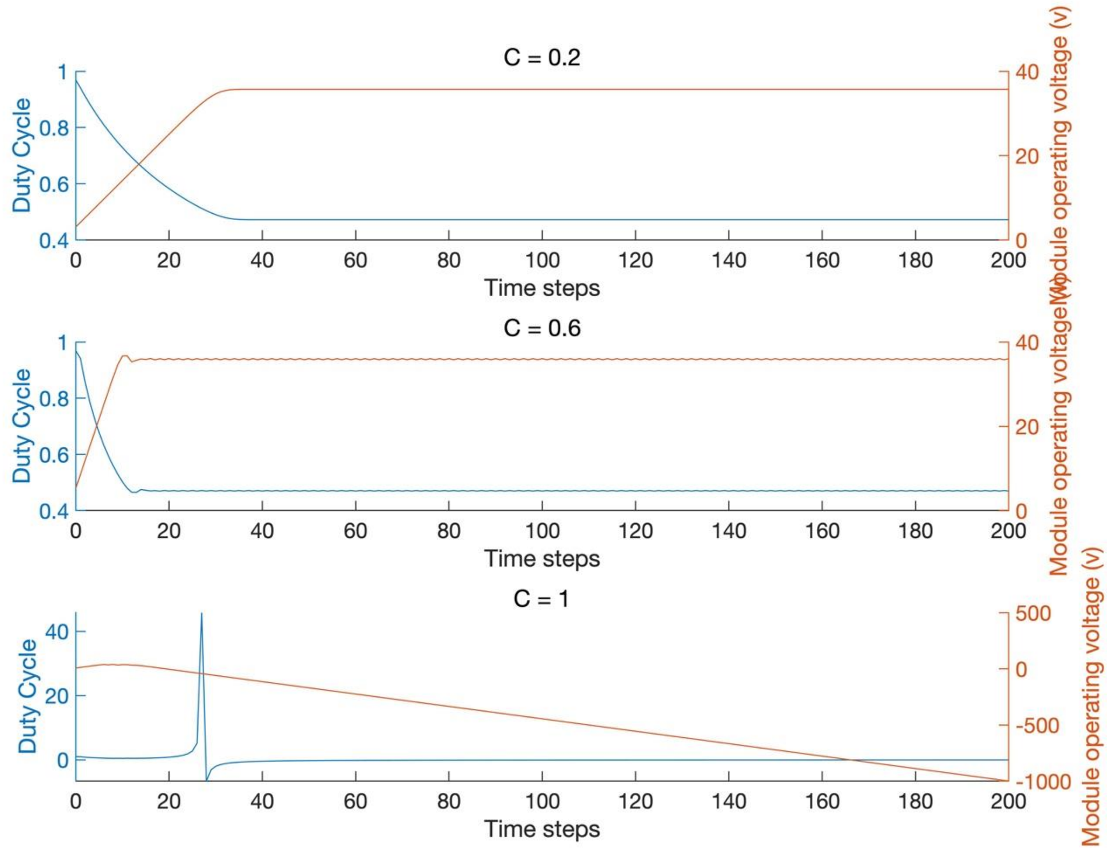
Code:

```
%% 8. Assignment
function Assignment_8()
    % S and T in standard condition
    S = 1; % [kW/m2]
    T = 25; % [C]
    C = [0.2 0.6 1];

    t_step = 1; % time step
    t_span = 0:t_step:2e2;
    D = zeros(length(C), length(t_span));
    V_new = zeros(length(C), length(t_span));
    for i=1:length(C)
        [D(i, :), V_new(i, :)] = PVAHC(S, T, t_span, C(i));
    end

    %% Plot
    figure(5);

    subplot(311)
    title('C = 0.2')
    yyaxis left
    hold on
    plot(t_span, D(1, :));
    ylabel('Duty Cycle')
    yyaxis right
    hold on
    plot(t_span, V_new(1, :));
    xlabel('Time steps')
    ylabel('Module operating voltage (v)')


    subplot(312)
    title('C = 0.6')
    yyaxis left
    hold on
    plot(t_span, D(2, :));
    ylabel('Duty Cycle')
    yyaxis right
    hold on
    plot(t_span, V_new(2, :));
    xlabel('Time steps')
    ylabel('Module operating voltage (v)')
```

```
    subplot(313)
    title('C = 1')
    yyaxis left
    hold on
    plot(t_span, D(3, :));
    ylabel('Duty Cycle')
    yyaxis right
    hold on
    plot(t_span, V_new(3, :));
    xlabel('Time steps')
    ylabel('Module operating voltage (v)')

end

function [D, V_new] = PVAHC(S, T, t_span, C)
    % Variables
    V_in = 1; % Start value
    V_out = 32;

    % Calculate Duty Cycle
    D = zeros(1, length(t_span));
    V_new = ones(1, length(t_span));

    for i = 1:length(t_span)
        D(i) = V_out ./ (V_in + V_out);
        if i == 1
            % Start values
            V_old = V_in;
            V_in = V_old + 1;
            V_new(i) = AHClimb(V_old, V_in, S, T, C);
            V_old = V_in;
        else
            V_in = V_new(i-1);
            V_new(i) = AHClimb(V_old, V_in, S, T, C);
            V_old = V_in;
        end
    end
end

function V_new = AHClimb(V_old, V_cur, S, T, C)
    I_old = PVmod(V_old, S, T);
    I_cur = PVmod(V_cur, S, T);

    P_old = I_old * V_old;
    P_cur = I_cur * V_cur;
    dP = abs(P_old - P_cur);
    dV = abs(V_old - V_cur);

    V_step = C * dP / dV;

    % Compare
    if P_cur > P_old
        V_new = V_cur + V_step;
    elseif P_cur < P_old
        V_new = V_cur - V_step;
    end
end
```

```
function I = PVmod(V, S, T)
    % Returns current of PV module under certain parameters.
    % Useful for calculating power.

    %% Variables
    n = 72;  % Number of PV
    V = V / n;
    I_SC = 5.5;  % Short circuit current [A]
    k_o = 65e-3 / 100;  % Temp coeff of I_SC [%]
    S1 = 1;  % STC irradiance [kW / m^2]
    T = T + 273;  % Current Temperature [K]
    T1 = 25 + 273;  % STC temperature [K]
    q = 1.6e-19;  % Elementary charge [C]
    K = 1.381e-23;  % Boltzmann Constant
    gamma = 1.3;  % Diode Quality Factor

    V_G = 1.12;  % Band gap voltage
    V_OC = 44.8 / n;  % Open curcuit voltage

    R_S = 0.007;  % Resistance in series
    R_SH = 530;  % Resistance of shunt

    %% Calculate I_L
    if (S == S1)
        I_L = I_SC;
    else
        I_L = I_SC * (S / S1) * (1 + k_o * (T - T1));
    end

    %% Calculate I_S
    I_S1 = I_L / (exp((q * V_OC) / (gamma * K * T1)) - 1);
    I_S = I_S1 * (T / T1) ^ (3 / gamma) * exp((-q * V_G * (inv(T) -
inv(T1))) / (gamma * K));

    %% Newton Iteration
    I = 0;  % Start value
    for i = 1:1:10  % 10 Iteration
        y = I_L - I ...
            - I_S * (exp(q * (V + I * R_S) / (gamma * K * T)) - 1) ...
            - (V + I * R_S) / R_SH;
        dy = -1 - (I_S * q * R_S / (gamma * K * T)) ...
             * exp(q * (V + I * R_S) / (gamma * K * T)) ...
             - R_S / R_SH;
        I = I - y ./ dy;
    end
end
```

Results (Plots and Calculation) including explanation:



Interpretation:

Varying C produces similar effect as varying the V_step directly. With smaller C value, the accuracy becomes better but the speed of convergence will become smaller. After a certain point, the V_step becomes too large and will lead to unexpected values. This is shown when C is equal to 1. The value does not converge to a stable value.